STOCHASTIC OPTIMISATION OF VEHICLE SUSPENSION CONTROL SYSTEMS VIA LEARNING AUTOMATA

by

Geoff Frost

A Doctoral Thesis

Submitted in partial fulfilment of the requirements
for the award of

Doctor of Philosophy of Loughborough University

October 1998

Abstract

This thesis considers the optimisation of vehicle suspension systems via a reinforcement learning technique The aim is to assess the potential of learning automata to learn 'optimum' control of suspension systems, which contain some active element under electronic control, without recourse to system models. Control optimisation tasks on full-active and semi-active suspension systems are used for the feasibility assessment and subsequent development of the learning automata technique.

The quarter-vehicle simulation model, with ideal full-active suspension actuation, provides a well-known environment for initial studies applying classical discrete learning automata to learn the controller gains of a linear state-feedback controller. Learning automata are shown to be capable of acquiring near optimal controllers without any explicit knowledge of the suspension environment. However, the methodology has to be developed to allow safe on-line application. A moderator is introduced to prevent excessive suspension deviations as a result of possible unstable control actions applied during learning. A hardware trial is successfully implemented on a test vehicle fitted with semi-active suspension, excited by a hydraulic road simulation rig.

During these initial studies some inherent weaknesses of the discrete automata are noted A discrete action set provides insufficient coverage of a continuous controller space so optima may be overlooked. Subsequent methods to increase the resolution of search lead to a forced convergence and hence an increased likelihood of local optima location. This motivates the development of a new formulation of learning automaton, the CARLA, which exhibits a continuous action space and a reinforcement generalisation.

The new method is compared with discrete automata on various stochastic function optimisation case studies, demonstrating that the new functionality of CARLA overcomes many of the identified shortcomings of discrete automata. Furthermore, CARLA shows a potential capability to learn in non-stationary environments. Repeating the earlier suspension tasks with CARLA applied, including an on-line hardware study, further demonstrates a performance gain over discrete automata

Finally, a complex multi-goal learning task is considered A dynamic roll-control strategy is formulated based on the semi-active suspension hardware of the test vehicle. The CARLA is applied to the free parameters of this strategy and is seen to successfully synthesise improved roll-control over passive suspension.

Acknowledgements

This work was conducted in the Department of Aeronautical and Automotive Engineering and Transport Studies (AAETS) at Loughborough University. It is based on research carried out as part of an EPSRC funded project (GR/J82652).

The author gratefully acknowledges the support of the EPSRC and also wishes to thank the Ford Motor Company for the financial support throughout this project, and for the provision of a research vehicle and testing facilities.

The research was supervised by Prof. Tim Gordon and I would like to extend my thanks to him for his supervision, guidance and patience.

Finally I would like to thank my colleagues in the AAETS department, and in particular Dr. Mark Howell and Dr. Matt Best, for their assistance and support throughout my time there.

Contents

Chapter 1 - Introduction

This thesis considers the application of reinforcement learning techniques, and in particular the learning automaton, to the synthesis of control laws for advanced vehicle suspension systems. The major emphasis is placed on developing the learning automaton methodology as a tool for on-line parameter optimisation of complex systems in practical application. Vehicle suspension control presents a naturally complex task, with a stochastic driving input from the road and inherent non-linearity from component characteristics, multi-component interactions and geometry effects.

In this chapter, the nature of vehicle suspension systems is first discussed, outlining the advances made in this area that make possible significant improvement in vehicle ride and handling characteristics. Reinforcement learning is identified as a prime method by which the performance improvements may be more rapidly accomplished during the vehicle design optimisation process. The general methodology and terminology of one suitable reinforcement learning technique, the learning automaton, is introduced with a discussion of its history and limited practical application to date A statement of the objective of this thesis is then given.

1.1 Overview

The majority of modern motor vehicles are fitted with a 'passive' suspension system, consisting of a spring and damper to control vertical forces, and linkages to control wheel angle and transmit cornering and braking forces as well as steering inputs from the driver. However, advances in transducers, microprocessors and force actuators have provided new opportunities in suspension design. Significant improvements in ride and handling performance are possible when the spring and damper arrangement of a conventional system is completely replaced by a hydraulic actuator, where the actuator is under full control of a microprocessor taking measurements from sensors to ascertain the current state of the vehicle Such a system is termed 'active'. Alternatively a 'semi-active' approach could be used, replacing the fixed characteristic damper in a passive system with a continuously variable damper The semi-active system also offers some potential for improved ride and handling performance, but requires no source of mechanical power, and is thus cheaper and simpler to implement than the corresponding active system; see for example Karnopp (1983) and Sharp & Crolla (1987).

The conventional approach to the development of suspension control systems involves extensive modelling, ranging from simple linear quarter car models, up to

sophisticated multibody simulation models based on computer codes such as ADAMS (see for example Kortum & Sharp, 1993). Even the simplest system will contain enough design parameters to make any model difficult to match accurately to hardware performance. Applying any form of active system will introduce even more variables and the design issues will now include synthesis of suitable control laws. Inevitably, more variables lead to more complexity and hence to longer development times Despite this design development time, an extensive period is also spent on prototype vehicles where suspension characteristics are subjectively tuned by a process of iterative development with experienced test drivers

Clearly it is attractive to consider techniques whereby the resources expended in doing such detailed modelling and subjective tuning could be reduced. Machine learning is one such technique that offers a capability of 'learning' how to achieve a pre-defined objective on-line with little, if any, prior knowledge of the system concerned. In conjunction with computer controlled active suspension systems, there is therefore the potential for a learning agent to acquire a control law on-line. This is in contrast to traditional methods of control law design that require system models for their synthesis off-line. As noted above, any model is an inevitable simplification of the real system, which then leads to the subjective tuning of the on-line system.

Current machine learning paradigms can be classified into three areas according to the form of feedback they receive from the world in response to their previous actions·

- *Learning by observation - unsupervised learning/learning by discovery* - Here there is no direct input available concerning the focus of the learning system The learner has no knowledge of positive/negative instances or which directions to follow in search of better descriptions. Instead the learner is only able to collect observations and derive generalised concepts according to its own internal rules. With no external guidance and no feedback unsupervised learning is inefficient and often only useful in specialised applications One example of unsupervised learning is a Kohonen net, originating from the image processing research field, which self-organises its internal state, in response to input observations alone, such the 'net' forms a representation of the shape described by the input values – see Kohonen (1984)

- *Learning from examples - supervised learning/learning with a teacher* - The learning algorithm is trained on a set of matched input/output data. The aim of the learner is to generate a data description that matches inputs from the training set with the correct output whilst also generalising the information effectively to

produce acceptable outputs for future unseen inputs. This method requires a very large amount of external guidance and the training depends upon a well-informed teacher with comprehensive and good quality data. Neural networks are a classic form of supervised learning technique, where data is input to the network. Required outputs for the given inputs are known and the error between those outputs and the network outputs is used in feedback training, suitably adjusting the weights of the network to reduce future error.

- *Reinforcement learning - learning with a critic* - Here the learner only receives a scalar signal as feedback. This reinforcement signal provides evaluation of performance with respect to pre-set goals No direct error information on the internal representation of the learning algorithm is given. Only the evaluative response to an action is provided as feedback on its performance.

It is this last learning category which is most applicable in a vehicle suspension setting.

The ability for a reinforcement learning algorithm to learn from only evaluative feedback removes any requirement for *a priori* knowledge of the agent's working environment Complex and noisy environments in which analytical study becomes excessively slow or intractable can be tackled by reinforcement learning techniques where an evaluative signal is available.

Early work in the area of reinforcement learning developed a stochastic automaton, able to work in an environment to 'learn' the best action from a set of possible actions it could apply. The stochastic automaton would initially randomly choose actions to test in the environment. For each test, the automaton would receive back from the environment a performance index, indicating how successful that action had been The automaton would update its internal state to increase the chance that successful actions would be tested again, while unsuccessful actions would be penalised. Successive trial/reward iterations lead to the automaton predominantly choosing only successful actions, and hence the overall performance of the system had been improved

In applying such an automaton to a vehicle suspension system, the environment can be identified as the whole vehicle hardware system including any interactions between vehicle, road and driver. The action an automaton may apply to a suspension system is a control law. An automaton can have a set of control laws from which the 'best' law in the given environment is to be identified. The evaluative feedback can be formed from sensor measurements Suitable choice of a cost function can give a relative measure of the performance of any action in comparison with previous action trials.

1.2 Advanced Vehicle Suspensions

Use of active and semi-active systems, under microprocessor control, considerably increases the control possibilities above those offered by passive systems. In this dissertation, attention will be restricted to optimisation of the ride function via such suspension systems, whereby road irregularities are filtered out and a comfortable environment is provided for the vehicle occupants

The stochastic nature of the road input suggests the application of linear optimal control theory for finding good control laws for active systems, and this method has been widely used (Wilson et al. 1986, Thompson 1984) However, the theory is based upon the assumptions that the suspension system is linear, and the stochastic input is white noise. Although such assumptions are reasonable to investigate the possibilities offered by active systems, any real system will be inherently non-linear because of practical considerations in component design, component interactions and suspension geometry. Also, investigation of the nature of road input has shown that white noise is a poor approximation (Robson 1979).

Semi-active suspension systems have been investigated as economical alternatives to full active systems. By replacing the fixed characteristic damper in a passive system with a variable characteristic actuator, many of the beneficial control aspects of full active systems can be maintained without the expense of providing an energy source (Karnopp et al., 1974); using a variable damper the semi-active system is purely dissipative with only small energy requirements for valve control within such an actuator. For these reasons semi-active suspension systems are attracting particular attention as an affordable improvement to passenger vehicle suspension refinement

Control law design for the semi-active system generally follows on from that of the full active system, with special consideration given to the constraints imposed (Karnopp, 1983). Synthesis of non-linear controllers has been undertaken by Gordon & Best (1994), but this relies on off-line optimisation techniques using a system model

On-line 'optimisation' of any suspension system has generally been limited to subjective development of passive systems in the final track testing of new vehicle designs, and is not reported in the literature beyond its principles (Sharp & Hassan, 1984)

1.3 Reinforcement Learning - Learning Automata

Early work in the area of automata models of learning originated in mathematical psychology. Consideration of the way in which animals appear to condition their response to various stimuli, and therefore 'learn' an improved response, led to a mathematical framework for the study of learning problems - (Bush & Mosteller 1958, Atkinson et al 1965, Iosifescu & Theodorescu 1969, Norman 1972)

Tsetlin (1961) introduced the concept of using deterministic automata operating in random environments as models of learning. Further work in the Soviet Union followed his ideas. An introduction to the Russian literature can be found in Tsetlin (1973).

Varshavskii & Vorontsova (1963) observed that the use of stochastic automata with updating action probabilities could reduce the number of states in comparison with deterministic automata. This idea was readily adopted and later research then focused on stochastic automata.

The general framework on which much of the learning automata development is based, is now given.

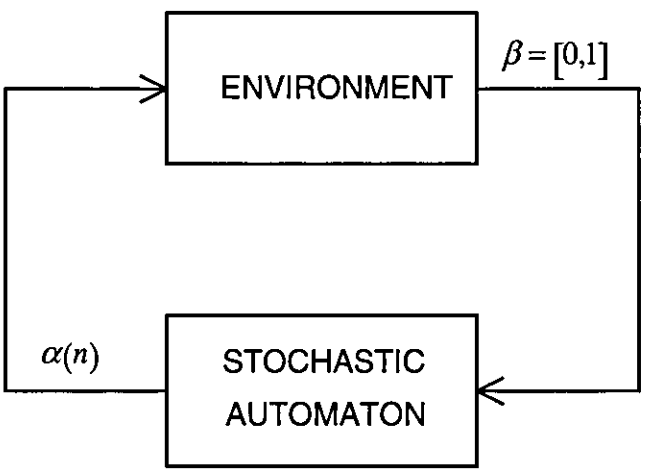


Figure 1.1 - Learning automaton

1.3.1 Stochastic automata

According to Narendra & Thathachar (1974), 'A learning automaton is a stochastic automaton that operates in a random environment and updates its action probabilities in accordance with the inputs received from the environment so as to improve its performance in some specified sense.' Figure 1.1 represents a learning automaton as defined above, with feedback connection of a stochastic automaton and a random environment The actions of the automaton form the inputs to the environment. The

responses of the environment in turn are the input to the automaton, which influence the internal state of the automaton.

A distinction between several models of learning automata, based on the nature of the input to the automaton, should be noted A model with a binary input set, e.g. $\{0,1\}$, is termed a P-model, whilst a model with an input set consisting of a finite number of values is called a Q-model. An automaton is of the S-model form if the input set lies in an interval $[0,1]$. Each of these models is seen as more or less appropriate for different applications.

The environment is assumed to exhibit random response characteristics. It has inputs (actions) $\alpha(n)=\{\alpha_1,...,\alpha_r\}$ and outputs (responses) belonging to a set β. The simplest response set to consider is binary, $\beta=\{0,1\}$, with zero signifying a reward response, one signifying a penalty response. The probability of selecting a penalty output response depends on the input and is denoted by c_i $(i=1,...,r)$. The c_i are called the penalty probabilities If the c_i do not depend on n, the discrete-time variable, the environment is said to be stationary, otherwise it is nonstationary. It is assumed that the c_i are unknown initially, as the problem would be trivial if they are known *a priori.*

A stochastic automaton is a sextuple $\{\beta,\phi,\alpha,p,A,G\}$ where β is the input set, $\phi=\{\phi_1,\phi_2,...,\phi_s\}$ is the set of internal states, $\alpha=\{\alpha_1,\alpha_2,\ ,\alpha_r\}$, with $r\le s$, is the output or action set, p is the state probability vector governing the choice of state at each stage (i.e. at each stage n, $p(n)=(p_1(n),p_2(n),...,p_s(n))^T$), A is an algorithm (also called an updating scheme or reinforcement scheme) which generates $p(n+1)$ from $p(n)$, and $G\ \phi\rightarrow\alpha$ is the output function. As the environment response is random, so the action probability vector is also random. G could be any function, but the majority of stochastic automata work assumes G to be deterministic and one-to-one (i e $r=s$, states and actions are regarded as synonymous)

The automaton models used throughout this thesis are based on the above stochastic automaton model, with $r=s$, and are referred to as learning automata from hereon.

1.3.2 Learning schemes

The basic operation carried out by a learning automaton is the updating of the action probabilities on the basis of the responses from the environment. The reinforcement scheme is thus central to the successful operation of a learning automaton. Much research in the late 1960s/early 1970s focused on the behaviour of learning automata

utilising various reinforcement schemes, and defined a number of terms relating to this:

The *average penalty* received by the automaton, conditional on the probability vector, is given by

$$\begin{aligned} M(n) &= E\{\beta(n)|p(n)\} \\ &= \sum_{i=1}^{r} p_i(n)\, c_i \end{aligned} \tag{1.1}$$

where n is the discrete time variable If no *a priori* information is available, and the actions are chosen with equal probability (i e. at random) the value of the average penalty is denoted by M_0

$$M_0 = \frac{c_1 + c_2 + \ldots + c_r}{r} \tag{1.2}$$

The use of the term learning automata can be justified if the average penalty is made less than M_0.

A learning automaton is called *expedient* if

$$\lim_{n\to\infty} \mathrm{E}[M(n)] < M_0 \tag{1.3}$$

It would be more desirable, however, if proper selection of actions could lead to minimisation of the average penalty. From (1.1) it can be seen that the minimum value of $M(n)$ is $\min_i\{c_i\}$.

A learning automaton is called *optimal* if

$$\lim_{n\to\infty} \mathrm{E}[M(n)] = c_m \tag{1.4}$$

where

$$c_m = \min_i\{c_i\}$$

Although optimality appears a very desirable property, implying that the action associated with the minimum penalty probability is chosen, asymptotically, with probability one, a slightly weaker condition is more beneficial in practice.

A learning automaton is called ε *-optimal* if

$$\lim_{n\to\infty} \mathrm{E}[M(n)] < c_m + \varepsilon \qquad (1\,5)$$

can be obtained for any arbitrary $\varepsilon > 0$ by suitable choice of parameters of the reinforcement scheme. ε-optimality implies that the performance of the automaton can be made as close to optimal as desired.

A learning automaton is called *absolutely expedient* if

$$\mathrm{E}[M(n+1)|p(n)] < M(n) \qquad (1.6)$$

for all n, all $p_j(n) \in (0,1)$ $(j = 1, ..,r)$ and all possible values of c_i $(i = 1,...,r)$. It is usually assumed the set $\{c_i\}$ has unique maximum and/or minimum elements

Using this framework of definitions to relate and compare different reinforcement schemes, a number of schemes have been proposed. If $p(n+1)$ is a linear function of the components of $p(n)$, the reinforcement scheme is said to be linear, otherwise it is non-linear. In general, when the action at discrete-time n is α_i, the reinforcement schemes take the form

$$\left.\begin{aligned} p_i(n+1) &= p_i(n) + \sum_{j\neq i} f_j(p(n)) \\ p_j(n+1) &= p_j(n) - f_j(p(n)), \quad (j \neq i) \end{aligned}\right\} \text{for reward}$$

$$\left.\begin{aligned} p_i(n+1) &= p_i(n) - \sum_{j\neq i} g_j(p(n)) \\ p_j(n+1) &= p_j(n) + g_j(p(n)), \quad (j \neq i) \end{aligned}\right\} \text{for penalty} \qquad (1.7)$$

where $f_j()$ and $g_j()$ are nonnegative continuous functions such that $p_k(n+1) \in (0,1)$, for all $k = 1,...r$.

The first scheme proposed, from the mathematical psychology background, is known as the linear reward-penalty (L_{R-P}) scheme where $f_j()$ and $g_j()$ are both nonnegative linear functions. Bush & Mosteller (1958) and Varshavskii & Vorontsova (1963) studied the L_{R-P} for a two state case, using the scheme

$$\left.\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) &= (1-a)\ p_j(n), \quad (j \neq i) \end{aligned}\right\} \text{for reward}$$

$$\left.\begin{aligned} p_i(n+1) &= (1-b).p_i(n) \\ p_j(n+1) &= b/(r-1) + (1-b).p_j(n), \quad (j \neq i) \end{aligned}\right\} \text{for penalty} \qquad (1\,8)$$

where $0<a<1$ and $b=a$. This work was extended to the multi-state case by McLaren (1966) and continued by Chandrasekaran & Shen (1968)

Another common scheme studied in the literature is the linear reward-inaction (L_{R-I}) scheme in which $f_j()$ is a nonnegative linear function and $g_j()=0$ The characteristic of this scheme is therefore that it ignores penalty inputs from the environment so that the action probabilities remain unchanged under these inputs. One such example of a L_{R-I} scheme is simply deduced from (1.8), hence

$$\left.\begin{aligned} p_i(n+1) &= p_i(n)+a\left[1-p_i(n)\right] \\ p_j(n+1) &= (1-a).p_j(n), \quad (j\neq i) \end{aligned}\right\} \text{ for reward}$$

$$\left.\begin{aligned} p_i(n+1) &= p_i(n) \\ p_j(n+1) &= p_j(n), \quad (j\neq i) \end{aligned}\right\} \text{ for penalty} \qquad (1.9)$$

Many other linear and non-linear schemes were conceived and investigated in the early 1970s Development of the theory and convergence proofs for these schemes are widespread throughout the literature, and many of these results are summarised and covered by Narendra & Thathachar (1974 and 1989), Lakshmivarahan (1981), Baba (1984) and Najim & Poznyak (1994).

In particular, expedient and optimal reinforcement schemes were compared by Viswanathan & Narendra (1972) in trying to ascertain which type of schemes are to be preferred in practical applications Three prominent schemes from the literature were compared: an expedient linear reward-penalty (L_{R-P}) scheme, an ε-optimal linear reward-inaction (L_{R-I}) scheme, and a "square law non-linearity" non-linear reward-penalty (N_{R-P}) scheme shown to be optimal under certain conditions and expedient otherwise. The schemes were compared on the basis of degree of expedience, speed of convergence, and variance associated with the convergence process. Extensive computer simulation of a ten-state problem (ten possible automaton actions, each with a pre-defined penalty probability) demonstrated that although the N_{R-P} scheme initially converged towards a solution faster than the other schemes, the maximum and final variance was greater, yielding a poorer final solution. The L_{R-I} scheme, as a special case of the considered L_{R-P} scheme, demonstrated the best quality of solution throughout, with faster convergence than the L_{R-P} scheme, and smaller measures of variance than either L_{R-P} or N_{R-P} schemes

1.3.3 Hierarchical automata

A hierarchical learning automata utilises a tree structure of simple learning automata, each with a small action set, to replace a single automaton that would require a much

larger action set. For example, Figure 1 2 shows a 3-level structure of automata, each with only three actions, that exhibits a 27 action set at the base level The automata in higher levels act only to activate an automata at the next lower level. The probability updating scheme for this hierarchical learning system needs only apply to 3 automata after each learning iteration; 9 probability values are updated, as opposed to 27 values in the equivalent single automaton. Thathachar & Ramakrishnan (1981) reported on a simulation study of a 7-level hierarchical system, with 2 actions per automata, and 128 output actions. Acting on an artificial random environment with pre-set penalty probabilities, a saving in convergence time compared to a single automaton system was demonstrated

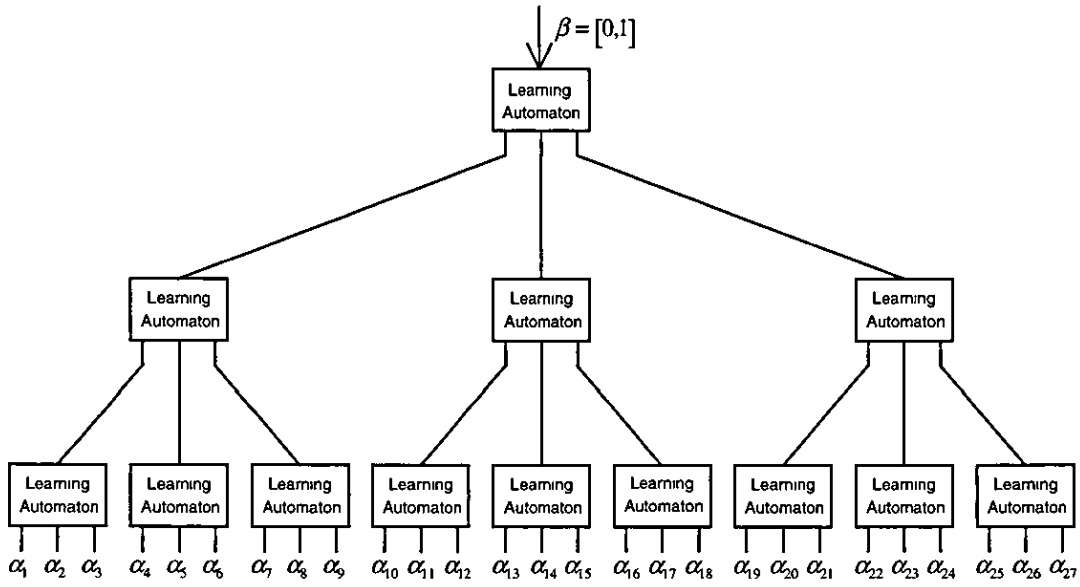


Figure 1.2 - Hierarchical learning automata

1.3.4 Interconnected automata

Thathachar & Ramakrishnan (1982) introduced an interconnected learning automata system. This uses a team of automata, and is best suited to high dimension problems caused by learning with a number of interacting action subsets, e.g. a multi-parameter optimisation problem Simultaneous actions from the individual automata form a single action applied to the environment. The environment replies with a single response, fed to all learning automata in the team. In this structure the automata are only linked together through the common environmental response - see Figure 1.3 - and otherwise have no knowledge of each other. A simple simulation study with two automata acting in an artificial random environment with pre-set penalty probabilities resulted in the convergence of an interconnected learning automata system.

