Application of Reinforcement Learning Methods to Computer Game Dynamics

David John Booth

School of Computing and Digital Media

London Metropolitan University

This dissertation is submitted for the degree of

Doctor of Philosophy

January 2018

Abstract

The dynamics of the game world present both challenges and opportunities for AI to make a useful difference. Learning smart behaviours for game assets is a first step towards realistic conflict or cooperation. The scope this thesis is the application of Reinforcement Learning to moving assets in the game world. Game sessions a generate stream data on asset's performance which must be processed on the fly. The lead objective is to produce fast, lightweight and flexible learning algorithms for run-time embedding. The motivation from current work is to shorten the time to achieve a workable policy solution by investigating the exploration / exploitation balance, overcome the curse of dimensionality of complex systems, and avoid the use of extra endogenous parameters which require multiple data passes and use a simple state aggregation rather than functional approximation. How action selection (AS) contributes to efficient learning is a key issue in RL since is determines the balance between exploiting and confirming the current policy or exploring an early less likely policy which may prove better in the long run. The methodology deploys the simulation of several AS using 10-armed bandit problem averaged over 10000 epochs. The results show a considerable variation in performance in terms of latency and asymptotic direction. The Upper Confidence Bound comes out leader over most of the episode range, especially at about 100. Using insight from action selection order statistics are applied to determine a criterion for the convergence of policy evaluation. The probability that the action of maximum sample mean is indeed the action of maximum population mean (PMSMMPM) is calculated using the 3 armed bandit problem. PMSMMPM reaches 0.988 by play 26 which provides evidence for it as a convergence criterion. An iteration stopping rule is defined using PMSMMPM and it shows plausible properties as the population parameters are varied. A mathematical analysis of the approximation (P21) of just taking the top two actions yields a minimum sampling size for any level of P21. Using the gradient of P21 a selection rule is derived and when combined with UCB a new complete exploratory policy is demonstrated for 3-arm bandit that requires just over half the sample size when compared with pure UCB. The results provide evidence that the augmented UCB selection rule will contribute to faster learning. TD sarsa(0) learning algorithm has been applied to learn a steering policy for the untried caravan reversing problem and for the kerb avoiding steering problem of racing car both using negative rewards on failure and a simple aggregation. The output policy for the caravan is validated as non jack-knifing for a high proportion of start states. The racing car policy has a similar validation outcome for two exploratory polies which are compared and contrasted.

Acknowledgements

I would like thank my supervisors Dr Vassil Vassiliv and Professor Karim Ouazzane for their helpful and wise guidance on the development of this thesis. They have been consistently supportive and focused on steering me to a coherent and integrated piece of work. I would like to extend my gratitude to Martin Wright of Gamelab (London) for the many informed and interesting discussions on the context of AI in games. I would like thank the RSPG and readers for their time and effort on my behalf. I include Doreen Henry whose patient professional advice has kept the thesis going during periods of difficulty.

Table of contents

List of Figures

Keyword definitions

action probabilities	*Action probabilities* give a direct probability for the selection rule.
action selection	An algorithm which outputs the next action to take.
action transition probabilities	Given any state and action, *s* and *a*, the probability $P^{a}_{ss'}$ of each next state, *s′*, is: $P^{a}_{ss'} \stackrel{\text{def}}{=} \mathbb{P}[s_{t+1} == s' \mid s_t == s , a_t == a]$.
transition probabilities	Given any state, *s* and policy π the probability $P^{\pi}_{ss'}$ of each next state, *s′*, is: $P^{\pi}_{ss'} \stackrel{\text{def}}{=} \mathbb{P}[s_{t+1} == s' \mid s_t == s]$.
action value	The expected reward conditional on choosing action *a* and state *s*.
action variable	Action variables characterise the problem directly from the environment.
actions	Actions are behaviours which affect the environmental *state.*
agent	An entity which can sense the environment and take *actions*.
Bellman Optimality Equation	The optimal action value at *(s, a)* under π*, Q*(s, *a*) is related to the expectation conditional on $s_t=s$, $a_t=a$ of the reward r_{t+1} and the discounted maximum over the actions *a′* of Q*(s', *a*') at the successor state *s′*.
bootstrapping	A calculation where estimates are updated on the basis of the values of successor states.
complete exploratory policy	A policy whose matrix is positive for every element.
deterministic policy	A policy which yields the next action to take from a given state.
eligibility traces	A learning algorithm which uses an extra parameter λ used to factor in a proportion of higher step returns.
epoch	The sequence of outcomes generated by a set of independent random variables as a basis for the reward.
exploitation	A process of choosing actions so as to improve its estimate of the reward.
exploration	A process of choosing non-optimal actions in order to minimise the regret of missing a better set of actions.
features	Properties of data cases which entail sufficient information for learning.
goal	A state or states chosen to achieve the appropriate learning behaviour.
greedy policy	A policy with no exploration and homes in on the first feasible solution.
irreducible	A Markov sequence is where it is possible to get to any state from any other state (Takahara and Hall, 2017).
learning rate	The proportion of the error term added to get the next increment.
Markov Decision Process	An MDP is defined by its state set S, action set A, action transition probabilities P, the conditional rewards R, and the action policy π.
Markov property	A property of the environment's response where the value at *t+1* depends only on the state and action at *t*.

Markov sequence	The sequence of states and actions, $(s_t, a_t \mid t \geq 0)$ where $a_t \sim \mathbb{D}[\pi(s_t, j), j= 1..A(s_t)]$ that obey the *Markov property.*
Monte Carlo	A method that repeatedly samples random inputs, simulates the system and collates the output distribution.
move and turn	A discrete approxmation to motion where the vehicle travels to a new position in Δt at velocity V in direction α_t. Then turns by $\Delta\alpha_t$.
Non-Player Characters	Moving assets of all kinds.
on-policy	The distribution of states encountered by the agent is the same as that used to update the state/action values.
optimal action value	The action value of the optimal policy.
plays to convergence	The first t, t_x where the PMSMMPM is greater then a critical value
PMSMMPM	The probability of the action of maximum sample mean being the maximum population mean.
Policy evaluation	An expression which improves the action value given a particular policy.
Policy improvement	An expression that uses the current action values to obtain a new greedy policy.
policy matrix	The probability for choosing an action given a particular state.
policy search	A process of seeking a policy solution directly by maximising the objective function of RL.
positive recurrent	A state s is called *positive recurrent* if the expected amount of time to return to state s, given that the sequence started in state s has finite expectation (Sigman 2006, Proposition 2.2).
Q-learning	A learning algorithm that uses an updated target based on the current approximation to the optimal Q.
reference reward	A variable that tracks the long run mean value.
Reinforcement Learning	RL, a tried and tested method of machine learning.
return	The discounted sum of the total rewards.
reward	A numerical measure of the achievement of the agent's *goal.*
round robin	The selection of each action in turn.
state	A sufficient statistic of the environment which entails all the information needed for learning.
state index	The state index is an index combining all of the state variable indices.
state path	The sequence of states embedded in a Markov sequence.
state space	The *state space* is the tuple of all the state variables.
state value	State value is the expected reward conditional on choosing a particular state
state variable	State variables characterise the problem directly from the environment

state variable sub-range	A small contiguous part of the range of a *state variable.*
state region	A multidimensional hypercube whose edges are sub-ranges of all the state variables and which correspond to a state index.
stationary in distribution	The joint distribution of signals at a set of distinct time points is the same as the joint distribution as those at time point displaced by any positive integer (Takahara and Hall, 2017).
steady state solution	a situation where the state variable do not change with time.
supervised learning	A learning process which is directed towards target examples.
Temporal Difference	A learning process which is driven by changes in temporally successive predictions.
trial-and-error	A search process which embodies selecting actions based on maximising the best information so far and using the error from the expected successor state to improve subsequent selections.
ϵ-greedy	A selection rule that choses the action of maximum reward with probability $1-\epsilon$ and choses a random action otherwise.

Chapter 1 ***Problem formulation and motivation***

1.1 **AI in the games context**

The task of human learning is highly diverse, multi-layered, and dynamic. It is diverse since it ranges from motor skills, e.g. walking and running, literacy and communication skills, numeracy, erudition, problem solving and planning to high levels of abstraction. It is multi-layered since higher learning like erudition depends on good reading skills. It is dynamic since circumstances in the real world change, new knowledge needs to be acquired, existing knowledge reviewed. Lastly it is embedded in a society that has a history of past learnt experience.

This Ph.D. is motivated by the sustained need to make *Non-Player Characters*(NPC) more intelligent, context aware and engaged with the game play, rather than static or at best reactive with limited behaviour. NPCs include opponents, buddies, weird animals and in this thesis moving vehicles. It is rare to see some common sense behaviours satisfactorily implemented. Indeed the next generation will need to show empathy and work hard to optimise the player game experience (Lucas, 2009). This marks a departure from the recent drive of just getting game characters cleverer. "There is a gulf to be crossed in making learning work well within commercial video games where it needs to be rapid, robust, and highly flexible – essential more human like" (Lucas, 2009). Tom Scutt stresses the inherent dramatic context and character based behaviour that makes NPCs plausible and even interesting (Scutt, 2009). Among the many desirable attributes for a NPC, learning from the game context was chosen as an important step towards this ambitious aim. Machine learning with its focus on experience appears as the place to start. A widely quoted definition of machine learning has been proposed: "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and a performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*" (Mitchell, 1997). In addition, several threads of investigation in machine learning offered potentially useful tools, the most tried and tested was Reinforcement Learning(RL). This claim is justified in terms of its origins, relationship to neighbouring approaches and breadth of RL applications.

1.2 **Reinforcement Learning as a candidate machine learning tool**

Reinforcement Learning was originally termed hedonistic learning and understood as "a system that wants something from an environment" (Sutton and Barto, 1998). In the natural world a lot of basic skill learning is done without a teacher but by a process of discovery and post experience analysis of consequences. RL introduces a simple goal

orientated *agent* which can sense the environment and take *actions* or behaviours, which affect the environmental *state*. The state is a sufficient statistic of the environment and entails all the information needed for learning.

The twin features of *trial-and-error* search embody selecting actions based on maximising the best information so far and using the error from the expected successor state to improve subsequent selections. The environment yields a numerical *reward* at each step to indicate the extent of the achievement of the agents *goal.* The *goal* has been chosen to achieve the appropriate learnt behaviour. The *return* is a discounted sum of the total rewards the agent receives after a sequence of actions. The RL task is to find that policy which maximise the return and thus learns the appropriate behaviour.

In contrast to *supervised learning*, where learning is directed towards target examples, RL is able to learn behaviour and choose actions in the light of the achievement of its *goal* (Barto and Dietterich, 2004). RL can chose actions based on what has been learnt so far so as to improve its estimate of the reward, called *exploitation*, or systematically try non-optimal actions in order to minimise the regret of missing a better set of actions, called *exploration*. Exploitation and exploration are intertwined. In order for exploitation to be productive the best actions must first have been discovered. In order for further exploration to occur the agent must be able to make a number of successful actions to reach an unexplored region.

RL is a useful paradigm for NPC learning and has inspired the work of Microsoft Research Lab, Cambridge (Graepel, 2005). It can offer some basic learning much earlier by a process of bootstrapping where estimates are updated on the basis of the values of successor states, and dynamic averaging. RL offers relatively simple update algorithms to the decision parameters based directly on the data stream as it arrives and so is feasible at run time. RL contrasts with sophisticated maximum likelihood or Neural Network approaches because they require extensive off-line computation before any outputs are obtained (Funge, 1999).

1.3 **Brief History of RL from a critical appreciation of the literature**

This section collates some of the key developments in RL approximately grouped by date order. RL is the outcome of several threads of research coming together. Bellman's (1957) work on optimal control in late 1950's using a notion of a system's state and a reward function, proposed the classic *Bellman Equation* (Bellman, 1957), and its stochastic version based on a Markov Decision Process (MDP) (Sutton and Barto, 1998). An MDP has a useful property that the environment's response at the next step depends only on the current state and action and not on any past state and action. Bellman's work did not

address the fact that most interesting MDPs do not have their internal structure accessible and the solution requires an iterative stochastic optimisation using sampling from trajectories in the environment to generate estimates of the expected reward. A trajectory is the trace of movement in the environment as an outcome of a simulation. Implicitly Sutton and Barto (1998) regard the incremental approach to the solution of a black box MDP as a metaphor for a learning by a process of repeated approximations.

Thorndike's (1911) law of effect linked trial-and-error learning with behavioural reinforcement depending on a positive or negative outcome and laid a foundation on "exploration as a route to discovery" exploited to this day. The trial-and-error is selectional in that alternatives are tried and a selection is made based on a comparison of their consequences. Minsky (1961) made the link to the *credit assignment problem* which seeks to assign one or more of a sequence of action decisions to the final credit/blame for improved / reduced reward. Such a result would provide the basis for selecting the best action decisions. The landmark pole balancing system studied by (Michie, 1974) and reworked by Barto et al. (1983) provides a useful example of learning a task under incomplete knowledge and shows an early departure from "simulated neural networks".

Widrow, Gupta, and Maitra (1973) modified the Least Mean Squares algorithm of Widrow and Hoff (1960) to produce a RL rule, described as "learning with a critic" instead of "learning with a teacher." "Although the hypotheses on which the model is based do not perfectly fit Blackjack learning, it is applied heuristically to predict adaptation rates with good experimental success". Whilst many learning researchers focused on supervised learning, Klopf (1982) with his novel but experimentally untested view that neurons "seek" excitation and "avoid" inhibition, was most responsible for reviving the trial-and-error thread to RL.

Sutton and Barto (1998), rehabilitated the notion *Temporal Difference* (TD) learning where learning is driven by changes in temporally successive predictions, known as the *TD error*. The TD error is used mostly in psychology but the language was adopted in early neuroscience models, (Gelperin, Hopfield and Tank, 1985). The components to one-step TD learning include for each step of the agent:

- an action selection usually based on the current action value table, Q;
- a simulation or experiment to generate the next step, next state and reward;
- calculation of the TD error which is the reward at the current step plus the discounted current Q value at the next step less the one step Q value;
- a new Q value for the current state based on the TD error factored by the learning parameter.

The output of TD learning is either a *deterministic policy* vector which yields the next action to take from a given state or a *policy matrix* which gives a probability for choosing an action given a particular state. By 1983 Barto, Sutton, and Anderson (1983) embedded TD in trial-and-error learning, called the *actor-critic* architecture, and applied it to the landmark pole balancing problem. Actor-Critic has been revised by both Sutton and Barto (1998) and Silver (2014). Actor-critic methods are TD methods that have a separate term to explicitly represent the policy (actor) independent of the value function (critic). The actor is modelled by a contemporaneously but separately leant policy vector which yields the next action to take from a given state. The actor uses the TD error as an input. The critic is the estimated long term reward using the consequences of actions taken and provides the TD error. Although an additional learning parameter is introduced for the policy vector, it does offer the modelling flexibility of a different learning rate, and a graceful feature of providing an improving policy from the outset.

The sampling of states for estimation can be either *on-policy* or *off-policy* in relation to the reward sequence generated from the using policy π. On-policy means that the distribution of states encountered by the agent is the same as that used to update the state/action values. On-policy state sequences are easier to generate and have better convergence results. In contrast in the *off-policy* case the policy used to generate behaviour, called the *behaviour* policy, may continue to sample actions unrelated to the policy that is evaluated and improved (i.e. being learned), called the *estimation* policy. Off-policy estimation has not been attempted in this study.

In 1989, Chris Watkins (1989) introduced *Q-learning* which integrated all the threads of RL. Watkins Q-learning uses an updated target based on the current approximation to the optimal Q, which is independent of the current behaviour policy being followed for step generation. Watkins Q-Learning is off-policy in the sense that the action selection policy is unrelated to the learned action value function that is evaluated and improved. Chris Watkins considers conditions for convergence in detail but does not look at the operationally difficulties for larger state action spaces.

Monte Carlo estimation has been used since the 1940's usually for systems that are intractable and is a mature method that repeatedly samples random inputs, simulates the system and collates the output distribution. In this case it simply derives the average return for each visited state for a set of episodes. An *episode* is a finite sequence of consecutive states ending with a terminating state, and which correspond to some natural finish in the real world. Since each episode must run to completion before the return is obtained there is a higher computational demand but this pays for several advantages: each state estimate is independent since no bootstrapping occurs, rewards are processed immediately so they can

be used directly for learning. Results compare favourably with the best TD, Tesauro and Galperin (1997).

Most interesting processes are continuous in time and because the continuous time version of RL presents difficult equations the approximation of discrete time intervals is used. CEWDoya (2000) compares the discrete time and continous time formulations of the classic cart pole problem. Normally the time interval has to be small in relation to dynamics of the system for accurate modelling. However the reward may occur at an arbitrary number of steps, n, later. If such an n is known, TD could be designed using the n-step (Sutton and Barto, 1998) return rather than the one-step return above. Eligibility traces introduces an extra parameter λ used to factor in a proportion of higher step returns. It provides a parameterised bridge between a Monte Carlo approach ($\lambda = 1$) and TD($\lambda = 0$). If the task is known to be non-Markovian or there are many steps per episode then eligibly traces produces significantly better results. However it does require a sequence of experiments to determine the optimal λ for any particular problem and there is a extra computation premium in updating all action values at each data step.

Since 2002 progress on RL has been a story of refinement and scaling up to larger more challenging complex domains.

Scaling up the number of Q values for a dimension, as particularly required where a continuous domain is modelled using a discrete number of ranges, rapidly leads to very large Q value matrices, each element requiring statistical estimation. The estimation task grows exponentially as the number of dimensions grows, an example of the *curse of dimensionality* foreseen by Bellman(1957) and a common problem across machine learning. For example a discrete approach to backgammon needs 10^{20} states (Tesauro, 1994).

Function Approximation (FA) gives an approximation to action or state values by using a considerably smaller set of orthogonal *domain features* each associated with a state and presumed to entail sufficient information for learning. A corresponding set of parameters is optimised to produce an approximation to each action value that satisfies the minimum squared TD error. Features share the properties of independent variables similar to those used in statistical regression. In principle any parameterised function can be used. In particular Neural Networks can represent any continuous function as deployed in TD-Gammon (Tesauro, 1994) subsequent to which considerable effort in FA has been undertaken with mixed results (Kober, 2013). FAs which are linear in features and parameters have the most useful mathematical properties and the most stable results, Silver (2014). The drawbacks include operationally the need to store and retrieve a feature vector,

and theoretically need for a non-learnt principle to guide the selection of features. However, as noted by Sutton and Barto (1998) of paramount interest is the optimal policy and not the best fitted action value.

The importance of addressing the problem at the episode level obscured the potential of using the step wise information to construct a model of environment, information which is thrown away after a Q update. Given a state and action the model aims to yield the next state and the reward either as a sample or as a distribution. Such models were used in the Bellman calculations. Normally the model is a unknown and it has to be learned using regression or a nonlinear function of inputs to outputs. Sutton and Barton (1998) demonstrated the *Dyna-Q* architecture which uses Q-learning, updates the model, adds N repetitions of the current model deployed from a random state and using the generated successor states applies N Q updates. They show an order of magnitude improvement with a toy maze problem of 63 states with 4 actions. Dyna-Q offers the advantage of leveraging past information to augment the step wise simulation when that is expensive. Sutton and Barton (1998) anticipated many current developments: partially observable MDP, state aggregation, trajectory sampling, modularity and hierarchy in problem decomposition some touched on below.

Forbes, building on his work with the Bayesian Automated Taxi project (Forbes et al., 1995), deploys RL to successively improve vehicle driving control policies through experience and feedback from the environment. The driving challenge is particularly difficult because the control variables are analogue, the multivehicle context presents a vast number of configurations and all kinds of road topology must be followed safely. He uses a two level decision theoretic architecture combining trip planning to identify routes with low level lane selection and speed control. At that time real time sensors were insufficiently reliable and so development was limited to a simulation environment. Addressed are several deficits of classic RL approaches: the curse of dimensionality which renders the tabular Q values too large for real time learning with continuous variables; the sampled state action space for on-policy is locally non-stationary with long periods of highway driving interfering with short periods of urban driving. Estimates suffer from that similar to over fitting in neural network models.

Forbes proposes an action value FA to Q values (Forbes, 2000). Rather than using features he stores each visited state and action with its current Q value estimate in a database. His *Instance Based Learning* (IBL) achieves lookup by using kd trees, which are binary trees that store k-dimensional data Bentley (1975). Instance averaging and instance deletion are used to keep the bound on database to a prudent level. The standard TD has an extra last stage: Take the current action to get the next state, calculate the next action based on ϵ-

greedy selection, calculate the TD error, update the state action Q values of the neighbours using the gradient ascent derivative of the weighting function and store the new Q value for the current state and action. Database query and update mechanisms are used to retrieve and update Q values.

Clearly a scaling up issue is finding the best sampling strategy given the vastness of the state action space. Forbes uses a sample approach that may be considered off-policy in the sense that action values used for Q estimation are in a different proportion that encountered in the input data stream. Forbes suggests that IBL will contribute to algorithms that complete in limited time, a requirement reinforcing an important objective of this thesis.

Jones and Crowe (2014) deploys a similar off-policy approach by re-using selected past data on failed trajectories to validate and if necessary alter the exploration and learning rate. The motivational principle is to focus learning where it is most effective and not necessarily the most recent. In general off-policy learning has more difficulty showing convergence if it is uncertain that all states will be visited.

Policy search was motivated by seeking a policy solution directly by maximising the objective function of RL rather than indirectly using large action values. The policy vector FA comprises of a differentiable function of features for each state action pair and a corresponding set of parameters to be optimised. The objective function is couched as the average reward per time step and can be constructed in terms of the policy vector, the stationary distribution of the Markov sequence and the average reward for each state action pair. The incremental update of the policy parameters using the gradient of the objective function produces the most stable improvement and also fits in with the iterative solution of RL.

Gradient methods to implement *policy search* include finite differences and likelihood ratio methods. A important requirement is the existence of a *score function* defined as gradient of the log of the policy vector. Silver (2014) generalises the expression for the 1-step gradient to the policy gradient theorem of the multistep MDP as the expectation of the score function and the action value for a particular state action pair. The classic TD learning algorithm above now has an added step for the policy vector update using an actor learning rate, the score function and the action values.

Policy gradient estimate is statistically inefficient and Silver (2014) introduces a critic in the form of a linear action value FA which uses a second set of adjustable parameters for the true action value. The classic TD learning algorithm above now has a further added step for the policy vector update using a critic, the TD error and the gradient of the linear action value FA . He present further refinements to these updates to mitigate the bias in the

policy parameters estimates, and introduces a compatibility constraint on the action value parameters and score function parameters. Silver (2014) has shown success by several orders of magnitude with a deterministic policy for several bench mark problems including high dimensional bandit and octopus arm of 20 actions and 50 state dimensions.

The use of endogenous parameters whether for less bias from the choice of time step as in the case of eligibility traces or the use of a constant α on the assumption that the reward processes might only be stationary in the very long term, require several passes over the data stream incurring addition data management resources.

The curse of dimensionality of complex systems generates a large number of actions values as the number of states expands as the product of the number of regions for each dimension. Each action value requires sufficient data to get statistical efficiency which motivates choosing the minimum state and action variables as necessary. The more powerful methods of policy search introduce further steps which are likely to be computationally more demanding as well as requiring decisions on underlying feature spaces for functional approximation. The games world requires multi-context applicability, few tuneable parameters and statistical efficiency for fast learning. This thesis will investigate how simple learning algorithms can be augmented to achieve rapid learning with relative little extra demand.

1.4 **Landmark applications**

Embryonic RL was introduced by Samuel checkers program (Samuel, 1959) who was probably the first to publish the term *machine learning*. A look ahead search was used to obtain a value for the current board position. The best move was based on a minimax procedure which assumed the opponent would always choose the minimum score. Rather than a modern evaluation function he used first a memoising of past board positions and then added a piece advantage index. Samuel was mainly concerned with how to generalise the evaluation function by adding board properties used at each self-play update. Samuel's checkers player did "better-than-average" play and was widely recognized as a significant achievement in artificial intelligence and machine learning.

RL has been applied to a wide variety of physical control tasks (Bernstein, 1927). One such task is the acrobat, a two-link, under actuated robot roughly analogous to a gymnast swinging on a high bar. The first joint (corresponding to the gymnast's hands on the bar) cannot exert torque, but the second joint (corresponding to the gymnast bending at the waist) can. The RL task is to determine the choices to the acrobat as to when to apply positive /negative or zero torque to his waist using his legs.

RL's major success with TD-gammon established it as a dominant approach to automated learning in an uncertain environment (Tesauro, 1994). Backgammon is a turn taking board game for two players. Each player aims to be first to move their checkers around the board to their home position and then to remove them to the side. Moves are directed by dice throws. The skill is the choice of which checkers to move at any time. TD learning receives an input from a representation of the Backgammon board and produces an output move. Because the number of board positions is enormous a Neural Network is used to process an input pattern into a state value FA. State value is the expected reward conditional on choosing a particular state. TD-gammon also introduces a reward delay using *eligibility traces*. The Neural Network is updated every learning step and a heuristic parameter λ is used to implement the delay. TD gammon was trained using a self-play protocol where an improved policy was developed from an "opponent" using the previous one. TD-gammon shocked the Back gammon elite players with par performance at the 1992 World Cup Backgammon tournament and provided an alternative to an opening gambit that had stood for 30years. Tesauro admits ignorance and surprise that TD Gammon learns so well, he conjectures that the dice rolls force sufficient exploration which no evaluation function would carry out.

The complex and open-ended context of modern video games requires real time learning by both players and game characters to achieve interesting game play. Graepel reports on the use of RL techniques to martial arts fight games. Often the moves are repetitive and implausible. By using a state-action formulation Q-learning (Peng & Williams 1996) with a reward to minimize hurt in both players he generated evasive and minimally aggressive play (Graepel, 2005).

Microsoft has put over a million pounds into applying RL methods to the solution of Go (Graepel, 2005). The goal of Go is to gain territory and take prisoners on a 19x19 board of black and white stones. Unlike chess there are over 200 moves per turn, strong global interaction which prevents local search, and difficulty in defining a good evaluation function. This work was superseded by the success of Monte Carlo Tree search combined with the use of the *Upper Confidence Bound* (UCB) to give par performance against Go masters (Enzenberger & Muller 2009). The breakthrough came in in April 2017 when DEEPMIND's alphaGo earned a Dan 9 victory over Mr Lee Sedol the strongest Go player in the world. Its innovative play overturned 100 years of received wisdom. The Go engine employs advanced tree search to model moves and a neural network to model the board input all embedded in a 'learning from mistakes' process here known as reinforcement learning (Hassibis, 2017). Demis Hassiabis co-founder of Deepmind commented on Desert Island Disks, in May 2017 (Hassibis, 2017b). "The most exciting breakthrough (in AI)

from our perspective is combining two different areas of AI together, there is one area called deep learning that uses a neural network to mimic what the brain does… And then the second part called reinforcement learning is about decision making. Given a model of the world and how it works then how do you use that model to make decisions?" (Hassibis, 2017b). .

1.5 **Scope of thesis and objectives**

The scope this thesis is to demonstrate the application of RL to moving assets in the game world to achieve realistic NPC learning for control. The particular constraints of engineering for games include algorithms that can provide a timely solution, have efficient learning gain from a single data steam and do not make large computing demands. The work first addresses the problem of exploration using the single state *n-armed bandit* problem. Although much studied it provides a simple controllable work bench on which to investigate selection strategies because it isolates the problem of exploration verses exploitation from the added complexity of choosing the state to state sequence in the course of maximising the reward. In parallel to this the focus is on finding a suitable case study and problems are looked at which are relevant to a range of games and which do not involve a disproportional large engineering effort in relation to the learning task. Next the dynamics of moving assets in the games world are studied in particular how to learn the steering task. Among the pattern of examples studied by Sutton and Barto (1998) and others a significant proportion are 2D spatial tasks and vehicle control e.g. race track driving, ship manoeuvring, orienteering, robot navigation, grid search and traversal, all presenting an inviting context for demonstrating learning algorithms. This is of relevance since there are hundreds of steering games on the market. Of interest to games designers is the interaction of ensembles of 'vehicles'. Graepel (2005) notes that game-based racing car teams follow like pearls on a string rather than jostle for overtaking / blocking moves. Craig Reynolds (1987) has a whole list of multi-vehicle behaviours which would need learned responses to specific contexts like arriving, queuing, avoiding and crowding following. Bevilacqua (2012) provides an online tutorial on modelling steering behaviours.

Subsequently the reversing caravan problem is explored since no studies from a learning viewpoint could be found on line. This work seeks a learnt steering policy which achieves a stable turn using the current steering angle and car-trailer angle as *state variables* and the *action variable* comprised of small changes in the steering angle. State and action variables characterise the problem directly from the environment. Jayakaran (2004) has offered a dynamic systems approach to the equation of motion and uses it to build a physical model with an additional trailer steering contol unit.

Five objectives are proposed as a guide to investigation:

Objective 1.0: Show how action selection contributes to efficient learning

From its earliest formulation to the present day the notion of exploration and exploitation remains an important issue discussed under various slogans: trial and error (Klopf 1982), learning from mistakes and test and learn. This objective drives the investigation into action selection, a key element of RL, because it implements how the balance of effort between exploitation and exploration can be made effective for learning.

Objective 2.1: To exploit and develop the implications of RL learning algorithm for the reversing caravan and its generalisation to related movement tasks.

The reversing caravan is an original problem from the RL viewpoint and provides a realistic example to study the effectiveness and limitations in the game context. It serves typical example of a nonholonomic vehicle manoeuvring under limited control.

Objective 2.2: Applying other known solutions to the reversing caravan problem.

This will provide a basis for cross comparison with classic solutions and establish the validity of the discrete simulation.

Objective 3.0. Explore and demonstrate the wider application of learning in games.

This objective builds on 2.1 and addresses another case study which looks at a more demanding dynamic task. In particular the focus is on steering to a target, e.g. Snake Domain (Silver, 2004, Lecture 7).

Objective 4.1: To produce fast, lightweight and flexible learning algorithms suitable for run-time embedding in current games.

The game engine requires fast algorithms that are both statistically efficient and have modest demands in terms of CPU and memory resources especially for mobile platforms.The games context requires a useable if approximate policy early in the session. A learning algorithm is considered fast if a usable policy can be obtained to a given probability even though the action value estimates are still converging. This in turn motivates investigation into the statistical properties of the policy expression. A fast algorithm will be eager in the sense that estimation will work incrementally from the first data item unlike Monte Carlo (Rubinstein and Kroese, 2017) which needs the complete input data.

A lightweight algorithm will not have any parameters that need pre optimising for either exploration e.g. UCB has none, Kocsis and Szepesvári (2006) , or learning parameters e.g. sample average has none (Sutton, 1998).

A flexible algorithm will be generalizable to many kinds of moving assets which preclude the use of FA since it requires asset specific domain features.

Objective 4.2: To **e**xplore at least one of the key theoretical problems arising from current theoretical advances.

4.2.1. Determine a convergence criterion that achieves a specified level of probability.

The early investigation covers the efficiency and interrelation of exploitation, exploration and action selection, since these determine the accuracy of output results. The experiments using n-armed bandit simulation for several selection rules show considerable variation of performance as the sample size for learning is increased. The latency, rates of increase and asymptotic limit show considerable differences and are still trending at samples of over 10^4. Some published work need vast data sets which are impractical for deployment in games engines. A neglected issue with several case studies is how to decide that learning has converged to a measurable criteria. Thus optimal selection is crucial to achieve useable estimates within a given samples size. The bandit problem has been revisited and a variance based measure for the convergence is proposed which also has the bonus of providing a more efficient exploration policy.

4.2.2. Investigate the properties of state aggregation.

State aggregation is trivial form of FA (Forbes, 2000) where each feature is a region of the aggregation. The feature value is 1 if the state variables lies in the region and zero otherwise and the feature parameter is just the Q value for that region.

4.2.3. Construct a single measure of convergence based on the measure for each state.

The n-armed bandit criterion for convergence of 4.2.1 is only applicable to one state. Multistate convergence requires a single measure which takes into account the relative importance of some states over others.

Objective 5.1: To investigate and develop metrics for the validation and evaluation of the performance of RL algorithms using a standard set of simulated problems.
The aim of validation is to check that the learning performance measures what was intended. Also a much more efficient validation test is proposed which avoids large random data set. In this context the output policy is simulated against a standard set of input data and the corresponding behaviour assessed for correctness.

Objective 5.2: Demonstrate and visualise the learning progress.
Both the learning and validation simulations are to be visualised both as static 2D graphs and as animations. The former will provide insight into the impact of exploration. The

latter will yield insight into the relationship between the RL reward and the shape of the learnt behaviour.

Appendix 3 shows a graph of the relationship of objectives to chapter and section headings. The first objective is to understand how actions are selected in a one state system. It is useful to focus on just the selection of a best action from a single decision and use a system which avoids the issue of locating the reward among sequence of decisions. The next chapter introduces the n armed bandit problem which provides a good work bench for the study of action selection.

Chapter 2 ***Bandit based exploration / exploitation***

The task of this chapter is to follow though on objective 1.0 and to investigate the effectiveness of several implementations of action selection. Action selection is a key issue in RL since it contributes to efficient learning by determining the best balance between exploitation for improved values and exploration for ignored values. The much studied *n-armed bandit* problem (Berry etal, 1985) is analogous to a slot machine but with *n* arms instead of one. Each play of an arm spins the dials and yields one of the jackpots. From the *n* arms to choose from the player faces the problem of learning from trials which arm gives the best jackpot. Here a play is generalized to an action and the jackpots correspond to the rewards. The *action value* of an arm is the expected value of the reward for choosing that arm. The solution followed below is to estimate action values using iterative sample estimates through a sequence of plays. CEWAn alternative evolutionary approach is followed by Koulouriotis and Xanthopoulos (2008) which is particularly appropriate for the case of non stationary rewards.

2.1 **Action value estimation methods and action selection rules**

At the start of a play an action a_t is selected from a set of A actions [1, 2,.. A]. At the completion of the play a_t a reward r_t is obtained. The jth selection of a particular *a* occurs at time i(*a*, j). The number of times *a* has been chosen in time interval [1..t] is k(t, *a*). The population action value and sample average of the reward for choosing action *a* are respectively (Sutton, 1998):

(2.1.1) $Q^*(a) \stackrel{\text{def}}{=} \mathbb{E}(r_t \mid a_t == a)$,

$\mathbb{E}(R \mid C)$ denotes the expectation of R over its distribution condition C.

(2.1.2) $Q_t(a) \stackrel{\text{def}}{=} (1/k(t, a) \Sigma_{j=1}^{k(t, a)} r_{i(a, j)}$

The core framework for learning the Bandits problem is to carry out a sequence of T independent plays. At each play t the *action selection rule* is deployed, a reward generated which has a random component, the action value expression updated and the number of successful plays are accumulated. The random component is based on based on T independent random variables $r_T = (r_t \mid t=1..T) \in \Re_T$ where $\Re_T$ is a T dimensional space of real numbers. An *epoch* is the sequence of outcomes generated by a set of independent random variables as a basis for the reward. The true value of any performance statistic p is $\mathbb{E}r_T[p]$. Using many epochs will control for any particular effect of the reward distribution for r_t. In practice a sample of X epochs, each with an independent r_T are generated and averages taken of p over all samples.

The performance statistic is the proportion of plays that choose the optimal action for a given epoch. The optimal action is that action which yields the maximum reward. At the end of each experiment of X epochs the average of the performance statistic is calculated as a percentage. The final output is the %optimal action, as function of episodes of T plays for T=2 to 10000, and of the learning parameter ζ.

The incremental update is known as the sample average method for estimating action values and becomes:

(2.1.3) $Q_t(a) = (1/k(t, a))\Sigma_{j=1}^{k(t, a)} r_{i(a, j)} = (1/k(t, a))(r_{i(a, k(t, a))} + \Sigma_{j=1}^{k(t-1, a)} r_{i(a, j)})$, for $a == a_t$

$= (1/k(t, a))(r_{i(a,k(t, a))} + (k(t-1, a))Q_{t-1}(a))$,

$= (1/k(t, a))(r_{i(a,k(t, a))} + (k(t, a) - 1)Q_{t-1}(a))$,

Now since $a == a_t$, then $i(a, k(t, a)) = t$ since the last time a is chosen it is t.

$= Q_{t-1}(a) + (1/k(t, a))(r_t - Q_{t-1}(a))$

The *action selection rule* is an algorithm which outputs the next action to take. The action value expression shows how the sample mean is calculated. The pseudo code below shows the algorithm used for all the learning methods of this chapter but with the appropriate action selection rule and action value expression embedded in the learning parameters, epoch and play loops. This section and sections 2.3 – 2.7 shows and compares the % optimal action for five alternative selection rules.

#loop over a learning parameter ζ, ranging from $\zeta_1, \zeta_2, .. \zeta_Z$

for $\zeta = \zeta_1, \zeta_2, .. \zeta_Z$

    set learning parameter ζ

#loop over epochs

    for x = 1:X

        CountOfCorrect actions(x, t) =0

#Set Q*(a) to a standard normal deviate

        Q*(a) ~ $\mathbb{N}(0, 1)$

#Find true best arm

        a* = arg max-with-random-tie-break(Q*)

        Q=0

#loop over plays

for t = 1:T

Deploy action selection rule for a_t

#Select the reward as a normal deviate of mean Q*(a)

$r_t \sim \mathbb{N}(Q^*(a), 1)$

Update action value expression for Q

CountOfCorrect actions(x, t) =+ (a_t == a*)

end t

end x

%optimal action(ζ ,t) = Σ_x CountOfCorrect actions(t, x) /X

end ζ

The action selection rule below is *ϵ-greedy* which choses the action of maximum reward with probability $1-\epsilon$ and choses a random action otherwise. The ϵ-greedy is designed to achieve both exploitation and exploration. The function maxWithRandomTieBreaks selects uniformly randomly the $Q_t(a)$ any a for which $Q_t(a)$ which is maximal. This extrapolates to the extreme case when the rewards of all the actions are identical. So the degeneracy in two or more tied actions is solved by using a random selection to provide feasible a_t and can be regarded as another component to exploration. For example if Q(1)=1 and Q(2)=1 and all other Q values were 0 then the function select action 1 and 2 with probability of 0.5. The tables below (2.1.1), (2.1.4),.. present for each learning algorithm the action selection rule and its corresponding action value expression.

(2.1.4) action selection rule	(2.1.5) action value expression
`if`($\rho <= \epsilon$) $\rho \sim \mathbb{U}[0, 1]$, ρ a uniform deviate	$Q_0(a_0)$ initialised, $k(0, a_t) = 0$
$a_t \sim \mathbb{U}[0, A]$ #$\mathbb{U}$ the uniform distribution	$r_t \sim \mathbb{N}(Q^*(a_t), 1)$
`else`	$k(t, a_t) = k(t-1, a_t) + 1$
a_t = arg maxWithRandomTieBreaks$_a$ $Q_t(a)$	$Q_t(a) = Q_{t-1}(a) + (1/k(t,a))(r_t - Q_{t-1}(a))$
`endif`	$Q_t(a') = Q_{t-1}(a'), \forall\ a' \# a_t$
	$k(t, a') = k(t-1, a'), \forall\ a' \# a_t$

Table 2.1.1 Action selection and value expression for ϵ-greedy selection

Experiments have been performed using GNU Octave (Eaton, 2012) which offers a mathematically based scripting and plotting language with large library of built-in functions using language that is mostly compatible with Matlab. The main strengths

include matrices as a fundamental data type and powerful built-in math functions, plotting function with simple GUI widgets, and extensive function libraries.

The epochs below use randomly generated rewards, each reward for action a is a normal deviate drawn from $\mathbb{N}(Q^*(a), 1)$ where $Q^*(a) \sim \mathbb{N}(0, 1)$ for $a \in [1..10]$. The average % optimal action is plotted against T=2,10000 in Figure 2.7.1, and Table 2.7.1, which shows and it reaches a learnt %optimal action of 87% by T=10000 after averaging over X=1000 epochs and uses the best value for ϵ of 0.1. Appendix 0 gives the detail on the derivation of ϵ based on those values used by Weatherwax (2005, chapter 2). The sections 2.3 – 2.7 show the % optimal action for five alternative selection rules.

2.2 **Optimistic or pessimistic initial values**

If the starting values $Q_0(a_0)$ are set much higher than expected subsequent rewards will most likely be less and cause the sample average to drop, in which case the greedy selection will choose another action and so on. In this way a sort of initial systematic or round robin exploration is achieved. If $Q_0(a_0)$ are set too low then expected action selection will follow that which is initially greater and exploration will be down to the ϵ probability of random selection. In either a large positive or negative initial value needed to ensure all estimates are bounded above or below will introduce a large initial bias in the incremental sample estimate. If $Q_0(a_0)$ is set to zero then no bias in introduced and a round robin selection can be introduced explicitly. By the action of maxWithRandomTieBreaks forces random selection from those actions whose (s, a) pairs have not been updated if a negative or zero reward is used with Q values initialised to zero, (2.2.1). For t > T it switches to (2.1.4):

(2.2.1)

if ($t \leq T$)

a_t = mod(t-1, A) +1

else

a_t = arg maxWithRandomTieBreaks$_a$ $Q_t(a)$,

endif

2.3 **Softmax Action selection**

The rationale here is to emulate natural stochastic systems and use a Boltzmann expression in each $Q(a)$ and temperature τ to give the action probability $\pi_t(a)$ of choosing action a (Gibbs, 1902). As $\tau \gg \max_a Q_t(a)$ the $\pi_t(a)$ all converge to 1/A. As $\tau \rightarrow 0$ $\pi_t(a) \rightarrow 1$ for a =

arg max$_a$ $Q_t(a)$ and zero for all other a. The selection rule is to choose a_t in proportion to the probabilities $[\pi_t(1), \pi_t(2),... \pi_t(A)]$ denoted $a_t \sim \mathbb{D}[\pi_t(1), \pi_t(2),... \pi_t(A)]$, i.e. $\mathbb{P}[a_t == j] = \pi_t(j)$.

(2.3.3) $\pi_t(a) = e^{Q_t(a)/\tau} \Big/ \sum_{b=1}^{A} e^{Q_t(b)/\tau}$, $\tau > 0$ is the temperature,

(2.3.4) $a_t \sim \mathbb{D}[\pi_t(1), \pi_t(2),... \pi_t(A)]$

The best %optimal action emerges at T =10000 for $\tau = 0.1$ (see Appendix 0 for comparison of other values of τ) and Figure 2.7.1 shows low long term increase finally reaching only 63%.

2.4 **Action value with Pursuit Selection methods**

Pursuit selection methods independently attenuate *action probabilities* $\pi_t(a)$ with additional parameter β to sample action values on the basis that they will have different long run behaviour (Sutton, etal (1998). *Action probabilities* give a direct probability for the selection rule, Table 2.4.1. Figure 2.7.1 shows the performance of pursuit.

(2.4.1) Selection rule	(2.4.2) Action probabilities
$a_t \sim \mathbb{D}[\pi_t(1), \pi_t(2),... \pi_t(n)]$	(2.1.4) for $Q_t(a)$ $\pi_t(a) = \pi_{t-1}(a) - \beta\pi_{t-1}(a)\ \forall a \neq a_t$ $\pi_t(a_t) = \pi_t(a_{t-1}) + \beta[1-\pi_t(a_{t-1})]$

Table 2.4.1 Action selection and value x expression for Pursuit selection

The best learnt %optimal action is 92% for $\beta = 0.01$ (See Appendix 0) and shows most rapid rise at t~200, see Figure 2.7.1.

2.5 **Reinforcement Comparison with softmax**

Reinforcement comparison deploys separate learning models for the actor and critic using parameters α and β respectively, Dayan, P. (1991). A *reference reward* $\bar{r}_t$ tracks the long run mean value and is updated using a linear update with parameter α on the deviation, $r_t - \bar{r}_t$. Base action probabilities $p_t(a_t)$ are generated with parameter β on the deviation, $r_t - \bar{r}_t$. Base action probabilities can range from below zero and greater than 1.0 and are converted to selection probabilities using a soft max expression (2.3.3). The probabilities provide a basis for action selection and hence fulfil the role of actor. The reference reward serves as a critic, Table 2.5.1.

(2.5.1) Selection rule	(2.5.2) Base action probabilities
$\pi_t(a) = \frac{e^{p_t(a)}}{\sum_{b=1} e^{p_t(b)}}$, $a_t \sim \mathbb{D}[\pi_t(1), \pi_t(2), \ldots \pi_t(n)]$	$\bar{r}_o = 1/n$, $p_0(a_0) = 1/n$ $p_t(a_t) = p_{t-1}(a_t) + \beta[r_t - \bar{r}_t]$, where $\beta > 0$ $\bar{r}_t = \bar{r}_{t-1} + \alpha[r_t - \bar{r}_t]$, where $0 < \alpha \leq 1$

Table 2.5.1 Action selection and value expression for Reinforcement Comparison

The best %optimal action is 96% for $\alpha = 0.2$ and $\beta = 0.01$ (See Appendix 0) but before t=250 the value is the lowest, see Figure 2.7.1

2.6 **Upper Confidence Bound action selection**

Hoeffding's inequality, Hoeffding (1963) provides a very useful bound of the sample mean. "For any sample size n, $\overline{X}_n$ is bounded below by the population mean μ plus a quantity t with probability δ, dependant on n and t".

(2.6.1) $\mathbb{P}[\overline{X}_n - \mu \geq t] \leq e^{-2nt^2}$

It means that the probability that the sample mean deviates from the population mean by an arbitrary quantity becomes increasingly smaller both as it and the sample size increase. In the bandit case t is the current play, a_t the current action, $k(t, a_t)$ the sample size of $Q_t(a_t)$ at t. The correspondence to above is $k(t, a_t) \stackrel{def}{\equiv} n$, $Q_t(a_t) \stackrel{def}{\equiv} \overline{X}_{k(t, at)}$, $Q^*(a_t) \stackrel{def}{\equiv} \mu$, $\hat{U} \stackrel{def}{\equiv} t$ and the bound $\delta_t = e^{-2k(t, a_t)\hat{U}^2}$, then (2.6.1) becomes:

(2.6.2) $\mathbb{P}[(Q_t(a_t) - Q^*(a_t)) \geq \hat{U}] \leq \delta_t$

Kocsis and Szepesvári (2006) state a symmetric result for $|Q_t(a_t) - Q^*(a_t)|$ (2.6.4) and suggest if δ_t is set to t^{-4} a bound can be determined as a function of t alone (2.6.3) which takes into account the two tails (2.6.4).

(2.6.3) $\hat{U}_t(a_t) \stackrel{def}{\equiv} \sqrt{(\log(1/\delta)/(2k(t, a_t))} = \sqrt{(\log(t)/(2k(t, a_t))}$

(2.6.4) $\mathbb{P}[Q^*(a_t) \geq -Q_t(a_t) + \hat{U}_t(a_t) \text{ OR } Q^*(a_t) \geq Q_t(a_t) + \hat{U}_t(a_t)] \leq 2t^{-4}$

The selection rule is now based on an estimate of the maximum value $Q_t(a)$ could be to a given probability, given the variation in Q_t . The upper bound $Q_t(a) + \hat{U}_t(a)$ from the second term of (2.6.4) gets the same probability of t^{-4} across all a and the maximum is used to provide the best a_t.

(2.6.5) $a_t = \text{argmax}_{a \in A}(Q_t(a) + \hat{U}_t(a))$

The simulation shown in Figure 2.7.1 deploys a UCB selection rule. It shows the best % optimal action of 98% after learning over 10000 plays.

2.7 **Constant α ϵ-greedy method**

The constant α approximation method introduces a departure from the sample average term 1/k(t, *a*) to a *learning rate* α_t that allows changes to the reward means Q*(*a*), to be tracked if non-stationary but at the risk of a failure to converge. Drawing from *stochastic approximation* theory (Robbins & Monro, 1951) laid down conditions for convergence to the population values, Q*(*a*):

(2.7.1) $\sum_{t=0}^{\infty} \alpha_t = \infty$ and

$\sum_{t=0}^{\infty} \alpha_t^2 \leq \infty$ and

If each (s, *a*) visited infinite number of times then $\mathbb{P}[\lim t \to \infty Q_t(a_t) = Q^*(a)] = 1$ (almost surely)

The update equation for action values is:

(2.7.2) $Q_t(a) = Q_{t-1}(a) + \alpha_t (r_t - Q_{t-1}(a))$

Figure 2.7.1 shows that using T=2,10000 plays with X=1000 epochs each, using the best value of α contingent on the best value of ϵ determined independently, (See Appendix 0) gives 87% optimal action.

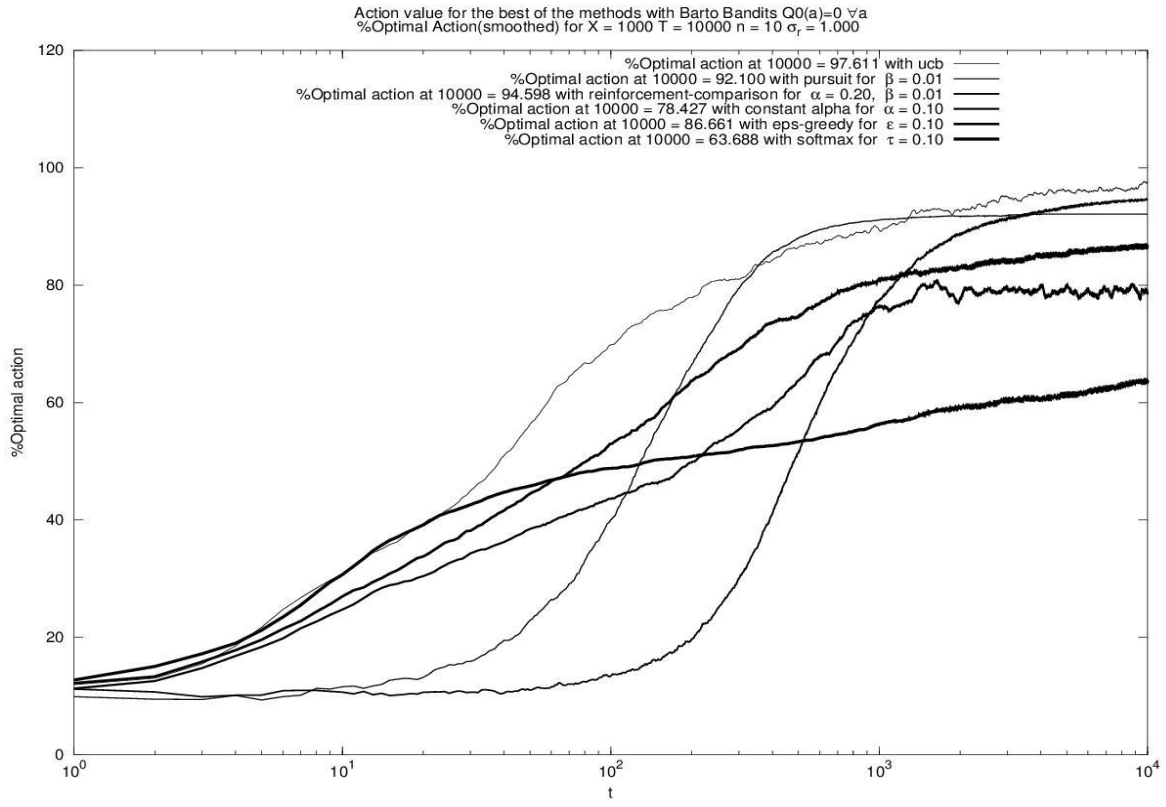


Figure 2.7.1 Plot of %optimal action with the number of plays for learning for a 10-armed bandit

Figure 2.7.1 above shows the % optimal action for the best parameters of each of the methods above and are rerun and plotted for 10000 steps. It can be concluded there is considerable variation in performance, latency and and asymptotic direction of the six selection rules above with some still trending at episodes of over 10^4. Apendix 0 shows

that the %optimal action depends, usually unimodally, on the learning parameters ζ whose values are those which maximise the %optimal action at T=1000 Weatherwax, (2005,chapter 2). Table 2.7.1 shows more clearly the best parameter values and the corresponding optimal action percentages for X=1000 and T=10000. The higher T value was introduced to reveal more long run behaviour which is still slowly increasing for some selection rules, in particular softmax, and does change the best parameters for Reinforcement Comparison. The value of n in the header is equivalent to A=10.

Selection Rule	Parameters and Values	%optimal
UCB		98
RC	$\alpha = 0.2, \beta = 0.01$	96
Pursuit	$\beta = 0.01$	92
ϵ-greedy	$\epsilon = 0.1$	87
Constant α	$\alpha = 0.1$	78
softmax	$\tau = 0.1$	64

Table 2.7.1 The maximising parameter values and the corresponding %optimal action

Although UCB does come out with the highest it has significant more variation than reinforcement comparison. It confirms the result of Silver (2014, Lecture 9 Exploration and Exploitation, Figure 9) that UCB will dominate for the most of the learning range. In the mid range pursuit is optimal for steps 500 to 1000 but falls away to UCB. CRAThe learning protocol is particularly inefficient in that the population means of the action rewards are chosen from a standard normal deviates with considerable duplication. Clearly the best selection rule has estimates of the average reward which are able to distinguish the most rewarding action form the next best and so on. Chapter 5 derives a more efficient validation. The distribution of the difference between the highest two sample means will determine the learning difficulty and therefore profoundly influence the % optimal. The validation of the learnt policy is more correctly the percentage of the predicted optimal actions that equal the population optimal a*. Using Table 2.1.1 the expression is Σ_e a*== arg maxWithRandomTieBreaks Qt over all the epochs of Qt.

2.8 **Definition and implementation of convergence**

The issue of lengthy convergence of action values is a serious bottleneck to fast lightweight learning algorithms. Many of the validation experiments have sample sizes ~$10^{6\sim7}$. The challenge is to address the problem of extimating convergence to a given level of probability and thus contribute to fast learning algorithms, Objective 4.2.1. The first step is to establish a definition for stochastic convergence on which to develop an expression for PMSMMPM. A useful definition of convergence is given by Mnih etal (2008). "Given any ϵ however small, Q_t *converges in probability* to Q* if we can always find a sampling set of size T such that for all $t \geq T$, $||Q_t - Q^*|| < \epsilon$ with probability $1-\delta$. T will depend on ϵ

and δ". Normally δ is a small number, conventionally 0.05, and relates to probability of non-convergence for some T. Convergence in probability is also the type of convergence established by the *weak law of large numbers* (Kendall and Stuart, 1997). Barto (1981) suggests the operational bound on successive differences $|Q^{\pi}_{T+1} - Q^{\pi}_{T}|_{\infty} < \epsilon$ and which is explored below.

The classical *confidence interval* (Walpole, 1982) conforms thus but does rely on knowledge of the population variance. This is an expression in T, δ and $Q^{\sigma}(a)$ the standard deviation of $Q_T(a)/\sqrt{T}$. The confidence interval is Mnih compliant since given any ϵ there is a T which satisfies $P(||Q_T - Q^*|| < \epsilon) = (1-\delta)$ or equilivantly $P(-\epsilon < Q_T - Q^* < \epsilon) = 1- \delta$. Since the probability in the ranges $]-\infty\ -\epsilon]$ $[\ \epsilon\ \infty\ [$ is equal, each has value of $(1-(1- \delta))/2 = \delta/2$, then:

$P(Q_T - Q^* < \epsilon) = 1- P(Q_T - Q^* > \epsilon) = 1- \delta/2$

Normalising with the standard deviation of Q_T which is $Q^{\sigma}(a) /\sqrt{T}$

$P(Q_T - Q^* < \epsilon) = P((Q_T - Q^*) / (Q^{\sigma}(a)/\sqrt{T}) < \epsilon/(Q^{\sigma}(a) /\sqrt{T})) = 1- \delta/2$. Introducing the cdf Φ

$\Phi(\epsilon / (Q^{\sigma}(a) /\sqrt{T}) = 1- \delta/2 \Rightarrow \epsilon\sqrt{T} / Q^{\sigma}(a) = \Phi^{-1}((1- \delta/2) \Rightarrow T = (Q^{\sigma}(a) / \epsilon)\Phi^{-1}((1- \delta/2)$ and

(2.8.1) $||Q_t - Q^*|| < \Phi^{-1}(1- \delta/2)\ (Q^{\sigma}(a)/\sqrt{T})$ with probability $(1 - \delta)$,

Also Mnih compliant is UCB (Hoeffding 1963) which uses an expression in t and δ. It has the advantage of independence from population parameters of any assumed underlying distribution and features only the sample size, T, so has general applicability.

(2.8.2) $||Q_t - Q^*|| < \sqrt{(\log(1/\delta)/2t)}$ with probability $(1 - \delta)$ from (2.6.4)

More recently Mnih etal (2008) have exploited Bernstein (1927) bounds on the sum of random variables. Mnih (2008) proposed an empirical similar to UCB but with terms that depends on an estimate of the sample variance, σ_t and range R.

(2.8.3) $||Q_t - Q^*|| < \sigma_t\sqrt{(2\log(3/\delta)/t) + (3/t)R\log(3/\delta)}$ with probability $(1 - \delta)$

The variance of the sample average will provide valuable information on its efficiency and this extra information is discarded in the classical estimates of sections 2.1 – 2.7. The next section explores how to use variance information to differentiate the action of maximum action value mean from noisy samples. To address the problem of stopping, sample estimates are calculated and stopped when a given level of probability has been reached.

2.9 **Probability based exploration**

This section takes objective 4.1 forward by formulating the probability distribution of the maximum of a several stochastic processes. True action selection is based on that action having the greatest population mean. Since the sample estimates have a distribution about

the population mean it is possible for the greater sample mean not to be greater population mean. Rather than exploit for the convergence of each $Q_t(a)$ it is only necessary to explore until the top two Q_t are different enough for the probability of the action of maximum sample mean being the maximum population mean (PMSMMPM) by at least $1 - \delta$, where δ is a small number. Forbes (2000) makes a related observation regarding his use of instance based learning. "An interesting future direction would be to design a heuristic for instance–averaging which takes into account the reduced utility of suboptimal Q values for a particular state".

Order statistics (Kendall and Stuart, 1997) are used to find an expression for PMSMMPM in terms of population parameters of the reward distribution for each action. At any t a mapping $d_t(a)$ that permutes $Q_t(a)$ into descending order $Q^S_t(a)$ can be found s.t. (2.9.1)

(2.9.1) $Q^S_t(a) \stackrel{\text{def}}{\equiv} Q_t(d_t(a))$ and $Q_t^S(1) \geq Q_t^S(2) \geq \ldots Q_t^S(A)$.

Let the random variable Z be the maximum of $\{Q_t^S(2),\ldots Q_t^S(A)\}$:

(2.9.2) $Z \stackrel{\text{def}}{\equiv} \max_{a\#1} Q^S_t(a)$, then the cumulative density function (cdf) of Z is $\mathbb{P}[Z \leq z]$. Since the sample estimates are independent RV's the cumulative density function (cdf) can be separated:

(2.9.3) $\mathbb{P}[Z \leq z] = \mathbb{P}[Q_t^S(2) \leq z \cap Q_t^S(3) \leq z \cap ..] = \mathbb{P}[Q_t^S(2) \leq z\ ]\mathbb{P}[Q_t^S(3) \leq z\ ]\ldots$

$= \prod_{a=2}^{A} F(z|\ Q^{*S}(a), Q^{\sigma S}(a))$, F is cdf of $Q_t^S(a)$.

$Q_t^S(1)$ has probability density function (pdf) $\mathbb{P}\ (Q_t^S(1) == z)) = f(z|\ Q^{*S}(1), Q^{\sigma S}(1))$, where the mean of $Q_t^S(1)$ is $Q^{*S}(1)$ and its standard deviation, $Q^{\sigma S}(1)$ given by the $\sqrt{\ } V^S(1)/k$, where $V^S(1)$ is the population variance for reward r for action d(1), $V^S(1) \stackrel{\text{def}}{\equiv} \mathbb{V}\text{ar}[r_t \mid a_t == d(1)] \stackrel{\text{def}}{\equiv} \mathbb{E}[\ (r_t - Q^*(d(1)))^2]$ and k is the sample size of $Q_t^S(1)$. So the probability that action 1 has the maximum population mean is $\mathbb{P}[Z \leq Q_t^S(1)]$.

(2.9.4) $\mathbb{P}[Z \leq Q_t^S(1)] = \int \mathbb{P}[Z \leq z \cap z \leq Q_t^S(1) \leq z+dz)dz = \int \mathbb{P}[Z \leq z]\ f(z|\ Q^{*S}(1), Q^{\sigma S}(1)]\ dz$

So $\mathbb{P}[\max_{a\#1} Q^S_t(a), \leq Q_t^S(1)]$ is the expectation of $\prod_{a=2}^{A} F(z|\ Q^{*S}(a), Q^{\sigma S}(a))$ over the density $f(z|\ Q^{*S}(1), Q^{\sigma S}(1))$.

(2.9.5) $\mathbb{P}[\max_{a\#1} Q^S_t(a) \leq Q_t^S(1)] = \int f(z|\ Q^{*S}(1), Q^{\sigma S}(1)) \prod_{a=2}^{n} F(z|\ Q^{*S}(a), Q^{\sigma S}(a))\ dz$.

A formal expression for PMSMMPM has been established in terms of the population parameters of the component distributions. To meet the convergence criterion for $\mathbb{P}[\max_{a\#1} Q^S_t(a) \leq Q_t^S(1)]$ requires:

(2.9.6) $\mathbb{P}[\max_{a\#1} Q^S_t(a) \leq Q_t^S(1)] > 1 - \delta$ for some t.

If δ is regarded as a significance level and is set to 0.05 the non-convergence criterion is $0 \leq \mathbb{P}[\ \max_{a\#1} Q^S_t(a) \leq Q_t^S(1)] \leq 1 - \delta = 0.95$. It means that there is a probability of 0.05 that the sample means obey $\max_{a\#1} Q^S_t(a) > Q_t^S(1)$ but the population means obey

$\max_{a\#1} Q^{*S}(a) \leq Q^{*S}(1)$. If the result $\max_{a\#1} Q^{S}_{t}(a) > Q_t^{S}(1)$ is taken as true then a statistical Type I error is made. As it stands (2.9.6) is intractable in closed form for A > 2 so simplifications and numerical approximations are required.

There are the cases where there is a tie between the top n action value means, $Q^{*S}(1) == Q^{*S}(2)$ for n=2, $Q^{*S}(1) == Q^{*S}(2) == Q^{*S}(3)$ for n=3 and so on. Equation (2.9.4) would need to be reworked to achieve and equivalent for (2.9.6) for each n.

2.10 **Exact solution for A=2 using population means and standard deviation**

In the case A=2 (2.9.5) becomes:

(2.10.1)P21 $\stackrel{\text{def}}{=} \mathbb{P}[Q^{S}_{t}(2) \leq Q_t^{S}(1)] = \int f(z|\, Q^{*S}(1), Q^{\sigma S}(1))\, F(z|\, Q^{*S}(2), Q^{\sigma S}(2))$

Given two RVs, X and Y where $X \sim \mathbb{N}(\mu_X, \sigma_X^2)$ and $Y \sim \mathbb{N}(\mu_Y, \sigma_Y^2)$ the probability that Y< X is obtained from an expression for the probability that $X-Y > 0$. The mean and variance of $Y-X$ is $\mathbb{E}[Y - X] = \mu_Y - \mu_X$ and $\mathbb{V}ar[Y - X] = \sigma_X^2 + \sigma_Y^2$ respectively.

The distribution of any linear combination of normal deviates has mean equal to the linear combination of the separate means and variance equal to the weighted sum of the separate variances:

$Y - X \sim N(\mu_Y - \mu_X, \sigma_X^2 + \sigma_Y^2)$

P[Y – X < 0] can be expanded if Y – X is standardised to a normal deviate and then an expression for P[Y – X < 0] in terms the standard normal cdf Φ is derived:

$P[Y < X] = P[Y - X > 0]$

$= P[(Y - X - (\mu_Y - \mu_X)/\sqrt{(\sigma_X^2 + \sigma_Y^2)} > -(\mu_Y - \mu_X)/\sqrt{(\sigma_X^2 + \sigma_Y^2)}] = \Phi[-(\mu_Y - \mu_X)/\sqrt{(\sigma_X^2 + \sigma_Y^2)}]$

$P[Y < X] = \Phi[(\mu_X - \mu_Y)/\sqrt{(\sigma_X^2 + \sigma_Y^2)}]$

Illustrating this result as follows:

If $\mu_Y \gg \mu_X$ then $\mu_X - \mu_Y$ is $\ll 0$ and so Φ [..] is very small. If the mean of Y is much greater than that of X it is very unlikely that Y is less than X.

If $\mu_X == \mu_Y$ then $\mu_X - \mu_Y == 0$ and Φ[..] =0.5. If the means are equal then it is equally probable whether X is either greater or less that Y.

Inserting $X = Q_t^{S}(1)$, $\mu_X = Q^{*S}(1)$, $\sigma_X^2 = Q^{\sigma S}(1)^2$, and similarly for Y into the RHS above yields a simple expression, denoted P12, in the population means and variances for the LHS of 2.10.1.

(2.10.2) P12 $\stackrel{\text{def}}{=} \mathbb{P}[\, Q^{S}_{t}(2) \leq Q_t^{S}(1))] = \Phi(\, (Q^{*S}(1) - Q^{*S}(2)) \,/\, \sqrt{(Q^{\sigma S}(2)^2 + Q^{\sigma S}(1)^2)}$

Looking at (2.10.2) let $\Delta Q = Q^S_t(1) - Q_t^S(2)$ and its standard deviation, $SQ = \sqrt{(Q^{\sigma S}(2)^2 + Q^{\sigma S}(1)^2)}$. Then at the critical region ΔQ is proportional to SQ with slope given by $\Phi^{-1}(1-\delta)$. This result implies that the 0.05 significance level is reached if the difference $\Delta Q \geq \Phi^{-1}(1-0.05) = 1.644$ SQ. It may be the case that taking the top two values and ignoring the rest will result in an expression which is faster to evalute.

It is straightforward to show that P21 converges if either of sample sizes go to infinity. The argument to Φ in 2.18.3 is monotonically increasing in either sample size. The cdf function is also a monotonically increasing function and bounded above. The RHS is therefore a monotonically increasing sequence with either sample size and bounded above. By the fundamental axiom (Scott and Tims, 1966, p110) it is concluded that P21 converges. It is plausible to extend this result to the case of A actions.

2.11 **Plotting the components of $\mathbb{P}[\max_{a\#1} Q^S_t(a) \leq Q_t^S(1)]$ for A=3**

To investigate the effect of a third action value the five components for $\mathbb{P}[\max_{a\#1} Q^S_t(a) \leq Q_t^S(1)]$ (2.9.5) are plotted in Figures 2.11.1 and 2.11.2 to show the effect of taking the maximum of $Q_t(2)$ and $Q_t(3)$. The curves 1, 2 and 3 below are plotted in Figure 2.11.1:

1. $F(z, Q^{*S}(3), Q^{\sigma S}(3))$,
2. $F(z, Q^{*S}(2), Q^{\sigma S}(2))$,
3. $\prod_{a=2}^{A} F(z, Q^{*S}(a), Q^{\sigma S}(a))$,
4. $f(z, Q^{*S}(1), Q^{\sigma S}(1))$,

and

5. $f(z, Q^{*S}(1), Q^{\sigma S}(1))\prod_{a=2}^{A} F(z, Q^{*S}(a), Q^{\sigma S}(a))$.

The product term 3 shows the combined probabilities of $Q^S(2)$ and $Q^S(3)$ and is systematically less than $Q^S(2)$ showing the influence of a competing value albeit one unit lower than $Q^S(2)$.

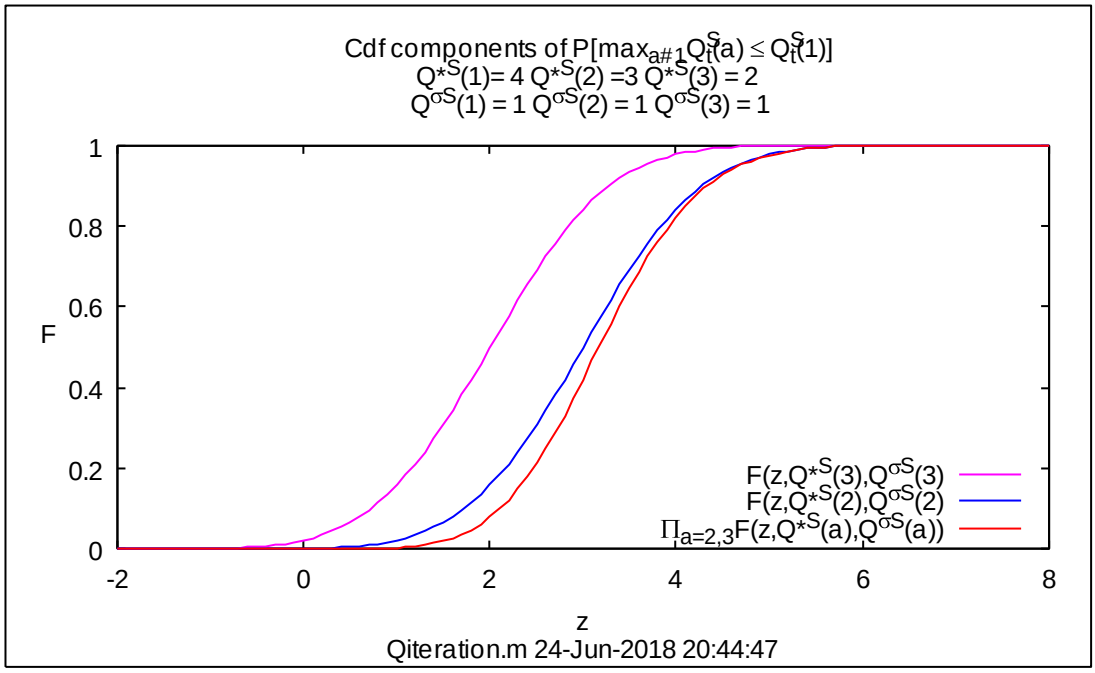


Figure 2.11.1 Plot of cdf of components of P[max$_{a\#1}$ Q(a) <Q(1)]

The pdf curve that corresponds to the density of $\mathbb{P}[\max_{a\#1} Q^S_t(a) \leq Q_t{}^S(1)]$, 5 above, shows a smaller area than the density of $Q^S(1)$, 4 above (Figure 2.11.2).

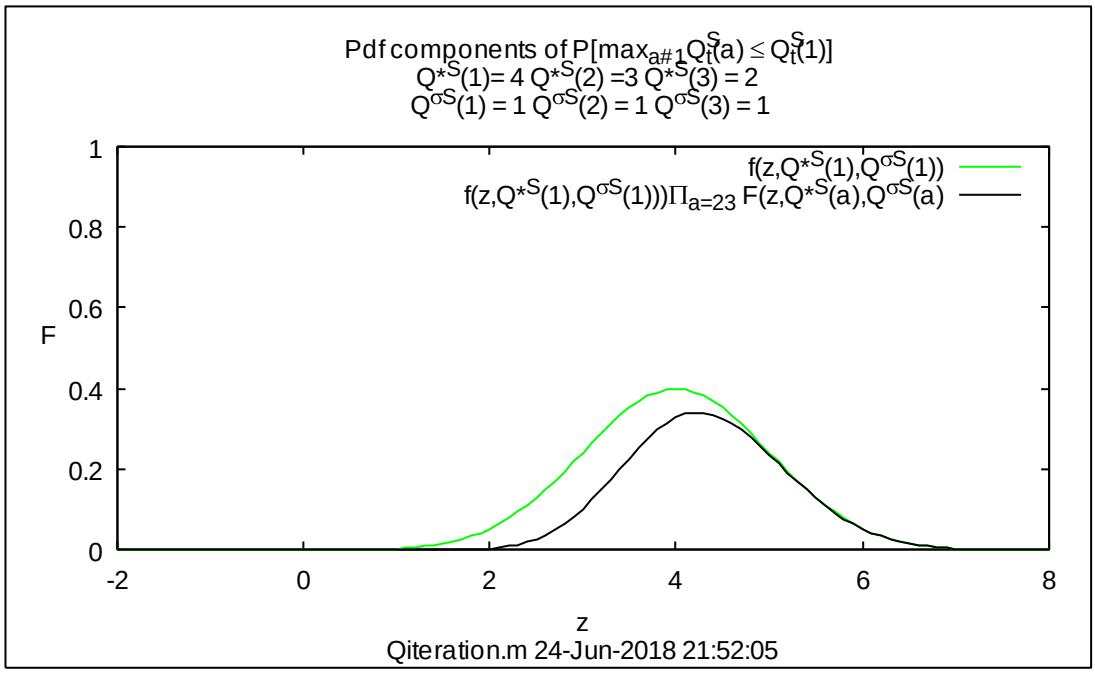


Figure 2.11.2 Plot of pdf of components of P[max$_{a\#1}$ Q(a) < Q(1)]

Q(1), component 4, is the upper curve of Figure 2.11.2 and is shifted to the right when convoluted with $\prod_{a=2}{}^A$ F(z, Q*^{S}(*a*), Q$^{\sigma S}$(*a*)), shown as the lower curve. The deviates of Q(2) and to a lesser extent Q(3) which have means less than 4, provide some probaility density for the condition Q(2) > Q(1) and Q(3) > Q(1) and which therfore reduce the denisty of Q(1) when convoluted with $\prod_{a=2}{}^A$ F(z, Q*^{S}(*a*), Q$^{\sigma S}$(*a*)). The effect on $\mathbb{P}[\max_{a\#1} Q^S_t(a) \leq Q_t{}^S(1)]$ of the third greatest term $Q_t{}^S(3)$ will determine whether it can be

ignored and just the A=2 case of (2.9.1) used. Table 2.11.1 shows $\mathbb{P}[\max_{a\#1} Q^S_t(a) \le Q_t^S(1)]$ calculated for a set of values of the population mean for the third greatest term a=3.