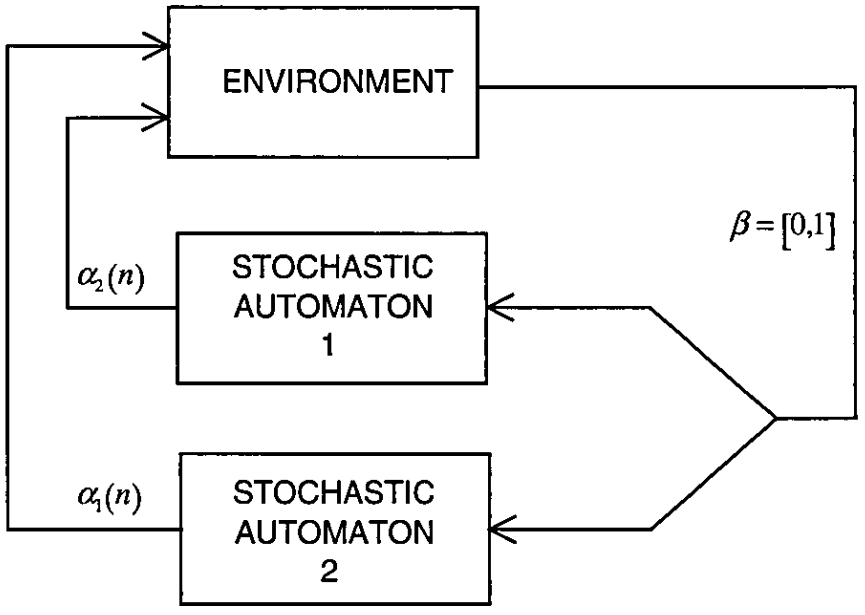


Figure 1.3 - Interconnected learning automata

1.3.5 Applications

The first non-trivial application of learning automata was in the optimisation field. McMurtry & Fu (1966) took a function of a single variable over a range containing a number of local minima with the aim of using a learning automaton to locate the global minima to within a subset of the initial range. Discretising the function into 10 distinct regions, it was shown that the learning automaton could distinguish the region containing the global minima, including cases where noise was added to all function evaluations carried out during the search

Shapiro & Narendra (1969) carried out similar experiments with P-model learning automata. Functions of a single variable, with local minima and additive noise (chosen to preclude the use of gradient type techniques such as stochastic approximation) were evaluated by the learning automaton at 10 discrete points within a pre-specified range. The aim of identifying the minimum function evaluation point, from the 10 points, was successfully demonstrated, including a case where the evaluated function was 'flat' in comparison to large-scale additive noise

Shapiro & Narendra also considered the problem of locating the optimal values for two parameters in an adaptive filter. The parameter space was discretised to 25 points, including the pre-known optimal point. Convergence to the optimal parameter values was demonstrated, but raised the question of how to deal with higher dimensional problems Two parameters, each discretised to just 5 values, gives a 25 point action set. One extra parameter, also discretised to 5 values, raises the action set to 125 points (5^3). Larger action sets, formed from high dimension problems or fine

discretisation of parameters, were noted as increasing convergence times in locating optimal actions. Hierarchical learning automata and interconnected learning automata were later proposed by Thathachar & Ramakrishnan (1981 and 1982) as possible aids to handling this dimensionality problem

Many early simulation studies run during development of the learning automata theory demonstrated their optimisation capabilities in stochastic environments, and yet little in the way of real world applicaton of this technique has since been reported. The earliest documented use of this technique, by Narendra & Mars (1983) considered a telephone traffic routing methodology, demonstrating likely efficiency gains to be had in employing learning automata as dynamic routers at nodes of a communications network.

Najim et al. (1990) applied learning automata to locate the optima and near-optima of a multi-modal function with constraints imposed. The technique was applied to find the optimal settings for a chemical process based on *a priori* deterministic knowledge of the cost per year of each possible setting.

Tang & Mars (1991 and 1993) used single automaton and interconnected automata strategies to learn multiple parameters of an adaptive IIR filter. Effectively an on-line optimisation problem, previous studies had shown multi-modal error surfaces would be present if the IIR filter was under or over parameterised in relation to the unknown system it was aiming to match. The global minima of the error surfaces was shown to be attainable through use of the learning automata methodology, as opposed to gradient search techniques which were prone to falling into local optima

Several authors have investigated applying a learning automata technique as a single parameter self-adjusting controller for a physical process. Najim (1991) considered the control of concentration of CO_2 in a chemical absorption column. A learning automaton was applied to adjust the absorbent flow rate into the column, thus controlling the CO_2 concentration to track a set-point concentration profile. The behaviour of the absorption column is influenced by random variation of various system parameters. A simulation study illustrated improved performance compared to a Ziegler-Nichols tuned PID controller. Valenzuela (1993) applied a hierarchical automata system to optimise the ore feed rate input to an autogenous grinding circuit in a mineral process. A dynamic simulator modelled the deterministic and stochastic behaviour of the grinding process The learning system was shown to converge to an optimal state with probability one, with output mass flow rate within the operating envelope required. Chidambaram (1994) used a single automaton in a similar manner

to learn the optimal dilution rate of a continuous stirred tank reactor. A deterministic model returned a binary response to the learning automata dependent on the relationship of output to a set-point. The automaton converged to a final value for the manipulated variable with probability one, exhibiting improved settling time characteristics during the learning process in comparison with a PI controller.

In all the above simulation studies, learning automata have dealt with optimisation of only one parameter, avoiding problems of high dimensionality. Discretisation of the parameter to only a small number of values leaves the possibility of the actual optima for any of the processes simulated lying in an interval between discrete points Also the processes all exhibit long time constants, so time required for computation of the learning automata is insignificant compared to response characteristics of the process.

Wu (1993) developed an extension to the learning automata methodology to enable automata to locate an optimum over a continuous area of the action space, even if that optima does not coincide with an action from the initial discrete action set, as may well be the case in many realistic applications. Wu proposed that automata may 'home in' to a smaller area of the action space once an action probability begins to dominate. Since the automaton should locate the minima of the discrete action set, it is reasonable to assume the action lies in a region of the action space close to an actual minimum of the continuous space. By suitable selection of a smaller area of the action space, centred on the previously located minima, a requantised action set with a reset automaton may be applied to search for a better defined minima - see Figure 1.4 Repeated reduction of the area of action space considered should lead, ultimately, to location of the optimum point, or something very close.

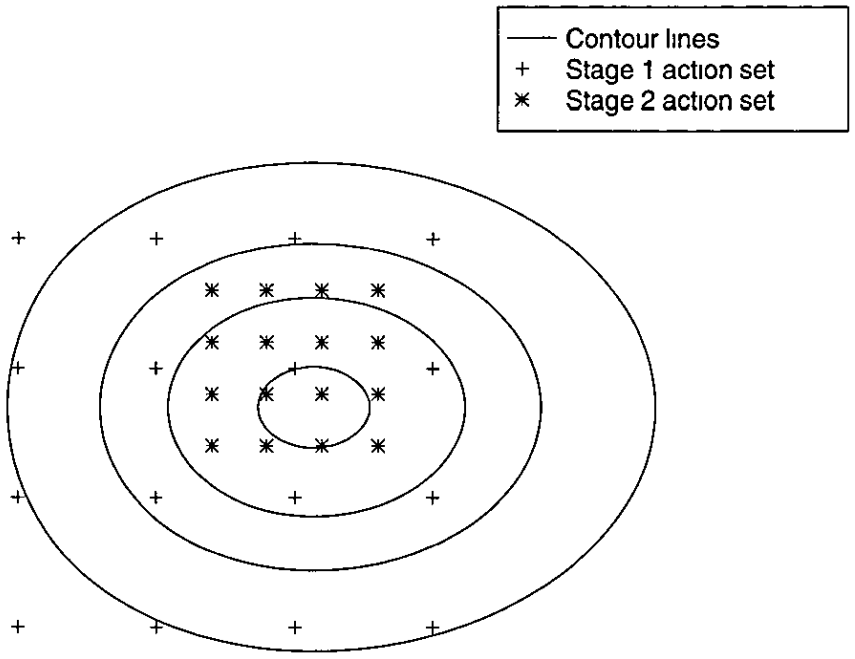


Figure 1.4 - Discrete action space reduction to locate global minima

Wu & Pugh (1993) applied the same technique to optimise a simulated turbogenerator power system controller. A single automaton was employed to learn three parameters of the pre-defined control function. Three stages of learning reduced the action space search area to 16% of the initial area Analysis of the average performance index shows the learning automata improving its performance at each stage. The final control values provided good control performance. Although only a simulation study, this work showed the potential for real-time on-line tuning of complex industrial systems in stochastic environments, without the need for system modelling. The action set requantisation methodology allows a more complete search of the parameter space, but in so doing also forces convergence. A poor action choice at an early stage may force the automaton away from the optimum in subsequent stages The reductionist nature of the quantisation scheme means recovery of the situation is not always possible and the methodology becomes inherently non-adaptive to any subsequent environmental variation.

One study, by Santharam et al (1994) has attempted to overcome problems caused in considering discretisation of an action space by using a continuous action set. It was proposed that the probability vector associated with a discrete action set could be replaced with a single distribution function, characterised by a small number of parameters, thus encompassing a continuous action set. Analysis of a learning algorithm using a Gaussian function as the distribution function was carried out. A reward or penalty response would yield an update to the distribution via variation of

the two descriptive parameters of a Gaussian distribution, the mean and variance. A single parameter optimisation problem, on a multi-modal function, illustrated the effectiveness of the algorithm in finding exact minima points. Unfortunately the algorithm often converged on a local minimum of the test function and a rerun would be required for the automaton to have a chance of finding the global minima. It was suggested that multi-parameter optimisation problems could be dealt with by applying the continuous action set learning automaton in an interconnected fashion, utilising a team of such automata, one per parameter. Although the simulated tasks showed up inherent weaknesses in the formulation as reported, the underlying philosophy proposed is significant The principle idea, to adapt the learning automata methodology to consider a continuous action space, is of particular note, and is picked up again in later chapters.

1.4 Objective of this Thesis

Many studies in the literature have viewed a learning automaton methodology as a possible optimisation technique, especially beneficial in random environments where traditional gradient type techniques would either not work, or at best generally fall into local optima Simulation studies have illustrated the effectiveness of employing learning automata for optimising single and multi-parameter functions including additive noise. Little real-world application of the technique has been reported however. The objective of this thesis may therefore be summarised as:

> *To investigate the application of a learning automaton methodology in the context of an on-line parameter optimisation in complex dynamic and stochastic environments, as exemplified by advanced vehicle suspension systems*

1.5 Outline of Thesis

Chapter 2 details the suspension system models used in the simulation studies of later chapters, introducing basic concepts and terminology The significant portion of this thesis is then broadly split into two main areas investigating two particular forms of learning automaton.

Chapters 3, 4 and 5 describe studies using classic discrete learning automata. The viability of applying a learning automaton technique to suspension systems is first investigated in Chapter 3, where two forms of automaton are tested on simple quarter-vehicle suspension tasks. However, a number of practical limitations are noted that

prevent safe on-line application. Chapter 4 addresses these concerns, developing the methodology to aid practical application. The culmination of this work is an experiment in hardware, presented in Chapter 5, in which the extended discrete learning automata methodology is applied directly on a test vehicle to perform a ride optimisation task

The remainder of this thesis introduces and investigates a new form of learning automaton, the Continuous Action-set Reinforcement Learning Automaton (CARLA)

Although the discrete automata studies produce some promising results, the simplification of an essentially continuous action space to a discrete set of actions is also seen to introduce a number of limitations itself. The CARLA is developed to overcome these issues Chapter 6 describes the concept behind the operation of CARLA and outlines its method of implementation The performance of this new automaton is investigated in Chapter 7 via analysis on basic tasks and comparison with discrete automata.

Chapter 8 returns to the vehicle suspension application, to repeat a number of tasks encountered during the discrete learning automata studies, but applying CARLA in their place for comparison. Approaches adopted earlier are seen to be equally applicable to aid on-line application of CARLA. The chapter concludes with a hardware experiment with CARLA applied to optimise the ride characteristic of a test vehicle.

A multi-goal learning task is presented in Chapter 9. A dynamic roll control algorithm is developed to minimise the roll response of a full-vehicle simulation including a realistic semi-active suspension model. The algorithm presents five free parameters available for learning Two teams of CARLA are configured to learn 'optimal' values for these parameters in simulation

During the development of CARLA some ideas for possible extensions to the methodology presented in Chapter 6 have arisen. Preliminary studies of two such extensions to the CARLA methodology are presented in Chapter 10

Chapter 11 concludes the thesis with a discussion of the work presented and possible areas of future research.

Chapter 2 - Suspension System Models

This chapter describes the suspension systems considered within this thesis, including both simulation models and vehicle hardware Simulation studies are implemented with application of 5th order Runge-Kutta integration routines, including variable step-size techniques to overcome discontinuities and non-linearities of the integrands. Digital control is implemented in simulation to match the control application in hardware considered later.

2.1 Quarter-vehicle Ride Model

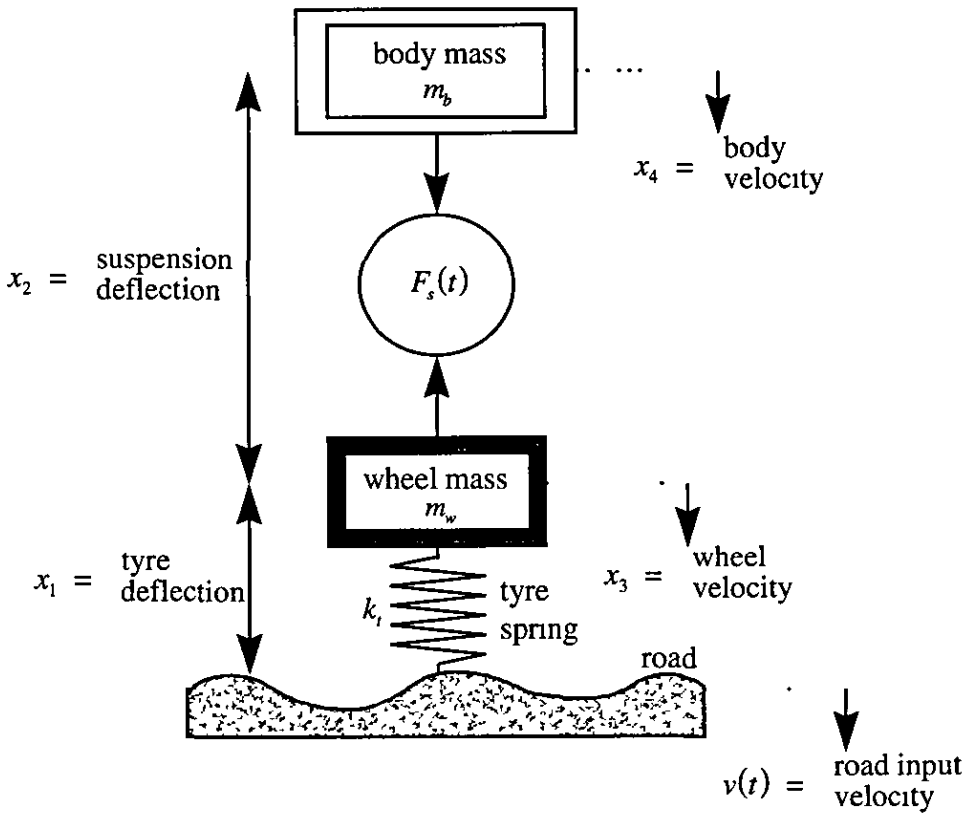


Figure 2.1 - Quarter vehicle model

Figure 2.1 shows a simple model relevant to the vertical motion at a single wheel-station of a vehicle. It consists of two masses, the sprung and unsprung masses, which represent the body and wheel masses respectively The wheel mass is isolated from the road by a tyre. The vertical dynamic load on a tyre, F_t, has been shown to be approximately proportional to the vertical tyre deformation (Sharp & Hassan 1986) and therefore a simple model of the tyre as a spring is sufficient in considering vertical motion alone. A suspension force, F_s, is applied between sprung and unsprung masses to provide vibration isolation for the sprung mass, maintaining a satisfactory

separation of the two masses and providing control of the dynamic tyre loads. F_s can be achieved in a number of ways, outlined later in this section.

The quarter vehicle model is widely used in the literature as it contains the most basic features of a vehicle for consideration of ride comfort. It includes a representation of the problem of controlling wheel load variations, and hence road-holding properties, and contains suspension system forces that are properly applied between the unsprung and sprung masses

Using the state variables shown in Figure 2.1, the equations of motion are.

$$\begin{aligned} \dot{x}_1 &= v(t) - x_3 \\ x_2 &= x_3 - x_4 \\ \dot{x}_3 &= \frac{F_t - F_s}{m_w} \\ \dot{x}_4 &= \frac{F_s}{m_b} \end{aligned} \qquad (2\ 1)$$

Here $v(t)$ is the vertical velocity of the tyre contact patch and the tyre force $F_t(t)$ is given by $F_t = k_t . x_1$

F_s may be defined in a number of ways to represent different suspension arrangements. Two particular arrangements are considered within this thesis in relation to quarter vehicle simulation, both providing reference for performance comparison with other systems

2.1.1 Passive suspension

In the passive suspension $F_s(t)$ is provided by a spring, of stiffness k_s, in parallel with a damper, so

$$F_s = k_s x_2 + F_d \qquad (2.2)$$

where the damping force

$$F_d = b_s(x_3 - x_4) \qquad (2\ 3)$$

and b_s is the damping rate. Although identification of the characteristics of an actual passive damper unit shows that some non-linearities and hysteresis are present, introduced by the physical characteristics of its components, the simple model in equation 2.2 gives a sufficient first approximation for modelling - see Best (1995) A passive system may be seen as the simplest form of suspension available to control the

sprung and unsprung masses to a satisfactory degree. No power is consumed by the system other than that derived from the vehicle's kinetic energy of forward motion

The linear passive quarter vehicle model exhibits two modes of vibration, commonly referred to as the *wheel-hop* and *body bounce* modes. Rearranging equations (2 1), (2.2) and (2 3) into the matrix form·

$$\dot{x} = Ax + Bv \tag{2 4}$$

and applying the following model parameter values, representative of a medium sized saloon car

$$\begin{aligned} k_s &= 20000\text{Nm}^{-1} \\ k_t &= 200000\text{Nm}^{-1} \\ m_w &= 40\text{kg} \\ m_b &= 300\text{kg} \end{aligned} \tag{2.5}$$

the modes of the model are characterised by the eigenvalues of A:

$$\begin{aligned} \lambda_w &= -25\,4 \pm 67.5\,\text{i} \\ \lambda_b &= -2\,9 \pm 7.5\,\text{i} \end{aligned} \tag{2 6}$$

From complex constants $\sigma + \omega\text{i}$, the undamped natural frequency and 98% settling time for each mode are then:

	Wheelhop - λ_w	Body Bounce - λ_b
$\omega_n = \frac{\omega}{2\pi}$	10 8 Hz	1.2 Hz
$T_s = \frac{4}{\sigma}$	0.16 s	1 4 s

The natural frequencies of each mode can be seen in the power spectral density response to white noise of the wheel and body velocity respectively - Figure 2.2 The lower plot shows the body motion is entirely dominated by the body bounce mode The upper plot, however, shows wheel motion as covering a much broader bandwidth, excited at both modes of the system, although the majority of the input power is transmitted at higher frequencies

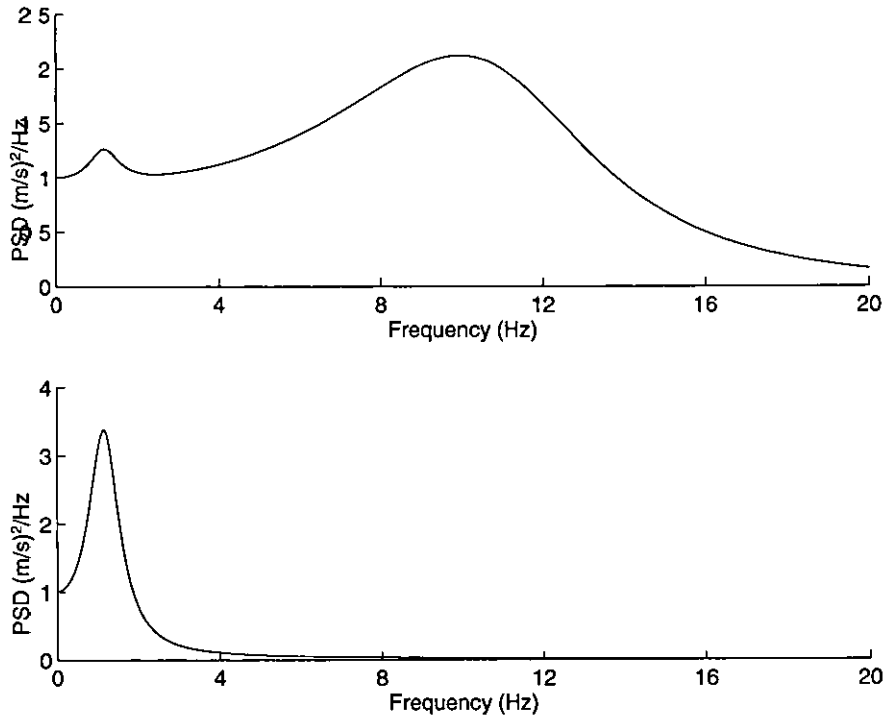


Figure 2.2 - Wheel velocity (upper) and body velocity (lower) power spectral densities

2.1.2 LQG active suspension

The second reference system is provided by considering an ideal actuator acting between sprung and unsprung mass, assumed to operate without error or time delay, so providing any requested force instantaneously. The suspension force is thus seen as the control variable.

$$F_s = u(t) \tag{2.7}$$

Assuming such an actuator, and applying a white noise signal as the road velocity input to a linear quarter vehicle model, there exists a theoretical technique to synthesise an optimal controller subject to a quadratic cost function.

The linear quadratic Gaussian (LQG) optimal control technique (Kwakernaak & Sivan, 1972) defines, for a linear system, a linear state feedback controller

$$u(t) = \boldsymbol{K}.\boldsymbol{x} \tag{2.8}$$

that optimises a quadratic performance index where the input to the system is defined in terms of zero mean Gaussian white noise processes. The gain vector $\boldsymbol{K}$ is found via solution of the algebraic Riccati equation.

For the quarter-vehicle model of Figure 2 1, an appropriate performance index to employ LQG optimal control is

$$J = a.x_1^2 + b.x_2^2 + \dot{x}_4^2 \tag{2.9}$$

The body acceleration term costs passenger discomfort, and road holding is maintained by suitably costing tyre deflection. The weightings, a and b are selected to tune the controller to return acceptable values for the three terms within the constraints of available workspace for suspension travel. Here, the weights have been set according to a study by Marsh et al. (1995)

$$a = 64000, \quad b = 750 \tag{2.10}$$

Using these values within the discrete-time LQG formulation, assuming a 500Hz sampling frequency (as used in later practical application) yields the feedback controller

$$u(t) = [\ -10406\ \ 8079\ \ 1029\ \ -2258\ ]\, \boldsymbol{x} \tag{2 11}$$

with associated performance index:

$$J_{opt} = 2\,6621 \tag{2.12}$$

2.1.3 Semi-active suspension

A third suspension arrangement is available by utilising semi-active actuators. A vehicle fitted with a semi-active suspension system has been used in practical application of the learning methods developed in this thesis The test vehicle is a Ford Granada fitted with continuously variable damper units and instrumented with sensors at each wheel-station. Representative models of the particular hardware actuators used were developed by Best (1995), and the derivation of the semi-active quarter-vehicle model is presented here as a precursor to developing a full vehicle model, described later.