$Q^{*S}(1)$	$Q^{*S}(2)$	$Q^{*S}(3)$	$\mathbb{P}[\max_{a\#1} Q^S_t(a) \le Q_t^S(1)]$
4.000	3.000	3.000	$0.63367^{(2)}$
4.000	3.000	2.000	0.72872
4.000	3.000	1.000	0.75560
4.000	3.000	-5.000	0.76022
4.000	3.000	Absent	$0.76025^{(1)}$

Table 2.11.1 $Q^{*S}(a)$ a=1,2,3 and $\mathbb{P}[\max_{a\#1} Q^S_t(a) \le Q_t^S(1)]$

Row five above shows the value of $\mathbb{P}[\max_{a\#1} Q^S_t(a) \le Q_t^S(1)]$ without $Q_t^S(3)$ and from (2.9.2) the value is:
$\Phi((Q^{*S}(1) - Q^{*S}(2))/\sqrt{(Q^{\sigma S}(2)^2 + Q^{\sigma S}(1)^2)} = \Phi((4-3)/\sqrt{(1+1)}) = \Phi(1/\sqrt{2}) = 0.76025^{(1)}$.
The worse case is when $Q_t^S(3) = Q_t^S(2)$ which does show $\mathbb{P}[Q^S_t(2) \le Q_t^S(1)]$ to be an overestimate of (.76025 - .63367)/0.76025 ≈ 17% since the value $\mathbb{P}[\max_{a\#1} Q^S_t(a) \le Q_t^S(1)]$ of is $0.63367^{(2)}$ from row one column four of Table 2.11.1. The result of investigating a tied third Q is drop in $\mathbb{P}[\max_{a\#1} Q^S_t(a) \le Q_t^S(1)]$, which could be mitigated if the variances of the Q's are reduced. The implication is that a greater sample size is needed to disambiguate them.

2.12 **Well condtioned incremental calculation of the sample variance**

To complete 4.2.1 the calculation of (2.9.5) needs well conditioned expressions for incremental estimates of the variance of sample means since both bandit and later MDP generate data as a stream. The calculation of (2.9.5) needs incremental estimates of the variance of the sample mean since both bandit and later MDP generate data as a stream. Like the sample mean an estimate of the sample sum of squares of deviations as each new data item arrives is needed. The population variance of the reward of taking action a is $V(a)$ and an unbiased estimate of the population variance of the reward of taking action a is $s_t^2(a)$.

$V(a) \stackrel{def}{\cong} \mathbb{V}\mathrm{ar}[r_t \mid a_t == a] = \mathbb{E}[(r_t - Q^*(a))^2]$

(2.12.1) $s_t^2(a) \stackrel{def}{\cong} (1/(k(t, a) - 1))\Sigma_{j=1}^{k(t, a)} (r_{i(a, j)}) - Q_t(a))^2 = (\Sigma_{j=1}^{k(t, a)} (r_{i(a, j)})^2 - k(t,a)Q_t(a)^2$

Introducing the sum of squares of deviations for action a, $S_t(a)$, and following Finch (2009) the well conditioned incremental expression for S_t is:

(2.12.2) $S_t(a) \stackrel{def}{\cong} \Sigma_{j=1}^{k(t, a)} (r_{i(a, j)})^2 - k(t, a)Q_t(a)^2$

(2.12.3) $S_t(a) - S_{t-1}(a) = (r_{i(a, k(t, a)} - Q_t(a)) (r_{i(a, k(t, a)} - Q_{t-1}(a))$

2.13 **Operational safeguard against premature stopping due to low sample size**

For vary small sample sizes the variance of the underlying $r_{i(a,\, k(t,\, a)}$ is not stable and particular extreme values will lead to premature high values of $\mathbb{P}[\max_{a\#1}(Q_t^S(a)) \leq Q_t^S(1]$. Sharma's (2008) lower bound to the sample variance has been implemented and Appendix 1 shows an empirical determination that at 5 cases the bound is stable. Let $\{y_i\}$ be a sample of N RVs having variance $\sigma^2_{Y,}$ then the arithmetic mean A and harmonic mean H provide bounds for σ^2_Y:

(2.13.1) $A\mu \stackrel{def}{=} (1/N)\Sigma_i\, y_i$, $H\mu \stackrel{def}{=} N/\Sigma_i\, 1/y_i$

(2.13.2) $\sigma^2_Y \leq y_{max}(A - H)(y_{max} - A)/\,(y_{max} - H)$, $\sigma^2_Y \geq y_{min}(A - H)(A - y_{min})/\,(H - y_{min})$

The implication is that five samples or more must be generated before applying any stopping rule based on sample variances.

2.14 **Bandits implementation with PMSMMPM stopping with round robin starts**

The algorithm below implements a Bandits simulation using incremental estimate of the action value means, sum of squares, sorted action values and $\mathbb{P}[\max_{a\#1}(Q_t^S(a)) \leq Q_t^S(1)]$. Action selection is initially Round Robin for RR cycles and then uses UCB to achieve action selection. The estimates $Q_t(a)$ for $Q^{*S}(a)$, and $s_{\mu t}(d_t(a))$ for $Q^{\sigma S}(a))$ are used for the calculation of $\mathbb{P}[\max_{a\#1}(Q_t^S(a)) \leq Q_t^S(1)]$. The output of the algorithm includes the count of plays to stopping at time t_x, Mean of t_x and Standard Deviation of mean of t_x. The key exercise is to assess whether (2.9.6) provides a stopping rule at $\delta = 0.05$.

#MRP, Learning and simulation constants

```
X=5; T=120;
```

```
for x = 1:X                    #X is the number epochs having different Q*(a)
```

#Initialise Q_0, k(0, *a*), S_0,s_0^2 , $s_{\mu 0}^2$, $\hat{U}_0$, RR

```
Q0=Qt=Qt=Qtm1=k =smut2=st2=Std= St=Stm1= zeros(1, A);
```

#Generate play t, T is the maximum number

```
for t=1:T                      #t is the play number, T is the maximum number
```

#Action selection

if($t \leq RR*A$) $a_t = \text{mod}(t\text{-}1, A) + 1$ #Force all actions to be tried for $t \leq RR*A$

```
if(t<=RR*A) at = mod(t-1,  A) + 1;
```

else $a_t = \arg\max_a (Q_{t-1}(a) + \hat{U}_{t-1}(a))$ #UCB $\hat{U}_0(a)$ only defined for k(t, *a*) > 0, $\forall\, a$

```
[Qm, adesc]= sort(Qtm1 .+ Ut, "descend");
```

```
at=adesc(1);
```

#Generate reward for playing action a_t

$r_t = Q^{\sigma S}(a)*R(t, a_t) + Q^*(a_t)$ #R(t, a_t) standard Normal RV ~ $\mathbb{N}(0, 1)$

```
rt=Qsd(a)*R(t, at) + Qpop(at);
```

#action count and action value mean update

$k(t, a_t) = k(t, a_t) +1$

```
k(at)+= 1;
```

$Q_t(a_t) = Q_{t-1}(a_t) + (r_t - Q_{t-1}(a_t))/k(t, a_t)$ #$Q_1(a_1) = r_t$ for any a_1

```
Qt(at) = Qtm1(at) + (rt - Qtm1(at))/k(at) ;
```

#Action value sum of squares update using the previous Q value for this action

$S_t(a_t) = S_{t-1}(a_t) + (r_t - Q_t(a_t))\,(r_t - Q_{t-1}(a_t))$ #$Q_0(a)$ is 0, but $r_0 = Q_t(a_0)$ so $S_1(a_t) =0$

```
St(at) = Stm1(at) + (rt - Qt(at))*(rt - Qtm1(at));
```

#Compute an unbiased estimate of the population variance of the reward of taking action *a*,

$s_t^2(a_t) = (1/(k(t, a_t) -1))S_t(a)$ #needs $k(t, a_t) \geq 2$

```
if(k(at)>=2)st2(at) = St(at)/(k(at)-1); endif
```

#Compute an unbiased estimate of the population variance of the action value of *a*

$s_{\mu t}^2(a_t) = s_t^2(a_t)/k(t, a_t)$ # $k(t, a_t) \geq 1$

```
if(k(at)>=1)smut2(at) = st2(at)/k(at); endif
```

#Sort Q_t in descending order into Q_t^S with sort order $d_t(a)$ $Q_t^S(a) = Q_t(d_t(a))$

```
[Qts, dts]= sort(Qt , "descend");
```

#Compute $\mathbb{P}[\max_{a\#1}(Q_t^S(a) \leq Q_t^S(1)]$ using $Q_t(a)$ for $Q^{*S}(a)$, and $s_{\mu t}(d_t(a))$ for $Q^{\sigma S}(a))$

$\mathbb{P}[\max_{a\#1}(Q_t^S(a)) \leq Q_t^S(1)] = \int \phi(z, Q_t^S(1), s_{\mu t}(d_t(1))) \prod_{a=2}^{A} F(z, Q_t^S(a), s_{\mu t}(d_t(a)))dz,$

```
f1FA = @(x) normpdf(x, Qts(1), smut2(ats(1))^0.5);#Bandits 7
for a=2:A
f1FA= @(x)f1FA(x).*normcdf(x,Qts(a),smut2(ats(a))^0.5);
endfor
if(t >= RR*A)PQs(1)= quadcc(f1FA , -8, 8); endif
```

#Compute $\mathbb{P}[Q_t^S(a)) \leq Q_t^S(1)]$, *a* # 1

$\mathbb{P}[Q_t^S(a)) \leq Q_t^S(1)] = \Phi\,((Q_t^S(1) - Q_t^S(a))/\sqrt{(s_{\mu t}^2(d_t(1))) + s_{\mu t}^2(d_t(a)))})$

```
for a=2:A
if(t>= RR*A)PQs(a)= normcdf((Qts(1)- Qts(a))/sqrt(smut2(ats(1))+
smut2(ats(a))));endif
```

```
    endfor
```

#Compute t_x the first t s.t. $t \geq A*RR$ and $\mathbb{P}(\max_{a\#1}(Q_t^S(a)) \leq Q_t^S(1)) > 1 - \delta$

```
    if(t>=RR*A & PQs(1)>1-delta & !Pqfound)tx= t; atx2= ats(1);
Pqfound=1;endif
```

#Calculate the UCB $\hat{U}_t(a) = \sqrt{}(2\log t / k(t, a_t)$, $k(t, a)$ is >1 so it is always conditioned

```
    Ut = sqrt(2*log(t)./k);
```

#Terminate plays if $t \geq A*RR$ and $\mathbb{P}[\max_{a\#1}(Q_t^S(a)) \leq Q_t^S(1)] > 1 - \delta$

```
if(t >= RR*A  & PQs(2) > 1-delta) break; endif

    endfor #t

 endfor #x
```

#Outputs

#Counts of plays to stopping at time t_x, Mean of t_x and Standard Deviation of mean of t_x

Figure 2.14.1 shows the calculation of PMSMMPM which achieves a value of 0.988 when t=26 at which convergence is assumed since it exceeds 1-0.05. A value of RR of 6 is used to ensure premature convergence does not occur.

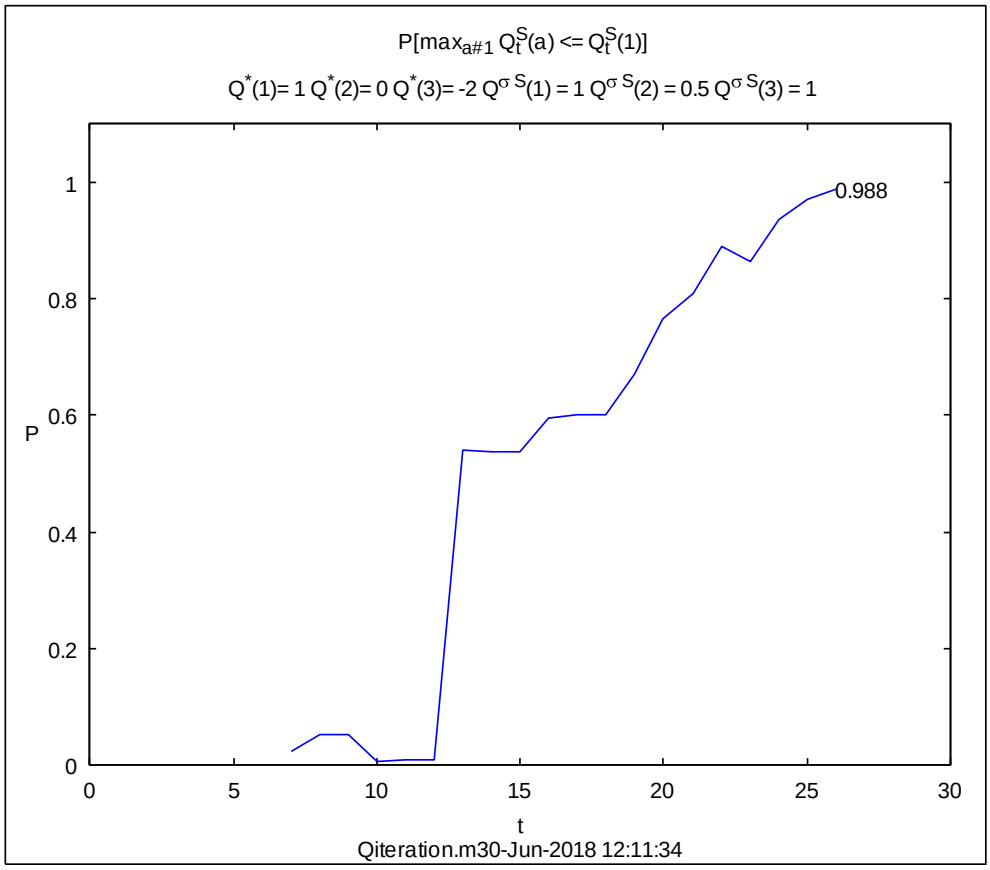


Figure 2.14.1 Plot of $P[\max_{a\#1}(Q_t^S(a)) \leq Q_t^S(1)]$ with t

The simulation of PMSMMPM using 3-armed bandit with shows monotonic rise to the threshold which confirms its suitability for a stopping rule.

The assumption is made that the sample estimates are independent normal RV's which might only be sound at very large sample size by the law of large numbers.

The classic student t-test (Gosset, aka Student, 1908), provides a statistic t, and a critical region for the acceptance of the hypothesis that $Q^{*S}(2) \leq Q^{*S}(1)$ based on estimates $Q_t^S(1) - Q_t^S(2)$, having estimated variances $s_{\mu t}^2(d_t(1))$ and $s_{\mu t}^2(d_t(2)$ and sample sizes k(t, dt(2)), k(t, dt(2)) respectively. It entails the calculation of the degrees of freedom based the estimates of the action value sample variances. Use of student t would require its cdf instead of Φ. Because the formulae for the t distribution is complicated and is well approximated by the standard normal distribution for samples over five (Pollard, 1998), it was decided to use a straight substitution of sample variances into 2.10.2.

Sutton & Barto (1998) suggests that a check on the absolute successive differences provide a stopping condition. In the Bandits case $|\Delta Q_t(a)| = \max_a |Q_t(a) - Q_{t-1}(a)|$. The Bandits update rule is:

$Q_t(a_t) = Q_{t-1}(a_t) + (1/k(t, a_t))(r_t - Q_{t-1}(a_t))$,

$\Delta Q_t(a_t) = (1/k(t, a_t))(r_t - Q_{t-1}(a_t))$

In this case only the Q for the selected action a_t is updated. Those for the non-selected action are unchanged. So the absolute deviation at t, ΔQ_t^* is:

(2.14.1) $\Delta Q_T^* = |Q^{\pi}_{T+1} - Q^{\pi}_T|_\infty = \max_a |Q_T(a) - Q_{T-1}(a)| = |(1/k(T, a_*))(r_T - Q_{T-1}(a_*))|$

The plot in Figure 2.14.2 below of single epoch of plays shows ΔQ_t^*, the lower curve and PMSMMPM, the upper curve against t. It confirms numerically that spot values of ΔQ_t^* at time t do not help to show where convergence has occurred. Successive differences are discontinuous between steps since a* may change which makes a difference formulae ill conditioned. The output of the run shows convergence to a probability 0.9524 occurs at t=142. This result will be sensitive to problem complexity which will change if the population parameters change which is explored below. PMSMPM is shown to provide a convergence criterion better than highly stochastic successive differences.

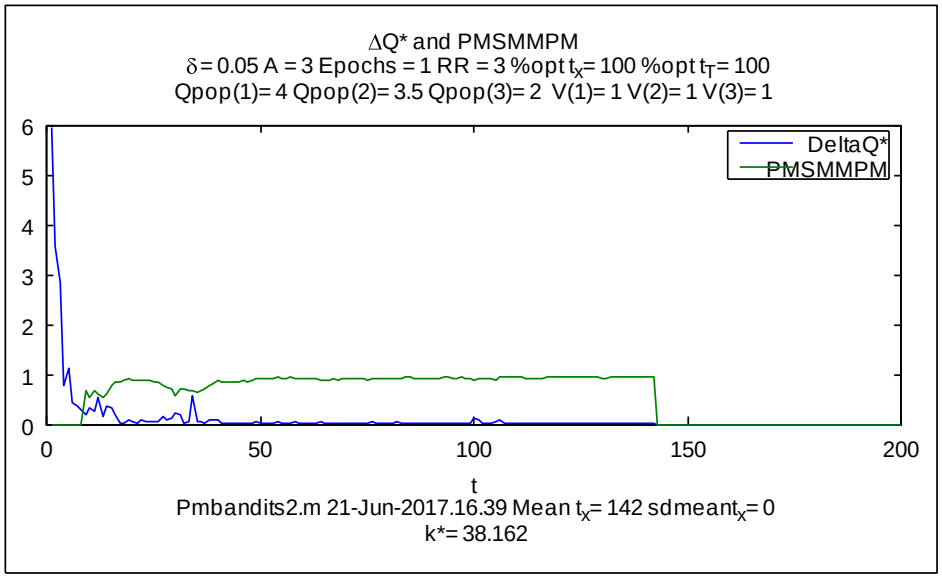


Figure 2.14.2 Plot of absolute successive differences and PMSMMPM with plays

2.15 **Exploring plays to convergence with RR=3**

Pursuing the objective 4.1 and having established the convergence criterion (Figure 2.14.0) it is now possible to define the number of plays to convergence. The number of *plays to convergence* (PTC) is the first t, denoted t_x s.t.

(2.15.1) $\mathbb{P}[\max_{a\#1}(Q_t^S(a)) < Q_t^S(1)] > 1-\delta$ from (2.9.6) such that $t \geq RR*A$.

The action a_{tx}, is defined as the action at time t_x given by (2.6.5). Following equation 2.1.2 the performance statistic is the proportion of plays that choose the optimal action at $t==t_x$ averaged over all epochs X as a percentage, denoted optx $\stackrel{def}{=} \Sigma_x(a_{tx} == a^*)/X*100$ where each epoch x is independent. In order to determine the trend with X Table 2.15.1 shows optx, optT the proportion of plays that choose the optimal action at t = T, the mean of t_x $\mu(t_x)$, the standard error of the mean of t_x $sdm(t_x)$, the theoretical minimum number of plays k* (2.19.8), the median of t_x and the standard deviation of the median (Stigler, 1973).

Experiment date	RR	X	optx	optT	$\mu(t_x)$	$sdm(t_x)$	k*	$median(t_x)$	$std(median(t_x))$
04-Nov-2018.20.49	5	100	98	98	58.8	6.4	38	32	5.00
04-Nov-2018.22.13	5	200	97	97	50.7	3.4	38	26	1.77
04-Nov-2018.22.36	5	300	95	97	52.9	3.0	38	28	4.33
04-Nov-2018.22.36	5	400	97	97	57.9	2.7	38	34	2.86
05-Nov-2018.09.55	5	500	97	97	59.1	2.4	38	36	3.73

Table 2.15.1 The performance statistics with epoch size

Table 2.15.1 uses values Q*(1)=1, Q*(2)=3.5, Q*(3)=2, V(1)=V(2)=V(3)=1 which are chosen so that ΔQ is of similar magnitude to the standard deviation of the difference, $\sqrt{(V(1) + V(2))} \approx 1.4$. It shows the performance statistics level out at 500, in particular optx and std(t_x) which provides empirical evidence that 500 epochs is sufficient for reliable estimation. However the median still shows a standard deviation which is too large to make strong conclusions. The value of RR is five following the conclusionof 2.1.3. A further confirmation of the effect of RR is in the Figure 2.15.1 below which shows the frequency counts for t_x using a value of RR of three. It shows heavy tailed behaviour with over 20 t_x values beyond the T=200 range, although this number may decrease for larger T. It shows 115 converged at t=9 confirming the conjecture regarding RR in 2.1.3. Until the sensitivity to RR is explored it is not easy to interpret this result.

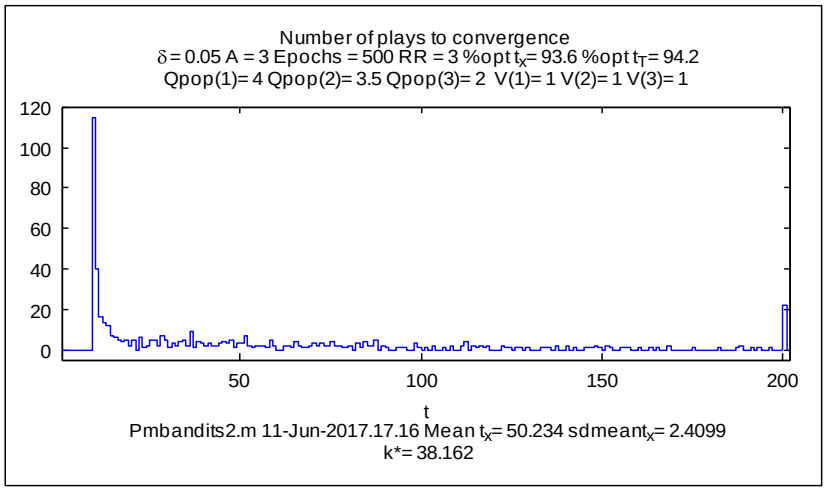


Figure 2.15.1 Plot of frequency of plays to convergence

2.16 **Exploring plays to convergence with RR**

Table 2.16.1 displays the sensitivity of the optx as RR is increased using the same Q and V as Table 2.15.1. A higher RR will stop over optimistic convergence and this should increase the percent convergence at the correct a*.

Experiment date	RR	X	optx	optT	μ(t_x)	sdm(t_x)	k*	median(t_x)	std(median(t_x))
12-Jun-2017.16.30	2	500	88	88	47.0	2.5	38	20	1.40
11-Jun-2017.17.16	3	500	93	94	50.2	2.4	38	27	5.59
12-Jun-2017.16.58	4	500	97	97	60.3	2.6	38	39	2.80
12-Jun-2017.17.50	5	500	97	97	58.1	2.2	38	38	3.73

Table 2.16.1 Shows experiments with increasing RR and the corresponding optimal t_x

It can be concluded that an RR of 5 will achieve 97% correctly learnt a_{tx} An RR of 2 or 3 achieves a lower performance. Experiments show the conditions for use of an early Round Robin selection mitigates volatile low sample sizes.

2.17 **Exploring plays to convergence (PTC) with the variation in Q* and V***

Section 2.10 shows that the dominant influences on $\mathbb{P}[\ Q^S_t(2) \le Q_t^S(1)]$ (2.10.1) are $\Delta Q = Q^S_t(1) - Q_t^S(2)$ and $SQ = \sqrt{(Q^{\sigma S}(2)^2 + Q^{\sigma S}(1)^2)}$ and intuitively that would be expected of $P[\max_a Q^S_t(a) \le Q_t^S(1)]$ which is used to generate t_x (2.15.1). Rather than generate a large set of random Q* it is more efficient just to systematically vary ΔQ and SQ. It is assumed that the terms $Q^S_t(3)$ and above can be neglected. Table 2.17.1, 2 and 3 shows optx and the same statistics and variances as (2.16.1) with systematic variation in Q* and V. In particular Table 2.17.1 shows μ(tx) increasess markedly as ΔQ gets samller with PMSMMPM**.**

date	Q*(1)	Q*(2)	Q*(3)	ΔQ	optx	optT	$\mu(t_x)$	$sdm(t_x)$	k*	$median(t_x)$	$std(median(t_x))$
13-Jun-2017.09.49	4	3	2	1.00	99	99	28.6	1.3	26	15	0.04
12-Jun-2017.17.50	4	3.5	2	0.50	97	97	58.1	2.2	38	38	3.73
13-Jun-2017.10.47	4	3.75	2	0.25	90	89	97.6	3.4	44	76	11.18

Table 2.17.1 Shows variation in optx as ΔQ is decreased

Table 2.17.2 shows the variation in optx with V(1) = 0.5 and 0.75 with the rest 1.0. The Q* are set to experiment 13-Jun-2017.09.49 of Table 2.17.1.

date	V(1)	V(2)	V(3)	optx	optT	$\mu(t_x)$	$sdm(t_x)$	k*	$median(t_x)$	$std(median(t_x))$
13-Jun-2017.09.49	1.00	1	1	99	99	**28.6**	**1.3**	**26**	**15**	**0.04**
13-Jun-2017.16.10	0.75	1	1	99	99	27.6	1.2	23	16	0.36
13-Jun-2017.15.34	0.50	1	1	99	99	26.9	1.2	19	15	0.04

Table 2.17.2 Variation of optx with V(1)

Table 2.17.2 and 3 both show decrease in μ(tx) as V(1), V(2) decrease with PMSMMPM respectively. Table 2.17.3 shows the variation in optx with V(2) = 1.00, 0.5 and 0.75 with the rest 1.0. The Q* are set to run 13-Jun-2017.09.49 of Table 2.17.1.

date	V(1)	V(2)	V(3)	optx	optT	$\mu(t_x)$	$sdm(t_x)$	k*	$median(t_x)$	$std(median(t_x))$
13-Jun-2017.09.49	1	1	1	99	99	**28.6**	**1.3**	**26**	**15**	**0.04**
13-Jun-2017.20.56	1	0.75	1	99	99	21.9	0.7	23	15	0.04
13-Jun-2017.16.23	1	0.50	1	99	99	19.6	0.4	19	15	0.04

Table 2.17.3 Variation of optx with V(2)

The variations with ΔQ and SQ derived from 2.9.6 confirm to their role in the expression for A=2, (2.10.2). The closer Q*(1) and Q*(2) the lower the optx and the longer the mean and median tx clearly showing the drop in optx as it becomes harder to discriminate between very close population means Table 2.17.1. The tighter the variance of either action reduces the mean PTC confirming that is easier to discriminate with less probability overlap Tables 2.17.2 and 3. The stopping rule achieves over 90% optx in all runs.

2.18 **Properties of $P[Q^S_t(2) \leq Q_t^S(1)]$**

The sensitivity of $P[Q^S_t(2) \leq Q_t^S(1)]$ to a change with either k(t, dt(1)) or k(t, dt(2)) is shown below. The result for two actions above in (2.10.2), $\mathbb{P}[Q^S_t(2) \leq Q_t^S(1)]$ can be expressed in terms of the counts of each action $k(t, d_t(1))$, $k(t, d_t(2))$. Let $\Delta Q = Q^S_t(1) - Q_t^S(2)$ and its standard deviation, $SQ = \sqrt{(Q^{\sigma S}(2)^2 + Q^{\sigma S}(1)^2)}$ as above. Now $Q^{\sigma S}(1)^2$ is equal to the sorted variance of the sample mean of size k for reward for action d(1), $V^S(1)$, divided by the size of the sample at time t which is $k(t, d_t(1))$.

(2.18.1) $V(a) \stackrel{def}{=} \mathbb{V}ar[r_t \mid a_t == a] = \mathbb{E}[(r_t - Q^*(a))^2]$

(2.18.2) $V^S(a) \stackrel{def}{=} V(d(a))$

Re-expressing (2.10.2)

(2.18.3) $\mathbb{P}[Q^S_t(2) \leq Q_t^S(1)] = \Phi(\Delta Q_t / \sqrt{(V^S(1)/ k(t, d_t(1)) + V^S(2)/ k(t, d_t(2)))})$

Here the sensitivity of ΔQ_t with sample size is explored by calculating the change in $k(t, d_t(a))$ would be needed to achieve the same $\mathbb{P}[Q^S_t(2) \leq Q_t^S(1)]$ if ΔQ_t were divided by n.

Let $\Delta Q_t' = \Delta Q_t / n$

$\mathbb{P}[Q^S_t(2) \leq Q_t^S(1)]' = \Phi(\Delta Q_t' / \sqrt{(V^S(1) / k(t, d_t(1))' + V^S(2)/ k(t, d_t(2))')})$

$= \Phi((\Delta Q_t /n) / \sqrt{(V^S(1)/ k(t, d_t(1))' + V^S(2)/ k(t, d_t(2))')})$

$= \Phi((\Delta Q_t) / \sqrt{(n^2 V^S(1)/ k(t, d_t(1))' + 4V^S(2)/ k(t, d_t(2))')})$

$= \Phi((\Delta Q_t) / \sqrt{((V^S(1)/(k(t, d_t(1))'/n^2 + V^S(2)/k(t, d_t(2))'/n^2))})$, $k(t, d_t(a))'/n^2 == k(t, d_t(a))$

$k(t, d_t(a))' = n^2 k(t, d_t(a))$.

In general if $\Delta Q_t' = \Delta Q_t/n$, then $k(t, d_t(a))' = n^2 k(t, d_t(a))$. It is concluded that dividing ΔQ_t by n requires multiplying the sample size by n^2 a much greater quantity.

2.19 **An optimal sampling strategy to achieve $P[Q^S_t(2) \leq Q^S_t(1)] > 1 - \delta$**

If the sampling effort is rationed in some sense (2.18.3) can guide the selection of action 1 over 2. Given that a finite number of actions k can be taken the problem is to find optimal $k_1 = k(t, d_t(1))$ and $k_2 = k(t, d_t(2))$ s.t. $k = k_1 + k_2$. From (2.9.2) and assuming $\delta = 0.05$ and ΔQ is known:

(2.19.1) $\Phi^{-1}(.95) = \Delta Q/\sqrt{(V^S(1)/k_1 + V^S(2)/k_2)}$

(2.19.2) $c \stackrel{def}{=} (\Phi^{-1}(.95)/\Delta Q)^{-2} = (V^S(1)/k_1 + V^S(2)/k_2)$

To find the optimum k_1 and k_2 the minimum of k s.t. (2.19.1) holds needs to be found. A Lagrangian $£(k_1, k_2)$ with multiplier λ is introduced:

(2.19.3) $£(k_1, k_2) = k_1 + k_2 + \lambda (V^S(1)/k_1 + V^S(2)/k_2)$

$0 = \partial £(k_1, k_2)/\partial k_1 = 1 - \lambda V^S(1)/k_1^2$
$0 = \partial £(k_1, k_2)/\partial k_2 = 1 - \lambda V^S(2)/k_2^2$

Eliminating λ
(2.19.4) $V^S(1)/k_1^2 = V^S(2)/k_2^2$

$k_1^2/V^S(1) = k_2^2/V^S(2)$

$k_2/\sqrt{V^S(2)} = k_1/\sqrt{V^S(1)}$
(2.19.5) $k_2 = \sqrt{V^S(2)}\, k_1/\sqrt{(V^S(1))} = \sqrt{(V^S(2)/V^S(1))}\, k_1$

With 2.19.5 the k_2 from any k_1 that makes $k_1 + k_2$ optimal can be derived. Solving for k*:

$V^S(2)/k_2 = \sqrt{V^S(2)}\sqrt{V^S(2)}/k_2 = \sqrt{V^S(2)}\sqrt{V^S(1)}/k_1$
$c = V^S(1)/k_1 + V^S(2)/k_2$ VV= $(V^S(1) + \sqrt{V^S(2)}\sqrt{V^S(1)})/k_1$, using above:

(2.19.6) $k_1^* = (V^S(1) + \sqrt{V^S(2)}\sqrt{V^S(1)})/c$, now from (2.19.4):
(2.19.7) $k_2^* = \sqrt{(V^S(2)/V^S(1))}k_1 = \sqrt{V^S(2)} / \sqrt{V^S(1)})(V^S(1) + \sqrt{V^S(2)}\sqrt{V^S(1)})/c$

$k^* = k_1^* + k_2^* = (V\sqrt{V^S(2)}\sqrt{V^S(1)} + V^S(1))/c + \sqrt{V^S(2)} / \sqrt{V^S(1)})(V^S(1) + \sqrt{V^S(2)}\sqrt{V^S(1)})/c$

$k^* = V^S(1) + \sqrt{V^S(2)}\sqrt{V^S(1)}) + \sqrt{(V^S(2)V^S(1) + V^S(2)}$

(2.19.8) $k^* = (V^S(1) + 2\sqrt{(V^S(2)V^S(1) + V^S(2))}/c$

The derivation of the stationary point of the Lagrangian $£(k_1, k_2)$ (2.19.3) above shows the existence of a minimal sampling size to achive a given P21 (2.19.8). The plot in Figure 2.19.1 of $\mathbb{P}(Q^S_t(2) \leq Q_t^S(1))$ based on k_1 and k_2 illustrates the need for an optimal decision rule.

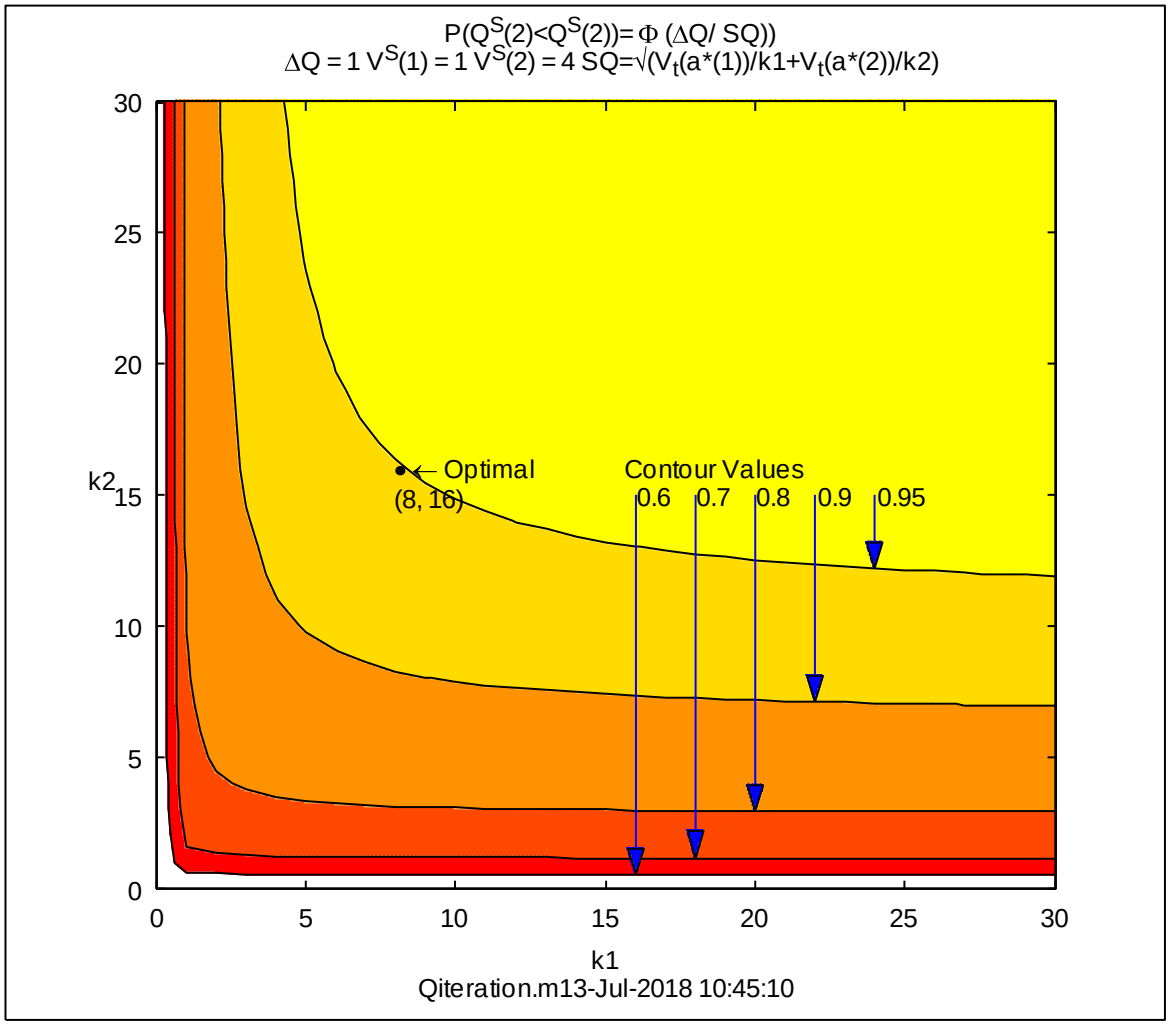


Figure 2.19.1 Plot of population P(Q2 < Q1) with sample size k1 and k2

The minimal k* = 8 +16 and is shown above at the bullet point (8, 16). It illustrates that a greater sample size of 16 is needed for action 2 with its $V^S(2) = 4.0$ than that for action 2 of size 8. It also suggests a decision rule that selects alternately a==1 and a==2 in a manner that maintains (2.19.4) will get to the nearest point on the 0.95 curve. Using a geometric argument it can be shown that k* is a minimum. The value (k_1*, k_2*) lies on the straight line k*= $k_1 + k_2$, and also on the contour $c = (V^S(1)/k_1 + V^S(2)/k_2)$. Since the straight line interests the contour once at (k_1*, k_2*) it is a tangent at (k_1*, k_2*). The line e + k*= $k_1 + k_2$, e > 0 will intersects the contour at two places violating 2.19.6,7 so e = 0 and the minimum ≤ k*. The line e + k*= $k_1 + k_2$, e < 0 does not intersect the contour at all violating 2.19.6, 7 so e=0 and the minimum ≥ k*. Hence k* is the minimum.

2.20 **Experiments to show convergence paths using sample estimates**

From an observation of (2.18.3) an investigation to show the convergence path of the counts k(t, at) follows. In order to properly assess (2.18.3) as a stopping criterion and subsequently as a selection rule the inherent randomness of using samples needs to be isolated by inserting population means and variances in the right hand side of (2.18.3):

(2.20.1) $\mathbb{P}[Q^S_t(2) \leq Q_t^S(1))] = \Phi(\Delta Q / \sqrt{(V^S(1)/k(t, d_t(1)) + V^S(2)/k(t, d_t(2))))}$, $\Delta Q = Q^*(a^*(1)) - Q^*(a^*(2))$.

where $a^*(1)$ denotes the action of maximum population mean and $a^*(2)$ the second action. A selection rule which seeks the maximisation of $\mathbb{P}(Q^S_t(2) \leq Q_t^S(1))$ can be obtained by choosing that action which has the greatest positive gradient with respect to k(t, d(1)) or k(t, d(2)).

Let $Q_2 = Q^S_t(2)$, $Q_1 = Q_t^S(1)$, $S_1 = V(d_t(1))$, $S_2 = V(d_t(2))$, $k_1 = k(t, d(1))$ $k_2 = k(t, d(2))$ and the selection rule can be written:

(2.20.2) If $\partial\mathbb{P}[Q_2 \leq Q_1] / \partial k_1 > \partial\mathbb{P}[Q_2 \leq Q_1] / \partial k_2$ then choose $d_t(1)$ otherwise $d_t(2)$

$\partial\mathbb{P}[Q_2 \leq Q_1] / \partial k_1 = \partial\Phi(y) / \partial y \, \partial(\Delta Q / \sqrt{(S_1 / k_t + S_2/k_2)} \, / \, \partial k_1$, where $\Delta Q = Q_1 - Q_2$

$= \phi(y) \, \Delta Q / (S_1/k_1 + S_2/k_2)^{3/2} \, (S_1 / k_1^2)$

$\partial\mathbb{P}[Q_2 \leq Q_1] / \partial k_2 = \phi(y) \, \Delta Q / (S_1/k_1 + S_2/k_2)^{3/2} \, (S_2 / k_2^2)$

Expanding (2.20.2)) can be rewritten as the selection rule (2.20.3):

$\phi(y) \, \Delta Q / (S_1/k_1 + S_2/k_2)^{3/2} \, (S_1 / k_1^2) > \phi(y) \, \Delta Q / (S_1/k_1 + S_2/k_2)^{3/2} \, (S_2 / k_2^2)$

$=> (S_1 / k_1^2) > (S_2 / k_1^2) => V(d_t(1))/k(t, d_t(1))^2 > V(d_t(1))/k(t, d_t(2))^2$

(2.20.3) If $V(d_t(1))/k(t, d_t(1))^2 > V(d_t(1))/k(t, d_t(2))^2$ then choose $d_t(1)$ otherwise $d_t(2)$

Clearly only UCB or PMSMMPM selection can be applied at any one step. A η-UCB selection rule is introduced which uses PMSMMPM a fraction η of the time, imitating the mixed approach of the tried and tested ϵ-greedy selection rule.

Starting with (2.19.1) expressions and code are presented for a^* based on population values:

Calculate $Q^*(a^*)$, a^*, k_1^* , k_2^* and k^*, where $k_1 = k(t, d_t(1))$ and $k_2 = k(t, d_t(2))$

$a^* = \text{DescendingSortOrder}(Q^*)$

$c \stackrel{\text{def}}{=} (\Phi^{-1}(.95) / (Q^*(a^*(1)) - Q^*(a^*(2)))^{-2}$

$k_1^* = (V(a^*(1)) + \sqrt{V(a^*(2))}\sqrt{V(a^*(1))})/c$

$k_2^* = \sqrt{(V(a^*(2)) / V(a^*(1)))} \, k_1$

$k^* = k_1^* + k_2^*$

From (2.20.3 The selection rule gives the corresponding pseudo code:

```
#Selection rule η−UCB based on PV
if  (0 == mod(t−1, η )
   if V(d_t(1))/k(t, d_t(1))^2 > V(d_t(1))/k(t, d_t(2))^2 then a_t = d_t(1) else a_t = d_t(2)
else
```

[Qm, a_t]= sort(Q $_{t-1}$.+ U_{t-1}, "descend"); #Obtain a new selection based on UBC

```
endif
```

Figure 2.20.1 below shows a contour plot of $\mathbb{P}[\ Q^S_t(2) \le Q_t^S(1)\ ]$ for $\Delta Q = Q^*(a^*(1)) - Q^*(a^*(2))$ and SQ based on population variances V(a*(i)) i=1, 2. The value of η was chosen to bring about a modest but discernible effect which could be easily compared to the pure UCB:

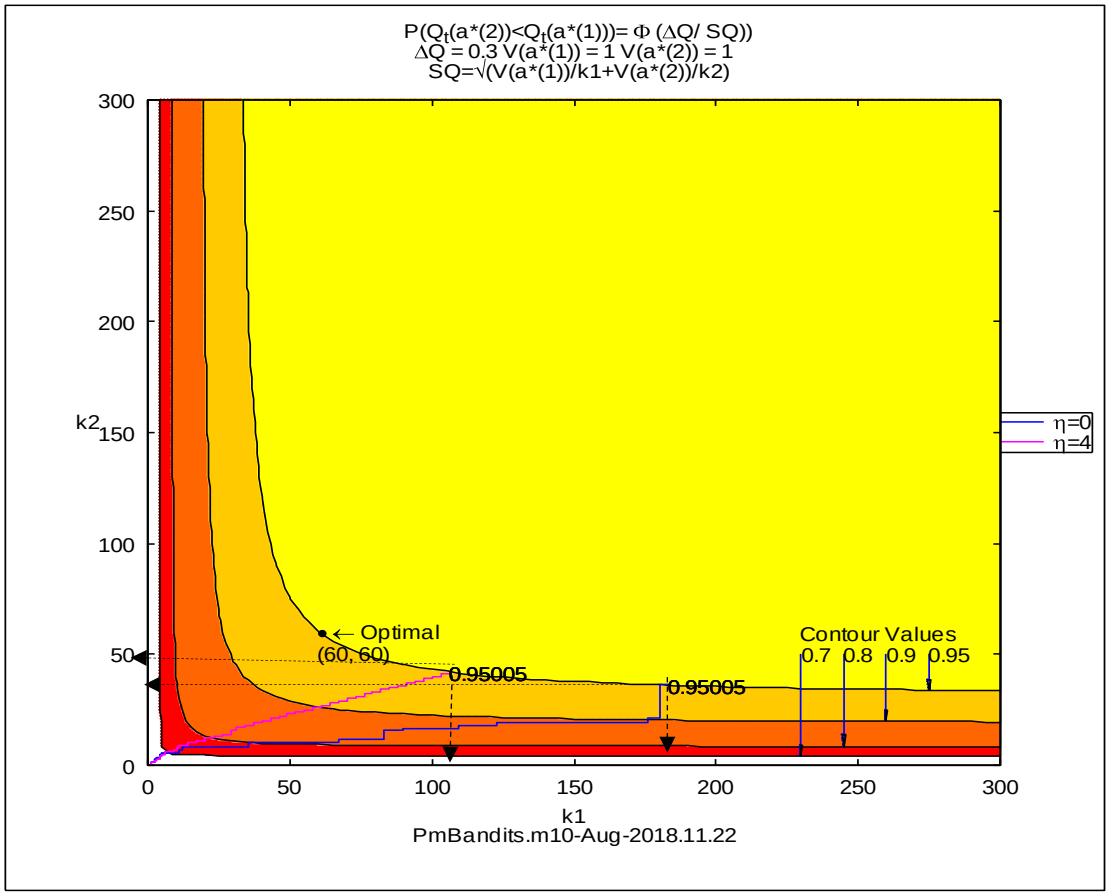


Figure 2.20.1 Contour plot of P[Q(a*(2) < Q(a*(1))] with k1 and k2

The lower stair case curve shows the k path of the pure UCB selection and shows the pursuit of action 1 with occasional action 2. Reading from the plot it reaches a $P[Q_t^S(a) \le Q_t^S(1)\ ]$ of 0.95005 at (183, 33) giving a k* of 183+33=216. The higher stair case curve depicts a 4-UCB selection rule which reaches 0.95055 at (107, 40) giving with a k* of 147. The optimal value, shown above at the "•" yields k_1*=60, k_2*=60, k* = 120. It can be concluded that the inclusion of η > 0 does decrease the estimate of k*.

Now expanding $V(d_t(1))/k(t, d_t(1))^2$ of (2.18.2) in terms of $s_{\mu t}^2(d_t(1))$ a new selection rule can be written (2.20.4)):

$S_1 / k_1^2 = s_t^2(d_t(2))/k_1/k_1 = s_{\mu t}^2(d_t(1))/k(t, d_t(1))$

(2.20.4) if $s_{\mu t}^2(d_t(1))/k(t, d_t(1)) > s_{\mu t}^2(d_t(2))/k(t, d_t(2))$ then choose $d_t(1)$ else choose $d_t(2)$

Adapting the code from the previous section:

#Selection rule η−UCB based on sample variance for sample means

```
if  (0 == mod(t−1, η )
```

if $s_{\mu t}^2(d_t(1))/k(t, d_t(1)) > s_{\mu t}^2(d_t(2))/k(t, d_t(2))$ then $a_t = d_t(1)$ else $a_t = d_t(2)$

```
else
```

$[Qm, a_t]$= sort(Q_{t-1} .+ U_{t-1}, "descend");

```
endif
```

The k paths for both the 0-UCB and 4-UCB are displayed in Figure 2.20.2 where the selection rules are calculated using sample variances (2.20.4). In addition it shows a plot of $\mathbb{P}[\ Q^S_t(2) \le Q_t^S(1)\ ]$ based on population means and variances. The estimated and actual P values are shown at the end of the paths.

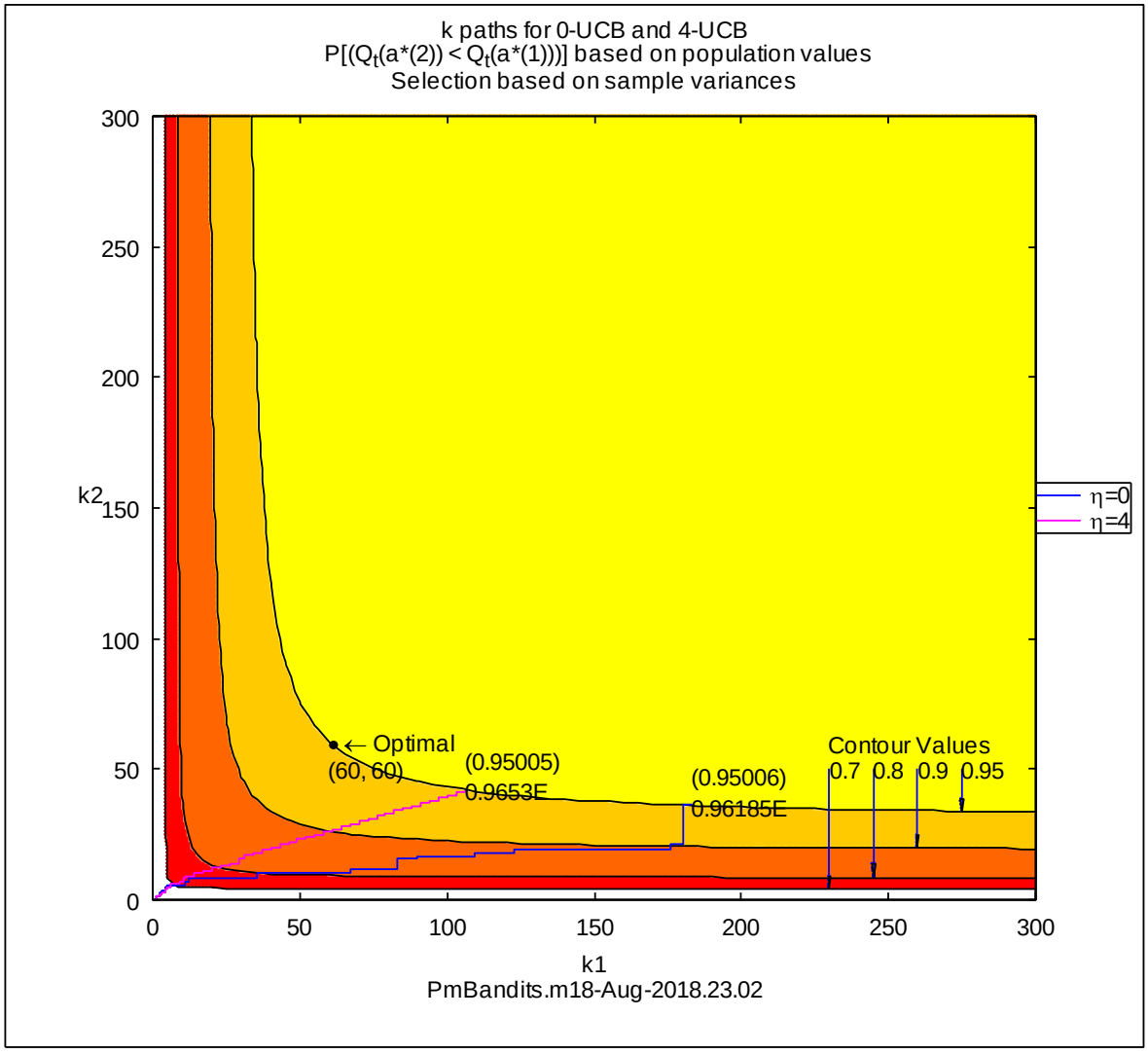


Figure 2.20.2 k paths of 0-UCB and 4-UCB

The 4−UCB rule shows the actions take a more direct route to the theoretical optimal level with a Manhattan distance of 106 + 42 = 148. The pure UCB selection rule has a distance of 183 + 36 =219 to achieve the population value for $\mathbb{P}[\ Q^S_t(2) \le Q_t^S(1)\ ]$.

The results of these experiments show one in four use of the PMSMPMP selection rule gives a sample size advantage and supports the claim that the η-UCB selection rule will contribute to faster learning algorithms suitable for run-time embedding. The speed enhancement comes from stopping convergence as early as possible.

A formal expression for PMSMMM has been derived and the properties of the simple case P21 established. The consequent properties of plays to convergence have been demonstrated. The latter provides a basis to show that a minimal sampling size to achieve a given P12 exists. In addition a selection rule has been derived and demonstrated which always nudges the sampling toward a given value for P21. This has been embedded in the η-UCB selection rule and which has been shown to contributes to faster learning algorithms. This result for the 4-UCB rests on a point estimate and to make a rigorous conclusion many experiments would be needed and average performances compared.

The bandit problem is a single state multi action problem. Once a play has been made the environment is reset to the one state. All dynamic systems of interest including dynamic NCPs, will move through a sequence of states, each having different properties. The next chapter presents and reviews the foundation of stochastic multi state systems.

.

Chapter 3 ***RL concepts and theorems***

In order to achieve objective 2.1 the basic concepts, theorems and definition of RL are set out culminating in the Bellman expectation and optimality equations which are the foundation for the RL TD algorithm.

3.1 **Agent, Environment, Goals and Reward**

A process can be in a state, $s_t \in$ S, where S is the finite set of possible discrete states. A(s_t) are the actions, a_t available from state s_t which take the process to s_{t+1}. Action selection is based on a stationary policy, denoted π(s, a), a mapping from each state, $s \in$ S, and action, $a \in$ A, to a number defined as the probability of taking action a when in state s:

(3.1.1) $\pi(s, a) \stackrel{\text{def}}{=} \mathbb{P}[a_t == a \mid s_t == s]$

In general having taken action a_0 from the initial state s_0 there is a probability that the subsequent state will be in any of s_1, s_1', s_1'' Figure 3.1.1. A *deterministic policy* has exactly one action for each state denoted π(s). Sutton implicitly requires that the state is *stationary in distribution* which implies that the joint distribution of states $s_{t1,}\, s_{t2,}.. s_{tn}$ for time points $t_1, t_{2,}.. t_n$ is the same as the joint distribution of $s_{t1+m,}\, s_{t2+m,}.. s_{tn+m}$. (Takahara and Hall, 2017).

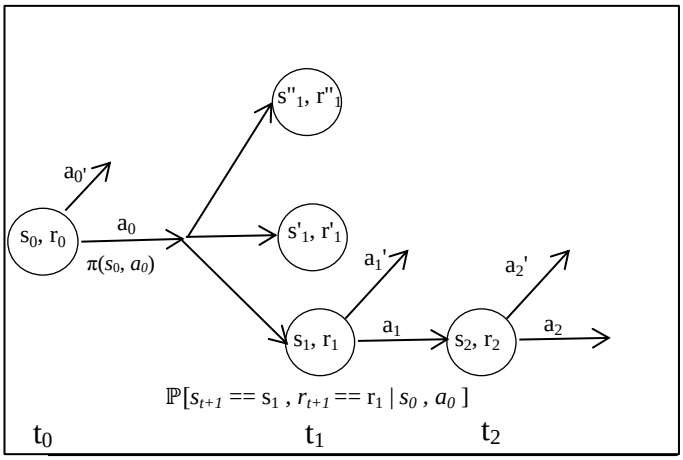


Figure 3.1.1 State space decision process.

The {S, A, and π } constitute a *state space decision process*. At the end of each time step, in part as a consequence of its action, the agent receives a signal considered to come from the environment in the form of a numerical *reward*, $r_t \in \Re$. In general r_t will be a random variable drawn from a stationary distribution. The purpose or goal of the agent is formalized in terms of the definition of the reward r_t which if positive, signals an achievement of the goal or if negative to reveals a failure.

3.2 **The Markov Property**

If the state signal has the *Markov property*, then the environment's response at *t+1* depends only on the state and action representations at *t* (Szepesvari, 2009) and the probability of $s_{t+1} == s'$ depends only on s_t:.

(3.2.1) $\mathbb{P}[s_{t+1} == s, r_{t+1} == r \mid s_t, a_t] = \mathbb{P}[s_{t+1} == s, r_{t+1} == r \mid (s_k, a_k, r_{k+1} \mid t \geq k \geq 0)], \forall s, r$

The Markov property is sometimes referred to as the independence of path property since all that matters is the current state and not the route taken to get there (Szepesvari, 2009). Together with a policy π (3.1.1) the sequence of states, actions and rewards generated by the agent environment interaction can be expressed as a *Markov sequence* $(s_t, a_t \mid t \geq 0)$ where $a_t \sim \mathbb{D}[\pi(s_t, j), j= 1..A(s_t)]$. The sequence of states embedded in a Markov sequence constitutes a *state path.* The Markov property makes the mathematics of RL tractable.

The best policy for choosing actions as a function of a Markov state is just as good as the best policy for choosing actions as a function of complete histories. A RL task that satisfies the Markov property embodies an MDP. A particular finite MDP, *M*, is defined by its state and action sets S, A, action transition probabilities P, the conditional rewards R, and the action policy π, $M = <S, A, P, R, \pi>$. In order to establish stationarity the conditional probability of the LHS of (3.2.1) is expanded in terms of the joint probability in s_{t+1} and s_t :

(3.2.2) $\mathbb{P}[s_{t+1}== s', r_{t+1} == r \mid s_t==s, a_t] = \mathbb{P}[s_{t+1}== s', r_{t+1} == r , s_t == s \mid a_t]/\mathbb{P}[s_t== s, r_{t+1} == r \mid a_t]$

Since s_t is stationary in distribution then both the denominator and numerator of the RHS are stationary corresponding to n=2 and n=1 respectively, which implies the LHS is stationary.

Given any state and action, *s* and *a*, the probability $P^a_{ss'}$ of each next state, *s'*, is:

(3.2.3) $P^a_{ss'} \stackrel{def}{=} \mathbb{P}[s_{t+1} == s' \mid s_t == s , a_t == a]$

$P^a_{ss'}$ are called *action transition probabilities*. Similarly, given any current state and action, *s* and *a*, together with any next state, *s'*, the expected value of the next reward, $R^a_{ss'}$ is:

(3.2.4) $R^a_{ss'} \stackrel{def}{=} \mathbb{E}_\pi [r_{t+1} \mid s_{t+1}=s', s_t=s, a_t=a]$

The *action transition probabilities* and the *conditional rewards* $R^a_{ss'}$ make up a model M of the MRP (Sutton and Barto, 1998).

3.2.1 Existence, uniqueness and utility of the stationary distribution

Section 2.9 states a convergence criterion for essentially a single state. For a multi state system there is a need to link each state measure of state convergence to a system measure in order to generalise PMSMMPM. As introduced earlier in gradient methods (Silver 2014) a useful property is the long run stationary distribution of states visited under a

Markov sequence using policy π. From it a stationary average of any state property can be obtained. For a stationary policy $\pi(s, a)$, $P^{\pi}_{ss'}$, called *transition probabilities*, can be defined (3.2.1.1):

(3.2.1.1) $P^{\pi}_{ss'} \stackrel{def}{=} \mathbb{P}[s_{t+1} == s' | s_t == s] = \sum_{a \in A(s)} \mathbb{P}[a_t == a | s_t == s] \mathbb{P}[s_{t+1} == s' | s_t == s, a_t == a] = \sum_{a \in A(s)} \pi(s, a) P^{a}_{ss'}$

The stationarity of π and $P^{a}_{ss'}$ together guarantee then the stationarity of $P^{\pi}_{ss'}$. The long run *stationary distribution* d of the Markov process is useful for the calculation of the average, $\overline{p}$, of any stochastic property p(s) of any state s. Such an average will provide a well founded basis for the global convergence of action stratetgies.

(3.2.1.2) $\overline{p} = \Sigma_s p(s)d(s)$

The condition for a stationary distribution *d* of visited states to exist and be unique is that the Markov sequence is *positive recurrent* (Sigman 2006, Proposition 2.2). A state s is called *positive recurrent* if the expected amount of time to return to state s, τ_{ss}, given that the sequence started in state s has finite expectation $\mathbb{E}(\tau_{ss}) < \infty$ (Sigman 2006, Proposition 2.2).

Every *irreducible* Markov sequence with a finite state space is in fact positive recurrent (Sigman (2006) . Since all the feasible systems encountered in games will have a finite states space it remains to establish irreducibility.

A Markov sequence is *irreducible* if it is possible to get to any state from any other state (Takahara and Hall, 2017). A sufficient condition is that for all (s, s') and (s', s) pairs $\exists$ a positive integer $n_{ss'}$ s.t. $(P^{\pi})^{n_{ss'}}{}_{ss'} > 0$ i.e. the (s, s') element of P^{π} raised to the power $n_{ss'}$. From 3.2.2 an exploration policy has $\pi(s, a) > 0$. To determine whether $P^{a}_{ss'} > 0$ the dynamics of the system need to be systematically investigated to detect islands of possible closed off behaviour in the state variable space. It is likely that games NPC designers would ensure an irreducible state variable space for continuity of game behaviour.

Takahara and Hall (2017) show that d can be determined from P^{π}, $d = dP^{\pi}$. An unbiased estimate of d is the proportion of the occurrence of $s_t == s$. So the proportion of visits to states over all epochs will be an estimate of d. Sigman (2006) shows that the population value is the limit of the sample estimate as $n \rightarrow \infty$ w. p. 1. Provided the state space is irreducible and finite an unbiased estimate of the stationary distribution can be obtained from the proportion of visits to states over all epochs.

3.3 **Returns**

RL seeks to maximize the *return*, R_t, itself a random variable, defined as the geometric weighed sum of the r_{t+1} in the sequence (s_t, a_t, r_{t+1}) $(s_{t+1}, a_{t+1}, r_{t+2})\ldots$ as a function of a discount rate γ.

(3.3.1) $R_t \stackrel{def}{=} \sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$, $0 \leq \gamma \leq 1$

If $\gamma = 0$ the agent is "myopic" being concerned only with maximizing the immediate reward. If $0 < \gamma < 1$ the infinite sum has a finite value as long as the reward sequence $(r_{t+1+k} | k > 0)$, is bounded. As $\lim \gamma \rightarrow 1$, R_t takes future rewards more strongly into account and eventually becomes the sum of all the rewards. The return (3.3.1) is appropriate for episodic tasks if the sequence finishes at the terminating state s_T, where T is the final step, and for which $r_{T+1} = 0$. Normally the terminating state has no successors and corresponds to a naturel conclusion of the *trajectory* in the task domain. The return in this case is:

(3.3.1) $R_t \stackrel{def}{=} \sum_{k=0}^{T} r_{t+1+k}$

The return is simple but useful way of modelling the aggregate reward over the state path and as formulated below yields graceful expressions for state values MRP (Sutton and Barto, 1998)..

3.4 **Value Functions**

Almost all RL algorithms are based on estimating cumulative rewards known as *value functions*, functions that estimate the expected return for states or state-action pairs. The *state value* of a state *s* under a policy π, denoted $V^{\pi}(s)$, is the expected return when starting in *s* and following π thereafter. For MRPs, define $V^{\pi}(s)$, as:

(3.4.1) $V^{\pi}(s) \stackrel{def}{=} \mathbb{E}_{\pi}[R_t | s_t=s] = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t=s]$

where $\mathbb{E}_{\pi}[]$ denotes the expected value of the R_t over all the $(r_{t+1+k} | k > 0)$, given that the agent follows policy π, and t is any time step. Note that the value of the terminal state, if any, is always zero. The function V^{π} denotes the state-value function for policy π.

Similarly, define the *action value* for policy π, denoted Q^{π}, as the expected return starting from *s*, taking the action *a*, and thereafter following policy π:

(3.4.2) $Q^{\pi}(s, a) \stackrel{def}{=} \mathbb{E}_{\pi}[R_t | s_t=s, a_t=a] = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t=s, a_t=a]$

Estimation of V^{π} and Q^{π} can be done by either Monte Carlo methods or an iterative process (Sutton and Barto, 1998).

3.5 **Bellman Expectation Equation for evaluation of Q^{π}**

A relationship for Q in terms of V is given by Bellman (1957):

$Q^{\pi}(s, a) \stackrel{def}{=} \mathbb{E}_{\pi}[R_t \mid s_t=s, a_t==a]$ from (3.4.2)

$= \mathbb{E}_{\pi}[r_{t+1} + \gamma R_{t+1} \mid s_t==s, a_t ==a]$

Introduce a partition over each successor state s' each with partial probabilities $P^a_{ss'}$

$= \mathbb{E}_{\pi}[r_{t+1} \mid s_t==s, a_t ==a] + \gamma \sum_{s'} P^a_{ss'} \mathbb{E}_{\pi}[R_{t+1} \mid s_t==s, a_t ==a, s_{t+1}==s']$

Generate the expectation over s' to produce an inner random variable $\mathbb{E}_{\pi}[R_{t+1} \mid s_{t+1}==s']$

$= \mathbb{E}_{\pi}[r_{t+1} \mid s_t==s, a_t ==a] + \gamma \sum_{s'} P^a_{ss'} \mathbb{E}_{\pi}[\mathbb{E}_{\pi}[R_{t+1} \mid s_{t+1}==s'] \mid s_t==s, a_t ==a]$

Since $\mathbb{E}_{\pi}[R_{t+1} \mid s_{t+1}==s']$ only depends on s' and not on s_t, a_t, replacing with V(s')

$= \mathbb{E}_{\pi}[r_{t+1} \mid s_t==s, a_t ==a] + \gamma \sum_{s'} P^a_{ss'} \mathbb{E}_{\pi}[V(s') \mid s_t==s, a_t ==a]$

Since s_{t+1} is s' with probability $P^a_{ss'}$ given s and a, writing $\sum_{s'} P^a_{ss'} V(s') = V^{\pi}(s_{t+1})$

$= \mathbb{E}_{\pi}[r_{t+1} \mid s_t==s, a_t ==a] + \gamma \mathbb{E}_{\pi}[V^{\pi}(s_{t+1}) \mid s_t==s, a_t ==a]$

(3.5.1) $Q^{\pi}(s, a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t==s, a_t ==a]$

Similarly a recursive relationship exists for Q Bellman (1957):

$Q^{\pi}(s, a) \stackrel{def}{=} \mathbb{E}_{\pi}[R_t \mid s_t==s, a_t==a]$ from (3.4.2)

$= \mathbb{E}_{\pi}[r_{t+1} + \gamma R_{t+1} \mid s_t==s, a_t==a]$

Introduce a partition over each *s'* and over *a'* for the second term:

$= \mathbb{E}_{\pi}[r_{t+1} \mid s_t==s, a_t ==a] + \sum_{s'} P^a_{ss'} \sum_{a'} \pi(s',a') \mathbb{E}_{\pi}[\gamma R_{t+1} \mid s_t==s, a_t==a, s_{t+1}==s', a_{t+1}==a']$

Generate the expectation over s', *a'* to produce $\mathbb{E}_{\pi}[R_{t+1} \mid s_{t+1}==s', a_{t+1}==a']$

$= \mathbb{E}_{\pi}[r_{t+1} \mid s_t==s, a_t ==a] + \sum_{s'} P^a_{ss'} \sum_{a'} \pi(s',a') \mathbb{E}_{\pi}[\gamma \mathbb{E}_{\pi}[R_{t+1} \mid s_{t+1}==s', a_{t+1}==a'] \mid s_t==s, a_t==a,]$

Since $\mathbb{E}_{\pi}[R_{t+1} \mid s_{t+1}==s', a_{t+1}==a']$ only depends on s' ,*a'* replacing with $Q^{\pi}(s_{t+1}, a_{t+1})$

$= \mathbb{E}_{\pi}[r_{t+1} \mid s_t==s, a_t ==a] + \sum_{s'} P^a_{ss'} \sum_{a'} \pi(s',a') \mathbb{E}_{\pi}[\gamma Q^{\pi}(s', a') \mid s_t==s, a_t==a,]$

The sum of expectations is the expectation of the sum:

$= \mathbb{E}_{\pi}[r_{t+1} \mid s_t==s, a_t ==a] + \gamma \mathbb{E}_{\pi}[\sum_{s'} P^a_{ss'} \sum_{a'} \pi(s',a') Q^{\pi}(s', a') \mid s_t==s, a_t==a,]$

Revoking the partition over s', a' a recursive relationship for Q (3.6.2) results:

$= \mathbb{E}_{\pi}[r_{t+1} \mid s_t==s, a_t ==a] + \gamma \mathbb{E}_{\pi}[Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t==s, a_t==a,]$

(3.5.2) $Q^{\pi}(s, a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t==s, a_t==a]$

This yields the Bellman Expectation Equation for Q^{π} (3.5.2) which is the basis of the TD update equation which is used in chapter 4 for the TD update equation. It is also straightforward to express $V^{\pi}(s)$ in terms of π and Q using (3.1.1) and (3.4.2)

(3.5.3) $V^{\pi}(s) = \sum_a \mathbb{P}[a_t == a \,|s_t == s]\mathbb{E}_{\pi}[R_t|\, s_t=s, a_t=a] = \sum_a \pi(s, a)\, Q^{\pi}(s, a)$

Bellman shows how (3.5.2) can be written in terms of $Q^{\pi}(s', a')$ and the model parameters $P^a_{ss'}$, $R^a_{ss'}$:

(3.5.3) $Q^{\pi}(s, a) = \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma \sum_{a'} \pi(s', a') Q^{\pi}(s', a')]$

The fixed point will be Q^{π}, Barto (1988) states that convergence is guaranteed under the same conditions that guarantee the existence of Q^{π}. This result provides a route to the solution of action values if the model parameters $P^a_{ss'}$ and $R^a_{ss'}$ are known, otherwise (3.5.2) is used to provide an update term in the incremental TD(0) algorithm used in chapter 4.

3.6 **Optimal Value Functions**

Solving a RL task means finding a policy that achieves the maximum reward over the long run. Since A(s) is finite ∀s there is a finite set of policies and there is always at least one policy that is better than or equal to all other policies and all the optimal policies are denoted by π^*. They share the same action value, called the *optimal action value*, denoted Q*, defined as:

(3.6.1) $Q^*(s, a) \stackrel{\text{def}}{=} \max_{\pi} Q^{\pi}(s, a)$, ∀s ∈ S and $a \in A$

Sutton and Barto (1998) use the intuition that the *Bellman Optimality Equation* expresses the fact that the optimal action value at *(s, a)* under π*, Q*(s, *a*) is related to the expectation conditional on $s_t=s$, $a_t=a$ of the reward r_{t+1} and the discounted maximum over the actions a' of Q*(s', *a'*) at the successor state s':

(3.6.2) $Q^*(s, a) = \mathbb{E}\,[r_{t+1} + \gamma \max_{a' \in A(s)} Q^*(s', a') | s_t = s, a_t = a]$ from (3.5.1)

or if the model is known:

(3.6.3) $Q^*(s, a) = \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma \max_{a'} Q^*(s', a')]$

The discounted maximum term can also be used to define V^* the optimal state value:

(3.6.4) $V^*(s) \stackrel{\text{def}}{=} \max_{a \in A(s)} Q^*(s, a)$

(3.6.5) $V^*(s) \stackrel{\text{def}}{=} \max_{a \in A(s)} \mathbb{E}\,[\, r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t=s, a_t=a\,]$, using (3.5.1).

3.7 **Policy improvement theorem**

In our context the model is not available and iterative methods are need to solve (3.6.3) for the optimal policy. The optimal stochastic policy is directly obtained from the action of maximum Q*. The definition (3.7.1) checks for up to n Q each satisfying the maximum and computes the least biased probability of (1/n) for π*(s, *a*).

(3.7.1) $\pi^*(s, a) \stackrel{def}{=} \{1/n \text{ if } \exists\ a_{1,} a_{2..} a_n \wedge a \in \{a_{1,} a_{2..} a_n\} \wedge Q^*(s, a_i) == \max_{a' \in A} Q^*(s, a')$;
$\{0$ otherwise.

In other words if there is a tie between two or more a values a_1, a_2,.. then $\pi^*(s, a_1) + \pi^*(s, a_2) +.. = 1$ will be optimal. If there is no tie then a strictly deterministic policy $\pi^*(s)$ can be derived:

(3.7.2) $\pi^*(s) \stackrel{def}{=} \text{argmax}_{a \in A} Q^*(s, a)$

Any deterministic policy $\pi(s)$ has a policy matrix $\pi(s, a)$ for which each state s has an action a of value 1 and the rest of the elements are zeros and are related:

(3.7.3) $\pi(s) = \sum_a a.\pi(s, a)$

(3.7.4) $\pi(s, a) = \delta_{\pi(s), a}$, where δ_{ab} is the discrete delta function

The action value for a discrete policy is related to its state value, V(s), and is a function of s only:

(3.7.5) $V(s) = \sum_a \pi(s, a)\, Q^\pi(s, a) = \sum_a \delta_{\pi(s), a} Q^\pi(s, a) = Q^\pi(s, \pi(s))$

A deterministic policy π' is defined to be better than or equal to a policy π if its expected return is greater than or equal to that of π for all states. Sutton and Barto (1998) propose a criterion to decide whether a policy change from π to π' will lead to an improved policy change for all states:

(3.7.6) $\pi' \geq \pi$ if and only if $Q^{\pi'}(s, \pi'(s)) \geq Q^\pi(s, \pi(s))$, $\forall s \in S$.