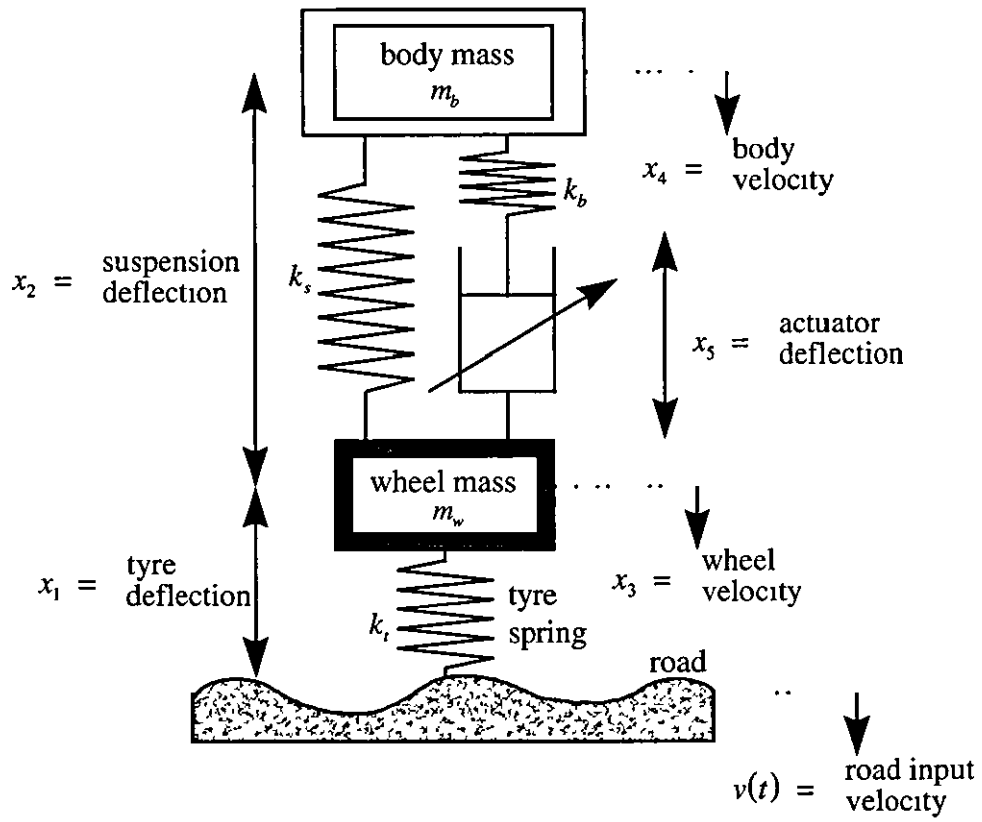


Figure 2.3 - Semi-active quarter vehicle model

The first four states in Figure 2.3 are the same as in the general layout of Figure 2.1, and the state equations in (2.1) apply. To derive state equations describing the application of control, and model the inherent actuator transients of the real semi-active damper, additional states are required Three extra states are defined to include such an actuator in the quarter vehicle model. x_5 is the instantaneous expansion velocity of the damper unit. x_6 and x_7 are state variables for a second-order transfer function which represents the transient behaviour of the damper actuator valve.

In the semi-active scheme, suspension force may still be described by equation (2 3), as in the passive system, but now F_d is the damping force of the continuously variable damper, which is a non-linear function of velocity x_5 and the dimensionless control valve position variable x_6

$$F_d = \mu(x_5, x_6) \tag{2.13}$$

x_6 is the actuator control variable, defined to have an operating range (0-100), with 0 implying minimum (0%) damping, and 100 thus implying maximum (100%) damping from the actuator. The non-linear maps of actuator force are derived from system identification on actual hardware (Best 1995). Figures 2.4 and 2.5 show damper maps from front and rear actuators. For simulation, only the outer two lines on these maps,

0% and 100%, are identified. Values between these boundaries are found by interpolation.

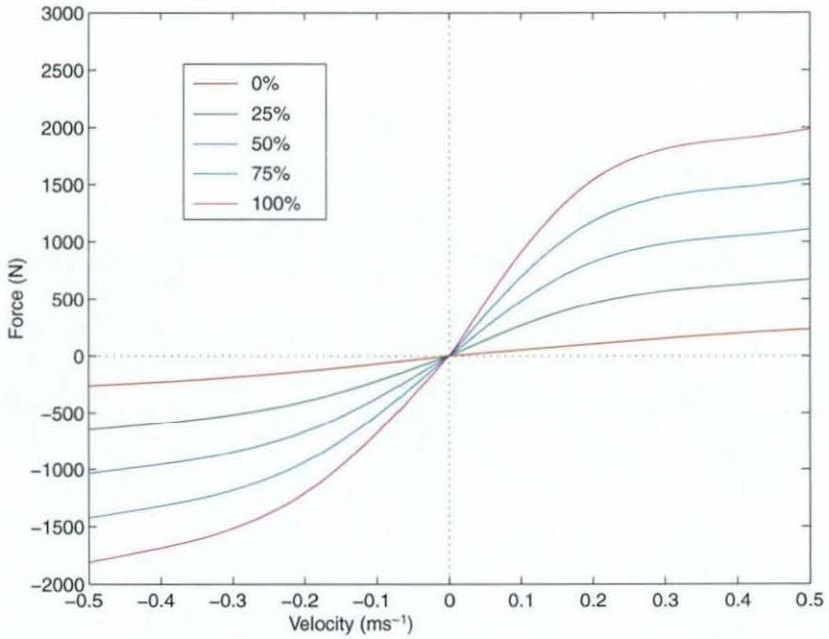


Figure 2.4 - Force velocity map of front actuator

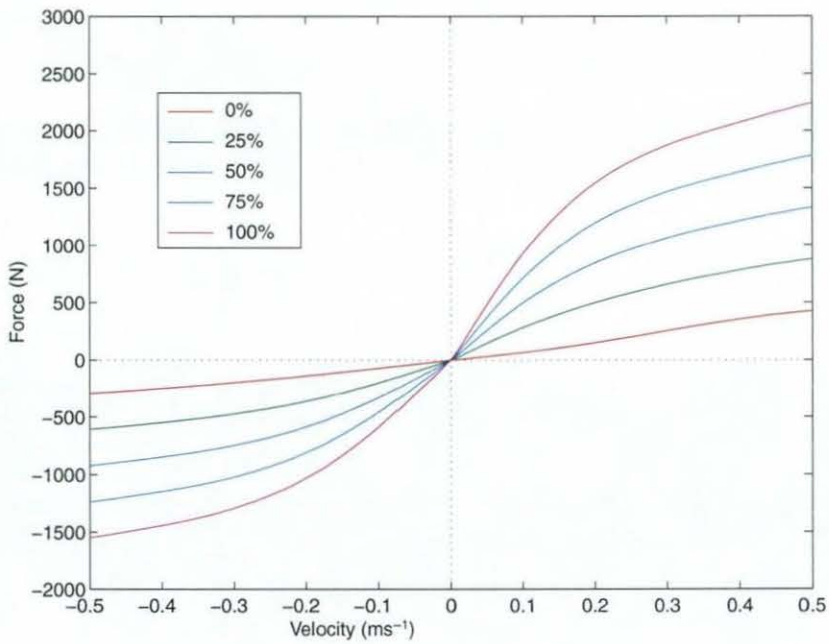


Figure 2.5 - Force/velocity map of rear actuator

The actuator filter equations, describing the transients in hardware, are then

$$\begin{aligned} x_6 &= x_7 \\ \dot{x}_7 &= -2\zeta\omega_n x_7 + \omega_n^2\left(u(t) - x_6\right) \end{aligned} \tag{2.14}$$

To derive the state equation for x_5, let k_b be the stiffness of the compliant 'bush' in series with the damper, this represents both the rubber mounting of the damper and its internal compliance. If $d(t)$ is the dynamic displacement of the bush, then equating forces at the damper and considering basic kinematics

$$\begin{aligned} k_b d(t) &= \mu(x_5, x_6) \\ d(t) &= x_2 - x_5 \end{aligned} \tag{2.15}$$

Differentiating the first of these with respect to time yields the fifth state equation

$$x_5 = \left\{\frac{\partial \mu}{\partial x_5}\right\}^{-1}\left\{k_b\left(x_3 - x_4 - x_5\right) - x_7\frac{\partial \mu}{\partial x_6}\right\} \tag{2.16}$$

which is clearly non-linear.

2.2 Rigid Body Model

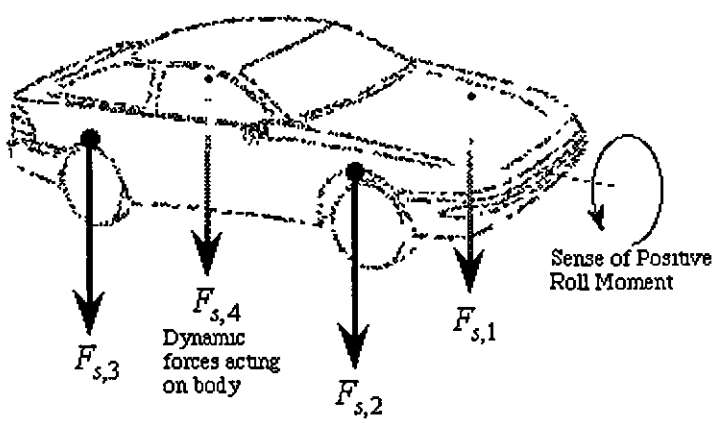


Figure 2.6 - Suspension force and moments acting upon the vehicle body labelling and sign conventions

A full-vehicle dynamic model, based on the test vehicle, is used in simulation studies of a roll control learning task in Chapter 9 This model assumes a rigid vehicle body fitted with a semi-active suspension unit, of the form described in Section 2 1.3, at each corner

To develop the set of state equations for the full vehicle model, Newton's second law is applied to deduce the vehicle body dynamics as follows, with reference to Figure 2.6 for sign convention:

$$\begin{aligned}
&\text{Bounce accel.} = \sum_i F_{s,i} \Big/ m_b \\
&\text{Roll accel.} = \left(p.\left(F_{s,2} + F_{s3}\right) - q.\left(F_{s,1} + F_{s4}\right)\right) \Big/ I_r \\
&\text{Pitch accel} = \left(r.\left(F_{s,3} + F_{s,4}\right) - s.\left(F_{s1} + F_{s,2}\right)\right) \Big/ I_p
\end{aligned} \qquad (2.17)$$

where p, q, r and s are lateral and longitudinal distances from the wheels to the vehicle's centre of gravity. I_r and I_p are the roll and pitch moments of inertia of the vehicle body, respectively.

Similarly, the dynamics of the unsprung mass at each corner are given by

$$\text{Unsprung corner mass accel.} = \left(F_{t,i} - F_{s,i}\right) \Big/ m_{w,i} \qquad (2.18)$$

These dynamic equations produce 14 state equations, and a further 12 are added to include realistic transients and compliance modelling for the semi-active actuator units. A comprehensive derivation and listing of the full vehicle dynamic and state equations are given in Appendix A.

The following parameter values have been used in simulation of the full-vehicle system

$$\begin{aligned}
k_{s,\text{front}} &= 22500\text{Nm}^{-1} \\
k_{s,\text{rear}} &= 21000\text{Nm}^{-1} \\
k_b &= 1200\text{Nm}^{-1} \\
k_t &= 160000\text{Nm}^{-1} \\
m_{w,\text{front}} &= 28\text{kg} \\
m_{w\,\text{rear}} &= 32\text{kg} \\
m_b &= 1400\text{kg} \\
I_r &= 380\text{Nms}^2.\text{rad}^{-1} \\
I_p &= 2400\text{Nms}^2.\text{rad}^{-1} \\
\zeta &= 0.7 \\
\omega_n &= 1257\text{rad s}^{-1}
\end{aligned} \qquad (2.19)$$

2.3 Stochastic Road Inputs

The only input to the vehicle model is the vertical road velocity considered to act at a point at the base of the tyre spring. All models in this thesis are simulated over

stochastic processes representing the random road velocities induced on a vehicle tyre as it traverses a road. Three types of single track 'road' have been employed with the quarter vehicle model, white noise, Robson road and measured road A dual track input is synthesised for the full vehicle model to include a realistic roll inducing component across the tracks.

2.3.1 White noise

The use of white noise allows the application of optimal control theory to synthesise a linear state feedback control law - see Section 2.1.2. This means that $v(t)$ is supplied as a Gaussian white noise process, and although this is not entirely realistic, it has been widely used in the literature (e g. Karnopp, 1983.) In comparison with real road spectra a white noise input spectra provides insufficient power at low frequencies.

Where white noise is applied in simulation studies in this thesis, independent samples of zero-mean white noise are taken. Using a zero-order hold period of $T = 0.01$ seconds, the signal RMS is set to $\sigma = 0.5\,\mathrm{m/s}$

2.3.2 Robson road

A more realistic road signal can be derived from a frequency shaped white noise process A widely used model is that suggested by Robson (1979). The vertical displacement power spectral density, S, of the surface is given by

$$S(f) = kU^{(w-1)} f^{-w} \qquad (2.20)$$

where f is the frequency in Hz, k is a roughness coefficient, and U is the forward speed of the car in m/s. Robson estimated the roughness coefficient of roads to typically lie between 3×10^{-8} for smooth motorway, and 3×10^{-5} for a rough minor road

This road model is used in practical studies to provide a reference input for hydraulic actuators acting on each wheel of a test vehicle The prime concern of a road description for such a rig is that the peak deflections stay within the working range of the actuators. For this reason, and assuming a 20 m/s forward vehicle velocity, a roughness coefficient of $k = 2.4\times10^{-6}$ is applied Also, to prevent large deflections from low frequency 'drift', all frequencies below 0 2Hz are removed by filtering in the frequency domain via fast Fourier transforms (FFT).

2.3.3 Measured road

A better approximation to real road spectra is achieved from taking measurements of different types of actual road. A number of roads around Loughborough (Leicestershire, UK) have been examined, with road height measurements of both left and right wheel tracks taken at 10cm intervals. Differentiation by FFT allows road velocity spectra to be produced for varying vehicle speeds. All simulation has assumed a forward vehicle velocity of 20m/s.

Two sections of measured road are used principally throughout this thesis: Breakback Road and Copt Oak Road. Figure 2.7 shows how the power spectral densities for these two roads are similar in shape, varying mainly in amplitude. Breakback Road is an undulating road with pronounced low frequency features and, as a C class road, also has a rough surface finish. Copt Oak Road presents a less severe response across the frequency spectrum; it is a B class road with a higher quality surface finish and generally less prominent features compared to Breakback road.

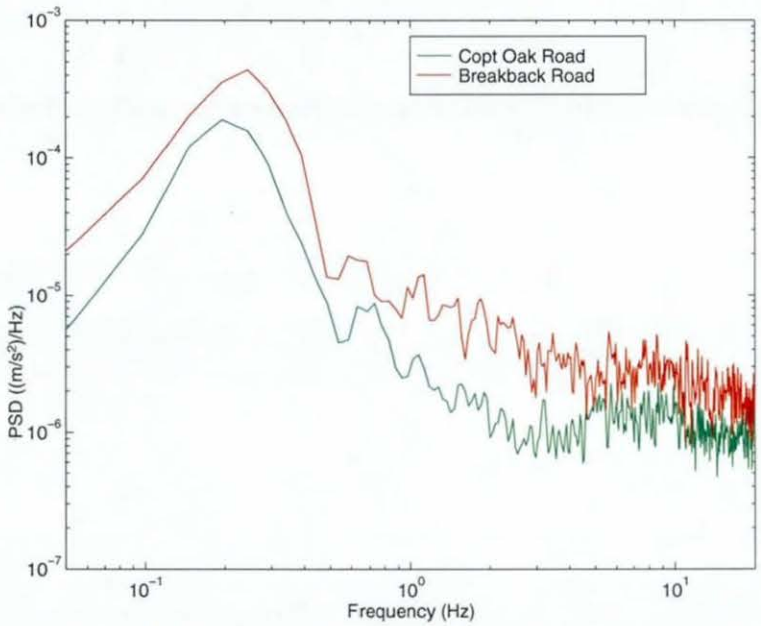


Figure 2.7 - Power spectra density of Breakback Road and Copt Oak Road

2.3.4 Dual track input for full body model

Input to the full vehicle model is an artificial road created to mimic the roll/bounce power spectral density relationship of a measured road surface profile. A 3000m continuous road is created as left and right tracks with vertical displacement of each track defined as a height component, plus or minus a roll component:

$$z_L = z + 6$$
$$z_R = z - \theta \qquad (2.21)$$

where

$$\dot{z} = n_1(t)$$
$$\dot{\theta} = -\xi\theta + n_2(t) \qquad (2.22)$$

and $n_1(t)$ and $n_2(t)$ are Gaussian white noise processes.

Figure 2.8 shows the roll/bounce PSD ratio relationship of a measured road profile. At low frequencies it is seen that left and right tracks have similar profiles, as expected as these long waves are defined by the lie of the land over which the road is passing. For higher frequencies the left and right tracks become increasingly unrelated. This phenomenon is simply modelled in the artificial road by high pass filtering the roll component in equation 2.22 to remove low frequency displacement differences between the two tracks of the road. A value for ξ of 12 6 results in a roll/bounce PSD ratio which closely matches that of the measured road, as shown in Figure 2 8

An amplitude for $n_1(t)$ and $n_2(t)$ of 3 was used. The resultant power content of the artificial road is significantly higher than the comparable Breakback Road at higher frequencies - see Figure 2 9. This 'rougher' road model will tend to induce a larger wheel hop response on the vehicle. Applying the artificial road, much more importance is then placed on any learning algorithm being able to learn adequate control in this harsher environment.

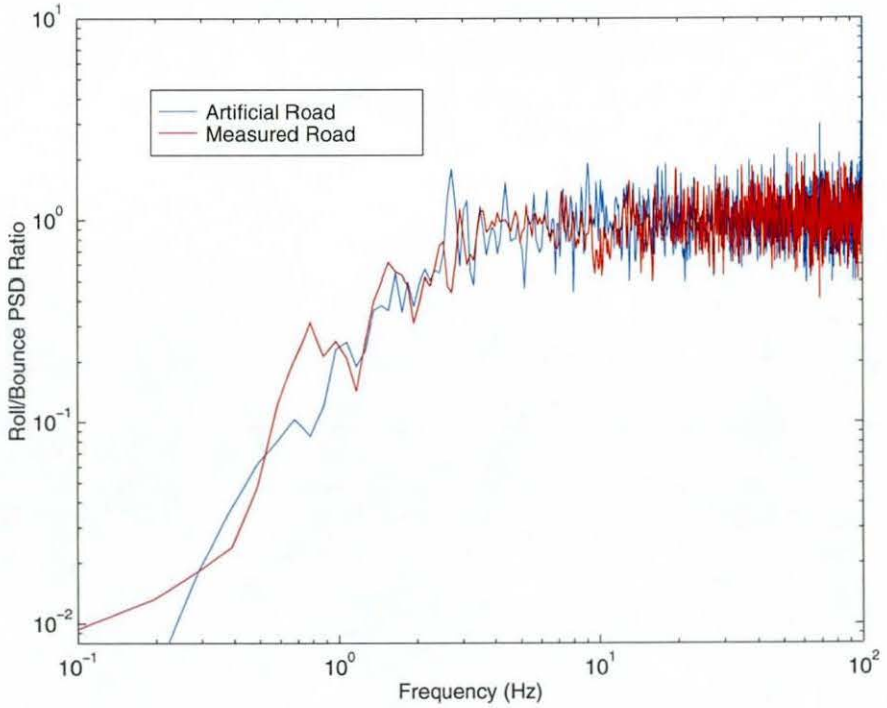


Figure 2.8 - Ratio of roll to bounce content of artificial road and Breakback Road

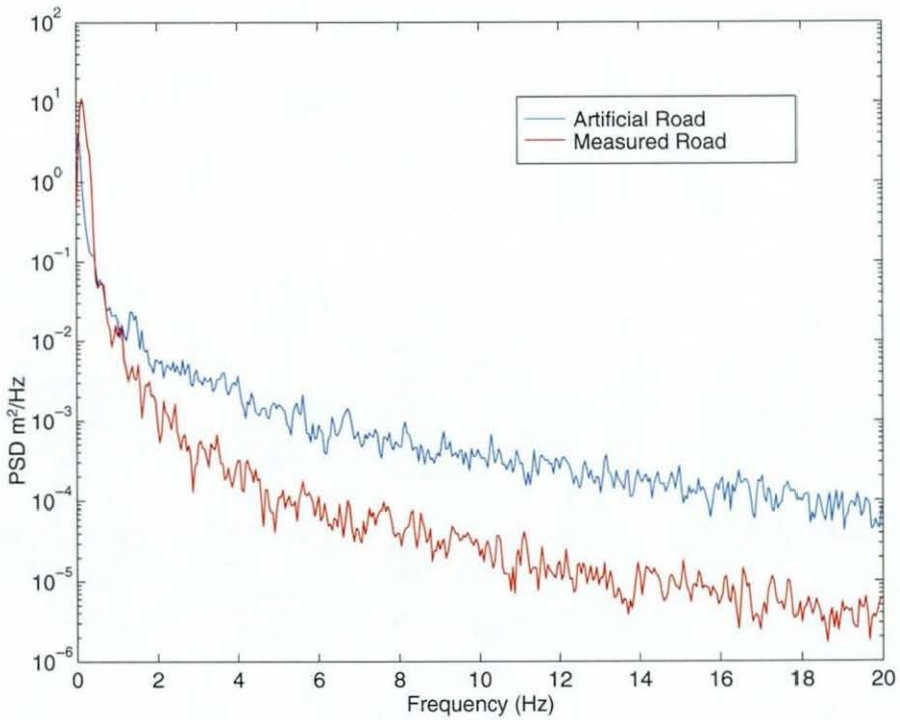


Figure 2.9 - PSD velocity content comparison of artificial road with Breakback Road

Chapter 3 - Discrete Learning Automata

This chapter takes the general mathematical framework of learning automata outlined in Section 1.2, and introduces further details of their implementation. A preliminary study of the possible application of a classic discrete automaton to learn a vehicle suspension controller is described, and, in so doing, two learning schemes are compared. This study was originally presented at the IUTAM Symposium, 1994 (Frost et al., 1994).

3.1 Introduction

A learning automaton may be termed 'discrete' when the action set of the automaton consists of a finite number of distinct actions, as is the case for the learning automata as first defined. This is well illustrated by considering an early application of the technique in the optimisation field (Shapiro & Narendra 1969). A discrete learning automaton was utilised to maximise a function of a single variable - Figure 3.1. Although the function is continuous across the considered range, the learning automaton has an action set of 10 distinct points at which the function is evaluated for the purpose of learning the action that returns maximum reward, $I(\alpha)$.

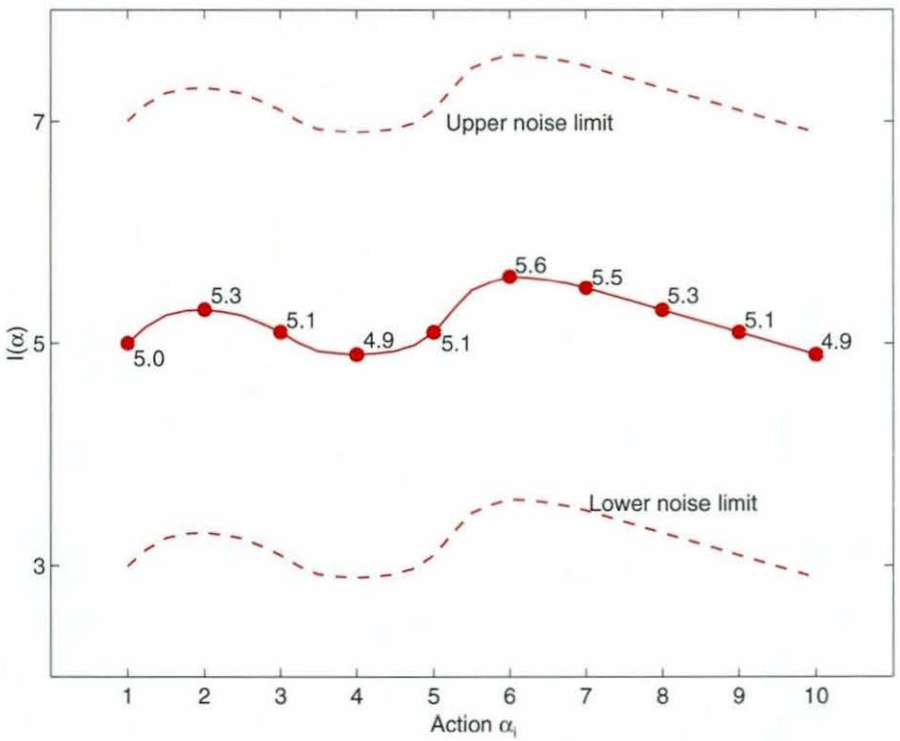


Figure 3.1 - Continuous function, discrete evaluation points

A natural extension of this approach maps the action set to discrete values of parameters in an environment Figure 3.2 shows an adaptive identification scheme, also from the study of Shapiro & Narendra (1969). A discrete learning automaton was utilised to learn the optimal values of the parameters *A* and *B*. Each parameter was discretised to 5 values, including the optimal values. The action set is then defined as the set of (*A*, *B*) value pairs covering all combinations of discrete parameter values, 5^2 = 25 in total for this case. Many subsequent studies employed similar parameter/action set mappings to cover a parameter space of the environment under consideration.

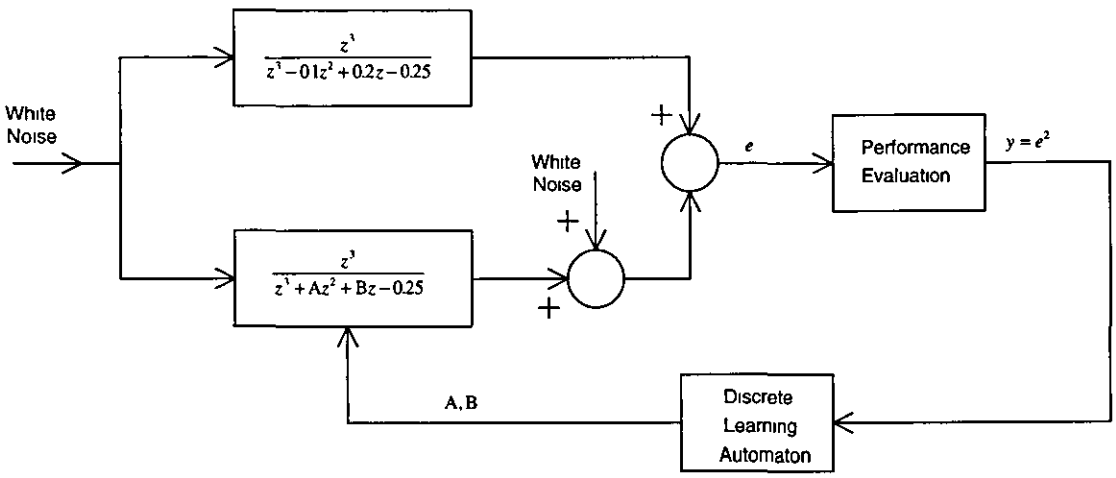


Figure 3.2 - Adaptive identification scheme

The further extension to this methodology (Wu 1993) allowed an automaton to 'home in' to a smaller area of the action space once a particular action begins to dominate. Upon achieving a level of convergence to any one particular action from its set, the proposed method allows the learning automata to re-start learning on an action set that covers a reduced area of the initial action space, centred on the successful action from the previous stage.

A preliminary study by Gordon et al. (1993) investigated the feasibility of a learning automata technique with regard to vehicle suspension control The suspension system under consideration was a quarter vehicle model, as defined in Section 2.1, with full bandwidth suspension force actuation and a white noise road velocity input. A theoretical optimal control law is readily available for such a system, from applying LQG optimal control theory, giving the control force as a function of the system states - Section 2.1.2. This provided a solution from which the relative performance of the learning automaton could be gauged. Without significant degradation in performance, the control law may be simplified to·

$$u(t) = \begin{bmatrix} 0 & k_2 & k_3 & k_4 \end{bmatrix} \boldsymbol{x} \qquad (3.1)$$

A discrete learning automata was applied, by Gordon et al., to learn values for the three gains, $\{k_2, k_3, k_4\}$, of (3 1). Simulation studies suggested that learning automata could indeed optimise such a control law, successfully learning capable controllers, on-line, with no explicit knowledge of the suspension system itself.

Section 3 2 extends the description of learning automata of Section 1.2.1 to form an algorithm describing their operation in general terms Individual parts of the general algorithm that describe a discrete learning automaton are then outlined in Section 3.3. In Section 3 4 the P-model learning scheme, adopted by Gordon et al. in their feasibility study, is introduced, including specific details pertaining to such a scheme. Section 3 4 then describes, in similar form, an S-model learning scheme. Simulation studies of both P-model and S-model schemes in Section 3.5 leads to a number of suggestions of possible improvements that would enable application in a real-world environment, and these are summarised in Section 3 6.

3.2 General Algorithm

Using the notation introduced in Section 1.2.1, a general algorithm for a learning automaton may be described by the following pseudocode

Initialise action set $\alpha(n)$
Initialise probability vector $p(n)$
$n = 1$
Repeat
    Select action $\alpha_i(n)$ stochastically according to probability vector $p(n)$
    Trial action $\alpha_i(n)$ in the environment
    Receive **reward/penalty response,** β, from environment
    Apply **reinforcement scheme** to produce $p(n+1)$
    $n = n+1$
Until **convergence criterion** attained

Figure 3.3 - General learning automaton algorithm

This algorithm constitutes a single stage of learning, with the automaton repeatedly selecting, executing and reinforcing actions until one action dominates and convergence to the 'best' action is deemed to have occurred.

The development of the learning automata methodology in this thesis is based upon the general description of the algorithm outlined in Figure 3 3. For this reason the major aspects of the algorithm which distinguish particular formulations of learning automata have been highlighted in bold

3.3 Discrete Learning Automata Algorithm

With reference to Figure 3.3, the major aspects of a discrete learning automaton algorithm are outlined here. In addition the extension to the methodology introduced by Wu (1993) is described.

3.3.1 Initialising the action set

A discrete learning automaton requires that the action space under consideration be discretised to a finite number of possible actions. Suppose the action space corresponds to a discretisation of a multi-dimensional parameter space If the environment has N parameters, each discretised to r equally spaced action values, the complete action set is formed from all possible combinations, $s = r^N$, of those actions, so the action set can be written as $\alpha = \{\alpha_1, \alpha_2, .., \alpha_s\}$.

3.3.2 Initialising the probability vector

Each action is associated with a probability of selection by the automaton. The probability vector is

$$p = \{p_1, p_2, ..., p_s\} \tag{3.2}$$

No prior knowledge of the performance is assumed and so each action is assigned an equal probability of selection. Subject to the natural constraint

$$\sum_{i=1}^{s} p_i = 1 \tag{3 3}$$

the initial probability is thus

$$p_i = \frac{1}{s}, \qquad i = 1,2,...,s \tag{3.4}$$

3.3.3 Action selection

The probability vector may be thought of as a discrete probability distribution across the action space, with a corresponding cumulative distribution function. For example,

consider the initialisation for an automaton covering the parameter range 0 to 10 with six actions. Each action is assigned a selection probability of 1/6 forming a discrete distribution as shown in Figure 3 4 The corresponding cumulative distribution function (c d f.) then consists of a series of discrete steps, shown in Figure 3.5.

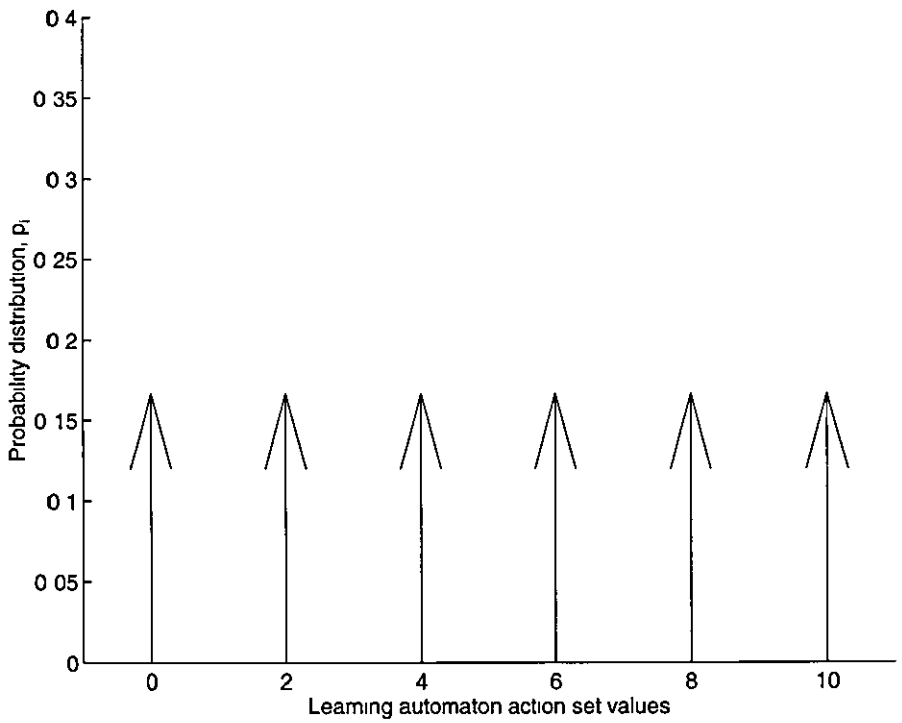


Figure 3.4 - Discrete probability distribution

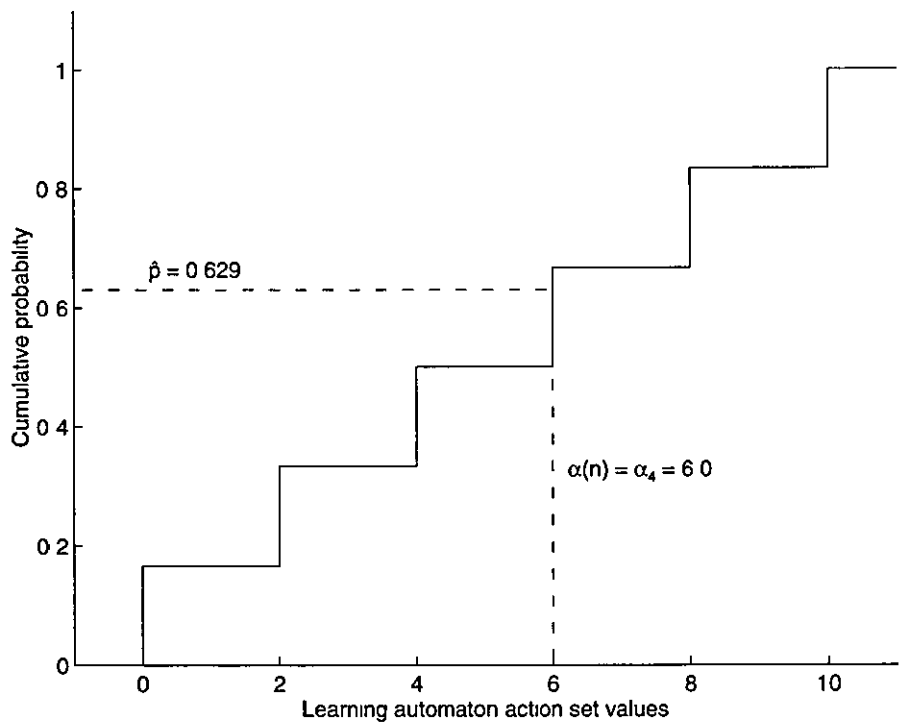


Figure 3.5 - Example of action selection on a discrete distribution function

At each iteration of the automaton a uniformly distributed pseudo-random number between 0 and 1 is taken, $\hat{p} \in U(0,1)$. Using this value, an action is then selected based on the current action probability vector. The cumulative distribution function formed from the probability vector is used in the selection process. Tracing $\hat{p}$ across to the point of intersection with the c.d.f, the action value at that point is taken as the chosen action For example, in Figure 3.5, $\hat{p} = 0.629$ gives $\alpha(n) = 6$.

3.3.4 Convergence criterion

Repeated reinforcement of an action through successful responses from trials in the environment will lead to the probability of selection of that action becoming dominant In the limit the learning automaton should converge towards a single action choice with probability 1. It is likely, however, that in a noisy environment where a number of actions return similar responses, the automaton will pick out these actions but be unable to distinguish between them sufficiently to converge to a single action choice. An automaton is thus deemed to have converged, in a practical sense, to an action choice if

$$\max_i \{p_i(n)\} > \eta, \quad i = 1,2,\ldots,s \tag{3.5}$$

where the convergence threshold η satisfies $1/s < \eta < 1$. This leads to a natural trade-off in learning between exploration and exploitation. A large value for η will lead to the automaton exploring the actions extensively to gather enough experience to converge to one action, with the possibility, as pointed out above, that no choice is made Too small a value for η and the automaton quickly 'exploits' an action through fast convergence, which may then be erroneous from 'jumping to conclusions'.

3.3.5 Extension to search action space

Use of a discrete action set naturally implies that areas of the parameter space are not explored - those that lie between action points. To effectively cover a parameter space and enable a full search of the region may require many finely spaced discrete actions. However, increased action set size invariably leads to increased learning times since more actions are available for trial and initial probability levels per action are also less.

Wu (1993) suggested the following method to enable the use of a small action set automaton to effectively search a large parameter space. A stage of learning as described in Section 3.2 takes place. Once the convergence criterion is met the action set is redefined about the successful parameter vector with a scale factor λ applied to

reduce the size of the search region and refine the choice of parameters. The learning automaton is then repeated, reinitialising the probability vector, to learn over the smaller action space

The single stage of learning from the general algorithm is now enclosed in an outer loop to include the above modification

$n = 1$
$m = 1$
Initialise action set $\alpha(n)$
Repeat
    Initialise probability vector $p(n)$
    Repeat
        Select and **Trial** action $\alpha_i(n)$
        Receive **reward/penalty response,**
        Apply **reinforcement scheme**
        $n = n + 1$
    Until **automaton convergence criterion** attained
    $m = m + 1$
    Reinitialise action set about successful action
     applying scale factor λ **to range of parameters**
Until **action space convergence criterion** attained

Figure 3.6 - Discrete learning automata algorithm with convergence

The learning automaton is deemed to have reached completion when the action space convergence criterion is achieved

$$\lambda^m < 0.01 \qquad (3\ 6)$$

where m is the number of learning stages completed; this corresponds to a reduction in the search region to 1% of the initial size.

3.4 Scheme 1: P-model Learning Automaton

The study by Gordon et al. (1993) employed a P-model learning automaton to investigate the feasibility of applying the method to learn a three parameter linear feedback controller applied to a simulated full-active vehicle suspension system - equation (3 1). Here, with reference to the pseudocode of Section 3.2, the defining sections of this automaton are described as applied in the earlier study.

3.4.1 Reinforcement scheme

A P-model reinforcement scheme works with a binary response, $\beta \in \{0,1\}$, from the environment, where $\beta = 0$ denotes a 'favourable' response, $\beta = 1$ an 'unfavourable' response. The particular P-model reinforcement scheme used by Gordon et al was a non-linear reward-penalty scheme (N_{R-P}) of the form:

$$\left.\begin{aligned} p_i(n+1) &= p_i(n) + \theta p_i(n)\big(1 - p_i(n)\big) \\ p_j(n+1) &= p_j(n) - \theta p_i(n)\big(1 - p_i(n)\big)/(s-1) \end{aligned}\right\} \text{ if } \beta(n) = 0$$
$$\left.\begin{aligned} p_i(n+1) &= p_i(n) - \theta p_i(n)\big(1 - p_i(n)\big) \\ p_j(n+1) &= p_j(n) + \theta p_i(n)\big(1 - p_i(n)\big)/(s-1) \end{aligned}\right\} \text{ if } \beta(n) = 1 \qquad (3.7)$$

where $j = 1,2,\ldots,s$ $j \neq i$, s is the number of actions comprising the action set and θ is a user-defined learning rate parameter, $0 < \theta < 1$. It is easily verified that this scheme maintains the constraint of (3.3) at each iteration.

3.4.2 Reward/penalty response from the environment

In many cases where learning automata are applied the environment alone is not capable of giving a critical performance response of the type required by the automaton. More readily a cost function is used to provide some measure of performance and then a performance evaluation routine is formulated to map the resulting cost, J, to the critical response, β.

This formulation utilises the following performance evaluation routine. The environmental response $J(n)$ is compared with a reference value of acceptable performance:

$$J_{ref} = (1+\delta)\bar{J} \qquad (3.8)$$

where $\bar{J}$ is the average measured performance index based on the previous H *favourable* responses The critical response, β, is then attained from

$$\begin{aligned} J(n) \leq J_{ref} \quad \beta{=}0 \quad &- \text{ favourable} \\ J(n) > J_{ref} \quad \beta{=}1 \quad &- \text{ unfavourable} \end{aligned} \qquad (3.9)$$

3.5 Scheme 2: S-model Learning Automaton

Whereas the P-model learning automaton acted on a binary environmental response, simply representing 'good' or 'bad', the S-model learning automaton takes values

within a continuous range as a response input. This type of automaton can be regarded as more applicable for problems of a continuous nature where it is not appropriate to simplify an action response to merely successful or unsuccessful. Instead a measurement of the 'degree of success' is more often available, in the form of a cost function The continuous range of automaton inputs allows an action to be rated in performance and gives more scope for evaluation of environmental responses than the binary rating used in Scheme 1.

3.5.1 Reinforcement scheme

The reinforcement scheme presented here is of the linear 'reward-inaction' form, demonstrated to exhibit good learning properties by Viswanathan & Narendra (1972). Such a scheme will reward a 'good' action, but the probability vector is left unchanged in response to a 'bad' action. Application of constraint (3.3) acts to penalise all other actions in response to a successful action. The critical response from the environment is now $\beta \in [0,1]$, with $\beta = 1$ being the most favourable response The particular scheme employed here to update the action probability distribution, in response to action α_i is:

$$\left.\begin{aligned} p_i(k+1) &= p_i(k) + \theta\beta(k)\big(1 - p_i(k)\big) \\ p_j(k+1) &= p_j(k) - \theta\beta(k)p_j(k) \end{aligned}\right\} \quad (j \neq i) \tag{3.10}$$

where θ is a learning rate parameter, $0 < \theta < 1$.

3.5.2 Reward/penalty response from the environment

The environmental response from a cost function, $J(n)$, is compared against previous values to gauge the level of success of the current action:

$$\begin{aligned} J_{\min} &= \min\{J(1), J(2), \ldots, J(n)\} \\ J_{\text{med}} &= \text{median}\{J(1), J(2), \ldots, J(n)\} \end{aligned} \tag{3.11}$$

$$\beta(n) = \max\left\{0, \frac{J_{\text{med}} - J(n)}{J_{\text{med}} - J_{\min}}\right\} \tag{3.12}$$

Hence $\beta = 1$ results from the latest action returning the lowest cost experienced during the current learning run, and $\beta = 0$ occurs if $J(n)$ exceeds the median cost of the available data The median of the cost history is used here in preference to the more obvious maximum statistic, as was used in a number of other studies, e.g. Viswanathan & Narendra (1973), Thathachar (1990), and Wu & Pugh (1993). The use of the

median value is motivated by the robustness of the median as a measure of central tendency. It allows the automaton to ignore any 'outlandish' values of J that would adversely skew the environmental response if the maximum statistic had been used.

3.6 Comparison of Learning Schemes

The description of the respective learning automata formulations in the preceding sections include all but one of the highlighted parts from the pseudocode of Figure 3 3, namely 'trial action in the environment'. Here, the particular environment of a vehicle suspension system is considered and the 'trial action' activity is discussed. The P-model and S-model schemes are compared in the context of this suspension environment.

3.6.1 Trial action in the environment

The 'environment' under consideration is the quarter-vehicle suspension system and its interaction with the road The road input supplied is Gaussian white noise velocity input to the base of the tyre.

The 'action' is a combination of parameter values, $\{k_1, k_2, k_3, k_4\}$, forming a linear feedback controller.

$$u(t) = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix} x \tag{3.13}$$

which is then applied to the environment for a period of time Δ Initial conditions at each iteration are set to zero so that no transients from 'previous' actions could interact with the effects of the 'current' action

An environment response, in the form of a cost function result J, is recorded for each selected action. This takes the form:

$$J(n) = \frac{1}{\Delta} \sum_{t-\Delta}^{t} \left(w_1 x_1^2 + w_2 x_2^2 + \dot{x}_4^2 \right) \tag{3.14}$$

where the three terms cost tyre deformations, suspension deflections and body accelerations respectively. Values for the cost are set at

$$w_1 = 64000, \quad w_2 = 750 \tag{3.15}$$

The goal of the above environment/action pair is therefore to learn optimal values of the four controller gains for the given cost function. The environment has been

defined as for the formulation for LQG optimal control theory application seen in Section 2 1 2 The 'solution' to this task is therefore already known and the values which the automata would ultimately be expected to learn are as given in equation (2 11) Performance of a learnt controller can also be gauged with comparison between the optimal cost of (2.12) and the theoretical cost obtained for the learnt controller.

The time period Δ has been selected at 16 seconds to enable the performance index to include low frequency effects of road surface unevenness. This time period is quite arbitrary, but its choice is justified from considering the variation of mean cost, J, with respect to Δ, for a set suspension controller. Here, the optimal controller given in Section 2.1.2 is applied in simulation, with various lengths of independent white noise samples as input Figure 3 7 shows how the coefficient of variation (a simple measure of noise to signal ratio defined as the ratio of standard deviation to mean) of the simulation cost J, averaged over 100 simulations, varies with simulation time. As would be expected for a random process, the smaller sample of the process offered by shorter simulation leads to a high level of variation As the simulation time is increased so the random process is more effectively averaged and the estimate of mean cost improves. Figure 3.7 suggests that a 16 second simulation time offers a reasonable compromise between high levels of cost variation and excessive learning times as the coefficient of variation falls below 10%.