The RHS of 3.7.6 says that choosing $\pi'(s)$ at step one and following π' thereafter yields a higher action value and hence higher expected return than choosing $\pi(s)$ at step one and following π thereafter. The Policy Improvement Theorem of Sutton and Barto (1998) is as follows: Let a new policy π' with $\pi'(s) = a \,\#\, \pi(s)$ and assume its better in the sense that :

(3.7.7) $Q^\pi(s, \pi'(s)) \geq Q^\pi(s, \pi(s))$, $\forall s \in S$

Sutton and Barto (1998) show that (3.7.7) implies $Q^{\pi'}(s, \pi'(s)) \geq Q^\pi(s, \pi(s))$ $\forall s \in S$ and hence $\pi' \geq \pi$ by 3.7.6) as follows:

$Q^\pi(s, \pi(s)) \leq Q^\pi(s, \pi'(s))$ using 3.7.7:

$= \mathbb{E}_{\pi'}[r_{t+1} + \gamma\, Q^\pi(s_{t+1}, \pi(s_{t+1})) \mid s_t = s]$, using (3.5.2), $Q^\pi(s, \pi(s)) = V^\pi(s)$

$\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma\, Q^\pi(s_{t+1}, \pi'(s_{t+1})) \mid s_t = s]$, using (3.7.7), and that expectation is monotonic

$= \mathbb{E}_{\pi'}[r_{t+1} + \gamma\, \mathbb{E}_{\pi'}[\, r_{t+2} + \gamma Q^\pi(s_{t+1}, \pi(s_{t+1}))] \mid s_t = s]$, using (3.5.2) $\pi'(s)$ at t+1 and $\pi(s_{t+1})$ at t+2

$= \mathbb{E}_{\pi'}[r_{t+1} + \gamma\, r_{t+2} + \gamma^2 Q^\pi(s_{t+1}, \pi(s_{t+1})) \mid s_t = s\ ]$, using $\mathbb{E}[X\ \mathbb{E}[Y\ ]\ |c] = \mathbb{E}[X, Y\ |c]$ where X,Y independent

$\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma\, r_{t+2} + \gamma^2 Q^{\pi}(s_{t+1}, \pi'(s_{t+1})) \mid s_t=s\ ]$, using (3.7.7),

(3.7.8) $Q^{\pi}(s, \pi(s)) \leq \mathbb{E}_{\pi'}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t=s\ ] = V^{\pi'}(s) = Q^{\pi'}(s, \pi'(s))$

So by $Q^{\pi}(s, \pi'(s)) \geq Q^{\pi}(s, \pi(s)) \Rightarrow Q^{\pi'}(s, \pi'(s)) \geq Q^{\pi}(s, \pi(s))$ hence $\pi' \geq \pi$. Any new policy where an improved action can be found for a single state will be improved for all states. The Policy Improvement Theorem provides the basis for policy update of the TD(0) algorithm of chapter 4.

3.8 **Calculating the optimal policy V* from the exact Q^{π}**

The *greedy policy* $\pi'(s) = \text{argmax}_a\, Q^{\pi}(s, a)$ will recommend action changes for all states. From (3.5.1):

(3.8.1) $\pi'(s) = \text{argmax}_a\, Q^{\pi}(s, a) = \text{argmax}_a\, \mathbb{E}[\, r_{t+1} + \gamma\, V^{\pi}(s_{t+1}) \mid s_t == s, a_t == a]$

By construction π' will be better than or as good as π for s so satisfies the Policy improvement theorem. If π' is as good as and not better than π the $V^{\pi} = V^{\pi'}$ From (3.8.1) there is a $V^{\pi'}$ that satisfies:

(3.8.2) $V^{\pi'}(s) = \max_{a \in A(s)} \mathbb{E}_{\pi'}[\, r_{t+1} + \gamma\, V^{\pi'}(s_{t+1}) \mid s_t=s, a_t=a\ ]$

Which is the same as the Bellman Optimality Equation for V*(s) so $V^{\pi'}(s) = V^*(s)$. If we have the correct action value the optimal policy is directly obtained.

3.9 **The importance of exploratory policies**

In practice only estimates of V^{π} in terms of Q^{π} are available and to ensure each *(s, a)* is visited an infinite number of times a policy with a non-zero probability of selecting any action at each state is needed. Such a *complete exploratory policies* has $\pi(s, a) > 0\ \forall s, a$ in contrast to strict policy which has unique action for each $\pi(s)$. The n-armed bandit exercises shows how a strict policy based on estimated Q will always find a sub-optimal Q* whereas an ϵ-greedy shows successively better long run results as $\epsilon \rightarrow 0, \epsilon > 0$, Figure 2.7.1.

(3.9.1) $\pi(s, a) = 1 - \epsilon + \epsilon/|A|$ if $a == argMaxWithRandomTiebreaks_{a'}\, (Q_t(s, a'))$

$= \epsilon/|A|$ if $a \neq argMaxWithRandomTiebreaks_{a'}\, (Q_t(s, a'))$

The reworking below of Sutton and Barto (1998) starts with the implicit definition of a deterministic policy π' based on a ϵ-greedy policy $\pi'(s, a)$ identical to (3.9.1) from which to establish the condition for the policy improvement section (3.7).

(3.9.2) $Q^{\pi}(s, \pi'(s)) \stackrel{\text{def}}{=} \sum_a \pi'(s, a) Q^{\pi}(s, a)$

$= (\epsilon/|A|) \sum_a Q^{\pi}(s, a) + (1-\epsilon)\, \max_a Q^{\pi}(s, a)$

Construct weights $W_a = (\pi(s, a) - \epsilon/|A|)/(1 - \epsilon)$, for $a=1..A$, $\Sigma_a W_a = 1$, then $\max_a Q^\pi(s, a) \geq \sum_a W_a Q^\pi(s, a)$

$$Q^\pi(s, \pi'(s)) \geq (\epsilon/|A|) \sum_a Q^\pi(s, a) + (1-\epsilon) \sum_a (\pi(s, a) - \epsilon/|A|)/(1 - \epsilon) Q^\pi(s, a)$$

$$= (\epsilon/|A|) \sum_a Q^\pi(s, a) - (\epsilon/|A|) \sum_a Q^\pi(s, a) + \sum_a \pi(s, a) Q^\pi(s, a)$$

$$= Q^\pi(s, \pi(s)) \text{ by 3.7.7}$$

By (3.7.6), the policy improvement theorem:

$Q^\pi(s, \pi'(s)) \geq Q^\pi(s, \pi(s)) \Rightarrow Q^{\pi'}(s, \pi'(s)) \geq Q^\pi(s, \pi(s))$ so $\pi' > \pi$

which shows that the new policy is an improvement for all states. The greedy policy entails no exploration and will eagerly home in on the first feasible solution. The chapter shows the detailed working behind the TD update equation used in Ch 4.0. The policy improvement theorem is explained and it is shown that a complete exploratory policy will select every state action pair and hence will converge almost surely to the optimum greedy policy by the policy improvement theorem and Robins Monro conditions.

Chapter 4 *Assessment and evolution of TD methods*

The scope of this chapter includes applying RL to a new problem in order to explore the effectiveness and limitations in the game context. The chapter addresses the task of objective 2.1 which entails applying a tried and tested RL learning algorithm to the determination of a steering policy for the reversing caravan.

The TD algorithms encompass the basic task of evaluating V^{π} where the sequence follows policy π, and from that to obtain V* when using an exploratory policy. The point of the work in this section is to succinctly bring together the key theorems which underpin the TD algorithm in a common format and notation.

Bellman equations state the core iterative TD algorithm and its convergence properties based on the Robins Monro conditions for estimation of $V^{\pi}$and Q^{π}. The structure of the TD algorithm, its complexity, and statistical convergence are subsequently presented. The online estimation of TD state/action values used above contrasts with the well founded offline or batch updating approach which provide a useful benchmark of the best estimates that can be obtained with a finite dataset.

Finally in the same format and notation the TD extension to Q estimation is presented known as *sarsa(0)* mimicking the quintuple (state , action, result, state, action) and the (0) indicates no eligibility traces.

4.1 **Foundations of the TD family of RL algorithms**

TD learning is a landmark idea in RL and uses framework of Monte Carlo and DP. Like DP, TD methods update estimates based in part on other learned estimates, without waiting for a final outcome, historically known as bootstrapping. Unlike DP only the successor states are used. From section 3.5, the Bellman Expectation Equation provides the founding definition for V^{π} and Q^{π} from (3.5.2)

$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[\, r_{t+1} + \gamma Q^{\pi}(s', a') \mid s_t == s, a_t == a]$, s', a' are successor state and action.

Given values of $r_{t+1} + \gamma Q^{\pi}(s', a')$ and $Q^{\pi}(s, a)$ for a Markov reward sequence generated by π an improved estimate $Q'^{\pi}(s, a)$ is obtained using the stochastic approximation equations:

(4.1.1) $Q'^{\pi}(s, a) = (1- \alpha)Q^{\pi}(s, a) + \alpha[r_{t+1} + \gamma\, Q^{\pi}(s', a')\,]$

$$Q'^{\pi}(s, a) = (1- \alpha)Q^{\pi}(s, a) + \alpha[r_{t+1} + \gamma\, Q^{\pi}(s', a')\,]$$

$$= Q^{\pi}(s, a) - \alpha Q^{\pi}(s, a) + \alpha[r_{t+1} + \gamma\, Q^{\pi}(s', a')\,]$$

$$= Q^{\pi}(s, a) + \alpha[r_{t+1} - Q^{\pi}(s, a) + \gamma\, Q^{\pi}(s', a')\,]$$

generalising the update from the incremental calculation of the mean of an RV using a learning rate parameter.

The Robbins-Monro conditions (2.7.1) (Robbins & Monro 1951) guarantee convergence with probability 1. Proof that the limit is $V^{\pi}(s)$ ∀ s and $Q^{\pi}(s, a)$ ∀ s ∀ *a* is provided by (Szepesvari 2009) and requires use of the Bellman operator. The TD algorithm below synchronises *policy evaluation* and *policy improvement*. Policy evaluation improves the Q value given a particular policy π using the TD update (4.1.1) which is underpinned by the RM conditions. Policy improvement uses the current Q values to obtain a new greedy policy based on (3.7.2) which will be equal or better by the Policy improvement theorem of 3.7. Sutton asserts that although policy evaluation and policy improvement mutually interfere they will converge to an optimal policy which is consistent with its action value function.

4.2 **TD Sarsa(0) estimating Q***

TD sarsa(0) is essentially a extension of TD to the estimation of Q(s, *a*). The implementation of TD sarsa(0) using a soft *behaviour policy* π is shown below:

Parameters

α State value learning parameter

γ Return discount parameter

ε Probability of a random action

Initialise *Q(s, a)* arbitrarily, Qn(s,a) to zero

Repeat for each episode *e*

Initialise s_0, a_0

Repeat for each step *t* of *e*

Take action *a* yielding reward *r* and next state *s′*

Choose *a′* from *s′* using a ε-soft behaviour policy based on Q # Policy evaluation

δ = [r + γQ(s', *a'*) – Q(s, *a*)]

Q(s, *a*) = Q(s, *a*) + α δ # Q Policy improvement

++Qn(s, a) #Increment count of visits to (s,a)

s = s'

a = *a*'

until s terminal

until last episode

Convergence and performance are similar to TD sarsa(0) and memory complexity is *O(|S| x |A|)*. TD algorithms have a graceful link to Monte Carlo methods. CEWIn Monte Carlo learning the complete episode is generated and the state value V(s) is the average of the undiscounted returns of the first visit to s over all the episodes, Rubinstein and Kroese (2017). On the other hand TD learning uses a one-step of the episode and an provisional estimate of the state value to yield an updated estimate.

4.3 **TD sarsa(0) learning and validation of the reversing caravan problem**

Pursuing objective 2.1 an incremental simulation model of a car and van is needed which replicates the effect of steering control on the configuration. It is understood that the game would need to have such a model anyway for the correct depiction of the asset's physics. A model for the dynamics of a car towing a caravan has been researched, developed and implemented, and RL applied to the caravan reversing problem. *Trajectories* which are pathways along the plane of the car and caravan for a given steering policy are generated and input into a TD sarsa(0) learning algorithm. A steering policy that generates a steady-state trajectory in the sense that it does not jack-knife is to be learnt using TD sarsa(0). The caravan reversing problem is a non trivial control task and as far as is known has been untried with RL. CEWOther known solutions include a control approach has been made by Jayakaran (2004). Using his work a formal solution is offered in Appendix 2.

Reversing a caravan or trailer with a car attached by a tow bar is a problem having meta-stable dynamics and compounded by the need to look rearwards. It involves judicious overshooting to orientate the caravan to the correct angle. Intuitively the steady state solution is when the lines though all the wheel axles meet at the same point known as the instance centre about which both the car and van trace out circle arcs. Jayakaran (2004) uses the geometry of the *steady state solution* to show that the car-van angle is a function of sterring angle for any in [-90 90].

Figure 4.3.1 shows the geometric model of the car and caravan ensemble co-aligned and overlaid by the configuration after one time step. The car centre line is FA with the tow bar at B. The van centreline BCD is at angle Ψ to the car. T is the instance centre of the turning circle of the car. The front wheels are at angle ϕ to the centreline.

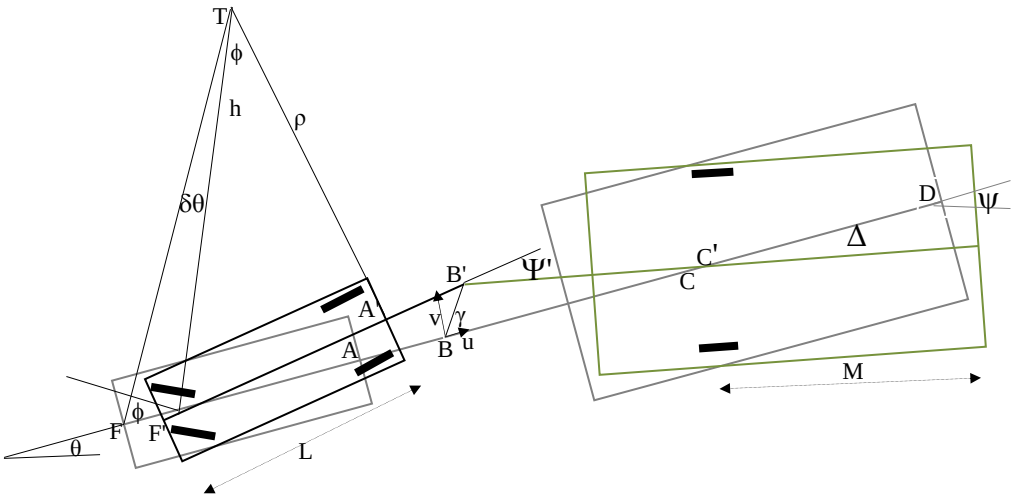


Figure 4.3.1 Car and caravan courtesy of LaValle, S. (2006)

The car and caravan system in 2D comprise of two displacement and one orientation degree of freedom each. These six degrees can also have rates of change making 12 degrees in all. The join of the tow bar makes two displacement and two time derivatives constraints leaving 8 remaining. If we regard the ensemble as a single object this will have six degrees of freedom leaving 2 internal which are the angle between the car and van, Ψ and its derivative. Jayakaran (2004) has offered a dynamic systems approach to the equation of motion for Ψ and Appendix 3 confirms in detail his result is equivalent to the geometric approach used below.

For simplicity and tractability the speed and acceleration are assumed fixed at this stage. In addition an infinite plane is available so the location and orientation of the whole ensemble is not needed. The *state space* for the problem is the tuple of all the state variables, in this case the pair of continuous state variables (ϕ,Ψ). The only action variable available is a change in the steering angle $\dot{\phi}$, which is constrained by the range for ϕ consequent on the vehicle steering lock. The jack-knife angle for Ψ is set to a nominal 7^{o} but a proper consideration of the jack-knife criterion for an actual car and caravan would be needed for an exact value.

Move then turn dynamics are deployed where the incremental movement is to model a move along a straight line as allowed by the wheels followed by a turn to a new orientation. The move will be along an arc of a circle. After each move the steering is nudged by the current best action or an exploratory random action. For the move dynamics over the time interval Δt the change in the car angle θ is given by (4.3.1) LaValle (2006):

(4.3.1) $\Delta\theta \equiv (q\sin\phi / L)\,\Delta t$

Using well conditioned expressions the displacement of point A is:

(4.3.2) $\underline{\Delta A}$ = [qcosϕ Δt cos(θ(t) +Δθ/2) {1 – $(\Delta\theta/2)^2/3!$ …}; qcosϕ Δt sin(θ(t) + Δθ/2) {1 – $(\Delta\theta/2)^2/3!$…}]

The change in van angle using displacement v is:

(4.3.3)$\Delta\psi = \sin^{-1}(v/M)$

where v is a complex expression in the displacement of the tow bar BB'. Appendix 2 presents the detail on the derivation of 4.3.3 and presents a transformation to the equation of motion for Ψ.

The change in car-van angle is:

(4.3.4) $\Delta\Psi = \Delta\psi - \Delta\theta$

Appendix 2 presents a transformation of the equation of motion for Ψ into a standard first order differential equation with known boundary conditions. The Wolfram|Alpha deductive engine is used to find a formal symbolic solution. The work in this thesis uses an incremental numerical approach to the solution for the equation of motion which provides a graceful input to the TD sarsa(0) learning algorithm.

For the turn dynamics the steering ϕ is updated by Δϕ which is based on the ϵ-greedy selection rule. A positive Δϕ is an anti-clock wise turn. A negative reward on failure is the most common scenario in the Weatherwax (2005) studies and in the cart and pole system of Barto etal (1983). A reward which will learns a policy which continuously steers away from a jack-knife over a set number of steps is defined as a stable policy. Accordingly the TD sarsa(0) reward r is set to zero for each step until jack-knife occurs when it is -1.

State aggregation divides the state space of a continuous variables into sub-regions each denoted by a state variable index. CEWBarto, A. G. and Sutton, R. S., Anderson, C. W., (1983) use the state aggregation scheme of Michie and Chambers (1968a, 1968b) 'boxes' model for the classic balancing the pole on a cart problem and uses a non-linear state aggregation which comprise of 162 regions covering a four dimensional space for the displacement, x and pole angle θ and their derivatives. Doya (2000) uses a 30 x 30 grid discretization of the state to achieve results comparable with Function Approximation. A detailed discussion of the interrelation of the granularity of the state aggregation and the probability of a state change is given in section 5.3 below.

A state aggregation scheme of 6 sub-regions for ϕ and 8 sub-regions for Ψ is used. Similarly for the single action an aggregation of 4 sub-regions for the steering nudge Δϕ is used. This choice gives two levels of steering nudge each way and results only in a total

state action space of 6 x 8 x 4 = 192. The exploration parameter $\epsilon = 0.2$ is based on a marginally more generous value than the best value for the bandit exercises (Figure 2.7.1) and the discount parameter set at $\gamma = 0.9$ based on the TD random walk of Weatherwax (2005). The value of Δt was determined as a compromise between simulation accuracy, which favours a smaller value and trajectory run length, which favours a larger value sufficiently long enough for trajectories to encounter near jack-knife conditions.

Table 4.3.1 shows the counts for visits to each state action pair (Qn, see 4.2) and reveals two orders of magnitude between visits to the central state-action elements and extreme elements of the state space. For example pair (3, 4) is visited 3228 times but (1, 2) only 28 times, see underlined values in Table 4.3.1. Clearly the learning process produces a policy which avoids actions leading to extreme states.

ϕ									Ψ								
below	$-\infty$		-7		-3		-1		0		1		3		7		∞
-15		1		2		3		4		5		6		7		8	
	1	0		28		10		176		87		136		108		10	
-6																	
	2	0		48		64		1472		591		366		118		2	
-2																	
	3	2		24		60		3228		227		303		108		4	
0																	
	4	0		44		81		3223		227		303		108		4	
2																	
	5	4		10		59		1684		193		914		245		0	
6																	
	6	1		11		14		247		308		485		156		0	

Table 4.3.1 Counts of visits to states (left column are ϕ bondaries)

The output deterministic policy, $\pi'(s)$, given by (3.8.1) yields an integer which is an index of the set of actions A(s). The change in steering $\Delta\phi$ is obtained from the aggregation vector [-4 -1 1 4] using the index. Table 4.3.2 shows the output deterministic policy which reveals the appropriate symmetry about the main diagonal

ϕ									Ψ								
below	-∞		-7		-3		-1		0		1		3		7		∞
-15		1		2		3		4		5		6		7		8	
	1	-4		-1		-1		4		4		4		4		-4	
-6																	
	2	-4		-4		-4		1		4		4		4		-4	
-2																	
	3	-4		-4		-4		1		1		4		4		-4	
0																	
	4	-4		-4		-4		-1		-1		4		4		-4	
2																	
	5	-4		-4		-4		-4		-1		1		4		-4	
6																	
	6	-4		-4		-4		-4		-4		-1		4		-4	

Table 4.3.2 Deterministic policy

For example for negative ϕ with state variable index 1 and positive Ψ with index 6, Δϕ shows that the car is turned by 4 to ϕ + 4 which is less negative and will reduce the increase in Ψ, see underlined values in Table 4.3.2. Similarly in the opposite quadrant the case of positive ϕ, and negative Ψ, a negative Δϕ, (-4) will mitigate the decrease in Ψ. With the exception of extreme Ψ the policy π' shows symmetry about the main diagonal which confirms the sense of Figure 4.3.1. Since exploration will introduce a stochastic element to the reward, 15 epochs each of 100 episodes and each episode potentially up to 150 steps have been carried out. These values required 8 minutes of run time on an Intel i5 @3.3Ghz and were considered adequate for the investigation.

Figure 4.3.2 below shows the average trajectory length following a clearly rising trend for Robbins & Monro compliant α of 1/Qn(s, a). It suggests a non-failing reversing trajectory is learnt by episode 40 from when most trajectories reach 150 steps, the maximum specified. At the early learning stages jack-knife occurs after about 25 steps when steering is more or less random.

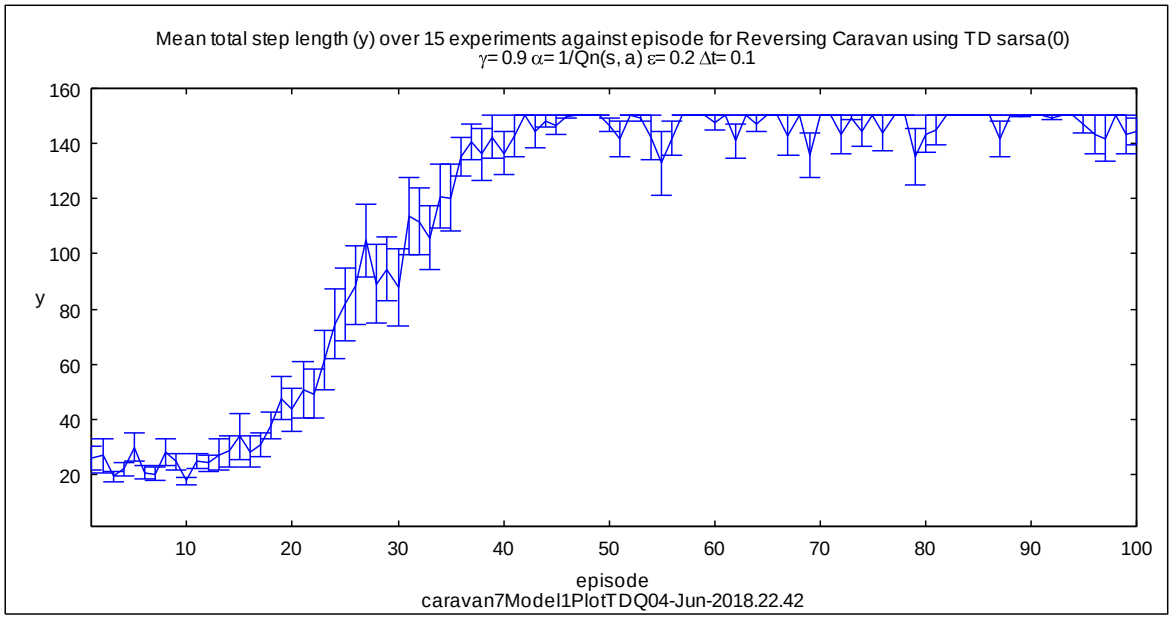


Figure 4.3.2 Average trajectory length with episode

Figure 4.3.3 shows the steady decline of the number of state action pairs with zero Q values. Despite all states are visited some Q values for some epochs are still zero. An explanation is they are visited early due to exploration and receive only the zero update as initialised but later paths avoid them due to learning and the jack-knife reward signal never propagates back to them.

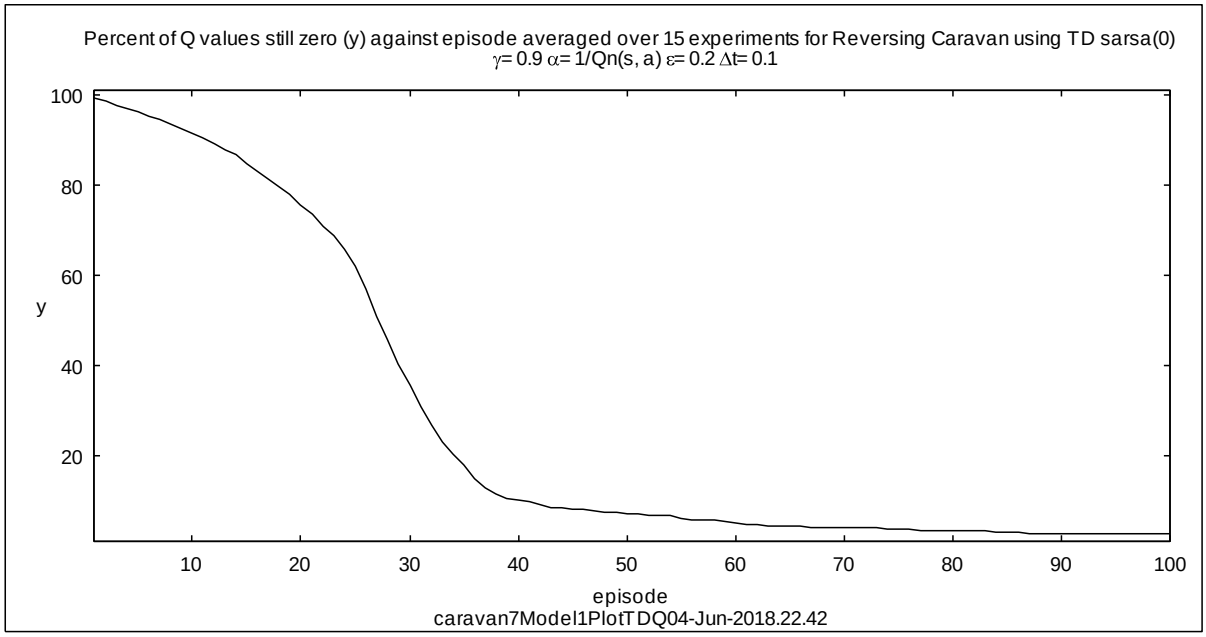


Figure 4.3.3 Percent of Q values still zero against episode

Although learning is suggested only a proper validation will show that the output policy is achieving the learning goal and that it can be generalised. The validation protocol proposed in this thesis is derived from a simulation of the learnt deterministic policy (Table 4.3.2) where all learning and exploration is switched off and the policy repeated using starting conditions each corresponding to a state (ϕ_i, Ψ_j) where ϕ_i, Ψ_j are state variable values. This validation protocol has the advantage of being repeatable, in contrast to some random

selection of starting conditions, as well as highlighting with equal probability extreme states. Table 4.3.3 shows for each starting state s0, the length of the trajectory t, the reward outcome r, the state indices (i, j) and start state variables (ϕ_i , $\Psi_{j,}$) the final state s1 and the simple unweighted Euclidian distance of (ϕ_i , Ψ_j) from the learning start state variable values ($\phi_i^L = 3$, $\Psi_j^L = 2$). For example from row 1: s0=35, $\Psi_6 = 2$, $\phi_5 = 4$ Distance = $\sqrt{((2-2)^2 + (3-4)^2)} = 1$

s0	t	r	i	j	ϕ_i	Ψ_j	s1	Distance
35	100	0.0	5	6	4	2	41	1.00
29	100	0.0	5	5	4	0	41	1.80
34	100	0.0	4	6	1	2	40	2.00
28	100	0.0	4	5	1	0	34	2.50
23	100	0.0	5	4	4	0	41	2.69
41	100	0.0	5	7	4	5	17	3.16
22	100	0.0	4	4	1	0	40	3.20
40	100	0.0	4	7	1	5	34	3.61
33	100	0.0	3	6	-1	2	33	4.00
17	100	0.0	5	3	4	-2	35	4.12
27	100	0.0	3	5	-1	0	9	4.27
16	100	0.0	4	3	1	-2	16	4.47
21	100	0.0	3	4	-1	0	15	4.72
39	100	0.0	3	7	-1	5	33	5.00
15	100	0.0	3	3	-1	-2	9	5.66
32	100	0.0	2	6	-4	2	8	7.00
11	100	0.0	5	2	4	-5	17	7.07
26	100	0.0	2	5	-4	0	8	7.16
10	100	0.0	4	2	1	-5	16	7.28
20	100	0.0	2	4	-4	0	32	7.43
36	100	0.0	6	6	10	2	36	7.50
38	100	0.0	2	7	-4	5	32	7.62
30	100	0.0	6	5	10	0	24	7.65
24	100	0.0	6	4	10	0	30	7.91
9	100	0.0	3	2	-1	-5	15	8.06
14	100	0.0	2	3	-4	-2	14	8.06
42	100	0.0	6	7	10	5	42	8.08
18	100	0.0	6	3	10	-2	42	8.50
8	100	0.0	2	2	-4	-5	14	9.90
12	3	-1.0	6	2	10	-5	6	10.26
31	18	-1.0	1	6	-10	2	1	13.50
25	15	-1.0	1	5	-10	0	1	13.58
19	15	-1.0	1	4	-10	0	1	13.73
37	3	-1.0	1	7	-10	5	43	13.83
13	15	-1.0	1	3	-10	-2	1	14.08
7	21	-1.0	1	2	-10	-5	1	15.21

Table 4.3.3 Validation of simulation from Table 4.3.2

Validation reveals steady-state trajectory to 100 steps for 80% (29/36) of the start state variable values. The failing start states are mostly extreme suggest that the selection rule learning is poor and steering is not nudging the van away from a jack-knife fast enough. An improvement to learningmay be obtained if a carefully chosen set of starting states was used which would have the effect of exposing extreme states to more visits and hence better estimation. Whist the performance of the learnt policy has been evaluated it is of interest to see how the policy performs for a particular start state.

Figure 4.3.4 shows a plot of successive ϕ and Ψ pairs for the validation of the simulation having policy of Table 4.3.2. The trace of Ψ shows that is maintained somewhere between -1.8 and 5.5 for 100 steps and no evidence this would change soon for further episodes. The steering oscillates between -15 and 19; the latter value is the steering lock plus four. The trajectory is steady-state in the sense that it oscillates about the learning start state with a very marginal drift in the positive Ψ direction.

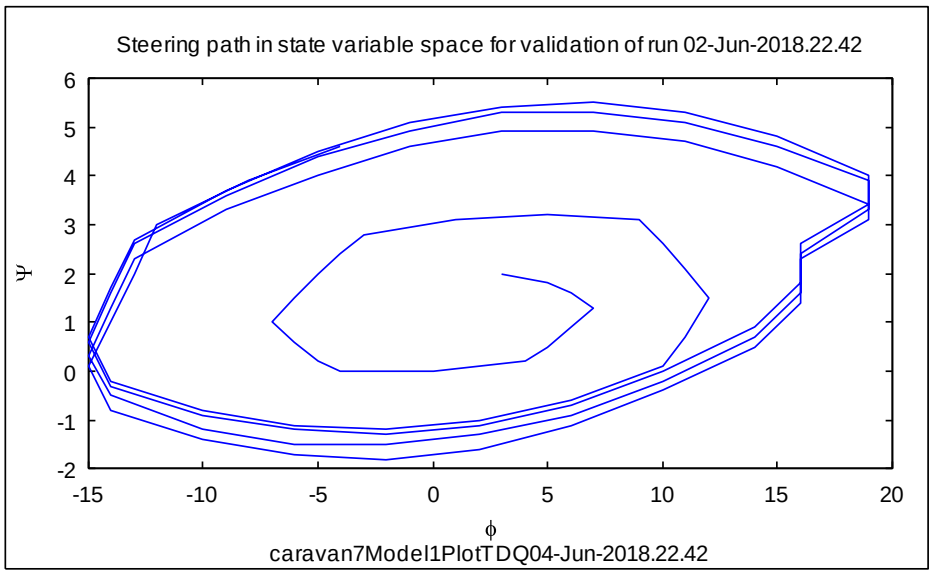


Figure 4.3.4 Successive state value pairs from Table 4.3.2

This reversing caravan problem completes objective 2.1. A state and action variable space that is sufficient for steering control has been found. A base line state aggregation is deployed which leads to a converged policy. The steering policy is successful in that a steady-state trajectory is obtained. The performance shows a clear learning phase which eventually plateaus out (Figure 4.3.2) and a falling number of unvisited states. Both of these pose the question as to whether there is a stopping rule based on properties of the Q values which would produce a feasible policy earlier. A sample average of 15 independent learning epochs ("experiments" in the figure) have been taken.

This chapter has solved the problem of maintaining the internal configuration of a dynamic system in this case the car-van angle and this needs to be complemented with learning how to steer to a target in a game world domain. The next chapter addresses this problem using a simple vehicle model.

Chapter 5 *The application of RL to a racing car game*

The motivation for this application builds on objective 2.1 and addresses another case study to follow through on the third objective and look at a more demanding dynamic task. In a racing car game road following is an essential requirement in order to solve the problem of constructing NPC's which can intelligently "drive" assets the game world. The performance of TD sarsa(0) algorithms is demonstrated for the learning of car steering based on local kerb avoiding. This offers minimal local information on which to base the steering decision. A successful outcome would complement the reversing caravan problem and make possible more complex steering task like parking into a bay.

5.1 **Application of TD algorithms to Racing Car problem**

A key decision for all learning is to identify the smallest set of state and action variables in order to mitigate the curse of dimensionality. Wang (2004) has successfully implemented a steering policy using the nearest kerb distance and its derivative. The simulated vehicle can successfully traverse loop circuits without leaving the track. The approach is used to provide a work bench for assessing the stopping rule proposed in (2.15.1). Silver (2014, lecture 7, slide 38) makes a similar two state variable approach to an RL solution to the simple actuated snake in a cranked course of straight and angled channels. He uses kerb normal distance to the head link and the angle of the kerb normal to the body axis on which to base its state variables. CEWThe simple vehicle model used here builds on the point vector model of Craig Reynolds, (Shiffman, 2012) where velocity change is essentially simulated using vector addition of the desired direction and the current direction but attenuated with appropriate upper and lower bounds. There is no attempt to model the underlying steering mechanism and its nonholonomic limitations.

A race track model data structure is built out of geometric rectangles and circular bends. The simple vehicle model represents the car at any time t by a point vector having position $\underline{p}_t$ and direction α_t. The state of the car *s*, comprise the local kerb distance *d* and velocity v from the kerb which can be analytically derived. The *move and turn* dynamics is as follows: the car travels to a new position in Δt at velocity V in direction α_t. On arrival at the new position the action available is to adjust the steering by an increment $\Delta\alpha_t$. The implications of *turn and move* dynamics are addressed below.

Intuitively it would be expected that a strong positive $\Delta\alpha$ (anti-clock wise, i.e. towards the kerb) would be needed if the car is traveling away from the left kerb and also a long way from the left kerb, conversely if near the left kerb and travelling toward it a strong negative steer is required.

The race track frame is described by X, Y coordinates. The car is at point in track shape n from which d and v are calculated. The environment also returns off track condition with n set to zero.

Following a common approach the terminal state is reached when the car has either come off the track or reached the finish line. Episodic learning is assumed with a reset to the start state if reaching a terminal state. A value of -1 is the reward on going off track and +1 if passing the finish line. Otherwise the reward is 0. The discounted reward uses $\gamma = 0.9$ obtained from the reversing caravan case study sction 4.3.

5.2 **Dynamic equations**

Below move and turn dynamics are made explicit for the motion of the car at each time step. From the initial state s_0 which entails $\underline{p}_0$ and direction α_0 the next position and orientation will be $\underline{p}_{t+1}$ and α_{t+1}.

#Initialise V, Δt

#Initialise $\underline{p}_0$, α_0 and $\Delta\alpha_0$

Repeat for t=0:T

#Update $\underline{p}_t$ α_t

(5.2.1) $\underline{p}_{t+1} = \underline{p}_t + \Delta t V[\cos(\alpha_t); \sin(\alpha_t)]$

(5.2.2) $\alpha_{t+1} = \alpha_t + \Delta\alpha_t$

#Derive $\Delta\alpha_{t+1}$ from $\underline{p}_{t+1}$ and α_{t+1} for next step

EndRepeat

In the case of a rectangular track the distance of the car to the left hand track is obtained by rotating anticlockwise the frame of reference, XOY to that of the track, SOT by θ. (Figure 5.2.1 below). This is equivalent to rotating clockwise the line OP by theta. The XOY position $\underline{p}$ is now $\underline{q}$ in the track frame of reference. The notation [d; e] is used to indicate the orthogonal components of $\underline{q}$ and subsequently for OP and $d\underline{p}$ /dt.

(5.2.3) $\underline{q} = [d; e] = R_{-\theta}[dx; dy] = R_{-\theta}\,\underline{OP}$, where [dx; dy] = OP which gives:

(5.2.4) $d = dx\cos\theta + dy\sin\theta$

The velocity of the car is $d\underline{p}/dt = [V\cos(\alpha); V\sin(\alpha)]$. The velocity v is the projection of this on the direction of d:

(5.2.5) $v = d\underline{p}/dt \,.\, [\cos\theta; \sin\theta] = V(\cos\theta\cos(\alpha) + \sin\theta\sin(\alpha)) = V\cos(\alpha - \theta)$

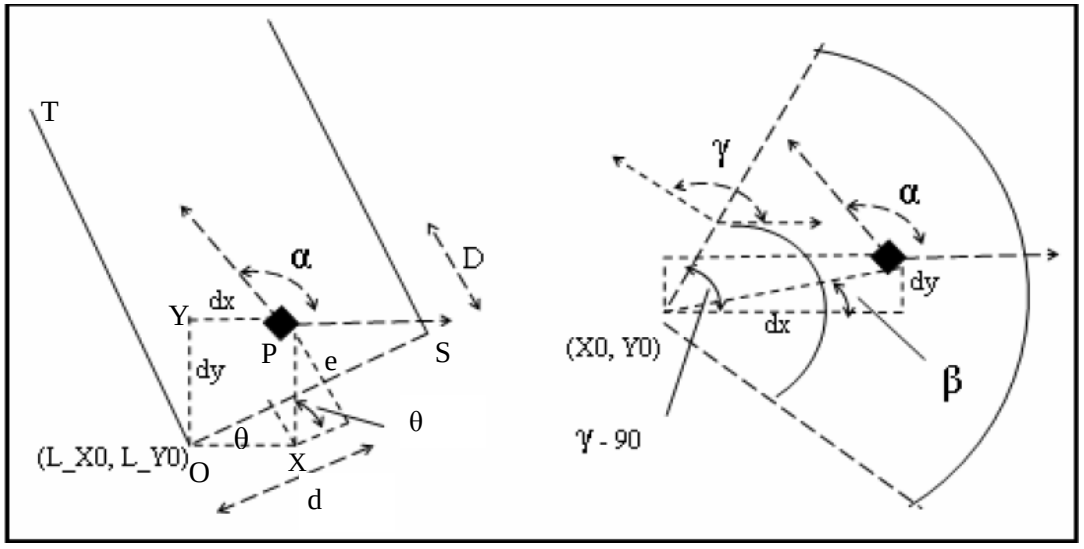


Figure 5.2.1 Rectangular and circular bends, Courtesy of Wang (2004)

For bends the right hand diagram in Figure 5.2.1 is used. If the kerb is nearest to the origin the distance to the kerb is:

(5.2.6) $d = \sqrt{((x - X0)^2 + (y - Y0)} - R$. The velocity is the projection of $d\underline{p}/dt$ onto this radius $V\cos(\alpha - \beta)$, $\beta = \arctan dy/dx$.

If the kerb is opposite to the origin $d = W + R - \sqrt{((x - X0)^2 + (y - Y0)^2)}$. The velocity is the negative projection of a vector of length V and direction α onto this radius $-V\cos(\alpha - \beta)$, $\beta = \arctan dy/dx$. In summary, following Wang(2004), the state variables have been defined and derived in terms of the car's x, y position coordinates.

5.3 **Investigating the bias in learning due to state aggregation**

In order to use action value learning algorithms the problem has to be fully discrete. Already introduced is the discrete time implicit in the Markov model by locating all change at a sequence of time step points. This section investigates the effect of state aggregation on the TD update expression and indirectly on the learnt policy. State aggregation is an approach to convert state variables d and v to state variable indices D_i and V_i according to the following: the range of d and v, dr = [0, W] and vr = [-V, V] are divided into equi-spaced sub-ranges totalling Nd and Nv. Dval are the Nd+1 boundary points of d and Vval the Nv+1 boundary points of v**.** Any $d \in dr$ has an index D_i, $D_i \in [1..Nd+1]$, indicating its enclosing range $[Dval(D_i), Dval(D_i+1)]$ and has value Dmid(i) which is the average of $Dval(D_i)$ and $Dval(D_i + 1)$. Any $v \in vr$ has an index V_i, $V_i \in [1..Nv+1]$, indicating its enclosing range and has value Vmid(i) which is the average of $Vval(D_i)$ and $Vval(D_i + 1)$.

For action aggregation the action variable Δα can take any value in the range ar = [-A, A]. For Na sub-ranges Aval denotes the Na+1 boundary points. Any $\Delta\alpha \in Aval$ has an action variable index $\Delta\alpha_i$ and an interpolated value $Aval(\Delta\alpha_i)$.

5.4 **Calculation of state to state probabilities**

Successive states s will have zero probability of being the same for any positive velocity and angle change.

(5.4.1) $\mathbb{P}[s'==s \mid s, \Delta\alpha\ ] = 0$ for state s' = (d', v') will always be true for a movement from (d, v) for $\Delta t > 0$

Linear state aggregation will coalesce a sub-region of d and v, having state variable indices D_i and V_i into one *state index* S_i. The state index is an index combining all of the state variable indices. The new sub-ranges D_i' and V_i' give the new Si'. If the vehicle move is outside the sub-region then Si' ≠ Si otherwise if it remains in the sub-region Si' == Si. State aggregation has introduced a non-zero $\mathbb{P}[S_i' == S_i, \mid S_i, \Delta A_i]$. Clearly the larger the sub-regions for Si the greater this probability and the lower the learning gain from the state paths. Using the action value update relationship (4.1.1):

(5.4.2) $Q(s, a)' = Q(s, a) + \alpha\ (r + \gamma Q(s', a') - Q(s, a))$

If (s', *a*') = (s, *a*) and its not terminal then r = 0. Inserting Q(s', *a*') = Q(s, *a*)) in 5.4.2:

(5.4.3) $Q(s, a)' = Q(s, a) + \alpha\ (\gamma Q(s, a) - Q(s, a)) = Q(s, a)(1 + \alpha(\gamma - 1) < Q(s, a)$

The value of Q(s, *a*) will be reduced and thus make action *a* less likely to be the policy. The smaller the sub-regions the greater the number of Q values that are required to be estimated. So it is important to understand the best ratios of track width, vehicle length, simulation time step and state aggregation granularity for the most effective learning. Figure 5.4.1 above shows the sub-regions associated with states (D_1, V_1), (D_1, V_2), … . The value of V as 1 is chosen to achive a step length $V\Delta t$ of a workable proportion to the track size W. To calculate the probability of a change to another state the critical value dc(1) is calculated. The Δt is set to 0.1s, the value used in the caravan study. The action variable is allowed to range from -40˚ to 40˚ to emulate a normal vehicle with a steering lock. The aggregation values Nd, Nv and Na are purely illustrative at this stage. For a left hand boundary it's the distance travelled to Dval(1) in Δt, given a velocity of v = Vval(1). Since this distance is in a negative direction it is dc(1) = |Vval(1)|Δt and similarly for v=Vval(2), dc(2) = |Vval(2)|Δt. Both critical values delineate a critical area for D_1 =1 shown as the trapezium area in the lower left hand corner.

If it is assumed that all values of v are equally likely the probability of a change in D_1 given the original state is in the sub-region D_1, V_1 , $\mathbb{P}[D_1' \# D_1 \mid D_1, V_1, 0]$ is given by the ratio of the critical area to the total area where critical area = 0.5*(base + top)*height and total area = ΔD * height.

(5.4.5) $\mathbb{P}[D_1' \# D_1 \mid D_1, V_{1,}0]$ = critical area / total area

= $0.5*\Delta t*(|Vval(1)| + |Vval(2)|)/ (Dval(2) – Dval(1)) = \Delta t*(|Vmid(V_1)| / (W/Nd)$

which for $\Delta t = 0.1$, $(W/Nd)= 0.166$ is $0.1*0.8/(0.5/3) = 0.4800$

This calculation is valid for all Δt until $dc(1) = Dval(2)$ giving $0 \leq \Delta t \leq Dval(2)/|Vval(1)|$ which at the top of the range $\Delta t = Dval(2)/|Vval(1)| = (5/3)/1.0=1.666$ and yields a probability of $0.166*0.8/(0.166) =0.8$. For $\Delta t = Dval(2)/|Vval(2)| =0.2$ no point remains the sub-region and so the probability of a state change is 1.0. An identical result is for all D_i. For V_3 two smaller triangles occur:

$\mathbb{P}[D_2' \# D_2 \mid D_2, V_{3,}\, 0] = \Delta t*|V(4)| / (W/Nd)$

Taking the compliment to get the probability of state unchanged:

$\mathbb{P}[D_2' == D_2 \mid D_2, V_{1,}\, 0] = 1 - \mathbb{P}[D_2' \# D_2 \mid D_2, V_{1,}\, 0] = 1 – \Delta t(|Vmid(V_1)| / (W/Nd))$

Similar formula derivable for $\mathbb{P}[D_i' == D_i \mid D_i, V_{1,}\, 0]$

Now the change introduced by $\Delta\alpha$ is derived. For S_i' to be the same as S_i then neither D_i nor V_j must change. Because d and v are independent variables the change in V_j is entirely due a change in α.

The condition for $V_1' == V_1$ is $\alpha_1 \geq \alpha + \Delta\alpha \geq \alpha_2$ which gives for positive $\Delta\alpha$, $\alpha_1 - \alpha_2 \geq \Delta\alpha \geq 0$ and for negative $\Delta\alpha$, $\alpha_2 - \alpha_1 \leq \Delta\alpha \leq 0$.

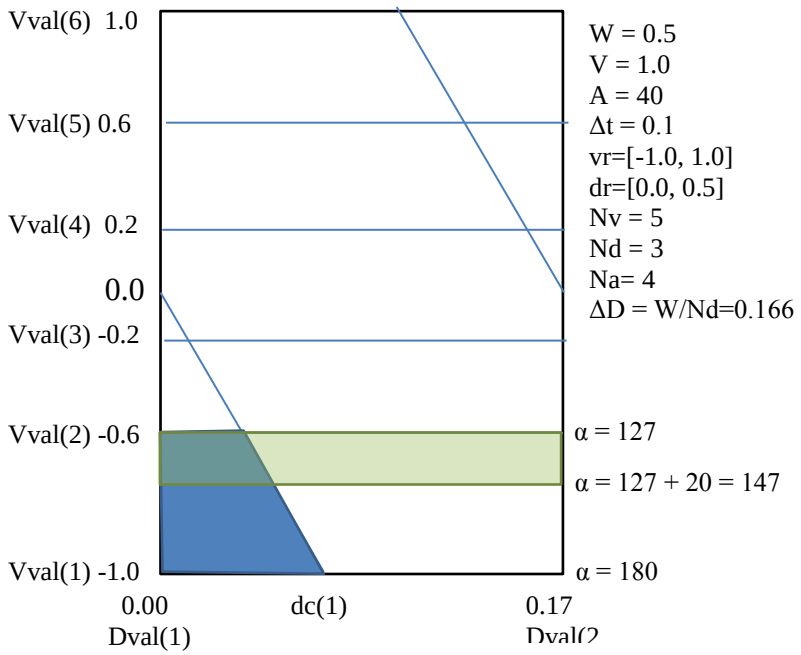


Figure 5.4.1 Showing critical area in d and v space

If it is assumed that all values of α are equally likely, the probability that the index V_j' from $\alpha + \Delta\alpha$ is the same as V_j from α, is that proportion of the α in the range $[\alpha_2 , \alpha_1]$ that are also in $[\alpha_2 + \Delta\alpha , \alpha_1]$.

An expression for $\mathbb{P}[V_1' == V_1 | V_1, \alpha_1 - \alpha_2 \geq \Delta\alpha \geq 0]$ is written:

$\mathbb{P}[V_1' == V_1 | V_1, \alpha_1 - \alpha_2 \geq \Delta\alpha \geq 0] = \mathbb{P}[\alpha \in [\alpha_2 , \alpha_1] \text{ AND } \alpha \in [\alpha_2 + \Delta\alpha, \alpha_1]]$

$= (\alpha_1 - \alpha_2 - \Delta\alpha)/(\alpha_1 - \alpha_2) = 1 - \Delta\alpha/(\alpha_1 - \alpha_2)$

For $0 \leq \Delta\alpha \leq 53$ the critical region occurs from 127 and above. E.g. for $\Delta\alpha$ =30:

$\mathbb{P}[V_1' == V_1 | V_1, 53 \geq 30 \geq 0] = \mathbb{P}[\alpha \in [127 , 180] \text{ AND } \alpha \in [127+30, 180]$

$= 1 - 30/(180 - 126.87) = 1 - 0.5647$

For a negative $\Delta\alpha$

$\mathbb{P}[V_1' == V_1 | V_1, \alpha_2 - \alpha_1 \leq \Delta\alpha \leq 0] = 1 + \Delta\alpha/(\alpha_1 - \alpha_2)$

The critical area corresponds to a rectangle in the upper part of sub-region (1,1). So in general:

(5.4.6) $\mathbb{P}(V_1' == V_1 | V_1, \alpha_1 \geq \alpha + \Delta\alpha \geq \alpha_2) = 1 - |\Delta\alpha|/(\alpha_1 - \alpha_2)$

Combining the results of 5.4.5 and 5.4.6 and using the complementary probabilities:

(5.4.7) $\mathbb{P}[S_i' == S_i | S_i, \Delta\alpha] = \mathbb{P}[D_i' == D_i \text{ AND } V_j' == V_1 | D_i , V_j, \Delta\alpha]$

$= \mathbb{P}[D_i' == D_i | D_i, V_1, \Delta\alpha] * \mathbb{P}[V_j' == V_j | V_j, \Delta\alpha]$ since they are independent

$= (1 - \Delta t(Vmid(V_j) / (W/Nd))(1 - |\Delta\alpha| /(\alpha_j - \alpha_{j+1}))$

Continuing the example for S_1, which is confirmed below:

$\mathbb{P}[S_1' == S_1 | S_1, -30] = 1 - \Delta t(|Vmid(V_1)| / (W/Nd))(1 - |30|/(180 - 126.87))$

$= (1-0.4800)*(1- 0.5647) = 0.2264$,(see underlined values in Table 5.4.1)

Table 5.4.1 shows the calculation of (5.4.7) for D_2 for all V_j and $\Delta\alpha$

	Δα:	Vmid	-30	-10	10	30
	1	-0.8	0.2264	0.4221	0.4221	0.2264
	2	-0.4	0.0000	0.4600	0.4600	0.0000
Vi	3	0.0	0.0000	0.4986	0.4986	0.0000
	4	0.4	0.0000	0.4600	0.4600	0.0000
	5	0.8	0.2264	0.4221	0.4221	0.2264

Table 5.4.1 ℙ[Si' == Si| Si, Δα] with Vi and Δα

Interpreting above ℙ[Si' == Si| Si, ± 30] shows zero for mid-range V_i. The value for ± 10 shows higher ℙ[Si' == Si| Si, ± 30] values for mid range of V_i , (-0.4 0.0 0.4) which correspond to moves parallel to the track.

The probability of no state change when there is a state variable change has been calculated for an example aggregation. The dependence on Δt, $\Delta\alpha$ and ΔD and ΔV has been established and the implication for learning drawn out. From (5.4.7) it can be concluded that $\mathbb{P}[Si' == Si|\ Si, \Delta\alpha]$ declines linearly as Δt, $\Delta\alpha$ and ΔD increase and will lower the learning bias. The is also a non-linear decline with ΔV due to the term in $1//(\alpha_1 - \alpha_2))$ where $\alpha_1 = \cos^{-1}(V_1 /V)$. However as Δt increases the step length increases which will make the dynamics less accurate to the point of unstable. A larger $\Delta\alpha$ will get lower $\mathbb{P}[Si' == Si|\ Si, \Delta A_i]$ but increase vacillation in trajectories. It is proposed to use this result as a guide to the choice of the Nd, Na, and Nv and adjust to lower $\mathbb{P}[Si' == Si|\ Si, \Delta\alpha]$. The next section illustrates this in part by the simulation of a fine grained policy which has larger Nd, Na and a linear expression for $\Delta\alpha$.

5.5 **Establishing the existence of a kerb-avoiding policy**

The rest of this section brings together the state aggregation, P21 monitoring, UCB or ϵ-greedy selection in order to complete objective 4.1 regarding fast, lightweight and flexible learning algorithms. To show that a policy does exist which will produce a steering behaviour that always moves forward and avoids kerbs a simulation has been constructed which tests out a intuitive hand crafted policy constructed below Table 5.5.1. At car position $\underline{p}$ a direction change $\Delta\alpha$ is generated based on the deterministic policy $\Delta\alpha^{\pi}$. The car then increments at velocity V in direction α for Δt seconds to a new position p' and then increment its direction by $\Delta\alpha$. As a starting point I have used a V range of [-6 6] and a d range of [0 3] to assist in the construction of a transparent expression for $\Delta\alpha^{\pi}$. A finer aggregation than above of Nv = 6, Nd = 9; $\Delta\alpha^*$= 9 is deployed in order to achieve a more sensitive steering control and thus show proof of concept. A linear expression in D_i and V_i gives a $\Delta\alpha^{\pi}$ as a proportion of the maximum shift $\Delta\alpha^*$ and the state indices D_i, V_i (5.5.1):

(5.5.1) $\Delta\alpha^{\pi}(D_i, V_i) = \Delta\alpha^*(-0.5 + V_i/(2*V) + D_i/W)$

Table 5.5.1 shows values for $\Delta\alpha^{\pi}(D_i, V_i)$ using (5.5.1)

D below			V across			
0.0	-5.0	-3.0	-1.0	1.0	3.0	5.0
0.15	-7.8	-6.2	-4.8	-3.3	-1.8	-0.2
0.50	-6.8	-5.2	-3.8	-2.2	-0.8	0.8
0.85	-5.8	-4.2	-2.8	-1.3	0.2	1.7
1.15	-4.8	-3.3	-1.8	-0.3	1.2	2.7
1.50	-3.8	-2.2	-0.8	0.7	2.2	3.8
1.85	-2.8	-1.3	0.2	1.7	3.2	4.7
2.15	-1.8	-0.3	1.2	2.7	4.2	5.8
2.50	-0.8	0.8	2.2	3.8	5.2	6.8
2.85	0.2	1.7	3.2	4.7	6.2	7.7

Table 5.5.1 $\Delta\alpha^{\pi}(D_i, V_i)$ against D_i, V_i

For example the value of $\Delta\alpha^{\pi}(1.15, -3.0)$ is -3.3.

#The octave code is below

#Implement (5.5.1)

function `da=astar(Di, Vi)`

da= `daStar*(-0.5 +Vmid(Vi)/(2*V) + Dmid(Di)/W)`

endfuntion

#Initialise V, Δt

```
V= 0.7
dt= 0.1
```

#Build track structure tk within members {O, θ , L, W..}

#Initialise $\underline{p}_0$, using Dmid, α_0 and $\Delta\alpha_0$

```
p0=p=[Dmid(Di)+6.0, 0.01];
a0=a=50;
```

#Compute (d, v) from $\underline{p}_0$, α_0, (5.2.5) and (5.2.4)

```
v = V*cosd(a - tk(n).theta)
```

#Compute (D_i and V_i) from (d, v) using #interpolation function I

```
Di= I(d, Dval, len(Dval));
Vi= I(v, Vval, len(Vval));
```

Simple steeringtrajectories for several Δα* with Δt = 0.1 and V = 0.7

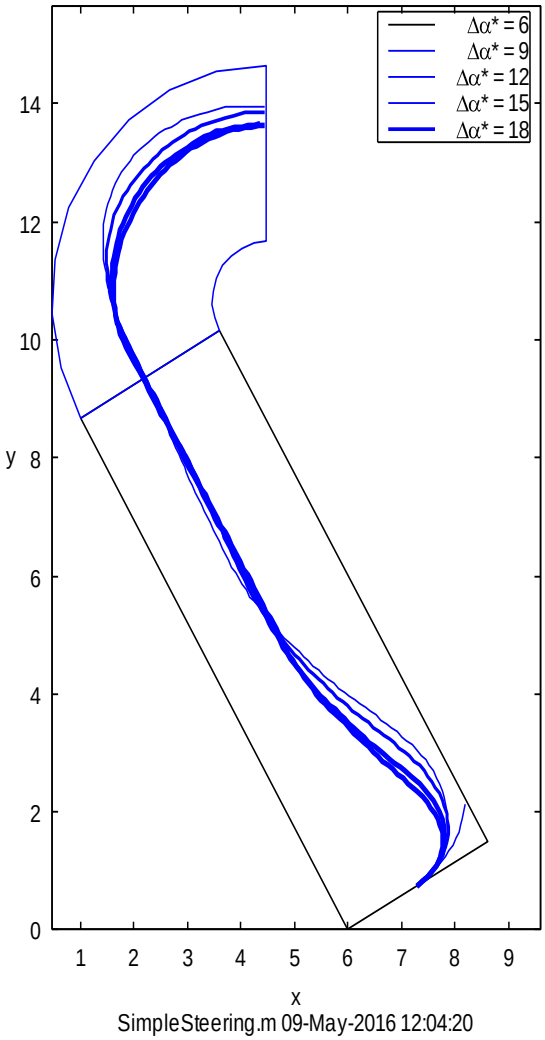


Figure 5.5.1 Simple Steering trajectories for hand crafted policy

#Compute $\Delta\alpha_0$ from $\Delta\alpha^{\pi}(D_i, V_i)$

```
da = astar(Di, Vi);
```

#Repeat for t=0:T

#Update $\underline{p}_t$, α_t

#(5.2.1) $\underline{p}_{t+1} = \underline{p}_t + V\Delta t[\cos(\alpha_t), \sin(\alpha_t)]$

```
p1 = p +[V*dt*cosd(a), V*dt*sind(a)];
```

#(5.2.2) $\alpha_{t+1} = \alpha_t + \Delta\alpha_t$

```
a1 = a + da;
```

#Derive $\Delta\alpha_{t+1}$ from (D_i and V_i) from (d, v), (d, v) from $\underline{p}_{t+1}$ and α_{t+1}

#$\Delta\alpha_{t+1} = \Delta\alpha^{\pi}(D_i , V_i)$

```
da1= astar(Di, Vi);
```

#Calculate reward for car finish (1), of track(-1) or just on-track(0)

#Update p, a and da

```
a=a1; p=p1; da=da1
```

#Exit if r#0

#end repeat

Simple steering trajectories for four states with Δt = 0.1 for coarse aggregation

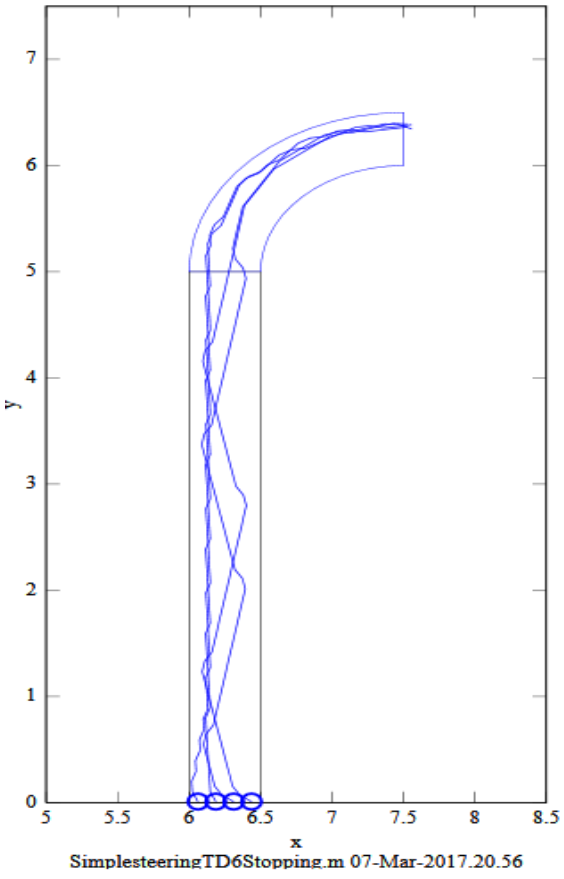


Figure 5.5.2 Simple Steering validation of coarse aggregation

The first simulation is to trial a hand crafted policy of a fine grained aggregation Nv= 6 and Nd= 9. Figure 5.5.1 shows five steering trajectories using increasing Δα* from 6 to 18. Only value 6 fails to turn early enough, the rest go beyond the first kerb encounter. It does convincingly show that a simple steering policy can be constructed from just very local information. This aggregation presents 6*9*4 = 216 Q values to estimate each may require over 100 values and has the potential to be too computationally demanding. For the purposes of learning a reference policy is needed which has properties of 5.5.1 against which to compare learnt policies.

A coarse but tractable aggregation (Na, Nv, Nd) = (3, 5, 5) is proposed in Table 5.5.2 below with a reference policy which is anti-symmetric about the antidiagonal and has proper behaviour at the kerbs. The state (1, 1) corresponding to state variables (0.05, -0.8) indicates a vehicle close to the kerb and heading towards it with velocity -0.8 and is given

a steering nudge of -21 to reduce α and thus bringing the vehicle away from the kerb. The three zero values correspond to maintaining current steering if the vehicle is parallel to the track.

			V across		
D below	-0.8	0.4	0.0	0.4	0.8
0.05	-21.0	-21.0	-21.0	21.0	21.0
0.15	-21.0	-21.0	0.0	21.0	21.0
0.25	-21.0	-21.0	0.0	21.0	21.0
0.35	-21.0	-21.0	0.0	21.0	21.0
0.45	-21.0	-21.0	21.0	21.0	21.0

Table 5.5.2 Handcrafted reference policy for coarse aggregation

Figure 5.5.2 show four trajectories from different values of D_i . The effect of the no change zone in the state space is manifested by straight lines with swerves when near to the kerb. The effect of the coarse aggregation is displayed in Table 5.5.3 using the probability P[Si' == Si| Si, Δα] for D_i=2. The effect of a coarser aggregation has reduced values at extreme indicies (1, -10) = 0.1247 at extreme indices compared to the calculation of Table 5.4.1 (1,-10) = 0.4221.

		P[Si' == Si| Si, Δα]	
Δα:	-10	0	10
1	0.1247	0.2000	0.1247
2	0.1263	0.6000	0.1263
Vi 3	0.1066	0.8000	0.1066
4	0.1263	0.6000	0.1263
5	0.1247	0.2000	0.1247

Table 5.5.3 Variation P[Si' == Si| Si, Δ] with Vi and Δα

Figure 5.5.3 below shows finishing trajectories starting from the mid points of all state sub-

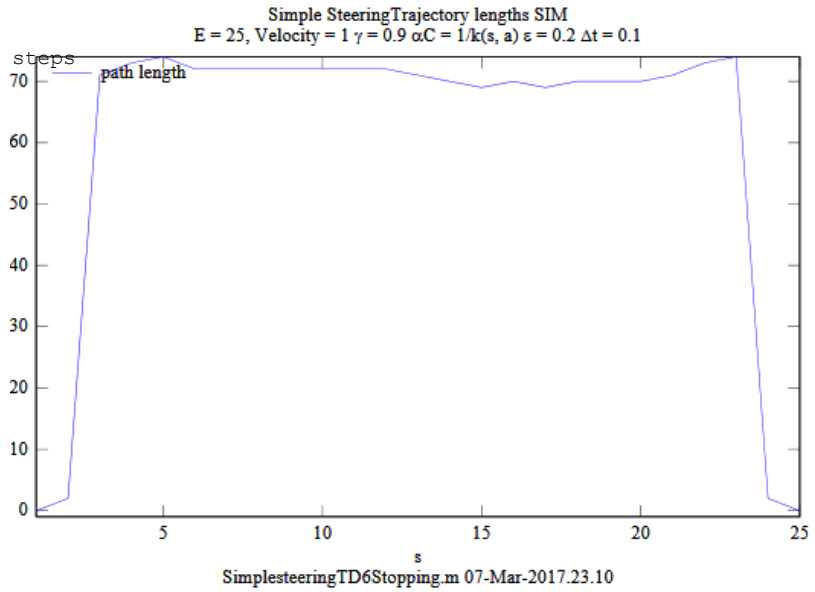


Figure 5.5.3 Simple Steering finishing trajectory length

regions except the extreme cases s=1,2, 24, 25. The path lengths of the finishing

trajectories are all over 66. The performance of finishing trajectories is (25 – 4)/25 = 84% of all trajectories. The extreme cases correspond to extreme proximity to the kerb and maximum velocity to the kerb and where in the first step the vehicle leaves the track.

5.6 **The application of TD sarsa(0) to a racing car steering**

The next task is to show that RL can provide an algorithm that can learn a feasible steering policy. As a base line it is necessary to show that TD Sarsa(0) for state indices will yield a feasible steering policy. Subsequently the application of the early stopping is investigated. The learning parameters are those of the caravan exercise. The implementation of TD sarsa(0) using the UCB / ϵ-greedy selection rule and the binary PMSMMPM based on the algorithm in 4.3 is reworked below:

#SimpleSteering4.m

#Track parameters W, V, L, R

```
W= 0.5; V= 1.0; L=1; R=1;
```

#Set aggregation parameters Nv, Nd, Na

```
Nv = 5; Nd = 5; Na = 3;
```

#MRP, Learning and simulation constants

```
XX=5; E=100; T=120;
```

#Set time and state partitions of Δt, Dval, Vval and Aval

```
dt= 0.1; Dval = 0 : W/ Nd : W; Vval = -V : 2*V / Nv : V;
```

#Generate race track in data structure tk

```
[tk, bounds, W, exit]=maketrack(pls);
```

#Set learning parameters γ, α_μ, ϵ

```
gamma = 0.9; epsilon = 0.2
```

#Initialise Q_0, k, S_0, s_0^2 , $s_{\mu 0}^2$, $\hat{U}_0$, RR

```
smu2=srt2=S=U=k=Q=zeros(Ns, Na);
```

#Repeat for E episodes

```
for e=1:E
```

#Initialise $\underline{p}_0$, and α_0

```
  p=[6.25, 0.1] ;a0=a=50;
```

#Generate n, the shape number enclosing $\underline{p}$, d , v, D_i, V_i and s_0

```
  [n, D2L, V2L] = gets(p, a, tk, V, Dval, Vval, prt);

  Di= I(D2L, Dval, len(Dval)); Vi= I(V2L, Vval, len(Vval));
```

```
s=sub2ind ([Nd, Nv], Di, Vi);
```

#Get the index ι_t from Round Robin using $k(0, s_0, a_0)$

```
Qsn=sum(k(s,: )+1; i=mod(Qsn-1, Na) +1;
```

#Get the real value of $\Delta\alpha_0$ by interpolating ι_0 in Aval, $\Delta\alpha_0$ = `X(`ι_0`, Aval)`

```
da = X(i, Aval);
```

#Generate next episode, where t is the step number for episode e

```
for t = 0:T
```

#Move to $\underline{p}_{t+1}$ along α_0 and change orientation to α_{t+1} , (5.2.2)

$\underline{p}_{t+1} = \underline{p}_t + V\Delta t[\cos(\alpha_t), \sin(\alpha_t)]$ #[x, y] is octave notation for a column vector

```
p1 = p +[V*dt*cosd(a), V*dt*sind(a)];
```

$\alpha_{t+1} = \alpha_t + \Delta\alpha_t$

```
a1 = a + da;
```

#Generate n, d' ,v', and calculate D_i', Vi' and hence s_{t+1}

```
[n, D2L1, V2L1] = gets(p1, a1, tk, V, Dval, Vval, 0);
D1i= I(D2L1, Dval, len(Dval)); V1i= I(V2L1, Vval, len(Vval));
s1=sub2ind ([Nd, Nv], D1i, V1i);
```

#Check on/off track and get r_{t+1}. Determine ι_{t+1} depending on learning setting

```
if(!n) #Car off track
    if(!isnan(norm(intersectEdges ([p, p1], [exit(1,:,
    exit(2,:])))))
        r = 1;   #Car travels trough finish Line
    else
        r = -1; #Car comes off early
    endif
    Qs1DA1i = 0 #since s1 is terminal. Following Barto,1983
else #Car still on track  r_{t+1}= 0, Choose ι_{t+1} on basis of learning method
    r = 0;
    if(s_{t+1}) has still has values in RR range, ι_{t+1} = RR(1)
        i1=inRR(1);                    #Choose ι_{t+1} based on RR
    elseif(UCB), ι_{t+1} = argmax_j (Q_u(s_{t+1} , j) + Û_u(s_{t+1}, j))
        [Qm, adesc]= sort(Q(s1, : ) .+ U(s1, : ), "descend");
```

```
                i1=adesc(1);
```

elseif (EGREEDY), $\rho \sim U[0,1]$, if($\rho > \epsilon$) ι_{t+1} = U[1..Na]

```
                if(rand()<epsilon)i1 = unidrnd(Na); #Exploration
                else, #ι_{t+1} = arg maxWithTieBreaks_ι (Q_u(s_t , •)
                [i1, ties] = randtiebrks(Q(s1,: ); #Exploitation
                endif
            endif
        endif #n
```

#Compute $k(u+1, s_t, \iota_t)$ and α_μ, $k(u+1, s_t, \iota_t) = k(u, s_t, \iota_t) +1$; $\alpha_\mu = 1/ k(u+1, s_t, \iota_t)$

```
  k(s,i) = k(s,i) +1;
  alphaC = 1/k(s,i);
```

#Calculate TD error $\delta_{t+1} = r_{t+1} + \gamma Q_u(s_{t+1}, \iota_{t+1}) - Q_u(s_t, \iota_t)$

```
  delta = r + gamma*Qs1DA1i - Q(s, i);
```

#Carry out TD update $Q_{u+1}(s_t, \iota_t) = Q_u(s_t, \iota_t) + \delta_{t+1} \alpha_\mu$

$Q_{u+1}(s_t, \iota_t) = Q_u(s_t, \iota_t) + \delta_{t+1} \alpha_\mu$

```
  Q1stDAi = Q(s, i) + alphaC*delta;
```

#Sums of squares update $S_{u+1}(s_t, \iota_t) = S_u(s_t, \iota_t) + (r_{t+1} - Q_{u+1}(s_t, \iota_t)) (r_{t+1} - Q_u(s_t, \iota_t))$

```
  S(s, i) = S(s, i) + (r - Q1stDAi)*(r - Q(s, i));
```

#Compute sample variance of the reward of ι_t, $s_{u+1}^2(s_t, \iota_t) = S_{u+1}(s_t, \iota_t)/(k(u+1,s_t, \iota_t) - 1)$

```
      srt2(s,i)= (1/(k(s, i) -1))*S(s, i);
```

#Compute sample variance of the action value of ι_t, $s_{\mu u+1}^2(s_t, \iota_t) = s_{u+1}^2(s_t, \iota_t) /k(u+1,s_t, \iota_t)$

```
      smu2(s,i) = srt2(s,i)/k(s, i);
```

#Compute $Q_u(s_{t+1}, \iota_{t+1})$ into local memory

```
  Q(s, i)= Q1stDAi;
```

#Sort $Q_{u+1}(s_t, \iota_t)$ descending magnitude with sort order $a_t(\iota)$ $Q_{u+1}^S(s_t, \iota_t) = Q_{u+1}(s_t, a_t(\iota))$

```
  [Qts, ats]= sort(Q(s, : ) , "descend");
```

#Compute $\mathbb{P}[Q_{u+1}(s_t, a_t(2))) \leq Q_{u+1}(s_t, a_t(1))]$ and if >0.95 mark as converged

```
  P21(s)=normcdf((Q(s,ats(1))-Q(s,ats(2)))/sqrt(smu2(s,ats(1))+smu2(s,
ats(2))));
  if(conv(s)==0 && P21(s) > 0.95) conv(s)=1; ; endif
```

#Calculate the UCB $\hat{U}_{u+1}(s_t, \iota_t) = \sqrt{(2\log((u+1) / k(u+1,s_t, \iota_t))}$

```
  U(s, i) = sqrt(2*log(u+1)/k(s, i));
```

#Get new $\Delta\alpha_{t+1}$ `= X(ιt, Aval)`

```
  da1 = X(i1, Aval);
```

#Increment u

```
  u=u+1;
```

#Exit if off track

```
  if(!n) break endif
```

endfor `#t`

endfor `#e`

#Get the stochastic policy $\Pi^*(s, \Delta\alpha_i)$ from Q (3.7.1)

#Get the deterministic control policy $\Delta\alpha^*(s)$ which is the expectation of Amid over the #corresponding stochastic policy, $\Delta\alpha^*(s) = \Sigma_\iota \Pi(s, \iota)Amid(\iota)$

```
astar = reshape(Pi*Amid, Nd, Nv);
```

The next section is to show that RL based on the coarse grained aggregation can learn a feasible steering policy and investigates the effectiveness of the stopping rule above (2.8.6).