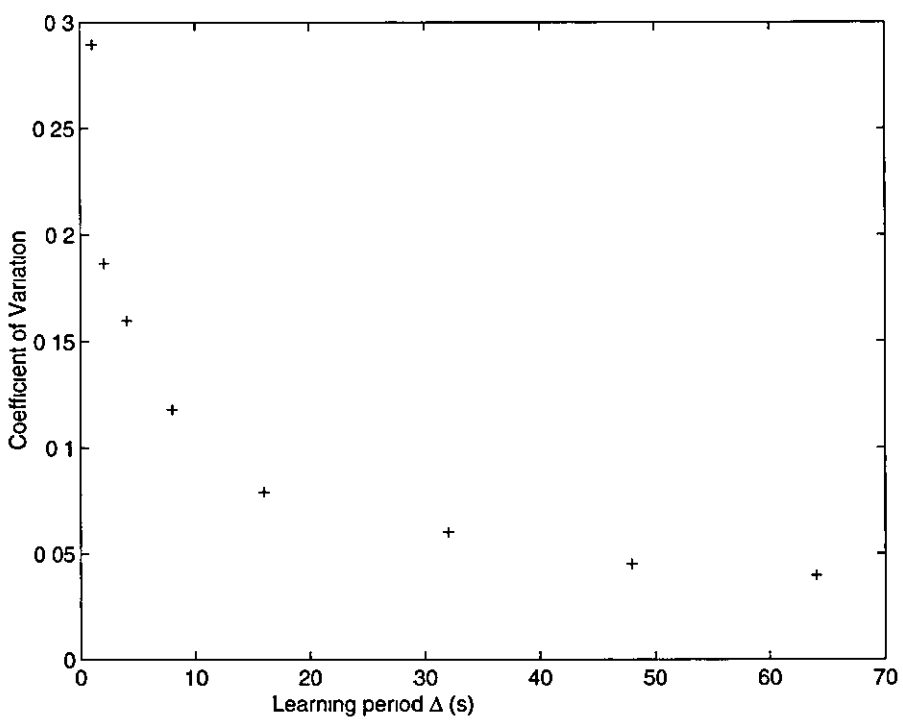


Figure 3.7 - Variation of simulation cost vs. simulation time Δ

3.6.2 Details of the learning schemes

At the start of the first learning stage the gains, k_i, were discretised within the ranges

$$\begin{aligned} k_1 &\in [-20000,0] \\ k_2 &\in [0,15000] \\ k_3 &\in [0,2000] \\ k_4 &\in [-4000,0] \end{aligned} \qquad (3\ 16)$$

These ranges then surround the known LQG optimal values, although it should be noted that similar ranges could be chosen from application of basic engineering knowledge of the system. In particular, the sign of the gains can be simply chosen from considering whether positive or negative feedback of each state moves the system to a more stable situation, i.e. positive 'spring stiffness' and 'damping' terms

Each parameter was discretised to three equally spaced values spanning the given range. Therefore, with $N = 4$ and $r = 3$, 81 possible actions are available. A stage of learning was deemed complete from (3.5) with $\eta = 0.5$ for both schemes. The scale factor $\lambda = 0.4$ was applied after each stage of learning to home in on a smaller region of the parameter space around the successful action from the previous stage. Final convergence, from (3.6), is then completed after six stages of learning.

The following parameter values, specific to the respective schemes, were used,

Scheme 1 - Automaton A: $\theta = 0.3,\ \delta = 0075,\ H = 10$

A suitable choice for δ depends on the disturbances being considered, larger values being required for less predictable environments A positive value for δ is essential in this scheme, to avoid the situation in which actions are reinforced *only* when the disturbance input is favourable simply by chance. The value of δ used was found to give best learning results from comparison with learning sets for $\delta = 0050$, and $\delta = 01$

Scheme 2 - Automaton B: $\theta = 0.1$

3.6.3 Performance analysis

Ten independent examples of learning were simulated for each scheme to gather a sample of automata results. It was seen, for both schemes, that the four-parameter controllers finally learnt from each simulation were close to the optimal values.

Figure 3.8 compares the results for Scheme 1, referred to as Automaton A, with the LQG optimal values. This shows both parameter values (mean ± one standard deviation) and theoretical expected costs evaluated via the system's Riccati equation. Although parameters vary quite markedly about the optimal values, the learning automaton achieves costs that are very close to optimal, confirming the results of Gordon et al. (1993) Table 3 1 summarises the cost results for Scheme 1, where the mean cost is seen to be 2.6990, only a 1 4 percent increase over the optimal value of 2 6621.

	Maximum	Minimum	Mean	Std dev.
Cost	2.7531	2.6793	2 6990	0.0266
Increase (%)	3 4	0.6	1.4	-

Table 3.1 - Automaton A cost performance

	Maximum	Minimum	Mean	Std. dev.
Cost	2.7024	2 6664	2.6837	0 0126
Increase (%)	1.5	0.2	0 8	-

Table 3.2 - Automaton B cost performance

Figure 3.9 and Table 3.2 record results of the same form taken for Scheme 2, referred to as Automaton B. The overall mean cost for Scheme 2 is just 0 8% above the optimal value and the cost variation for the 10 trials is half that of Scheme 1. One would expect that reduced cost variation is a result of reduced variation in the parameter values learnt. This is indeed the case for k_2, k_3, and k_4, but k_1 still shows a large variation about the mean. However, this is of little concern as a large variability of k_1 was anticipated, this being a relatively insensitive parameter for suspension control (Sharp & Crolla, 1987).

Though not apparent from Figure 3 9, the results show a correlation between the k_2 and k_4 parameters, with high values of k_2 being associated with low values of k_4 This anomaly is investigated further in Chapter 4.

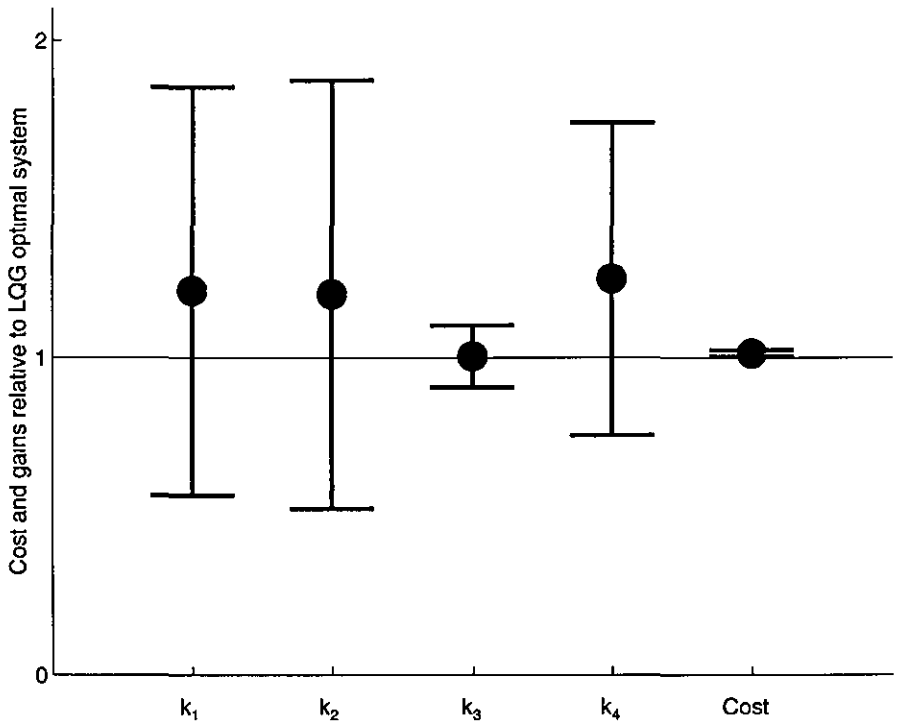


Figure 3.8 - Automaton A results relative to LQG values

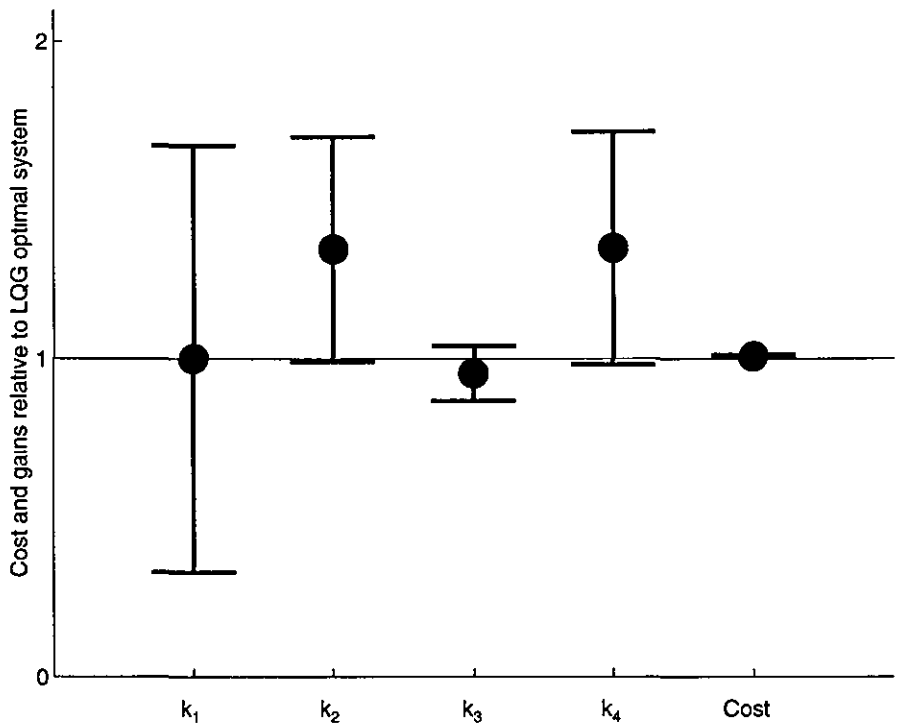


Figure 3.9 - Automaton B results relative to LQG values

3.7 Discussion

As also noted by Wu (1993), Scheme 2 performs better than Scheme 1 Not only is improved control achieved, but also the time to final convergence is considerably reduced under Scheme 2. Average 'real-time' learning for Scheme 1 was 11.5 hours compared to 8.3 hours for Scheme 2.

Scheme 1 also suffers from the number of free parameters in its definition - two more than for Scheme 2. In particular, parameter δ in equation (3.10) has to be set very carefully for successful learning under Scheme 1, whereas Scheme 2 has no such sensitive parameters.

As the action set is defined it is possible for either scheme to enter unstable regions of the control space. Each parameter being learnt is given an initial range of specific sign, with one extremum on a stability boundary at zero. It is then possible for the first stage of learning to 'choose' an action on the edge of the stable region, that action having at least one zero parameter value. The next stage of learning will centre its actions around this action, and hence some actions lie beyond the stability boundary Such a scenario could be avoided through reducing the learning rate parameter, θ, of either scheme so the automata spend a longer period assessing actions and thus have more chance of selecting an action away from stability boundary worries for the next learning stage. For general application, however, tuning of parameters alone to try and avoid stability concerns is not an option; the best action in a set could be one close to a stability bound. Another technique is required to handle such a situation satisfactorily, preferably without requiring any iterative parameter tuning

There is also scope for a possible reduction in the free parameters used to define the specific learning task considered here. A successful LQG formulation of optimal ride control requires additional terms in the performance index to constrain tyre deflection and suspension workspace usage. The relative weights applied to these terms decide the degree to which each aspect is controlled, and are generally attained through an iterative trial and error process, with the designer tuning the values subject to resultant system characteristics. Without recourse to the LQG technique, variation of the weights based on the results of iterations of learning runs would be a lengthy and unwieldy process. It would be more natural if the workspace usage terms could be removed from the cost function, incorporating a desired workspace usage as part of a specification of an acceptable controller. The cost function could then concentrate the learning task to improve ride performance as initially desired.

Chapter 4 - Development for On-line Implementation

The previous chapter has shown that the learning automata methodology is capable of learning good controllers from optimising for a given cost function As a consequence of these preliminary studies, two major revisions to the learning automata methodology are introduced in this chapter The first revision deals with the possibility of unstable actions being selected during learning. A second revision then allows the removal of 'constraint' terms from the original cost function so learning can concentrate on the primary optimisation for ride performance. These revisions, first presented in a paper by Frost et al (1996), take the learning automaton methodology from being primarily a simulation tool, to being suitable for application on a hardware task.

4.1 A Moderator

In the previous chapter it was suggested that if the learning automaton selects an action close to stability bounds as the 'best' action at the end of one learning stage, then the quantisation of a new action set for the next stage may result in unstable actions being available to the automaton. This situation has been avoided thus far through careful selection of the learning parameters A moderated learning scheme is now introduced which addresses this problem

For a general physical system, a basic engineering knowledge of the system affords some idea of the normal operational limits expected when the system is under stable control. In the case of a suspension system, for instance, the designer will know the limits of acceptable suspension deflection. If an applied control action results in the acceptable range being exceeded then the controller is clearly failing to meet its specification. In particular, if an unstable action were applied then it is likely that extreme limits will be exceeded very rapidly.

Considering the four states of the quarter-vehicle model used previously, an operational envelope can be identified as

$$\left.\begin{array}{ll} \text{tyre deflection:} & |x_1| < 25\text{mm} \\ \text{suspension deflection:} & |x_2| < 100\text{mm} \\ \text{wheel hub velocity:} & |x_3| < 2.5\text{m/s} \\ \text{body velocity:} & |x_4| < 1.25\text{m/s} \end{array}\right\} \quad \Omega \subset \mathbf{R}^4 \qquad (4.1)$$

These values are based on physical measurements from a typical car and define wide, yet reasonable, limits on the state variables. Any excursion beyond these limits can be considered as an instant failure of the controller, especially where an unstable controller is being applied, and this failure needs to be signalled to the learning automaton. This can be achieved by returning $\beta = 0$ to the automaton directly Effectively a further cost has been added to the environmental response function, J

$$J' = J + L \tag{4.2}$$

where L could be any additional costing function. Here however, any action that causes (4.1) to be violated is considered to be an 'unstable' action and should automatically 'fail', hence

$$L = \begin{cases} 0 & x \in \Omega \\ \infty & x \notin \Omega \end{cases} \tag{4 3}$$

A limit violation may occur at any time within the action trial period and the suspension is deemed to be in a potentially unstable state. In the practical case when this occurs, the system must be returned to a stable state quickly. To achieve this a moderating control action is needed. For the suspension control problem, a suitable control action is easily supplied by using a conventional passive suspension law as described in Section 2.1.1. The particular stabilising control employed here is:

$$u_0(t) = \begin{bmatrix} 0 & k_s & b_s & -b_s \end{bmatrix} x \tag{4 4}$$

with spring stiffness, $k_s = 20000\,\mathrm{N/m}$ and damping rate, $b_s = 2000\,\mathrm{N/ms}$.

The moderator is thus defined as the overseeing control that, upon observing a possible unstable situation during learning, will signal a 'fail' of the offending action to the automaton and apply a moderating control action to re-stabilise the system. It can be thought of as a 'panic button' hit by an overseeing supervisor to recover from very poor choice of actions, especially during the early stages of learning when unstable actions are most likely. Physically the moderator could be activated from transducer measurements of the state of the system. In the real world, activation could also result from an actual panic button supplied to a test engineer!

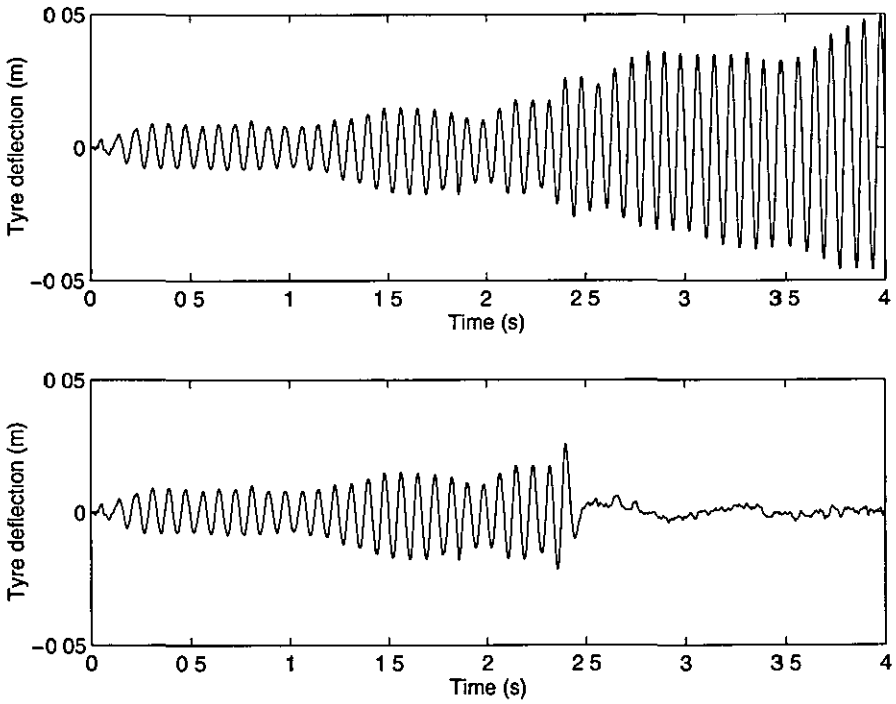


Figure 4.1 - Effect of the moderating control

An example of the effect of the moderator is shown in Figure 4 1. The upper plot shows the first five seconds of an unmoderated learning interval. The wheel displacement quickly becomes highly oscillatory. Continued testing of this controller is of little use, as it is obviously highly sub-optimal. In practical terms, continued testing of the controller would also lead to hardware failure as physical limits of the suspension are encountered. The lower plot in Figure 4.1 shows the same controller, but with the moderator taking over at around 2.4 seconds, where the pre-defined limit of tyre deflection is reached. The vehicle is returned to a more stable condition before the commencement of the next testing interval, and so there is little effect, if any, on subsequent learning. The effect of the moderator on the learning merely allows the automaton to ignore such unstable actions, and thus these actions are indirectly avoided at later stages of learning as they are never reinforced in the learning automaton.

Ten independent examples of learning are simulated to ensure the addition of a moderator does not disturb the learning process. This set of examples, referred to as Automaton C, include the use of the moderating control as defined in equation (4.1) and a comparison is made with Automaton B from Chapter 3 The learning automaton and learning task are kept the same as for Automaton B with two minor changes to test the moderator notion.

Primarily the initial gain ranges are shifted to include unstable control actions, c.f. equation (3.18):

$$\begin{aligned} k_1 &\in [-18000,2000] \\ k_2 &\in [-1500,13500] \\ k_3 &\in [-200,1800] \\ k_4 &\in [-3600,400] \end{aligned} \tag{4.5}$$

However, the learning environment is also made more challenging in a second respect. For Automaton B, each 16 second simulation was independent, with initial conditions set to zero on each iteration Automaton C introduces continuous simulation across iteration bounds, whilst still using a 16 second interval to test each action. Each action then 'inherits' a certain amount of dynamic response from the previous action via the initial conditions, which tends to increase the environmental noise.

Comparing the results of Automata B and C in Figures 3 6 and 4.2, it is seen that they perform very similarly; the mean cost for C is 2.6827, an increase of 0.77% over J_{opt}. Although there appear to be minor systematic changes in the parameter range obtained, the inclusion of unstable actions in the initial set has not led to any degradation in the learnt control system performance. Moving to the more realistic and challenging conditions offered by continuous simulation from one 16 second iteration to the next, has also had little, if any, effect on the learning process.

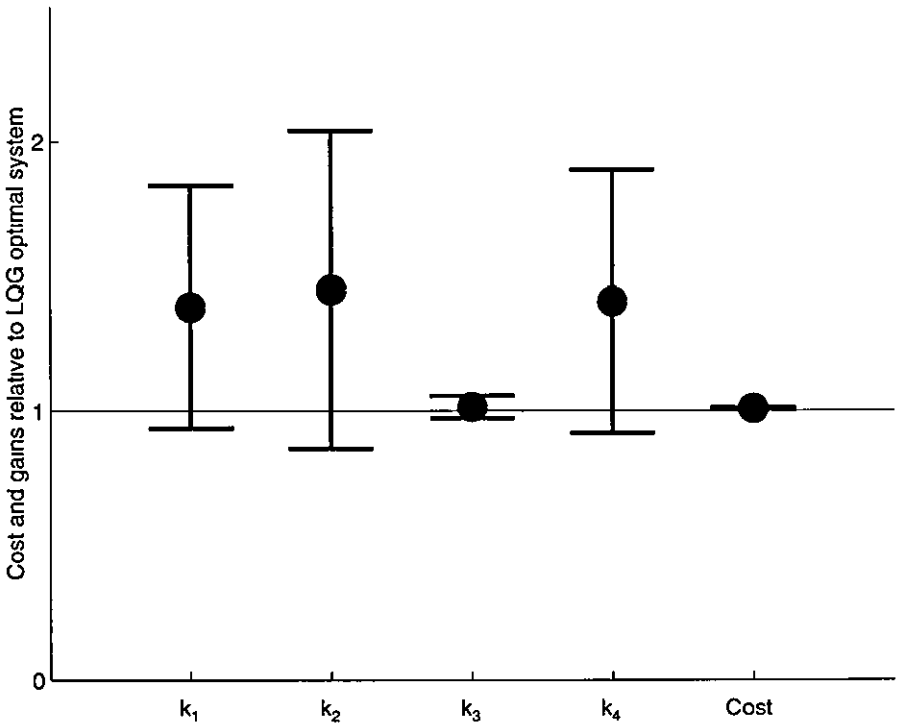


Figure 4.2 - Results for Automaton C

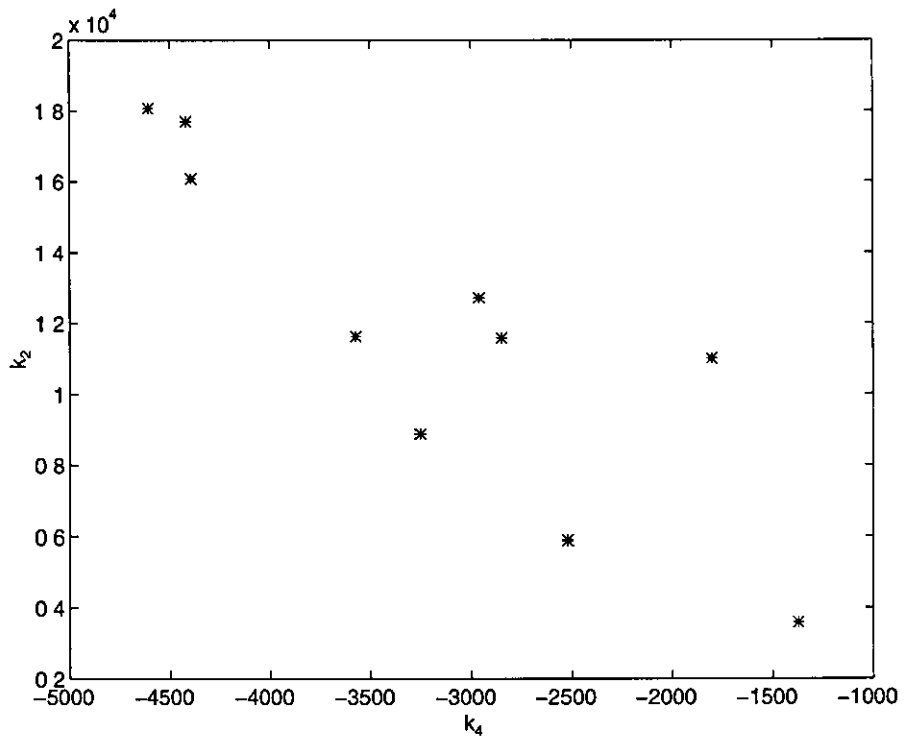


Figure 4.3 - k_2 vs k_4 from Automaton C

4.2 Parameter Correlation

Closer inspection of the gains from Automaton C controllers again reveals an apparent correlation between the k_2 and k_4 parameters, as noticed previously in the results of Automaton B. Figure 4.3 clearly shows this correlation with high values of k_2 being associated with high negative values of k_4.