5.7 **Learning experiments**

Figures 5.7.1(a) and 5.7.2(b) show a visualisation of the learning process for ϵ-greedy and UCB respectively. The TD sarsa(0) learning algorithm has been implemented for the simple steering vehicle model on a track of a straight section and a right hand curve. The

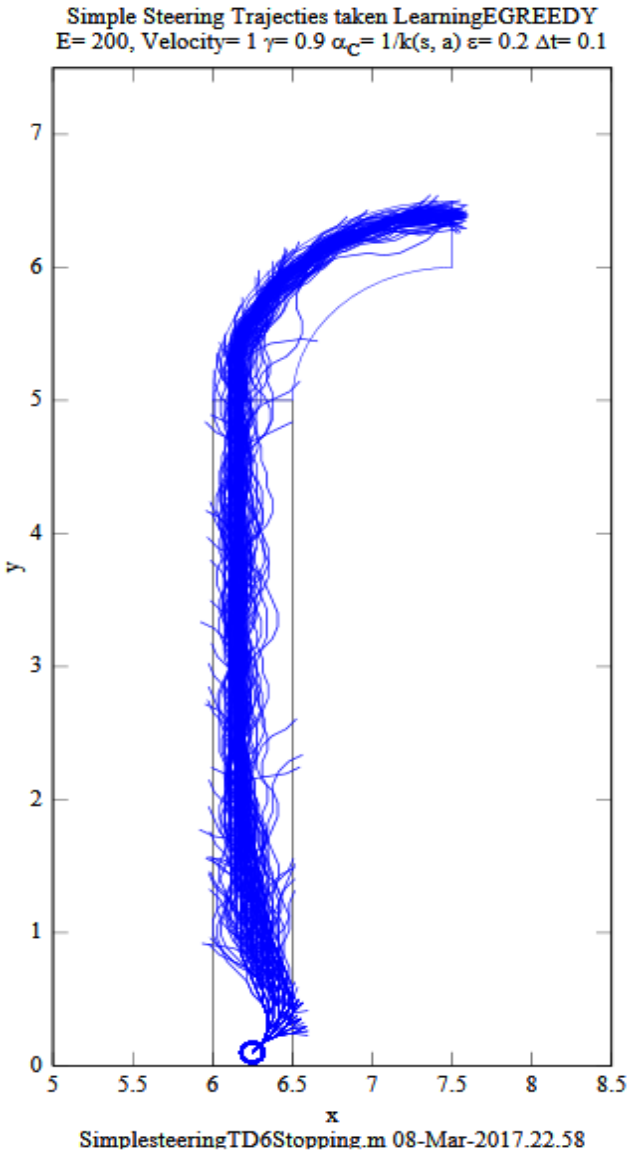


Figure 5.7.1(a) Simple Steering trajectories during ϵ-greedy learning

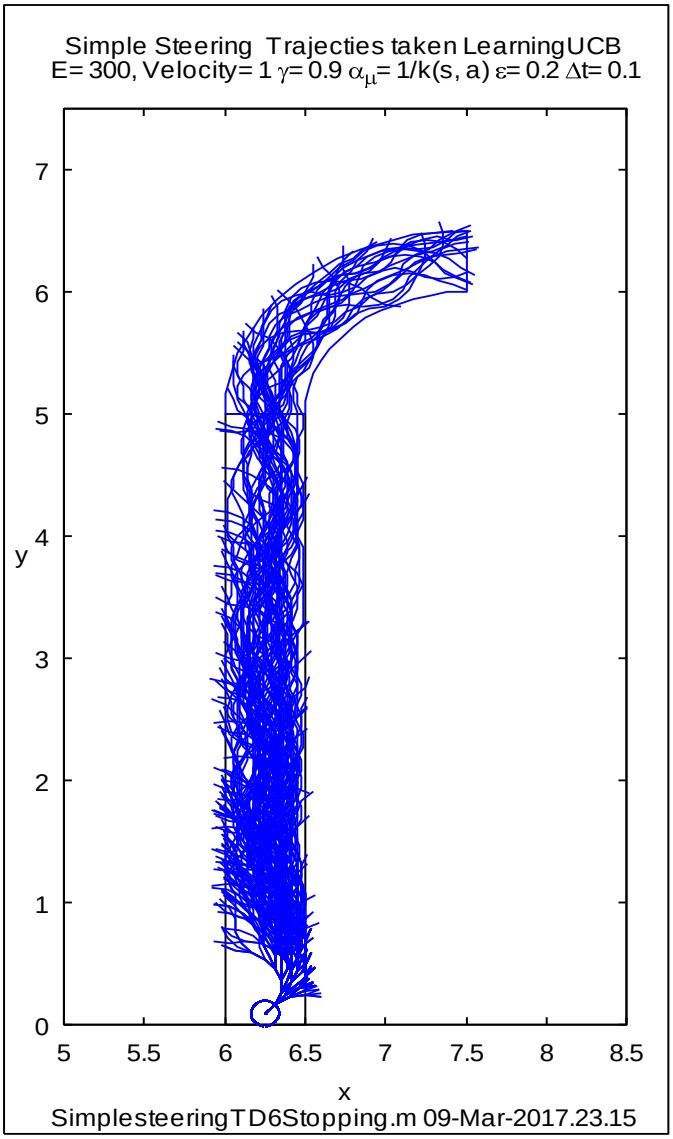


Figure 5.7.1(b) Simple Steering trajectories during UCB learning

reference aggregation of 5.5.1 has been used (Na, Nv, Nd = 3, 5, 5) with V=1 and the nudge steering having values 0 and ± 20. The P[Si' == Si| Si, Δα] are those of Figure 5.5.1.

Figure 5.7.1(a) shows 200 trajectories and uses a 0.2-greedy selection rule. CEWThe values chosen here are informed by Wang's research (2004). The trajectories show more exploration than those of Wang(2004) possibly due to a larger exploration parameter. The 200 episodes generate 9035 learning steps which are apportioned on average at 120 over each of the 75 (s, ι) pairs, but with about 15% of them below six. This is lower than that needed for a reliable estimate of the variance and confirms that the on-policy data stream does not find and visit all states equally.

Figure 5.7.1(b) above shows 300 trajectories below which generate 6309 steps at an average of 84 per state action pairs but only 4% are below 6. Compared to the 15% for ϵ-greedy, UCB does an improved job of exploration. The pattern of trajectories is more spread out than ϵ-greedy which suggests a broader range of exploration in the data stream.

For ϵ-greedy the learning performance in Figure 5.7.2(a) below shows the trajectory lengths by episode and displays a rising trend which plateaus out at around episode 55. The dips thereafter are due to the 0.2-greedy random steering set at 20% of all action selections used to achieve exploration. For UCB the learning performance of in Figure 5.7.2(b) below shows the trajectory lengths by episode and displays a rising trend of peak values which plateaus at around episode 150 much later than ϵ-greedy.

Figure 5.7.2a, b shows the convergence of $\mathbb{P}[Q^S_t(2) \leq Q_t^S(1)]$ (P21) on a rising or approximately level trend for both ϵ-greedy and UCB.

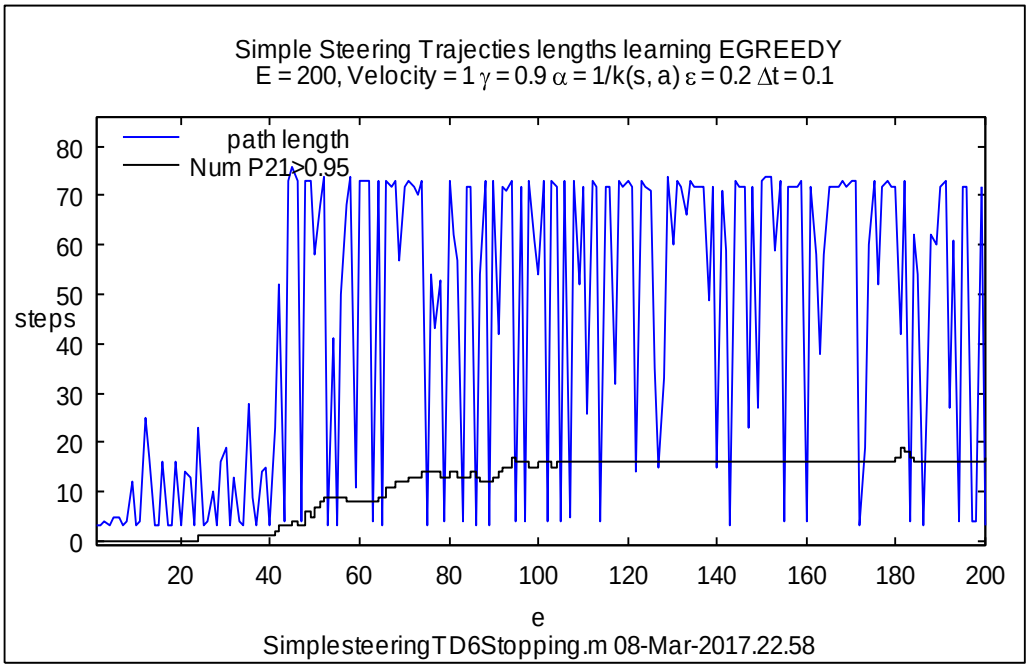


Figure 5.7.2(a) Simple Steering trajectory lengths during ϵ-greedy learning

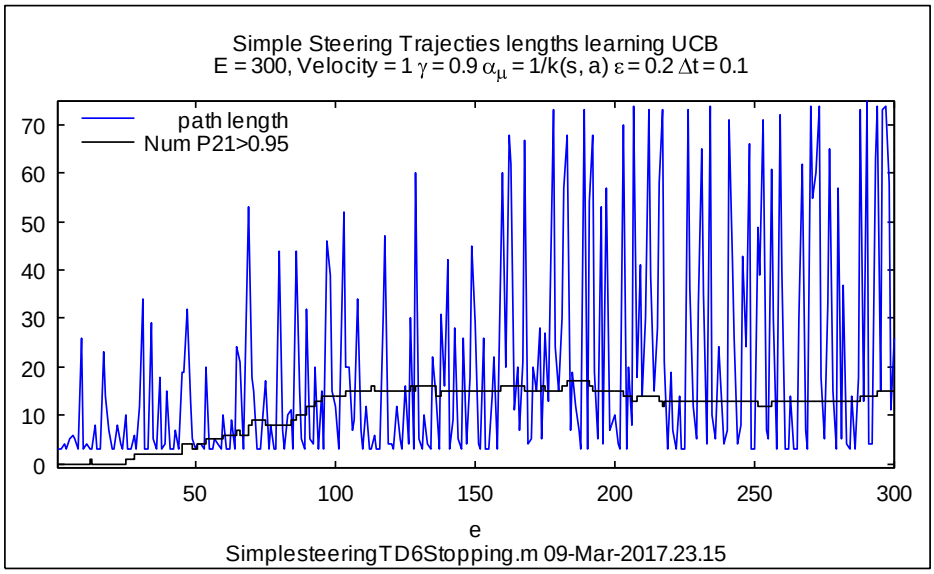


Figure 5.7.2(b) Simple Steering trajectory lengths during UCB learning

For ϵ-greedy the $\mathbb{P}[Q^S_t(2) \leq Q_t{}^S(1)]$ (P21) below (Table 5.7.1(a)) show mostly values over 0.95 with 17 qualifying for convergence at episode 200. Among the 8 that are yet to pass three remain at zero which is confirmed by the state action visits being two low. For UCB the $\mathbb{P}[Q^S_t(2) \leq Q_t{}^S(1)]$ (P21) below (Table 5.7.1(b)) show mostly values over 0.95 with 15 qualifying for convergence at episode 300. All the $\mathbb{P}[Q^S_t(2) \leq Q_t{}^S(1)]$ values are greater than 0.5 reflecting the better data count at extreme states than for ϵ-greedy:

	D=	0.05	0.15	0.25	0.35	0.45
	-0.8	0.793	0.986	0.992	1.000	0.000
	-0.4	0.933	1.000	1.000	1.000	1.000
V=	-0.0	0.997	0.596	0.999	1.000	1.000
	0.4	0.836	1.000	1.000	0.999	0.867
	0.8	0.000	0.000	1.000	0.963	0.996

Table 5.7.1(a) P21 for ϵ -greedy

0.05	0.15	0.25	0.35	0.45
0.908	0.894	0.999	1.000	0.995
0.955	1.000	1.000	0.640	1.000
0.723	0.806	0.999	0.969	0.999
0.792	0.683	0.927	1.000	0.999
0.593	0.813	1.000	0.999	0.996

Table 5.7.1(b) P21 for UCB

For ϵ-greedy the output policy shown in the table below (Table 5.7.2(a)) shows differences in four cells from the reference policy (see section 5.5) denoted R. As intuitively expected it shows anti-symmetry about the approximate antidiagonal. For UCB the output policy shown in the Table (5.7.2(b)) below shows differences in six cells from the reference policy (see section 5.5) denoted R.

	D=	0.05	0.15	0.25	0.35	0.45
	-0.8	-20	-20	-20	0.0R	20
	-0.4	-20	-20	0	20	20
V=	-0.0	-20	-20	20.0R	20	20
	0.4	-20	-20	20.0R	20	20
	0.8	-20	0.0R	20	20	20

Table 5.7.2(a) Output Policy for ϵ -greedy

	D=	0.05	0.15	0.25	0.35	0.45
	-0.8	-20	-20	-20	0R	0R
	-0.4	-20	-20	-20R	0R	-20R*
V=	-0.0	-20	-20	0	20	20
	0.4	-20	-20	0	20	20
	0.8	-20	0R	20	20	20

Table 5.7.2(b) Output Policy for UCB

The validation protocol is derived in the same manner as section 4.3 and the policy deployed using a standard set of starting conditions each corresponding to a state (D_i, Vi). This validation protocol has the advantage of being repeatable, in contrast to some random selection of starting conditions, as well as highlighting with equal probability extreme states. For ϵ-greedy Figure 5.7.3(a) provides a visual validation of the learnt policy. Gratifyingly the trajectories eventually converge to a common route independent of the start position. For UCB Figure 5.7.3(b) shows the performance is 84% finishing trajectories with a mean length of 59.4, very marginally lower than ϵ-greedy. The output policy generates two dominant trajectories unlike the single one for ϵ-greedy.

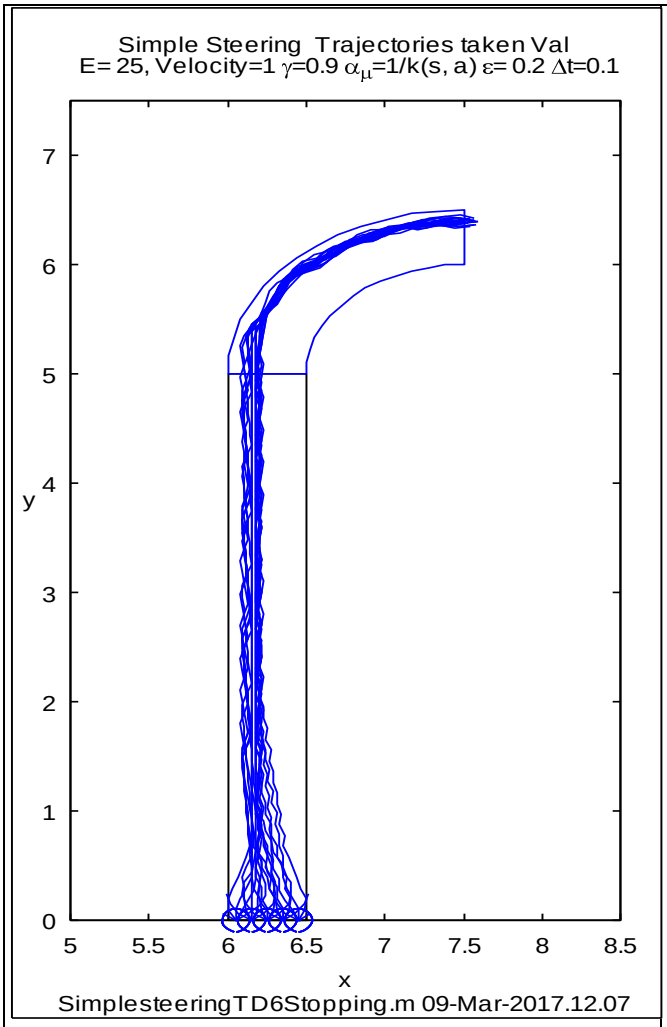


Figure 5.7.3(a) Simple Steering trajectories for ϵ-greedy validation

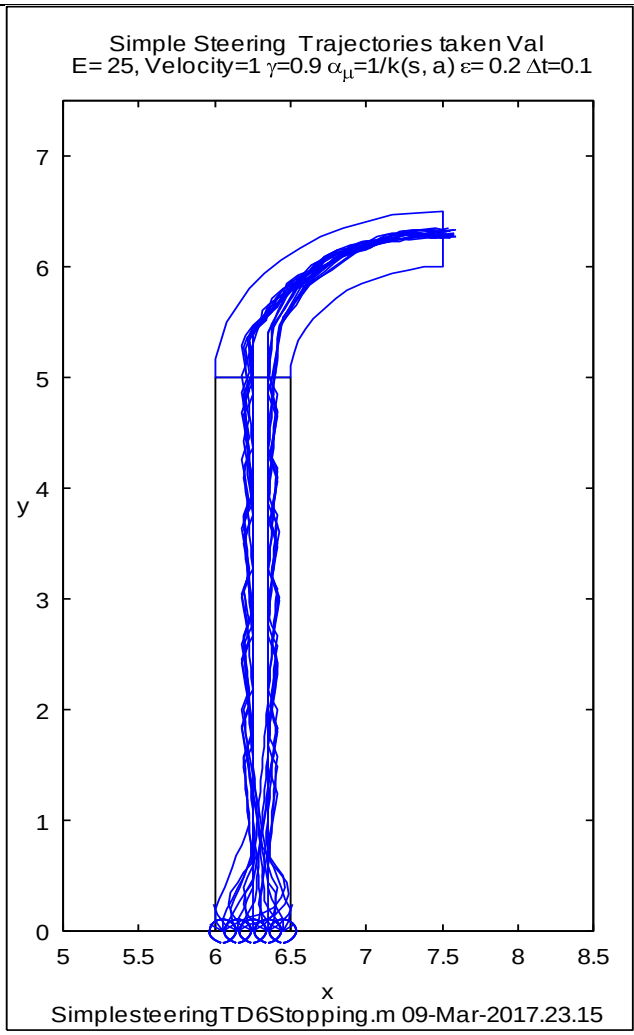


Figure 5.7.3(b) Simple Steering trajectories for UCB validation

Figures 5.7.3(a) and 5.7.3(b) show a visualisation of the validation for ϵ-greedy and UCB respectively. The performance variable is the % of finishing trajectories and derives directly from the reward intention. For both ϵ-greedy and UCB Figures 5.7.4(a) and (b) shows 84% of trajectories finishing achieves with a mean length mean trajectory of over 70 each starting from a different state.

The four failures at s=1,2, 24, 25 correspond to extreme states where the policy does not make the car turn sufficiently quickly.

The problems with states 1 and 25 are investigated here. The turning circle radius, R, of a vehicle at p steering in direction α, and advancing ΔtV in one time step, and then turns Δα clockwise is:

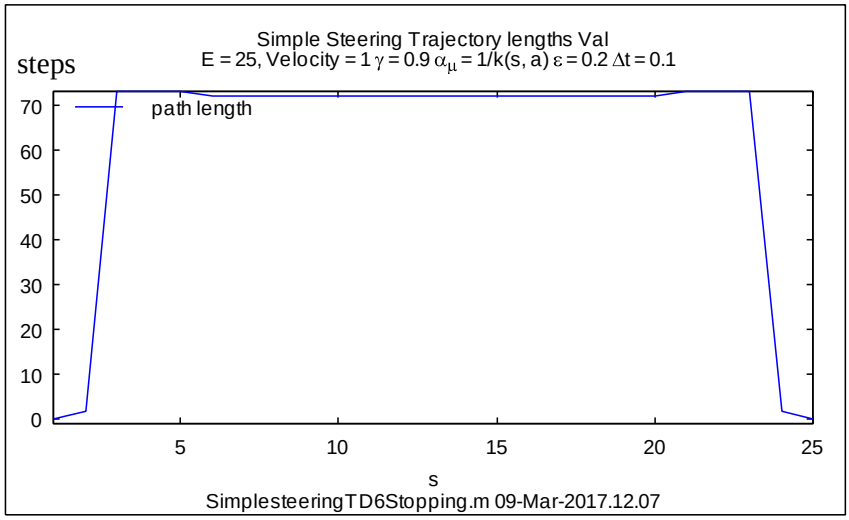


Figure 5.7.4(a) Simple Steering trajectory lengths for ϵ-greedy validation

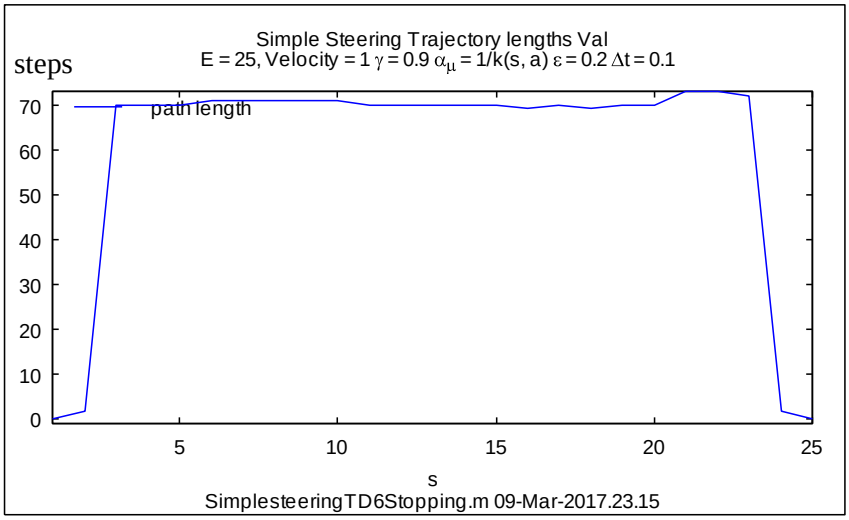


Figure 5.7.4(b) Simple Steering trajectory lengths for UCB validation

(5.7.1) $R= (\Delta t*V/2)/(\sin(\Delta\alpha/2))$

The condition for a car d from the edge and orientation α to steer away from the edge is:

(5.7.2) $d + R\cos\alpha \geq R$,

For $\Delta t = 0.1$, V=1, $\Delta\alpha = 20$, then inserting in (5.7.1), R= 0.288. With α =143, d=0.05 and s=1 and inserting into (5.7.2), 0.05 + 0.288*cosd(143) = -0.18 which is not greater than R. Thus there is no policy that can avoid a failure if the car starts in states 1, 2, 24 or 25. A point vector model for a car has proved sufficient to provide a state and action variable space that is sufficient for steering control. A simple state aggregation has been found which offers a basis for a kerb avoiding reference policy. The learned steering trajectory lengths reach the max possible at episode 55 and 150 for ϵ-greedy and UCB respectively. The proportion of converged PMSMMPM rise steadily and plateau at 17/25 around 100 episodes for both selection rules. The output policies generate validation trajectories which show a common route at the finish line except for four extreme start states.

5.8 **An aggregate stopping rule**

Table 5.8.1 shows a percentage of P21 that have met the convergence criterion. This section compares several aggregate measures of the P21 matrix which result in a single figure. A single figure which embodies the correct trade offs provides an effective aggregate stopping criteria.

A simple aggregate stopping rule is just the proportion that met the individual criterion, P21 > δ. Table 5.8.3 shows values 0.68 (17/25) and 0.60 (15/25) for egeedy and UCB respectivly. There is no obvious simple relationship between δ and the proportion above and so not readily interpretable or generalizable.

The P21(s) of each state is a function of the ΔQ and SQ (2.18) of the top two actions and the number of visits. An aggregate measure using the sum of each P21(s) weighted by d(s) has an interpretation as the expected probability of a correct action at the next step, explained as follows: A set of E epochs and trajectories T(e) e ∈ 1..E generate a set of steps from which d is estimated, and from which a random subset of size S is extracted which gives the expected number of visits to a state s, Sd(s). The expected number of correct actions from state s is Sd(s)P21(s). The expected number of correct actions at the next step is ΣSd(s)P21(s) . The expected probability of a correct action at the next step is = (Σ Sd(s)P21(s))/S = Σ d(s)P21(s). It also has the property that the most visited states influence the measure the greatest. This reflects their importance in the dynamics of the learnt policy. The unvisited states have by definition zero P21 and are correctly omitted. The infrequently visited sates will have a poor estimate of P21 and will be discounted by having a low d value. On the other hand any P21 that is well over the threshold will compensate for below threshold states and hide important differences of P21 across states. Using the estimated stationary distribution the average P21 is given by (5.8.3). Table 5.8.3 shows that UCB does come close to the 0.95 threshold but the implication is that both need more episodes. The higher value of $\overline{P21}$ for UCB correlates with its more episodes and its higher unweighted average P sum. In fact this counts for 0.90792-0.83464 = 0.07 of the difference.

(5.8.3) $\overline{P21} = \Sigma_s P21(s)d_s$

run	selection rule	Proportion > 0.95	$\overline{P21}$	episodes	Σ P21/25
08-Mar-2017.22.58	ϵ-greedy	0.68	0.82416	200	0.83864
09-Mar-2017.23.15	UCB	0.60	0.93	300	0.90792

Table 5.8.3 Aggregate selection rule for ϵ-greedy and UCB

Chapter 6 ***Critical appraisal and conclusions***

The scope this thesis is to demonstrate the application of RL to moving assets in the game world to achieve realistic NPC learning for control. In order to make a critique the original objectives from chapter 1 are restated.

Objective 1.0: Show how action selection contributes to efficient learning.

Objective 2.1: To exploit and develop the implications of RL learning algorithms for the reversing caravan and its generalisation to related movement tasks.

Objective 2.2: Applying other known solutions to the reversing caravan problem.

Objective 3.0: Explore the wider application of learning in games.

Objective 4.1: To produce fast, lightweight and flexible learning algorithms suitable for run-time embedding in current games.

Objective 4.2: To **e**xplore at least one of the key theoretical problems arising from current theoretical advances.

4.2.1. Determine a convergence criterion that achieves a specified level of probability.

4.2.2. Investigate the properties of state aggregation.

4.2.3. Construct a single measure of convergence based on the measure for each state.

Objective 5.1: To investigate and develop metrics for the validation and evaluation of the performance of RL algorithms using a standard set of simulated problems.

Objective 5.2: Demonstrate and visualise the learning progress.

Motivated by objective 1.0 the investigation into action selection reveals the interrelation of exploitation, exploration and the efficiency of action selection, since these determine the accuracy of output results.

As evidenced from the %optimal action over the first 200 episodes there is considerable difference in learning power and trends only become apparent over 5000 episodes. The ideal action selection is one which is has high %optimal action through the sample range. From Figure 2.7.1 this is UCB except between 500 and 1000 episodes when it is Reinforcement Comparison.

An explanation lies with the structure of the distribution of the rewards and with the greediness of the action selection. Regarding the structure of the reward, each epoch uses a reward drawn from a normal distribution whose mean is Q* and variance is 1. Q* is also drawn from a unit normal distribution. It implies that 40% of the means are bunched in in the range [-.52 .52], and the means have standard deviation 1/√ (sample size), so it takes

100 samples to distinguish means that are at least 2 x $1/\sqrt{(100)} = 0.4$ apart. Greater samples will be needed in the case of closer and therefore harder to distinguish means. Those action selections that focus on the most likely actions rather than over exploring the least likely will be successful in finding the optimal.

It would be illuminating to have calculated a measure of exploration defined as the proportion where the action chosen is not the current greedy action, Σ_e a $\neq$ arg maxWithRandomTieBreak Q_t. The greediness of the action selection is the trend to reduce exploration as the number of episodes increase. Soft max %optimal only reaches 64% and it may be due to the expression for the probabilities is very insensitive to small differences in Q and so weaker actions will be over favoured. An unexploited action in UCB will eventually produce a growing U value large enough to supersede the maximum but only to fall back down when it is selected and its k value increased, so exploration continues even when an action is a clear maximum. The output to objective 1.0 is the decision to use UCB as the favoured action selection and to compare it to ϵ-greedy as a reference.

The evidence of Figure 2.7.1 reveals the crucial role action selection plays in achieving fast learning and motivates the investigation of the greedy action selection expression. Action selection must address the following cases: a high variance in the best action value or bunched action values that will require more exploration to distinguish them.

A variance based approach to convergence has been implemented with a focus on deciding when convergence has occurred and what form the convergence criterion should take. Using a proper definition (Mnih etal, 2008), the investigation of convergence of a random process concludes that successive differences are ill-conditioned, UCB is sound but ignores the reward variance information, the confidence interval is sound and needs the variance and the Bernstein bounds (Mnih etal, 2008), need both variance and range. A relevant observation is that all of these measures ignore that what is needed is only the convergence that confirms the maximum. All that is needed is only to explore until the top two action values are different enough for the probability of the action of maximum sample mean being the maximum population mean (PMSMMPM) to be a number arbitrarily near to 1.0.

A probability based measure of convergence PMSMMPM has been derived from first principles. This contributes to objective 4.2.1 and provides a stepping stone to a convergence criterion. A sound formulae for the convergence of policy evaluation, PMSMMPM, based on order statistics and which exploits the semantics of the policy evaluation expression, has been proposed, implemented and tested using an n-armed bandit simulation.

A Bandits simulation has been implemented using an initial round robin of 2 cycles for each action, and using UCB action selection and numerical integration to output $\mathbb{P}[\max_{a\#1}(Q_t^S(a)) \leq Q_t^S(1)]$ at each play t. A sample epoch using a 3-armed bandit shows a monotonically rising PMSMMPM to a value of 0.988 at play 26 thus passing the convergence criterion of 0.95 and which provides evidence for its feasibility for a stopping rule. In summary PMSMMPM is a sound and interpretable measure of convergence.

The classic student t-test (Gosset, aka Student, 1908), provides a t statistic and it is relevant to comment its validity for very small sample sizes and thus enable omission of the round robin step. For P12 values based on student t estimates with samples as low as 2 can only be supported if the difference in the means is large, the randomness is normal and the variances are comparable (De Winter, 2013) but since these cannot be assummed the round robin safeguard has been deployed.

The stopping rule defined here is to quit plays at the first t such that PMSMMPM greater than $1 - \delta$, where interpret δ is interpreted as a significance level as used in statistical estimation.

Further simulations show the mean of PTC increases markedly as the top two action values get closer but decrease as their variance gets smaller reinforcing the intuition that more samples are needed for estimation of close or high variance Q population means. The step when the sample maximum is the population maximum to a given probability, described as PTC is simulated and it properties investigated as the underlying population parameters vary. The frequency distribution of the number of plays to convergence based on PMSMMPM shows a high peak at the first allowed episode and a very long tail, a result that shows that the stochastic reward sequence can produce a wide range of sample average behaviour. Objective 4.2.1 has been substantially met with the development and testing of an iteration stopping rule and which will contribute to faster learning algorithms. The results show that PMSMMPM with an initial RR of 5 will provide a robust measure of convergence.

Objective 3.2 is to develop fast, lightweight, algorithms and so a simpler expression for PMSMMPM is desirable. The effect on PMSMMPM of ignoring the third and subsequent action value shows that at the worse case of a tie between second and third introduces an error of 17% which rapidly fades as they move apart. PMSMMPM is reasonably approximated by P21 which is an expression that uses standard library functions and does not require numerical integration and theoretically should be faster than PMSMMPM.

A minimal sampling size is of interest since it will promote faster learning and thus contribute to Objective 4.1. To that end an expression for the total sampling effort based on

population variances subject to the achievement of a given value for P21, is minimised and yields an expression for the theoretical minimum sampling sizes

The results of a simulation of P21 show lower %optimal action as the population means of the action values come closer and it is harder to discriminate between very close population means. The tighter the variance of either action reduces the mean PTC confirming that is easier to discriminate where action values have less probability overlap. The validity of using P21 is confirmed since the theoretical minimal PTC based on the A==2 case with population parameters closely tracks the empirical mean PTC based on sample estimates.

The expression for P21 shows an explicit dependence on the respective samples sizes. P21 can be increased the most by selecting which ever action has the greatest positive gradient. Based in on explicit expression for the derivative of P21 a selection rule to decide whether to choose action 1 or action 2 is derived. Since choosing action 1 or 2 is not a complete exploratory policy the policy η–UCB combines UCB with P21 every ηth step to achieve completeness. It is shown that the 4-UCB produces 68% less sampling than 0-UCB using either population variances or sample estimates which supports the claim that the η-UCB selection rule will contribute to faster learning. This result rests on a point estimate and to make a rigorous conclusion an average over many epochs would be needed.

In summary the expression for the simplest case of just two actions provides a basis to show there is a theoretical minimum sample size that could achieve any desired level of probability. A selection rule is proposed and demonstrated based on P12, the tractable approximation of PMSMMPM that chooses that action which yields the higher marginal convergence probability of taking either action 1 or action 2 at step t. A new *complete exploratory policy* is proposed and simulated based on augmenting UCB with the *selection* rule from P12.

The foundation of TD is the modelling of dynamic change using a stationary discrete Markov Decision Process characterised by a set of states, actions, action transition probabilities, and conditional rewards. The system constraints for a the existance of a stationary distribution are derived and the implication for NPC assets is that places of no return in the game world should be designed out. If not the places of no return will need a separate learning exercise.

In part as proof of concept and in part as a learning exercise TD sarsa(0) was applied to learn a steering policy that will solve the caravan reversing problem. Although animations are available as far as online searches reveal this problem has been untried as a machine learning exercise. It presents a sufficiently complex problem which is relevant to game

dynamics. In particular it learns the steering behavour that has the latency of a human operator in contrast to following a predifined trajectory of traditional animations. A geometric model for the configuration of a vehicle and caravan has been built which realistically captures steering motion given a steering angle and indicates the completion of objective 2.1.

The steering angle and the car-van angle have been determined as the state variables and numerically well conditioned incremental dynamic equations for displacement of the ensemble over the time interval Δt derived, based on move then turn dynamics. In a game world asset dynamics would be done as a matter of course. The action variable is the steering angle change reflecting the normal experience of incremental steering turn and will be determined by the selection rule.

A state aggregation scheme has been used to translate variable values to state variable indices. Similarly an aggregation of sub-ranges for the steering angle change deployed. The output of the TD sarsa(0) is a learnt steering change policy for any pair of state variables which displays the appropriate symmetry but with deviations at the edge columns due the reason as follows: the distribution of visited states shows the bulk of the volume at the central sub-regions of the state index space underlining again the difficulty in estimating extreme state configurations. The exercise shows that TD sarsa (0) can learn a successful steering policy that will enable a reversing caravan to find a trajectory arc that does not jack-knife.

A validation exercise which simulates the ensemble starting from the state variables at each of the state indices and which switches off all learning and exploration shows 19% jack-knife trajectories during 100 steps. A plot of a trajectory by ϕ and Ψ shows overlapping circular paths revealing the learnt behaviour is more jack-knife avoidance than finding the steering angle which achieve a steady state.

Control engineering offers a solution to the reversing caravan problem, (objective 2.2). Building on the dynamic systems approach of Jayakaran (2004) the equation of motion the reversing caravan problem is derived and shown to be equivalent to the geometric approach of this thesis. A transformation of the equation of motion into a standard first order differential equation with known boundary conditions is solved for a symbolic solution using the Wolfram|Alpha deductive engine. In the games context most NPC motions will not have an exact solution whereas the incremental numerical approach is generalizable.

The caravan reversing problem has been complimented with a general kerb avoiding steering problem of a racing car and expands objective 3.0. In the interest of avoiding the

curse of dimensionality Wang's (2004) small set of state and action variables are followed and a point vector model for the car is deployed using move then turn dynamics. Of course kerb avoiding will fail at junctions or dead ends where trajectories need to be determined using other contextual information. A NCP will need a route following policy to make decisions at junctions. If racing is the objective then close kerb following may be a better reward than just kerb avoiding.

State aggregation is a key approach to transforming the continuous problem to a discrete one but in making any approximation there is a price to pay in modelling accuracy. The investigation into the bias effect of state aggregation on the state action estimates shows that a illustrative aggregation of 60 values does introduce a maximum probability of 0.5 of the state being unchanged consequent on a single step. The implication is that for 50% of the steps the action value is diminished and ultmately its action may be less likely to be the policy action. The expression for this probability declines linearly as each of Δt, $\Delta\alpha$ and ΔD increases up to an upper bound. However increasing Δt will increase the step length and make the incremental dynamics less accurate and a larger $\Delta\alpha$ will increase vacillation in trajectories. This suggests that a larger time interval is used for learning and is separate from that used for incremental dynamics. To assist easy implementation it could be a small multiple of the dynamic time interval. Recapitulating state aggregation has been shown to introduce a bias downwards on action values for steps that remain in the same state action region and it receives no adjustment from action values of successor state action regions. It implies that only those steps which produce a state value region change will contribute to effective learning and by implication more state action visits and hence more episodes will be needed to compensate.