Figure 4 4 shows a contour plot of expected costs against k_2 and k_4, holding k_1 and k_3 constant at their LQG optimal values. The plot includes 5 contour lines at each of 1%, 2%, 3%, 4% and 5% above the minimum optimal point that is pin-pointed with a marker. Here it is seen that, with white noise input the k_2, k_4 cost surface shows a sizeable 'flat valley' around the optimal point. Both Automata B and C generally manage to locate a controller within this valley, without being sensitive enough to locate a cost minimum especially close to the theoretical optimum

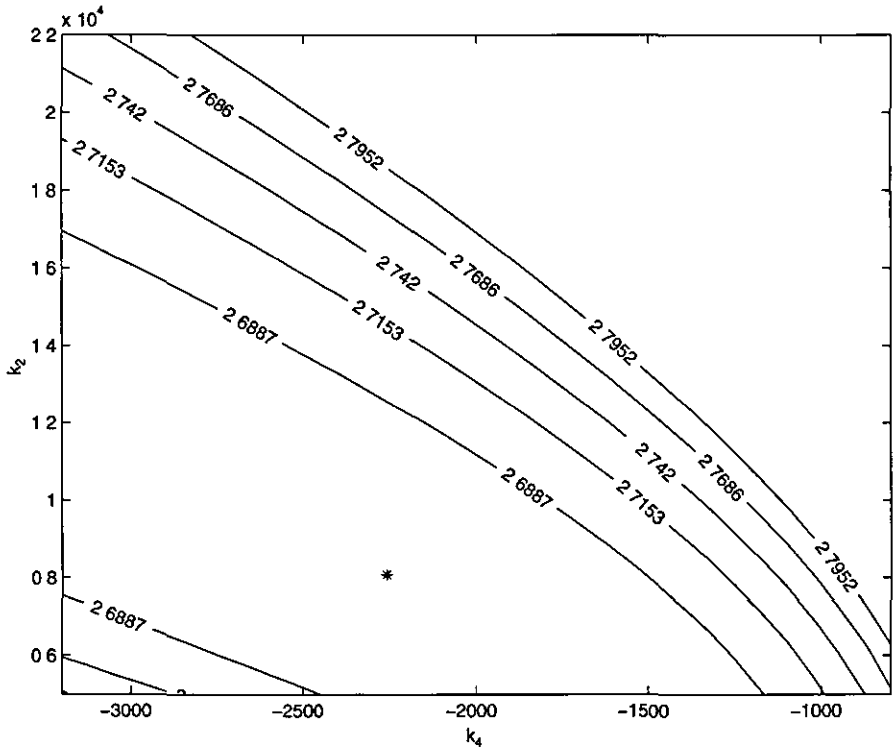


Figure 4.4 - Contour plot of cost vs (k_2, k_4) for white noise input

A high k_2 value in the controller is analogous to a stiff spring being applied in the suspension. A large negative k_4 value applies strong skyhook damping These terms evidently can act together for a wide range of values to return similar costs, although controllers from around a cost contour may produce significantly different system characteristics. For example, the sprung and unsprung mass PSD responses for two controllers from opposing ends of a 2.67 cost contour line are shown in Figures 4 5 and 4.6 respectively. The controller of Figure 4.5 is very 'stiff' and so exhibits no body

bounce resonance. Figure 4.6 illustrates, conversely, a 'soft' controller that allows a clear body resonance and a larger response around the wheel hop resonance. However, it also filters out higher frequency response, seen in the lower plot of Figure 4.6 as the body response drops off sharply beyond the resonant frequency. As both controllers result in the same cost, it is this available balance between resonant and high frequency response that accounts for the parameter correlation, with white noise input.

If real roads were 'white', the choice of controller one could employ would therefore be just one of taste in terms of the favoured body response. As mentioned earlier however, white noise is only a simple approximation to real road spectra, not possessing as much power at low frequencies. The suspension system has an inherent low frequency resonance of body bounce on which any ride optimisation should have most effect. By using learning automata for optimisation of suspension control, there is no longer a restriction limiting the driving input to a white noise process, as was the case to enable application of the LQG optimal control technique. Instead, there is the opportunity to apply a more realistic road spectra that could significantly alter the parameter relationships. Figure 4 7 confirms this, showing a cost contour plot, produced similarly to Figure 4 4, but using a measured Breakback Road spectra input in the place of white noise. It is seen that a parameter correlation is still evident, but to a much smaller degree. A cost minimum is now more clearly evident.

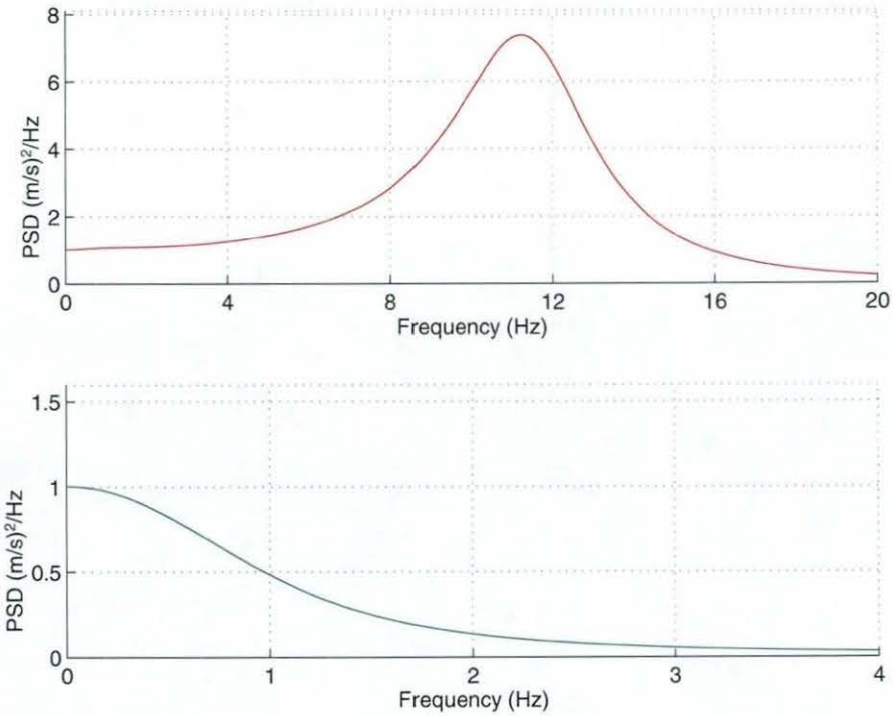


Figure 4.5 - PSD wheel velocity (upper) and body velocity (lower) responses with fullactive controller: $k_2 = 18342,\ k_4 = -4521$

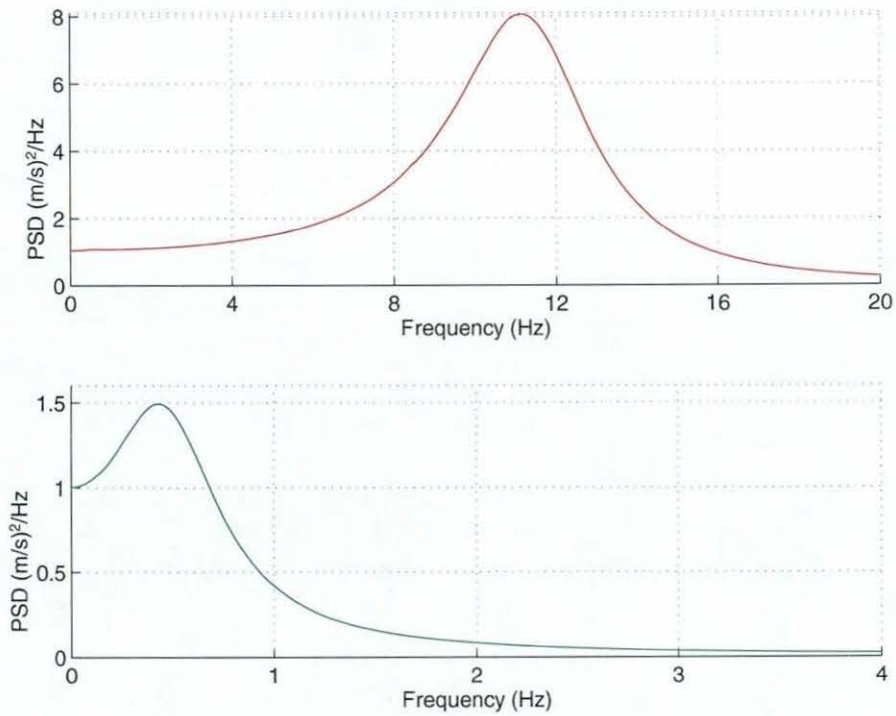


Figure 4.6 - PSD wheel velocity (upper) and body velocity (lower) responses with fullactive controller: $k_2 = 4045,\ k_4 = -1368$

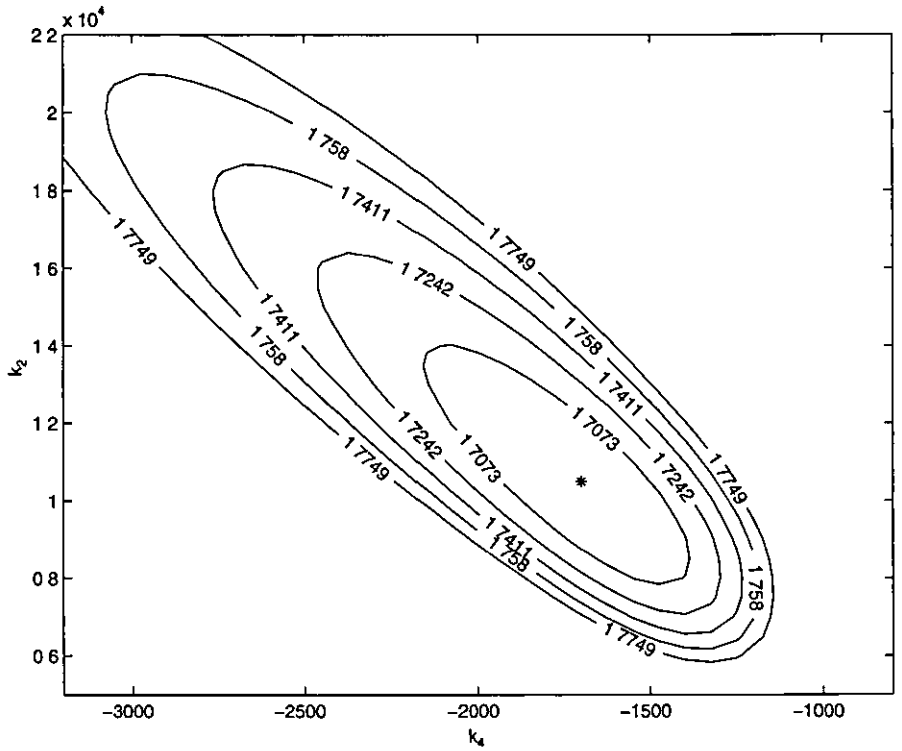


Figure 4.7 - Contour plot of cost vs (k_2, k_4) for Breakback road input

4.3 Learning on 'Real' Roads

A set of ten independent learning examples is taken on Breakback Road, referred to as Automaton D. The parameter results are summarised in Figure 4.8 A definite change in characteristic of the learnt parameters can be seen in comparison with the earlier results from white noise learning. k_1 and k_3 are similar to before, with k_1 showing wide variation as an insensitive gain for control purposes whilst k_3 has least variation Now, however, the previous relationship between k_2 and k_4 has been altered and Automaton D has learnt values for these parameters with reduced variation, as expected from the observation made on Figure 4.7.

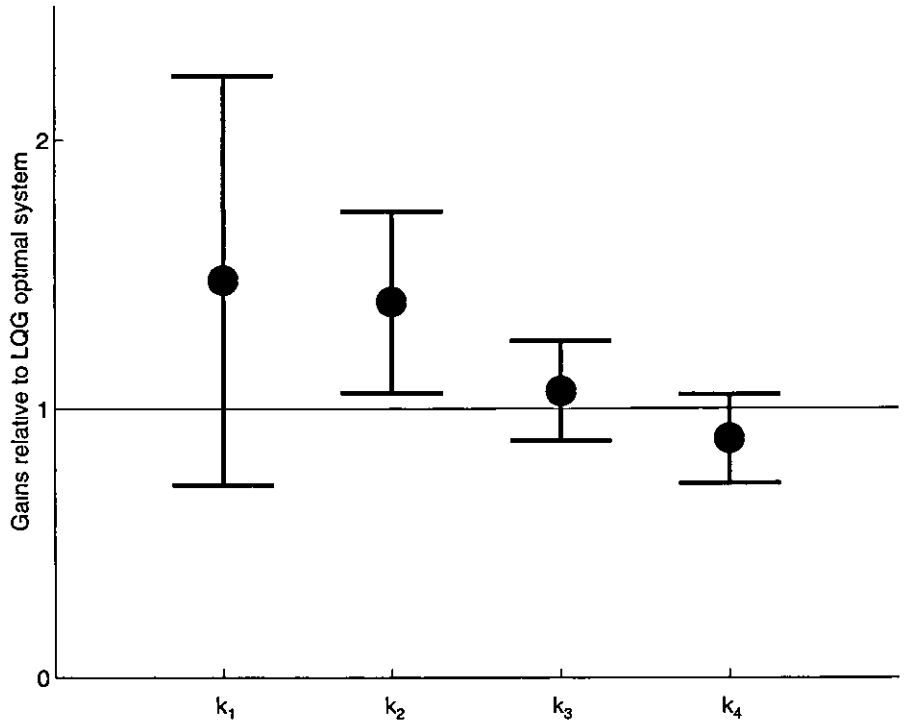


Figure 4.8 - Automaton D results - road spectra learning

To assess system performance for road spectra learnt controllers, it is no longer a simple matter to provide a theoretically optimal cost, cost evaluations over asymptotically large times are also very expensive to obtain. Instead the suspension systems from Automaton D are evaluated via their dynamic costs obtained from a single complete run along an independent section of road - Copt Oak Road from Section 2 2.2 A similar set of simulations over Copt Oak Road was also taken for Automaton C and comparison of the results is shown in Figure 4 9. Controllers obtained under Automaton D learning are seen to perform significantly better than those of Automaton C. This would seem quite natural; Automaton D, having experienced real road spectra during its learning phase, performs well on a similar input spectrum. Conversely, Automaton C, trained on a quite different input spectrum, namely white noise, then struggles to perform on the unfamiliar road spectrum

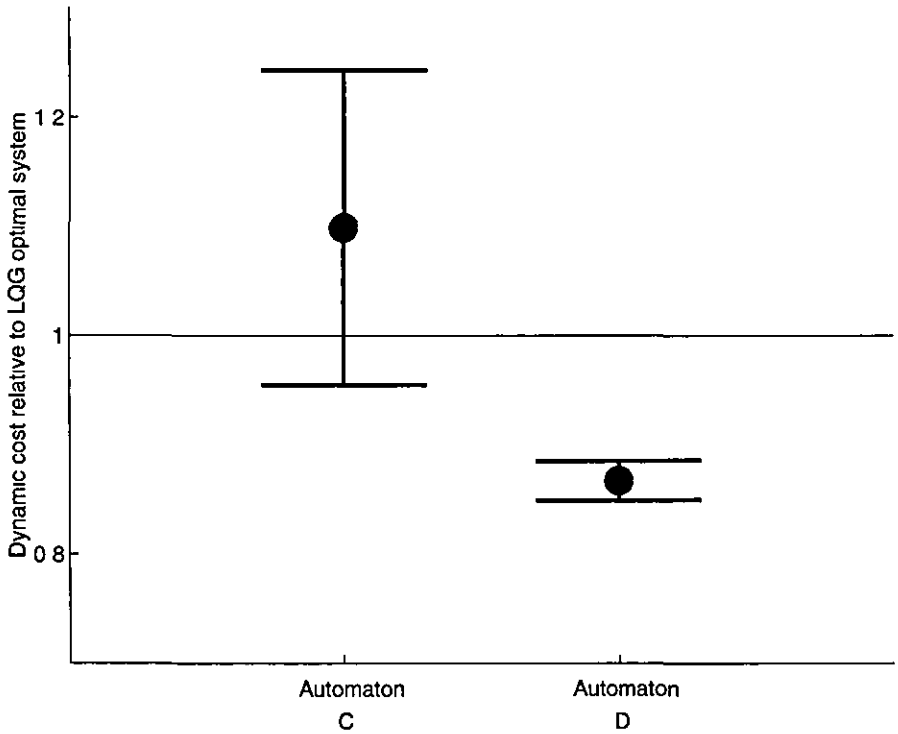


Figure 4.9 - Performance of learnt controllers on Copt Oak road

Analysis of the relative proportions of terms making up the dynamic cost reveals another side to this story. Automaton D has achieved the improved cost primarily through reduced suspension workspace usage - see Table 4.1 below. Automaton C actually gives better ride performance than Automaton D, as measured by r m s body accelerations.

RMS Response	System S1	System S2	Automaton C	Automaton D
x_1 (mm)	1.7	1.9	1.9	1.9
x_2 (mm)	6.0	18.9	21.1	13.4
$\dot{x}_4$ (m/s^2)	0.78	0.52	0 52	0 56

S1: Nominal passive: $k_1 = 0$, $k_2 = 20000$, $k_3 = 2000$, $k_4 = -2000$.
S2: LQG optimal full-active: $k_1 = -10406$, $k_2 = 8079$, $k_3 = 1029$, $k_4 = -2258$.

Table 4.1 - Controller evaluation on Copt Oak Road - RMS responses.

4.4 Reduced Cost Functions

A beneficial side-effect of the moderator is that limits of workspace usage are now restricted by something other than the cost function during learning. Costing

suspension and tyre deflections to indirectly constrain workspace usage is no longer required If deflections under the influence of any particular action become excessive then that action is 'failed' by the moderator and it is unlikely that any such action will achieve success with the automaton. Moderated learning thus allows a simplification in the cost function originally acquired through formulation of the optimal control problem. The cost function terms used for constraining workspace can be simply omitted, and the supervision and limitation of the system deflections is safely left to the moderator. Workspace can be utilised freely unless the unacceptable limits are exceeded The performance index is now simply

$$J(k)=\frac{1}{\Delta}\sum_{t-\Delta}^{t}\left(\dot{x}_4^2\right) \tag{4 6}$$

Note that the LQG methodology cannot be applied with such a cost function since the constraints cannot be imposed separately and formal optimisation would yield the null controller

$$u(t)=\begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}\mathbf{x} \tag{4 7}$$

Zero force to the sprung mass will indeed produce zero body accelerations, but this results in absurd unconstrained motions in the suspension and tyre. It is not a physically realisable or meaningful controller.

Once again, ten independent examples of learning were undertaken, with the performance index (4.6) applied, and using Breakback Road as input, referred to as Automaton E. Figure 4.10 summarises the parameter results, where a different characteristic is observed; k_3 is significantly removed from the LQG value

Dynamic performance results are shown in Table 4.2 alongside the results from Automata C and D for comparison.

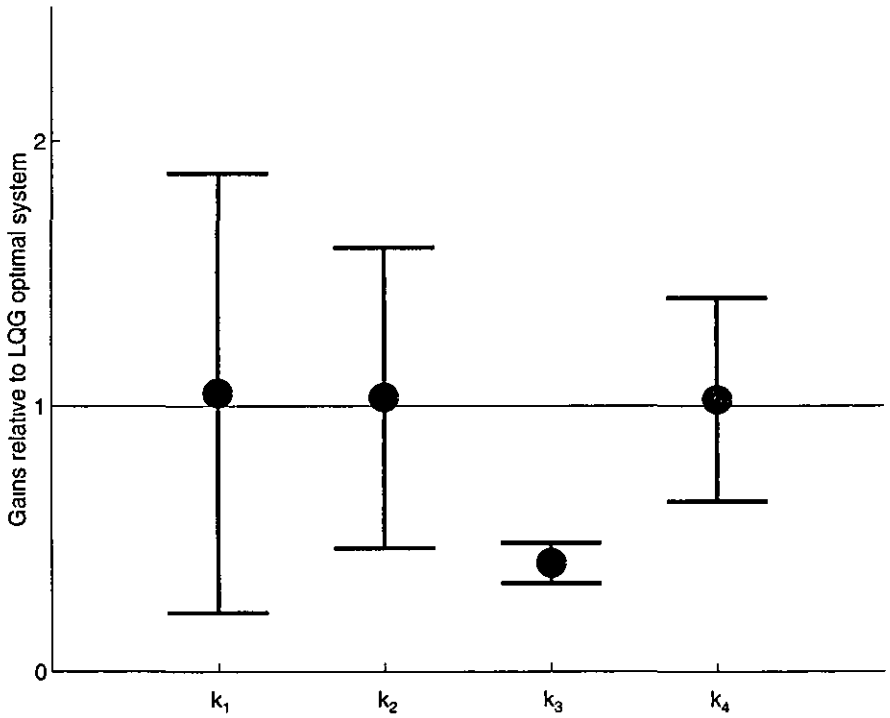


Figure 4.10 - Automaton E results - reduced cost function

RMS Response	Automaton C	Automaton D	Automaton E
x_1 (mm)	1.9	1.9	2.8
x_2 (mm)	21.1	13.4	29 1
$\dot{x}_4$ (m/s^2)	0.52	0.56	0.45

Table 4.2 Controller evaluation on Copt Oak Road - RMS responses.

The move to a reduced cost function is clearly beneficial. Body acceleration in simulation with Automaton E controllers is markedly reduced Both tyre deflection and suspension deflection usage has increased, a significant part of which may be explained by the reduced k_3 values corresponding to decreased damping between sprung and unsprung mass. The moderator has freed learning from the concern of workspace usage and hence a different form of result has been discovered. Meanwhile the workspace usage is supervised by the moderator to keep this usage well within the extreme limits it imposes

4.5 Discussion

The learning automaton methodology as originally applied to vehicle suspension control proved its viability as an optimisation technique in such a stochastic environment, but was flawed if the technique was to be considered beyond simulation studies for on-line application. The possibility of directly applying unstable actions to the environment without constraint was of particular note Initial formulation of the learning task around a cost function derived from LQG theory also meant that iterative adjustment of this function would still be required to tune the system to attain a specified characteristic.

Addition of a moderator has overcome both of these drawbacks and really moves the methodology towards being a useful practical tool. The moderator allows the automaton to concentrate on its learning task in the safe knowledge that an overseer is dealing with any bad situations before they get out of hand 'Concentrating on the learning task' for the task considered here means that workspace usage terms from the cost function can be removed and left to the moderator to keep in check. The simulation studies clearly show the learning automaton is then able to acquire controllers which perform admirably in comparison with passive and 'optimal' LQG controllers. Body acceleration and hence ride performance is reduced significantly whilst suspension and tyre deflections are less constrained but kept at reasonable levels by the moderator

Some initial engineering knowledge of the system has been applied to set the state limits for the moderator. These limits are easily found from physical limits imposed by hardware geometry and by measurement of peak values during normal operating conditions with a known stable passive controller. Acquisition of a suitable moderator for application of the technique to other systems would be possible in a similar manner. In this way, a certain amount of human knowledge and intuition can be directly built into the learning task and thus time spent tuning the terms in a cost function to produce controllers that return the required characteristics can be reduced.

Chapter 5 - Initial Vehicle Experiment

The development of the learning automaton methodology in the previous chapter paves the way for the first on-line application to a suspension system in hardware. The initial experiments described here made use of a vehicle fitted with semi-active suspension mounted on a four-post hydraulic shake rig. With a hydraulic post at each wheel station to provide vertical excitation to the suspension, experiments were run to test discrete learning automata on the physical system. This chapter documents these tests.

5.1 Vehicle and Rig Hardware

The test vehicle is a Ford Granada. It is essentially standard except that it is fitted with prototype continuously variable dampers, and instrumented with sensors at each wheel-station to provide a semi-active suspension system. The controllable dampers provide variable damping rate via actuation of an internal solenoid valve. This valve effectively varies the size of an orifice through which oil passes during extension/compression of the damper, thus varying the damping rate provided by the unit.

A standard method of controller implementation for such a semi-active system is to apply a 'clipped' active control law (Tseng & Hedrik, 1994.) The semi-active suspension is only capable of supplying a control force to oppose the relative velocity across the actuator, and then only within the actuator's operating envelope imposed through hardware effects. 'Clipping' the active control law refers to attempting to

apply the active control request within the constraints of the semi-active system. To apply the active control law the spring force is first subtracted to give a desired damper force, F_d If F_d lies within the damping force envelope, then the required damping rate lies between the softest and hardest settings, and is deduced from linear interpolation. Where F_d lies outside the damper velocity-force envelope the control is 'clipped' to 0% or 100% damping rate, as appropriate, as the nearest achievable value.

A PC fitted with a TMS320C30 digital signal processing (DSP) board enables control of the damper units. The PC and DSP access a shared dynamic memory area via which the PC interacts with the controlling program, running on the DSP, for on-line variation of control parameters and data acquisition The control action for each damper is applied by the DSP process via a damper drive module This module converts control voltage signals from the DSP board to pulse-width modulated (PWM) current signals supplying the solenoid valve in each damper. This signal modulates the damping rate, whilst, additionally, the high frequency oscillation of the PWM drive signal acts to vibrate the solenoid sufficiently to prevent sticking of the valve.

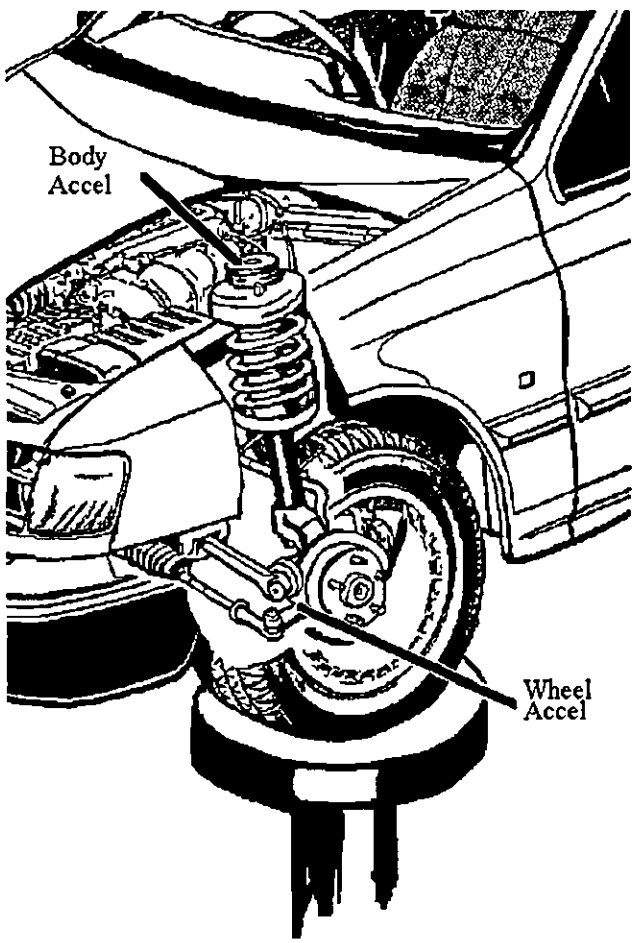


Figure 5.1 - Front corner of vehicle: sensor location

The sensor set at each corner of the car comprises two piezo-resistive type accelerometers. These accelerometers are rigidly mounted in the vertical plane at the wheel hub and top of each damper pinchbolt respectively - see Figure 5.1 The study by Best (1995) documents the instrumentation of this vehicle more thoroughly. Details of system identification carried out on the vehicle to ascertain vehicle parameters and characteristics of the variable dampers are also given there. Furthermore, development of Kalman filters to provide on-line state estimation is described.

The vehicle is mounted on a four-post hydraulic shake rig to excite the suspension systems. Freedom of movement is maintained by liberally greasing the tyre contact patch. This minimises lateral tyre forces that may restrict freedom of vertical wheel movement introduced from the geometry of the suspension throughout its stroke A minimal constraint arrangement is applied to prevent the vehicle from falling off the rig.

The drive signals for the hydraulic rig are shaped as a Robson road as described in Section 2 3.2. Storage constraints of the rig's computer control system limited the length of the drive signals that could be stored to 200 seconds at 204 8Hz. Prior to rig operation, drive files are prepared describing signals with the above characteristics Each drive signal is shaped with a ramp function at each end, to steadily bring the rig up from, and back down to, zero displacement after each drive segment. In this way a drive signal can be used repeatedly to give a continuous 'test track' for learning. Each rig actuator is driven by an independent drive signal to maximise independence of learning at each corner.

5.2 Learning System

A single discrete learning automata, as developed in the preceding chapters, is applied at each corner of the vehicle to learn the parameters of the three term variant of the familiar linear feedback control law

$$u(t) = \begin{bmatrix} 0 & k_2 & k_3 & k_4 \end{bmatrix} x \qquad (5.1)$$

where $u(t)$ is the required control force. k_1 is set to zero as it has been shown from theory to have little effect on control performance, for a full-active system (Sharp & Crolla, 1987). Also feedback of tyre deflection is a high frequency effect in relation to the bandwidth of the actuators, so little is lost in discarding this term.

Each automaton acts independently of the others to learn its own control gains, so maximising the available rig time with four learning automata running concurrently. A level of interaction will occur between wheel-stations via the body dynamics, but this interaction is small enough for each corner to be considered as independent of the others.

The initial range of the learnt parameters is as for the earlier full-active simulation studies

$$\begin{aligned} k_2 &\in [0,15000] \\ k_3 &\in [0,2000] \\ k_4 &\in [-4000,0] \end{aligned} \tag{5.2}$$

with each gain quantised to 3 values, giving 27 actions per corner learning automaton. The learning automaton parameters are also as before, $\delta = 0.1$, $\eta = 0.5$, $\lambda = 0.4$.

A moderator is applied, with limit values set as before (see equation 4.1), although it is unlikely that it will come into effect. A semi-active suspension is stable by its very nature, with any control setting of the actuator still resulting in dissipative vibration reduction, therefore the moderator, originally developed to catch potentially unstable situations, would only be activated at the most violent suspension excitation with the softest damper settings. Prudent choice of the driving rig input, primarily to protect the rig itself from excessive actuation, tends to remove the possibility of moderator involvement in the learning process.

The system architecture for a single corner of the vehicle is shown in Figure 5.2. Two timing loops are present: the control loop operating at 500Hz and the learning loop operating on the 16 second learning iteration over which any action is trialled.

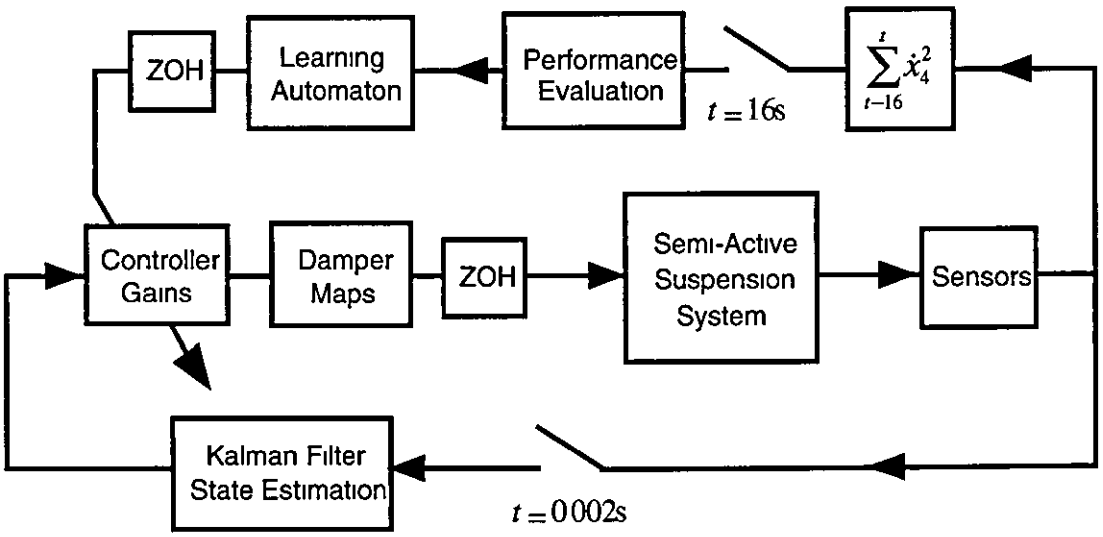


Figure 5.2 - System architecture

5.3 Controller Performance

Three trials were run for approximately 1000 iterations each, representing less than five hours of rig time per trial. A different randomised Robson road was generated for each trial. All the automata achieved two stages of convergence and would probably have converged further had more rig time been allocated.

Figure 5.3 shows a typical reduction in the cost that is achieved during learning, the occasional rise coinciding with the re-quantising of the learning automaton The periodic nature of the cost plot is explained by the wrapping of the rig input every 200 seconds

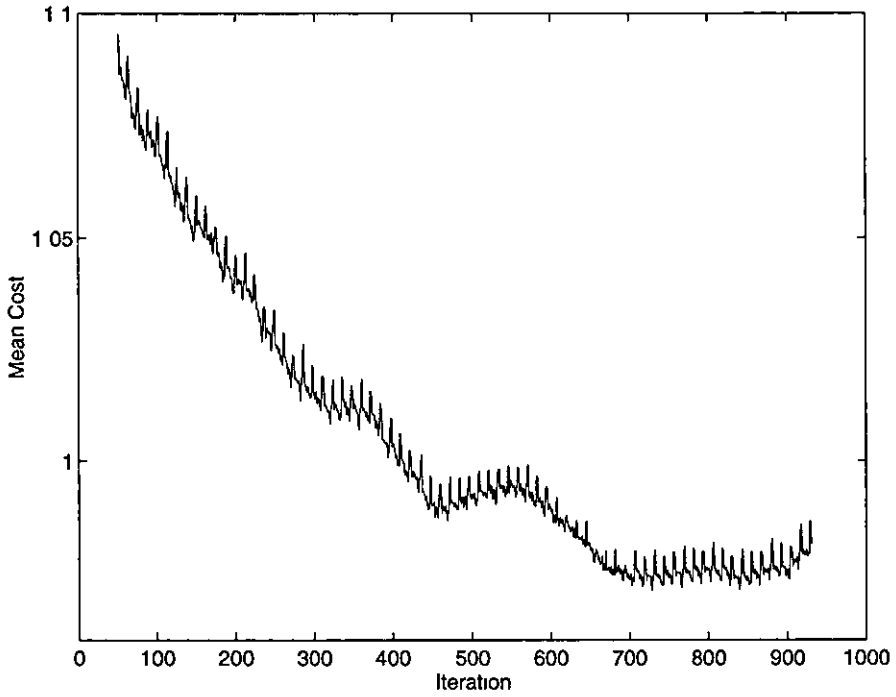


Figure 5.3 - Typical mean cost reduction over time

	Corner A	Corner B	Corner C	Corner D
Firm	-49 2	-39.8	-46 6	-32.9
Soft	2.1	-5.2	-10.5	-0.4
Automaton 1	7.8	6 0	2.5	8 6
Automaton 2	7.6	6 2	4.8	8 2
Automaton 3	6.6	4.7	4.9	6.5

Table 5.1 - Percentage improvement in RMS body acceleration over nominal passive damper setting (- sign for degraded performance)

To ascertain the relative performance of each learnt controller, the body acceleration was measured during a complete pass of the vehicle over an independent section of road, with the same controller gains applied at all four corners. Three tests for comparison were also carried out with the suspension control set to passive damper settings: nominal, equivalent to the vehicle being fitted with standard production dampers, firm, with the maximum damping rate set; and soft, with the minimum damping rate set. Table 5.1 gives the percentage improvements in r.m s. body acceleration compared to the nominal passive setting. A typical power spectral density

(PSD) of body acceleration for a learnt controller is compared with that of the nominal passive setting in Figure 5.4 where it is apparent that the learning automaton has identified a controller which significantly reduces the body bounce response at around 1Hz.

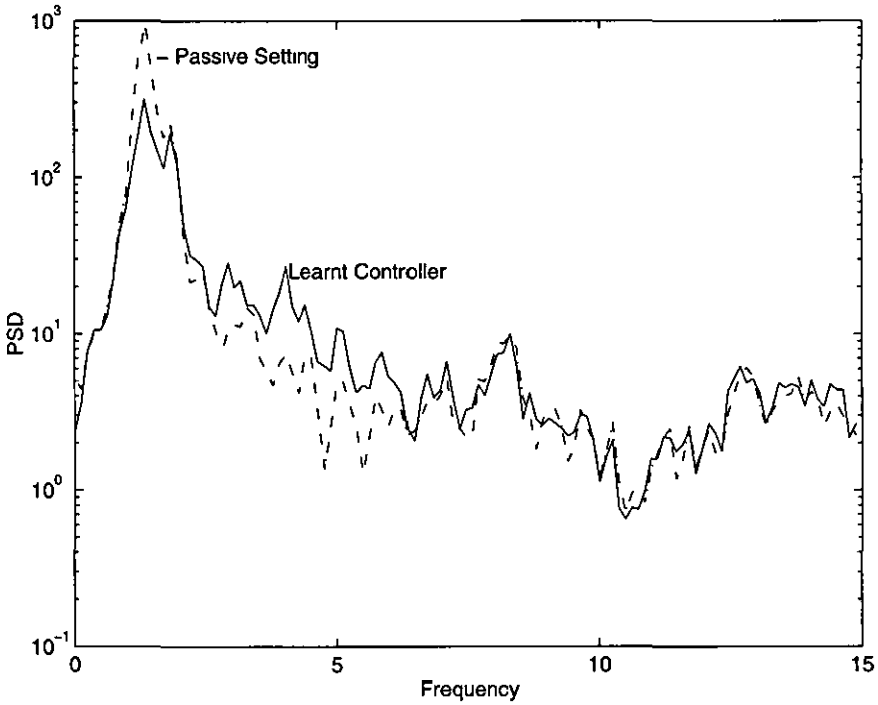


Figure 5.4 - A comparison of body acceleration PSDs

5.4 Discussion

This practical study has confirmed the promise of the previous simulation studies. From Table 5 1 it is seen that the learnt controllers consistently provide reduced r m s response compared with the nominal passive controller. Also an improved control was attained in comparison with setting the damping rate to soft. If the rig input had been of insufficient power to excite the body modes of the vehicle then the best level of control would result from simply setting the damping rate to soft at all times.

The discrete learning automaton has been shown to be capable of learning despite the high level of noise inherent in the hardware implementation. Even with only two stages of learning complete, the automata have identified regions of the action space that produce controllers able to significantly reduce the body bounce response in comparison with the passive control.

Chapter 6 - Development of the CARLA

Studies in the previous chapters have all made use of traditional learning automata with action sets consisting of a finite number of discrete actions These studies have shown that the learning automata methodology can be successfully applied as an optimisation technique in the presence of high levels of uncertainty and noise. Its success has been demonstrated on-line in learning ride optimising controllers for a semi-active suspension system on a road-going vehicle. However, a number of limitations of the discrete learning automaton have been noted. In particular the discrete action set limits the thoroughness of search over an action space. Using the discrete learning automaton as an optimisation tool, it is quite possible that optima may be missed as they lie between action points. Also, by increasing the number of actions to more densely cover the parameter space, or by considering higher dimension learning tasks, the action set size quickly becomes large enough to significantly slow the learning process, possibly to the point of learning becoming inconclusive. Interconnected or hierarchical automata may be used to ease this problem, but these still suffer from the inherent limitations of the discrete nature of the methodology

The technique of multiple stage learning with action space reduction at each stage (Wu 1993), implemented in the studies of previous chapters, has enabled increased thoroughness of parameter space search, but has itself introduced other limitations. At completion of each stage of learning a new action set is formed around the 'successful' action of the last stage, covering a reduced area of the parameter space. In this manner a more thorough search of a region is made, however this has also forced a convergence. This formulation of learning automata can be shown to suffer for this; should the automaton hastily converge on an action away from the global optimum, possibly towards a local optimum, then it is unlikely, if not unable, to correct its early error and find the global optimum Enforcing convergence also reduces the effectiveness of application in non-stationary environments. Were the environment to vary between stages of learning then the automaton is again unlikely to adapt to the change and locate a shifted optimum.

Santharam et al. (1994) proposed an alternative learning automaton formulation, which went some way to overcoming a number of the inherent limitations of discrete learning automata outlined above. The automaton they proposed employs a continuous action probability distribution in place of the discrete action probability vector of discrete automata The continuous distribution maps to a continuous region of the action space Random action selection based on such a distribution then gives, in discrete

automaton terms, an action set with an infinite number of members, immediately inferring a complete search capability. The particular formulation used by Santharam et al described the whole probability distribution function via just two descriptive variables, the mean and variance of a Gaussian distribution. Although this learning automata was shown as capable of locating minima in a stochastic optimisation application it was prone to locating local minima.

This chapter introduces the Continuous Action Reinforcement Learning Automaton (CARLA), a new formulation of a continuous action probability learning automata with a generalised representation of the probability distribution designed without prior knowledge of the work of Santharam et al.. The benefits of this algorithm over discrete automata enable

- a complete search of a parameter space for the global optimum, within a pre-defined range
- an action set size which increases proportionately with parameter space dimension
- full adaptability in non-stationary environments.

This chapter explains the concept of CARLA more thoroughly and describes the implementation methodology adopted to maintain efficient coding and subsequent execution of the algorithm. Chapters 7 and 8 will subsequently demonstrate the capabilities of this new automaton formulation, including comparison with the CALA devised by Santharam et al.

6.1 The CARLA Concept

The key variation between traditional discrete learning automata and CARLA lies in the representation of the action selection probability distribution, replacing the discrete probability vector describing the state of the traditional automaton with a continuous probability distribution. The motivation for this can be informally described with reference to Figure 6.1. Recalling the general algorithm of a learning automaton - Figure 3 3 - the automaton output, an action α, is selected based on a probability distribution. In the discrete case the automaton has a finite number of actions from which to choose and so the probability distribution is discretised over those actions (Section 3.3). Through reinforcement the 'best' action is chosen as its associated probability of selection tends to one. Efficient learning is experienced when there is a small number of actions available to the automaton. For example in Figure 6.1(a) six actions cover a parameter range of 0 to 10. However, a small action set does not

provide a thorough search across the range; optima may easily lie at values between those selected as action values. By increasing the number of actions to cover the range more thoroughly, as in Figure 6 1(b), it is found that a discrete automaton has an increasingly difficult task to distinguish between actions sufficiently to converge to any single action; for a typical stochastic environment a number of actions around any optima may give very similar responses.

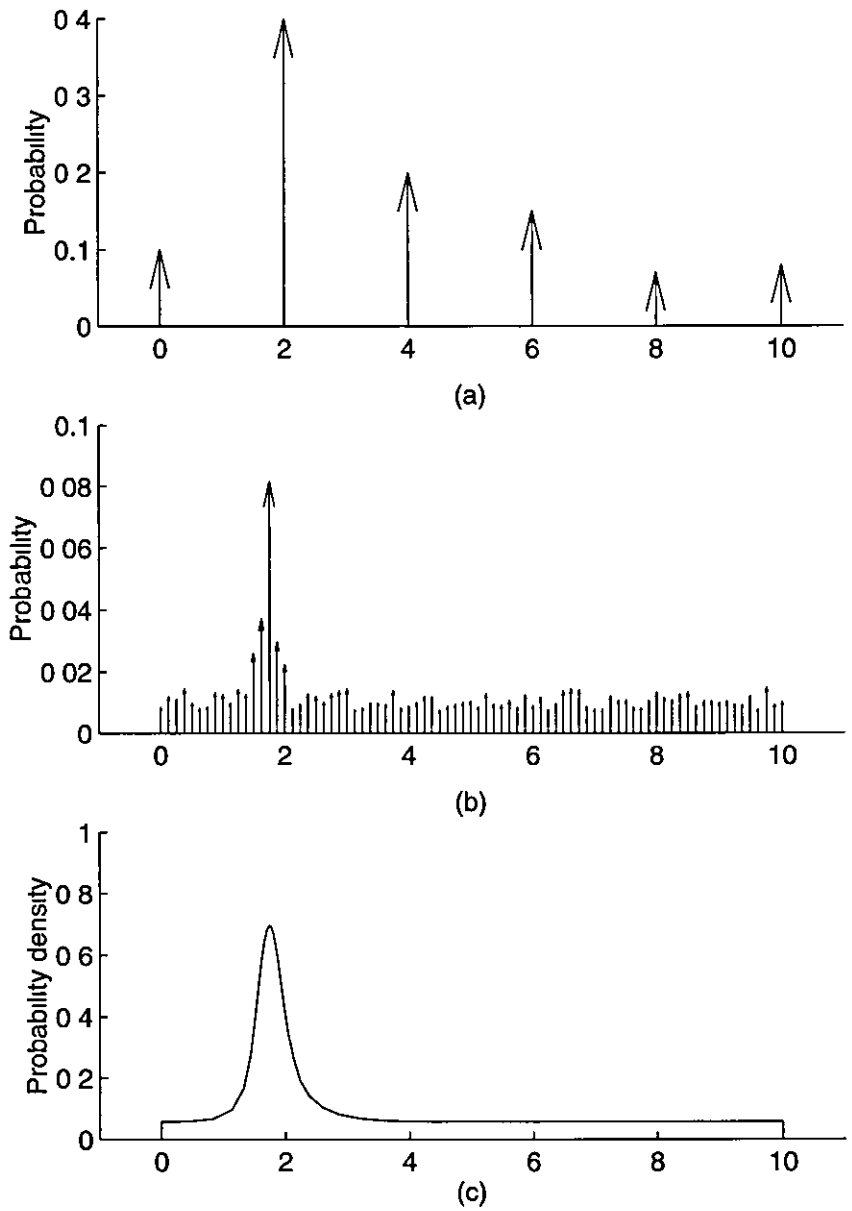


Figure 6.1 - Motivation for continuous probability density function

As action space coverage is improved by increasing the number of actions, so, in the limit, complete coverage is achieved by an infinite number of actions. This can in fact be easily implemented, for the single dimension case, by applying a continuous probability distribution as the automaton state. In so doing, the 'stepped' nature of the

discrete cumulative probability function will be replaced by a continuous monotonically increasing curve Now action selection on uniform random numbers between 0 and 1 will result in an action set with effectively an infinite number of members within the limits of the action space defined

The discrete automaton rewarded a single action to the penalty of all others. The reward of a single action in a similar way for a continuous action space becomes intractable for implementation Instead a generalisation can be applied which aids implementation whilst also seeming intuitively 'correct' It would seem reasonable to assume that in a region of the action space around any action tested in the environment a similar level of performance could be expected, and so similar reward may be given. Less confidence in this assumption will be had for actions lying further away from that trialled, and so less reward should be applied. This can be implemented by applying a reward function which applies the most significant level of reward for the trial action with monotonically decreasing levels of reward for more remote actions. The reward function used in a CARLA is a Gaussian distribution reward function, of the standard form

$$\frac{1}{\sigma\sqrt{2\pi}}\, e^{\frac{(x-\mu)^2}{2\sigma^2}} \qquad (6.1)$$

describing a 'bell' shaped function with mean μ and variance σ Addition of such a curve to a continuous probability function, with its mean centred on the trial action, implements the generalisation of reward outlined above, with maximum reward at the trial action and reward falling away for more distant actions.