A deterministic policy based on a function of the state variables with the intuitively correct antidiagonal symmetry using a fine aggregation scheme of 216 elements demonstrates feasible trajectories and so confirms the validity of the point vector model. For the purpose of a less demanding approach feasible trajectories are shown which are based on a reference policy with appropriate antidiagonal symmetry and based on an aggregation of 75 elements.

The ϵ-greedy and UCB selection rules to racing car problem using simple steering model with a 2 segment track are compared. For both the course aggregation of the reference policy and its steering change are used. The experiments show more spread in the trajectories than in Wang's (2004) study, possibly due to a larger exploration factor.

From an implementation of objective 5.2 an animation and visualisation of the learning progress of the UCB shows even more failures throughout the track and as a consequence

generates considerable fewer state/action visits but does show fewer pairs unvisited. The learning performance of ϵ-greedy shows a clear rise to a plateau whereas UCB shows a much slower rise punctuated by many dips. An interpretation is that this is due to the higher level of exploration implicit in UCB. In both cases the PMSMMPM achieves convergence in almost 70% of all states.

The output policy of ϵ-greedy conforms better when compared to the reference, however the respective validations show similar proportion of finishing trajectories which suggests that the final policy need only be near optimal to satisfy the learning task. It is worth observing that there is no left hand turn yet the top left hand corner of the output policy conforms to the reference policy exemplifying that some generalisation has been achieved. Besides demonstrating that TD sarsa(0) can produced a succesfull steering policy for the racing car the PMSMMPM has been monitored and evidence shows it serves as a valid stopping rule in this context as well.

As with the reversing caravan objective 5.1 results in a validation protocol which standardised the testing procedure and provides evidence of the power of the output steering policy to generate finishing trajectories. In both learning case studies the start state is constant for all epochs which may well reflect a constraint on NPC's in the game scenario. Game designers would need to vary start states to be biased towards extreme states in order to increase their visit their count and reduce their sampling error.

The stationary distribution developed above is employed to produce an aggregate convergence measure (objective 4.2.3).

The $\overline{P21}$ provides a candidate aggregate stopping criteria and the evidence so far suggests that with more episodes it could be reached.

For the purposes of positive play it is better to have a policy entry for an extreme state which has a lower than specified convergence probability than a completely random action which will be the case where no visits occur.

Summarising the contribution to the science offered is the derivation, implementation and demonstration of a probability measure to a given level convergence of the maximum action value of a TD learning process. The learning of a simple steering problem has been modelled, implemented and validated using two selection rules. Two selection rules are compared and their respective convergence properties reveals complementary differences in the final policy. The probability of convergence is reaches the target level for a majority of the states and an aggregate probability of convergence as the expected probability of a correct action at the next step is derived and compared between the selection rules. The

contribution to intelligent NPC's is the demonstration of a rapid, robust and flexible (Lucas, 2009) learning tool to small if challenging problems but with additional insight on how to scale up and generalise to more demanding behaviours.

References

Barto, A. G. and Sutton, R. S. (1981) *Goal seeking components for adaptive intelligence: An initial assessment*, Technical Report AFWAL-TR-81-1070, Air Force Wright Aeronautical Laboratories/Avionics Laboratory, Wright-Patterson AFB, OH.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983), *Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems*, IEE TRANS ON SYSTEMS, MAN, AND CYBERNETICS Vol. SMC-13, No. 5, September/October.

Barto A. and Dietterich T. (2004), *Reinforcement Learning and its Relationship to Supervised Learning*, University of Massachusetts, Amherst, MA and Oregon State University, Corvallis, OR.

Bellman, R. E. (1957), *Dynamic Programming*, Princeton University Press, Princeton, NJ.

Bentley, J. (1975), Multidimensional binary search trees used for associative searching, Communications of the ACM 18, 509–517.

Bernstein, S. N. (1927), "*Theory of Probability*" (Russian), Moscow.

Berry, D. A. and Fristedt, B. (1985), *Bandit Problems. Sequential Allocation of Experiments.* Monographs on Statistics and Applied Probability. Chapman and Hall, London/New York.

Bevilacqua, F. (2012), Understanding Steering Behaviours, http://gamedevelopment.tutsplus.com/series/understanding-steering-behaviors--gamedev-12732, [visited 2018-12-23].

Dayan, P. (1991), *Reinforcement comparison*, In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., and Hinton, G. E., editors, *Connectionist Models: Proceedings of the 1990 Summer School*, pages 45-51. Morgan Kaufmann, San Mateo, CA.

De Winter J., C., F. (2013), *Using the Student's t-test with extremely small sample sizes*, Practical Assessment, Research & Evaluation, Volume 18, Number 10, August 2013.

Doya K. (2000), *Reinforcement Learning in Continuous Time and Space*, Neural Computation, 12(1), 219-245.

Eaton, J. W. (2012), *Gnu octave, version 3.6.2*, http://www.gnu.org/software/octave/.

Enzenberger, M. and Muller, M. (2009*), Fuego – an open-source framework for board games and go engine based on monte-carlo tree search*, Technical Report TR 09-08, Dept. of Computing Science., University of Alberta, Canada.

Finch, T. (2009), *Incremental calculation of weighted mean and variance*, University of Cambridge Computing Service.

Forbes, J. R. and Andre, D.(2000), *Real-time reinforcement learning in continuous domains*, Computer Science Division, University of California, Berkeley, CA.

Forbes J., Huang, T., Kanazawa, K., and Russell, S., (1995), *The BATmobile: Towards a Bayesian Automated Taxi*, Computer Science Division, University of California, Berkeley, CA 94720, USA.

Funge, J. D., (1999), *AI for Games and Animation: A Cognitive Modeling Approach*, A. K. Peters. Natick, MA.

Gelperin, A., Hopfield, J. J. and Tank, D. W. (1985), The logic of *limax* learning. In Selverston, A., editor, *Model Neural Networks and Behavior*. Plenum Press, New York.

Gibbs, J. W. (1902), *Elementary Principles in Statistical Mechanics*, New York: Charles Scribner's Sons.

Gosset, W. S. 1908, "*The Probable Error of a Mean*", Biometrika. **6** (1): 1–25, doi:10.1093/biomet/6.1.1.

Graepel, T. (2005), *Machine learning for games mlss*, Microsoft Research, Cambridge. http://videolectures.net/mlss05au_graepel_mlg/. 1.2 .

Hassibis, D. (2017) https://deepmind.com/research/alphago/, visited 24th June 2017.

Hassibis, D. (2017b), [radio], *Desert Island Disks*, 2017, BBC Radio 4, 21 May 2017, 11:15am.

Hoeffding, W. (1963), '*Probability inequalities for sums of bounded random variables'*, *Journal of the American Statistical Association* **58**(301),13 – 30.

Jayakaran, A. (2004), Enhanced Trailer Backing, Msc. thesis, Graduate School of the University of Florida. cimar.mae.ufl.edu/CIMAR/pages/thesis/jayakaran_a.pdf, visited: 24/2/2014.

Jones, C. and Crowe, M. (2014), Reluctant Reinforcement Learning, *in* M. Bramer, M. Petridis & A. Hopgood, eds, 'Research And Development In Intelligent Systems', p.p. 85 – 101.

Kendall, G. M. and Stuart, A. (1997), *The Advanced Theory of Statistics: Distribution Theory*, Macmillan, NY.

Klopf, A. (1982), *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence*, Hemisphere, Washington D. C.

Koulouriotis D. E. and Xanthopoulos A. (2008) Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems , Applied Mathematics and Computation 196 913–922.

Lucas, S. M. (2009), editorial. IEEE Transactions on Computational Intelligence and AI in Games. Vol. 1. 1. March.

LaValle, S. (2006), 'Planning algorithms'. http://planning.cs.uiuc.edu/node658.html [Accessed June 2013].

Michie, D. (1974), *On Machine Intelligence*. Edinburgh University Press.

Michie D. and Chambers R. A. (1968a) *BOXES: An experiment in adaptive control*, in Dale E. and Michie D., (Eds.), Machine Intelligence 2, Edinburgh: Oliver and Boyd, pp137 – 152.

Michie D. and Chambers R. A. (1968b) *'BOXES' as a model of pattern formation*, in Towards a Theoretical *Biology, Vol 1, Prolegomena, Waddington C. H., (Ed.) Edinburgh: Univ Press, 1968. pp. 206 – 215.*

Minsky, M. L. (1961), Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49:8-30. Reprinted in E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw-Hill, New York, 406-450, 1963.

Mitchell, T. (1997), *Machine Learning*. McGraw Hill. p.2. *ISBN0-07-042807-7*

Mnih V. , Szepesv´ari C., Audibert J. Y., (2008), *Empirical Bernstein Stopping*, In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland.

Peng, J. and Williams, R. J. (1996), '*Incremental multi-step q-learning'*, *Machine Learning* **22**(1-3).

Pollard, D., (1998), *Introduction to Statistics*, Lecture 7, Statistics Department Yale University, Connecticut.

Reynolds, C. (1987). *Flocks, Herds, and Schools:* A Distributed Behavioral Model, *Computer Graphics*, 21(4), pp. 25-34.

Robbins, H. and Monro, S. (1951), 'A stochastic approximation method', *The Annals of Mathematical Statistics* **22**(3),400.

Rubinstein, Y. and Kroese, D. (2017), *Simulation and the Monte Carlo Method*, Wiley.

Samuel, A. (1959), '*Some studies in machine learning using the game of checkers*', *IBM J. Res. Develop* **3**, pp211–229.

Scott, D. B. and Tims S. R. (1966), Mathematical Analysis An Introduction, Cambridge University Press.

Scutt. T. (2009), *Artificial Intelligence, Artificial stupidity*, Keynote speaker at the AI games network meeting June 2009, Imperial College.

Sharma R. (2008), *Some more inequalities for arithmetic mean, harmonic mean and variance*, Journal of Mathematical Inequalities, Volume 2, Number 1 (2008), 109–114.

Shiffman, D. (2012), *The Nature of Code: Simulating Natural Systems with Processing*, creative commons.

Sigman, K., (2006), Communication classes and irreducibility for Markov chains, IEOR Department, Columbia University, www.columbia.edu/~ww2040/4701Sum07/4701-06-Notes-MCII.pdf, [visited 2018-02-26].

Silver, D. (2014), '*Advanced topics in reinforcement learning'*. http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. and Riedmiller, M. (2014), *Deterministic Policy Gradient Algorithms*, Proceedings of the International Conference on Machine Learning.

Stigler, S. (1973), "*Studies in the History of Probability and Statistics. XXXII: Laplace, Fisher and the Discovery of the Concept of Sufficiency*", Biometrika. **60** (3): 439–445.

Sutton, R. S. and Barto, A. G. (1998), *Reinforcement Learning: An Introduction*, MIT Press, Cambridge. http://webdocs.cs.ualberta.ca/sutton/book/ebook/the-book.html, [Accessed 9 September 2011].

Szepesvari, C. (2009), *Algorithms for reinforcement learning, in* 'Synthesis Lectures on Artificial Intelligence and Machine Learning', Morgan & Claypool Publishers.

Takahara, G. and Hall, J. (2017), Stochastic Processes STAT 455, Set3, Queens University at Kingston, ON, Canada, http://www.mast.queensu.ca/~stat455/lecturenotes/set3.pdf, [visited 9-Sept-2018].

Tesauro, G. J. (1994), 'TD-gammon, a self-teaching backgammon program, achieves master-level play', *Neural Computation* **6**(2),215–219.

Tesauro, G. J. and Galperin, G. R. (1997). On-line policy improvement using Monte-Carlo search. In *Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference*, Cambridge, MA. MIT Press.

Thorndike, E. L. (1911), *Animal Intelligence*. Hafner, Darien, Conn.

Walpole, R. E. (1982) Introduction of Statistics. Third Edition, Macmillan Publishing Company, Inc., New York.

Wang, Z., (2004), *Car Simulation Using Reinforcement Learning*, Department of Computer Science University of British Colombia.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England.

Weatherwax, J. (2005), Code and Results for Chapter 6, http://waxworksmath.com/Authors/N_Z/Sutton/WWW/chapter_6.html

Widrow, B. and Hoff, M. E. (1960). *Adaptive switching circuits*. In 1960 WESCON Convention Record Part IV, pages 96-104. Reprinted in J. A. Anderson and E. Rosenfeld, Neurocomputing: Foundations of Research, MIT Press, Cambridge, MA, 1988.

Widrow, B., Gupta, N. K., and Maitra, S. (1973). *Punish/reward: Learning with a critic in adaptive threshold systems.* IEEE Transactions on Systems, Man, and Cybernetics, 5:455-465.

Wolfram|Alpha (2009). retrieved February 25, 2015, from http://www.wolframalpha.com/ with query y'(t) -Dsin(y(t) + G) –C = 0, y(0) = p

Appendix 0 Experiments to derive the best parameters for the %optimal action for selection rules

Experiments were carried out to compute the %optimal action for several paramter values for each selction rule with X=1000 and T=1000 and the results are presented below. The best parameters are those which maximise the %optimal action at T=1000 and are marked as (*):

Selection Rule	Parmeters	Values						
ϵ-greedy	ϵ =	0	0.1*	0.2	0.4	0.6	1.0	
	%optimal	35	81*	74	59	43	10	
softmax	τ =	0.0	0.1*	0.5	1.0	1.5	4.0	9.9
	%optimal	11	57*	53	31	24	14	11
Reinforcement comparison	α =	0.1*	0.2	0.3				
	β = 0.01*	86*	83	84				
	β = 0.05	70	70	68				
	β = 0.30	41	35	51				
Pursuit selection	β =	0.0	0.01*	0.1	0.2	0.4	0.6	0.8
	%optimal	10	91*	58	45	38	36	36
UCB	%optimal		91					
Constant α, ϵ = 0.1(best)	α =	0.0	0.1*	0.2	0.4	0.6	0.8	1.0
	%optimal	10	77*	72	64	59	53	46

Appendix 1 Implementation of Sharma's (2008) lower bound to the sample variance

Below is shown an implementation Sharma's (2008) lower bound to the sample variance and empirically determine that at 5 cases the bound is stable. Let $\{y_i\}$ be a sample of N RVs having variance σ^2_Y then the arithmetic mean A and harmonic mean H provide bounds for σ^2_Y.

$\sigma^2_Y \leq y_{max}(A - H)(y_{max} - A)/ (y_{max} - H)$

$\sigma^2_Y \geq y_{min}(A - H)(A - y_{min})/ (H - y_{min})$, where

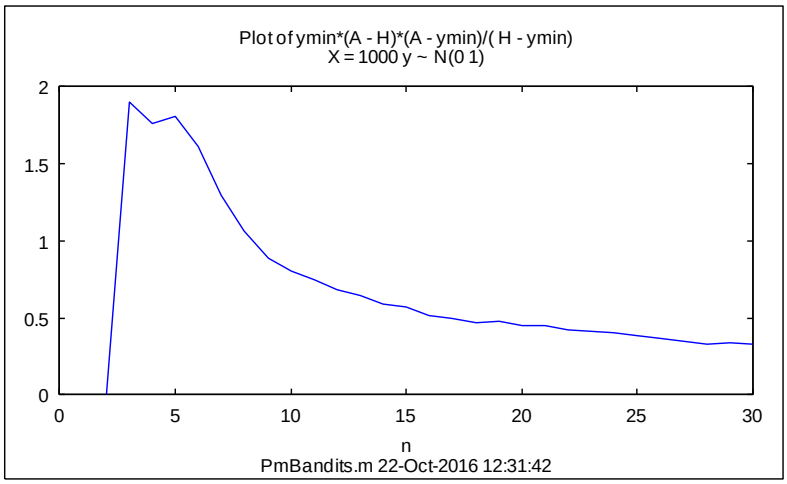
$A \stackrel{def}{=} (1/N)\Sigma_i y_i$

$H \stackrel{def}{=} N/\Sigma_i 1/y_i$

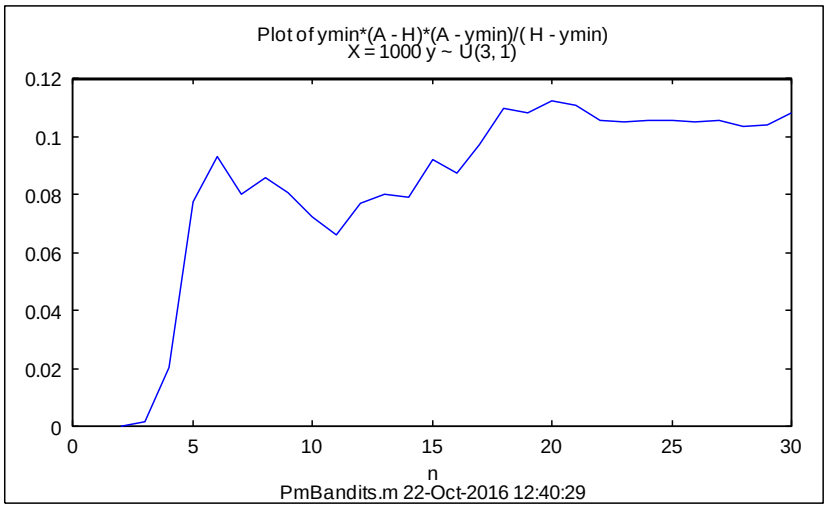
```
#Ex Bandits 3 Use of SHARMA's lower bound to st2
N=30
x=1000
sigma=1.0
Y=rand(N, 1) +3;
Y=[0.1+3, 0.11+3 , Y']
for x=1:X
 for n=1:N
  y=Y(1:n);
  ymin=min(y);
  ymax=max(y);
  A=mean(y);
  yi = 1.0./y;
  H = n/sum(yi);
  VLB = ymin*(A - H)*(A - ymin)/( H - ymin);
  VLBav(n)= VLBav(n) + (1/n)*(VLB -VLBav(n)); #Accumulate averages
 endfor #n
endfor #x
plot(VLBav)
title (['Plot of ymin*(A - H)*(A - ymin)/( H - ymin)', "\n",…
 'X = ', num2str(X), ' y \~ U(3, 1)']);
xlabel(['n', "\n", "PmBandits.m ",datestr(now()) ]);
```

```
return
```

Below the averaged variance lower bound for a standard Normal distribution with modest assumptions for ymin is plotted.



Below the averaged variance lower bound for a standard uniform distribution is plotted



Again there is a sharp rise and subsequent plateau or decline from n=5. This will support a forced round robin exploration for n=5.

Appendix 2 Equation of motion for car and van

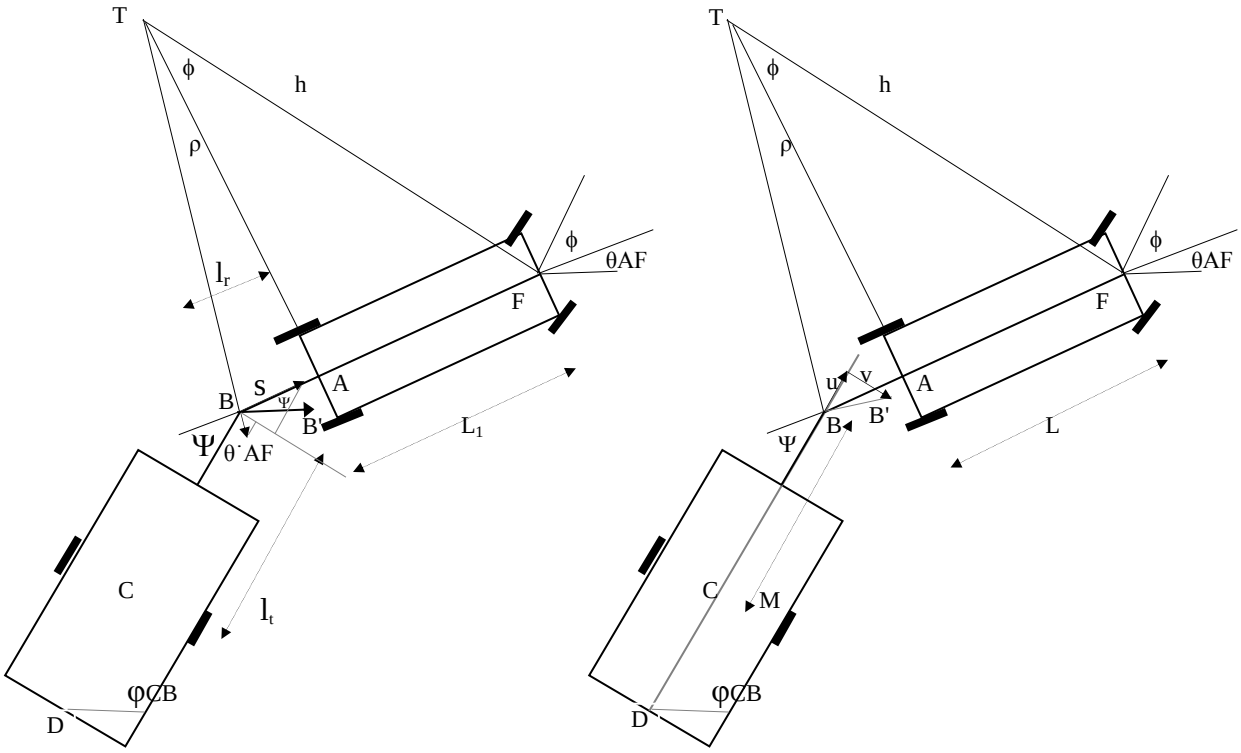


Figure A3 1.0 Jayakaran left, thesis right

This appendix presents the derivation of the equation of motion for the car-van angle using a canonical diagram from the dynamic approach of Jayakaran (2004) and the geometric approach in this thesis (see section 4.3) respectively. In Figure A3 1.0 all the angles are positive $\theta_{AF} = \angle AF > 0$, $\varphi_{CB} = \angle CB > 0$, $\Psi > 0$ $\phi > 0$. The car is represented by vector AF and the trailer by CB, $s > 0$

$\phi > 0$ Steer to the left

$s > 0$ The speed under A, positive means forwards.

$q > 0$ The speed under wheels at F

Measuring anticlockwise wise $\theta_{AF} + \Psi = \varphi_{CB}$

Distance travelled by F is $|FF'| = h\delta\theta = q\delta t$ where q is the speed of F. From the diagram

(A2.0.1) $h\sin\phi = L$, if $q > 0$ then $h > 0$

(A2.0.2) $\dot{\theta}_{FA} = \lim_{\delta t \to 0} \delta\theta_{FA}/\delta t = q/h = (q/L)\sin\phi$, if $\phi > 0$ then $\dot{\theta} > 0$ i.e. anti clockwise

The equations of motion of A from LaValle follow $\delta y / \delta x = \tan\theta_{FA}$ giving Pfaffian constraint (A2.0.2)

(A2.0.3) $\dot{x}\sin\theta_{FA} + \dot{y}\cos\theta_{FA} = 0$ which has solution $\dot{x} = s\cos\theta_{FA}$, $\dot{y} = s\sin\theta_{FA}$,where

$s = \rho\dot{\theta}_{FA} = L/\tan\phi\ (q/L)\sin\phi = q\cos\phi$

A2.1 Dynamics of p$\underline{A}$

p$\underline{A}$= [x; y] = [cos θ_{FA} ; sin θ_{FA}]

(A2.1.1) p$\dot{\underline{A}}$ = [$\dot{x}$; $\dot{y}$] = [q cos ϕ cos θ_{FA} ; q cos ϕ sin θ_{FA}], using (A2.0.3)

$\underline{pB}$ = pA + b$\underline{\hat{A}B}$

$\underline{pB}$ depends on pA and θ_{FA}

$p\dot{\underline{B}}$ = $\dot{x}$d/dx ($\underline{pA}$ +b$\underline{\hat{A}B}$) + $\dot{y}$ d/dy ($\underline{pA}$ +b$\underline{\hat{A}B}$) + $\dot{\theta}_{FA}$d/dθ_{FA} ($\underline{pA}$ +b$\underline{\hat{A}B}$), now $\underline{\hat{A}B}$ = −$\underline{\hat{A}F}$

$p\dot{\underline{B}}$ = p$\dot{\underline{A}}$ + (q/L)sin ϕ d/dθ_{FA} ($\underline{pA}$ +b$\underline{\hat{A}B}$) = p$\dot{\underline{A}}$ + (q/L)sin ϕ d/dθ_{FA} ([-cos θ_{FA} ; -sin θ_{FA}])

(A2.2.1) $p\dot{\underline{B}}$ = p$\dot{\underline{A}}$ + (q/L)sin ϕ [sin θ_{FA} ; -cos θ_{FA}]

For θ_{AF} >0 then the second turn causes a further x component but a negative y component

A2.2 Dynamics of pB

The expression for BB' and hence A2.2.1 can be derived

B' = $R_{\delta\theta_{FA}}$ (B –A) +A + ΔA = $R_{\delta\theta_{FA}}$ (B – A) + A +$\underline{AB}$ - $\underline{AB}$ + ΔA = $R_{\delta\theta_{FA}}\underline{AB}$ + B - $\underline{AB}$ + ΔA

Let B' ≡ B + ΔB and ΔB = ($R_{\delta\theta_{FA}}$– I)$\underline{AB}$ + ΔA then

$pB' - pB = \delta pB = (R_{\delta\theta_{FA}} - I)b\underline{AB} + \Delta A$, $R_{\delta\theta_{FA}} = \begin{bmatrix} \cos(\delta\theta_{FA}) & -\sin(\delta\theta_{FA}) \\ \sin(\delta\theta_{FA}) & \cos(\delta\theta_{FA}) \end{bmatrix}$

$d(R_{\delta\theta_{FA}} - I)/dt = d\begin{bmatrix} \cos(\delta\theta_{FA})-1 & -\sin(\delta\theta_{FA}) \\ \sin(\delta\theta_{FA}) & \cos(\delta\theta_{FA})-1 \end{bmatrix}/dt = d\begin{bmatrix} \cos(\delta\theta_{FA})-1 & -\sin(\delta\theta_{FA}) \\ \sin(\delta\theta_{FA}) & \cos(\delta\theta_{FA})-1 \end{bmatrix}/d\theta_{FA}$

$d\theta_{FA}/dt$

$$\begin{bmatrix} -\sin(\delta\theta_{FA}) & -\cos(\delta\theta_{FA}) \\ \cos(\delta\theta_{FA}) & -\sin(\delta\theta_{FA}) \end{bmatrix}\dot{\theta}_{FA}$$

$p\dot{\underline{B}} = \lim_{\delta t\to 0} \delta B/\delta t = \lim_{\delta t\to 0} (R_{\delta\theta_{FA}} - I)b\underline{AB}/\delta t + \lim_{\delta t\to 0} \Delta A/\delta t = \lim_{\delta t\to 0} (R_{\delta\theta_{FA}} - I)b\underline{AB}/\delta t$ +p$\dot{\underline{A}}$

$$p\dot{\underline{B}} = d(R_{\Delta\theta_{FA}} - I)b\underline{AB}/d_{\Delta\theta_{FA}} \lim_{\delta t\to 0} \Delta\theta_{FA}/\delta t + p\dot{\underline{A}} = \begin{bmatrix} -\sin(\delta\theta_{FA}) & -\cos(\delta\theta_{FA}) \\ \cos(\delta\theta_{FA}) & -\sin(\delta\theta_{FA}) \end{bmatrix} b\underline{AB}\,\dot{\theta}_{FA}$$

+p$\dot{\underline{A}}$

$p\dot{\underline{B}}$ = b[-sin($\delta\theta_{FA}$)ABx –cos($\delta\theta_{FA}$)ABy; cos($\delta\theta_{FA}$)ABx –sin($\delta\theta_{FA}$)ABy]$p\dot{\underline{B}}$ + p$\dot{\underline{A}}$

$p\dot{\underline{B}}$ = b $\dot{\theta}_{FA}$ [-sin($\delta\theta_{FA}$)(-cos(θ_{FA}))–cos($\delta\theta_{FA}$)sin(θ_{FA});]=b $\dot{\theta}_{FA}$ [-sin(θ_{FA}+$\delta\theta_{FA}$); cos(θ_{FA}+$\delta\theta_{FA}$)] + p$\dot{\underline{A}}$

$\underline{p\dot{B}}$ = b $\dot{\theta}_{FA}$ [sin($\delta\theta_{FA}$)(cos(θ_{FA})) – cos($\delta\theta_{FA}$)sin(θ_{FA});]=b $\dot{\theta}_{FA}$ [sin(θ_{FA}+$\delta\theta_{FA}$); -cos(θ_{FA}+$\delta\theta_{FA}$)] + p$\underline{\dot{A}}$

Now $\lim_{\delta t\to 0}\ \Delta\theta_{FA} = \lim_{\delta t\to 0}\ \theta_{FA} + \delta t\,\dot{\theta}_{FA} = \theta_{FA}$

(A2.3.1) $\underline{p\dot{B}}$ = $\lim_{\delta t\to 0}$ b $\dot{\theta}_{FA}$ [sin(θ_{FA}+$\delta\theta_{FA}$); -cos(θ_{FA}+$\delta\theta_{FA}$)] = b $\dot{\theta}_{FA}$ [sin(θ_{FA}); -cos(θ_{FA})] + p$\underline{\dot{A}}$

A2.3 Derivation of BB' and p$\dot{B}$

BB' is now projected in components // to CB u and left handed ⊥ to CB (Turn by +90)
Using the frame rotation RF:

$$\begin{bmatrix} u \\ v \end{bmatrix} = RF(\varphi CB)\ p\dot{B} = \begin{bmatrix} \cos(\varphi CB) & \sin(\varphi CB) \\ -\sin(\varphi CB) & \cos(\varphi CB) \end{bmatrix} \begin{bmatrix} B\dot{x} \\ B\dot{y} \end{bmatrix} = \begin{bmatrix} \cos(\varphi CB) + \sin(\varphi CB) \\ -\sin(\varphi CB) + \cos(\varphi CB) \end{bmatrix} \begin{bmatrix} \sin(\theta FA) \\ -\cos(\theta FA) \end{bmatrix} b\dot{\theta}FA$$
+RF(φCB)p$\underline{\dot{A}}$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos(\varphi CB)\sin(\theta FA) - \cos(\theta FA)\sin(\varphi CB) \\ -\sin(\varphi CB)\sin(\theta FA) - \cos(\theta FA)\cos(\varphi CB) \end{bmatrix} b\dot{\theta}FA = \begin{bmatrix} \sin(\theta FA - \varphi CB) \\ -\cos(\theta FA - \varphi CB) \end{bmatrix} b\dot{\theta}FA + RF(\varphi CB)p\underline{\dot{A}}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sin(\theta FA - \varphi CB) \\ -\cos(\theta FA - \varphi CB) \end{bmatrix} b\dot{\theta}FA + \begin{bmatrix} \cos(\varphi CB) & \sin(\varphi CB) \\ -\sin(\varphi CB) & \cos(\varphi CB) \end{bmatrix} \begin{bmatrix} \cos(\theta FA) \\ \sin(\theta FA) \end{bmatrix} q\cos\phi$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sin(\theta FA - \varphi CB) \\ -\cos(\theta FA - \varphi CB) \end{bmatrix} b\dot{\theta}FA + \begin{bmatrix} \cos(\varphi CB)\cos(\theta FA) + \sin(\varphi CB)\sin(\theta FA) \\ -\sin(\varphi CB)\cos(\theta FA) + \cos(\varphi CB)\sin(\theta FA) \end{bmatrix} q\cos\phi$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sin(-\Psi) \\ -\cos(-\Psi) \end{bmatrix} b\dot{\theta}FA + \begin{bmatrix} \cos(-\Psi) \\ \sin(-\Psi) \end{bmatrix} q\cos\phi$$

(A2.4.1) $$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sin(\Psi)b\dot{\theta}FA + q\cos\phi\cos(\Psi) \\ -\cos(\Psi)b\dot{\theta}FA - q\cos\phi\sin(\Psi) \end{bmatrix}$$

A2.5 Dynamics of φ and Ψ

Now sin $\delta\varphi_{BC}$ =v/M. Since v < 0 for values on the diagram δφ is <0 i.e. clockwise.

$\dot{\varphi}_{BC}$ = $\lim_{\delta t\to 0}\delta\varphi_{BC}/\delta t = \lim_{\delta t\to 0}\sin^{-1}(v/M)/\delta t = d(\sin^{-1}(v/M)/d(v/M))\lim_{\Delta t\to 0}(v/M)/\delta t$

$=\lim \delta t_{\to 0}\ 1/\sqrt{(1-(v/M)^2)}(1/M)\ (v/\delta t)$

$\lim_{\delta t\to 0}(v) = \lim_{\delta t\to 0}[-\cos(\Psi)b\dot{\theta}FA - q\cos\phi\sin(\Psi)]\delta t = 0$

$\lim_{\delta t\to 0}(v/M)\delta t = \lim_{\delta t\to 0}[-\cos(\Psi)b\dot{\theta}FA - q\cos\phi\sin(\Psi)]\delta t/M$

$=[-\cos(\Psi)b\dot{\theta}FA - q\cos\phi\sin(\Psi)]/M$

Now Ψ = φCB – θAF

$\dot{\Psi} = \dot{\varphi}_{CB} - \dot{\theta}_{AF} = -\cos(\Psi) b\dot{\theta}FA/M - q\cos\phi\sin(\Psi)/M - (q/L)\sin\phi$

(A2.5.1) $\dot{\Psi} = -(bq/L/M)\sin\phi\cos(\Psi) - q/M\cos\phi\sin(\Psi) - (q/L)\sin\phi$

A2.6 Jayakaran approach

Jayakaran's specification is

The steering angle $\phi > 0$, and this causes a steer to the left.

The vehicle wheel base is L (L_1).

The length of off-hook is b (l_1).

Orientation of the vehicle with the x axis θAF.

Angle between the vehicle and the trailer Ψ.

Turning radius ρ.

Instance centre T.

Speed under A s.

Angular velocity about T $\dot{\theta}_{FA} = s/\rho = s\tan\phi/L = q\cos\phi\tan\phi/L = q/L\sin\phi$.

$\varphi_{CB} = \theta_{AF} + \Psi$ from the geometry.

(A2.6.1) $\dot{\varphi}_{CB} = \dot{\theta}_{AF} + \dot{\Psi}$.

The velocity at B is the sum of the forward velocity s and that tangent to the arc traced out by the off hook having angular velocity at A, $b\dot{\theta}_{AF}$ and which is perpendicular to AF. In order to preserve the direction convention the perpendicular projection on a vector has positive value for the anticlockwise side. The first projects a component $b\dot{\theta}_{AF}\cos(\Psi)$ on negative perpendicular to the trailer and similarly the second a component $s\sin(\Psi)$. Added together an expression for $\dot{\varphi}_{CB}$ is:

$\dot{\varphi}_{CB} = -1/M(b\dot{\theta}_{AF}\cos(\Psi) + s\sin(\Psi))$.

The diagram confirms that both components of the velocity of B cause a clockwise turn to the trailer. With (A2.6.1) an expression for $\dot{\Psi}$ is obtained:

(A2.6.2) $\dot{\Psi} = \dot{\varphi}_{CB} - \dot{\theta}_{AF} = -b/Mq/L\sin\phi\cos(\Psi) - q/M\cos\phi\sin(\Psi). -q/L\sin\phi$.

A2.7 An exact solution for Ψ

Re stating (A2.6.2) and introducing constants A, B C and D:

$\dot{\Psi} = A\sin(\Psi) + B\cos(\Psi) + C$

$A = -(q/M)\cos\phi$

$B = -(b/L)(q/M)\sin\phi$

$C = -q\sin\phi$

Let $D \stackrel{\text{def}}{=} \sqrt{(A^2 + B^2)}$ solving for that $\gamma \in [0, 360]$ such that $\cos\gamma = A/D$, $\sin\gamma = B/D$ gives an ODE (A2.7.1)

$\dot{\Psi} = D(A/D \sin(\Psi) + B/D \cos(\Psi)) + C$

$\dot{\Psi} = D\cos\gamma\sin(\Psi) + D\sin\gamma\cos(\Psi) + C$

(A2.7.1) $\dot{\Psi} = D \sin(\Psi + \gamma) + C, \Psi(0) = \Psi_0$

Recasting into the ODE format: $\dot{\Psi} - D \sin(\Psi + \gamma) - C = 0, \Psi(0)=p$ and inserting $y'(t) - D\sin(y(t) + G) - C = 0, y(0) = p$ into Wolfram|Alpha (2009) the formal solution is:

$$y(t) = 2\tan^{-1}\left(\frac{1}{C}\left(\sqrt{(C-D)(C+D)}\tan\left(\frac{1}{2}\left(\frac{2\sqrt{(C-D)(C+D)}\,\tan^{-1}\left(\frac{C\sqrt{(C-D)(C+D)}\,\tan\left(\frac{G+p}{2}\right)+D\sqrt{(C-D)(C+D)}}{C^2-D^2}\right)}{\sqrt{C^2-D^2}} + t\sqrt{(C-D)(C+D)}\right) - D\right) - G\right)\right)$$

The dynamics of pA, pB, pḂ, φ have been presented. An ODE formulation for $\dot{\Psi}$ is written out and a formal solution for Ψ over time obtained using Wolfram|Alpha.

Appendix 3 Graph of objectives and section headings