Figure 6.1(c) then shows the culmination of the above points with a continuous function representing the internal state of the automaton, a continuous probability distribution. A parameter range defines the limit of a continuous action set mapping one-to-one to the probability distribution. Any trial response of a single action is generalised to an area of the action space around it. Figure 6.1(b) shows a strong region of performance around $\alpha = 1.75$ but the discrete automaton struggles to converge to a single action in preference to the near neighbours In contrast, Figure 6 1(c) also shows such a strong region, but is not required to converge to any single action; the CARLA has generalised the learning process to locate an area of the action space which performs well.

6.2 CARLA Algorithm

A CARLA follows the same basic algorithm given for a learning automaton in Figure 3 3, and is similar in many aspects to the discrete automata described in previous chapters. The differences lie in the application of a continuous function to describe the automaton state vector; the action selection probability distribution.

Applying a continuous probability distribution leads to difficulties when considering higher dimension (N>1) action spaces where a suitable mapping between the two is required. For the discrete automaton the action set is formed from all combinations of discrete action values and then the probability distribution is simply generated by assigning each discrete action a probability of selection. The action set could easily consist of higher dimension actions, but the discretisation process reduces these to an action vector, easily mapped one-to-one with a probability distribution vector.

Trying to achieve a similar implementation for a continuous case is not viable, as it would require a mapping of an N-dimensional action space to a probability distribution of a single dimension. An alternative is a one to one mapping of the N-dimension action space to an N-dimension probability distribution This is also not a viable solution however, as it would require a representation of a continuous probability distribution that can satisfy equation (6.2) after applying an N-dimensional reward to this function at any iteration.

$$\int_{N}\int\int p.\mathrm{d}\alpha = 1 \qquad (6.2)$$

The approach adopted overcomes this difficulty by only implementing CARLA for the 1-D case, i.e. only taking an action set of a single variable Learning tasks of a higher dimension are easily accommodated by utilising the interconnected arrangement of multiple automata, described in Chapter 1.

The formulation of a CARLA is now described, with reference to the major aspects of the general learning automaton algorithm of Figure 3.3

6.2.1 Initialisation

As the CARLA is to operate on only a single parameter in an action space it only requires the maximum and minimum values, α_{min} and α_{max}, to define its action set, the continuous parameter range over which it will operate. Initialisation consists of generating the continuous probability density function between these values. In satisfying (6 2) the specific case is

$$\int_{\alpha_{min}}^{\alpha_{max}} p \, dx = 1 \tag{6.3}$$

As for the discrete case, no *a priori* knowledge of the action space is to be assumed, so the initial probability of selection is to be equal for any action. From (6.3) it follows that the initial probability magnitude across the range is therefore

$$p_{init} = \frac{1}{(\alpha_{max} - \alpha_{min})} \tag{6.4}$$

An example of CARLA initialisation is shown in Figure 6.2. A parameter range is defined over the range $\alpha_{min} = 0$, $\alpha_{max} = 10$. Satisfying (6 3), $p_{init} = 1/10 = 0\,1$ and so a uniform probability distribution function is defined, with associated cumulative probability function shown in the lower plot.

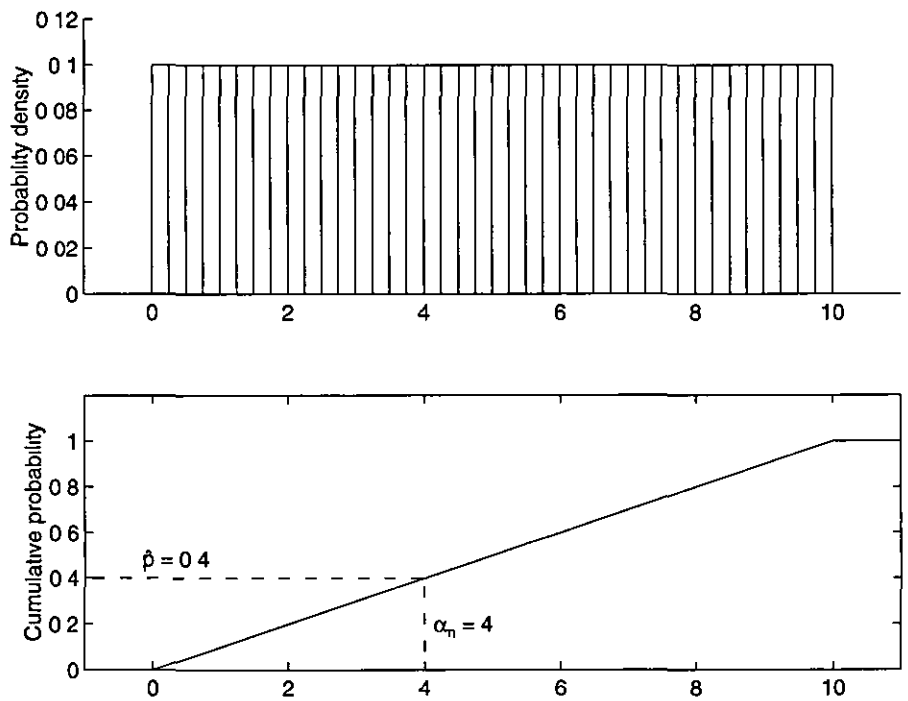


Figure 6.2 - Initial continuous probability density function and associated cumulative probability function

6.2.2 Action selection

Action selection is achieved via the cumulative probability function in much the same manner as for discrete automata. A uniformly distributed pseudo-random number between 0 and 1 is taken and the action corresponding to this value of cumulative probability is selected as the action for trial at iteration n, i.e. for random probability $\hat{p}$, action $\alpha(n)$ is found via

$$\hat{p} = \int_{\alpha_{min}}^{\alpha(n)} p \, dx \qquad (6\,5)$$

Figure 6.2 shows an example action selection for $\hat{p} = 0.4$.

6.2.3 Reinforcement scheme

The reinforcement scheme implements a generalisation of the environment response to apply reward across a region of the action space, centred on the last trial action and diminishing with distance from the action

The reinforcement is applied via addition of a reward function to the probability distribution. The CARLA reward function is defined as the Gaussian function

$$\frac{\beta . g_h}{(\alpha_{max} - \alpha_{min})} . e^{-\frac{1}{2}\left[\frac{\alpha - \alpha(n)}{g_w (\alpha_{max} - \alpha_{min})}\right]^2} \qquad (6.6)$$

and hence the reward is centred on the action tested at iteration n. β is the performance index received from the environment. g_w and g_h are dimensionless parameters that determine the relative width and height of the reward function. They are user-defined, and kept constant throughout a learning run.

A reward-inaction scheme is used by the CARLA following the success of such a scheme for discrete automata. No penalty is therefore applied for actions returning a poor environment response.

Learning generalisation, g_w

g_h defines the 'spread' of any reinforcement that the automaton can apply to the probability distribution around an action to effect a reward. Comparing (6.6) and (6.1) it is seen that $\sigma \equiv g_w (\alpha_{max} - \alpha_{min})$, and so g_w is a dimensionless parameter that describes the standard deviation of the reinforcement Gaussian distribution function as a fraction of the parameter range, $(\alpha_{max} - \alpha_{min})$.

A small value for this parameter will give a thin 'spike' of reinforcement, very local to the tested action. In this case the automaton will act much like a discrete automaton with a large action set; many reinforcements will be required in a successful region of the action space for the rewards to conglomerate sufficiently for that region to dominate.

A large value for this parameter will over-generalise any reward to apply reward across a wide area of the action space, the automaton will continue to explore the action space extensively as any reinforcement will give little differentiation between the relative performance of neighbouring actions.

Empirical tuning of this parameter has found that a value of $g_w = 0.02$ gives a good balance in the above trade-off. This value is not critical to learning performance, or particularly sensitive, and so the above value is used in all CARLA studies from hereon.

Learning rate, g_h

g_h acts as a learning rate parameter, defining the basic magnitude of reinforcement that can be applied at each iteration This parameter is comparable to 6 from discrete automata, and selection of a suitable value for g_h follows similar guidelines, i e. it is somewhat dependent on the task being considered, and is chosen to control the learning rate such the automaton is not 'jumping to conclusions' and erroneous decisions through rapid learning, nor taking too long so learning becomes ineffective because of indecision. The stochastic characteristics of the performance index will also have some bearing on the learning rate chosen.

A typical value of $g_h = 0.3$ has been found from empirical tuning to perform satisfactorily in the majority of situations, including most of the CARLA studies in this document. A demonstration of the effect of an excessive value of g_h will be seen in Chapter 7

6.2.4 Reward/penalty response from the environment

The environmental response from a cost function, $J(n)$, is compared against previous values to return a performance index β as before. The critic used here to produce the performance index is as defined for the S-model discrete learning automata of previous chapters, so, for a minimisation task

$$\begin{aligned} J_{\min} &= \min\{J(1), J(2), .., J(n)\} \\ J_{\text{med}} &= \text{median}\{J(1), J(2), ..., J(n)\} \end{aligned} \tag{6.7}$$

$$\beta(n) = \max\left\{0, \frac{J_{\text{med}} - J(n)}{J_{\text{med}} - J_{\min}}\right\} \tag{6.8}$$

6.3 Convergence

The discrete automaton of earlier chapters included a convergence property to facilitate a more thorough search of the action space. This technique required multiple stages of learning to converge to a successively smaller action space, concluding a 'learnt' action when a convergence limit was attained. For the CARLA, the whole action space is considered from the outset of learning; explicit convergence in an attempt to refine learning is not required. Instead, the learning process can progress towards a more 'natural' convergence as successive application of the reward function around any region allows that region to dominate. For example, Figure 6.3 shows an automaton in which two regions of the parameter space are beginning to dominate, around $\alpha = 2.5$ and $\alpha = 6$, giving increased probability of action selection in that region Ultimately, as the automaton learns the optimal action region the probability distribution will be dominated by a single peak in density.

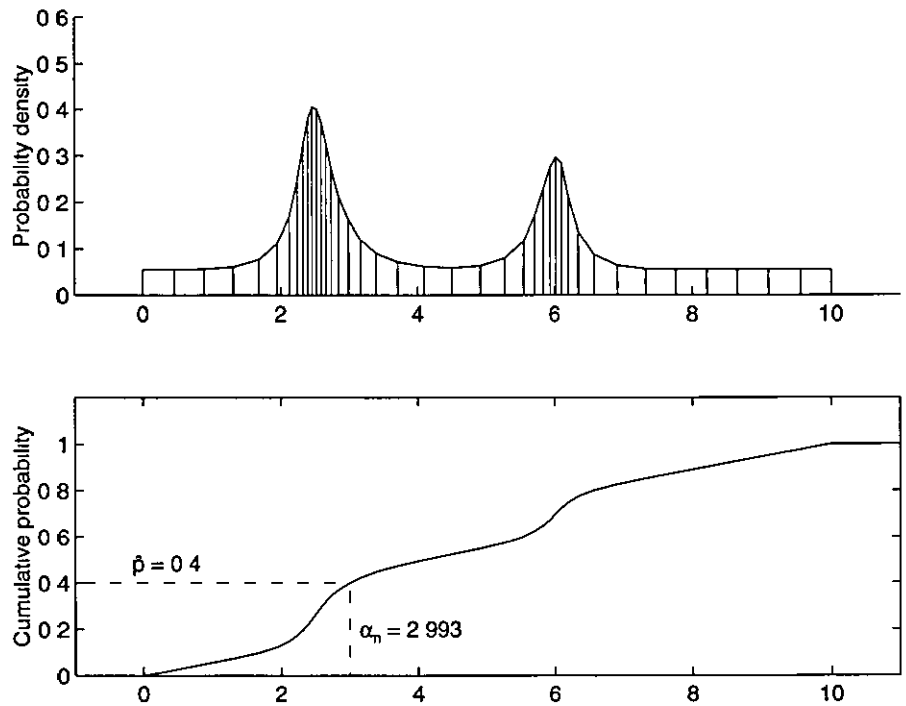


Figure 6.3 - Active CARLA probability density function and associated cumulative probability function

6.3.1 Convergence measures

Under ideal reinforcement, applying full reward at the optimal action on each iteration, the probability distribution will change as shown in Figure 6.4. Each curve exhibits a dominant region around the rewarded action, so it would now be useful to have some measure of convergence to distinguish the degree to which a CARLA has 'learnt'. The simplest measure of this is the peak value of probability density. As more rewards are

applied in a given region the peak probability density will rise. However, that statistic is dependent on the parameter range, and so a more general statistic is provided by its normalised version

$$C_m = \frac{\max\{p\}}{(\alpha_{\max} - \alpha_{\min})} \tag{6.9}$$

In Figure 6.4 each distribution line is separated by an equal number of reinforcement applications, and yet it is seen that the difference between successive peak values is becoming smaller. Plotting C_m against iteration for ideal reinforcement - Figure 6.5 - reveals that the CARLA convergence is self-limiting. In the limit, the probability distribution becomes the same shape as the reward function and imposing the constraint of equation (6.3) after a reward at each iteration acts to give no change overall. The limit value of C_m is dependent on the particular parameters g_w and g_h used, which define the shape of the reward function and hence the limiting form of the probability distribution.

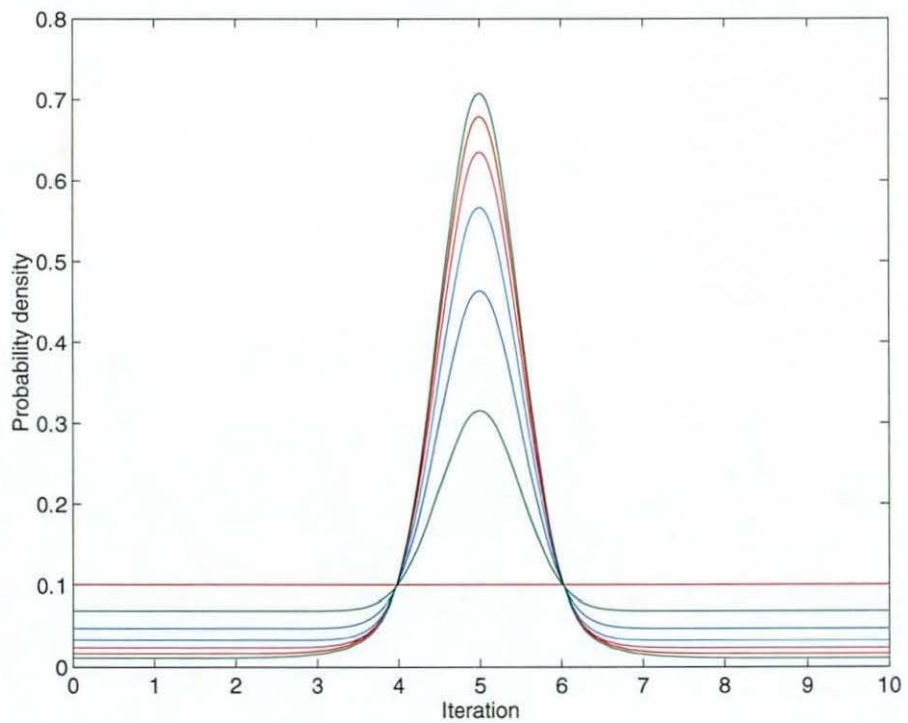


Figure 6.4 - Probability density function under ideal reinforcement

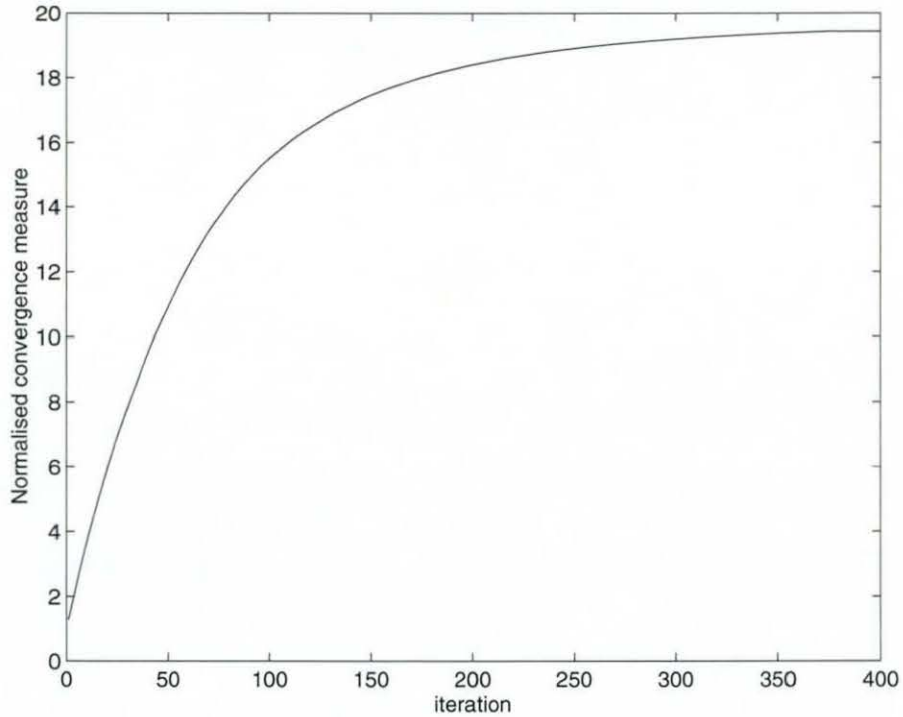


Figure 6.5 - Self-limiting convergence of the CARLA

6.3.2 Assessing the learnt action

In practical situations, where a stochastic task is considered, it is unlikely that maximal C_m will be reached within a reasonable time scale, if at all. 'Convergence' could be deemed to have taken place well before a near maximal value of C_m is achieved when the probability distribution shows a strong trend in one region of the action space. Consider Figure 6.6 that shows an iteration history of the probability distribution during one learning run. It is evident that two regions of the x parameter action space are exhibiting strong reward responses from the environment. Initially the automaton has applied reward around $x = -1.5$ before later moving its attention to around $x = 1.5$. Figure 6.7(a) shows that throughout the learning run maximal C_m was not approached despite the evident trends. It is therefore useful to define the 'learnt action' for a given state of the CARLA; what is the action that the CARLA has determined as the most likely to return a reward from the environment?

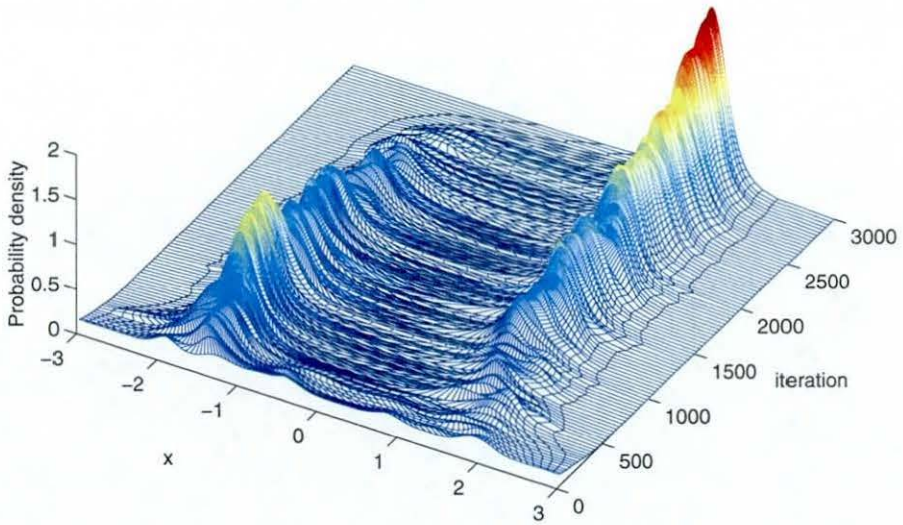


Figure 6.6 - Iteration history of a CARLA probability distribution

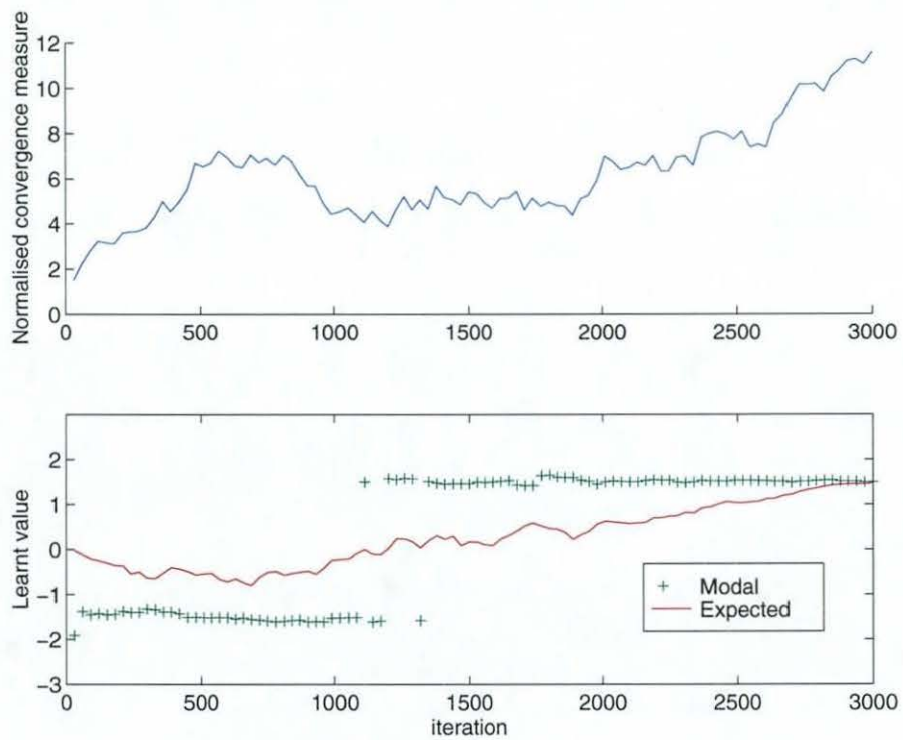


Figure 6.7 - Learning measures (a) C_m (b) learnt values:

expected (red) and modal (green)

Expected value

One statistic to consider in determining the learnt value from a CARLA probability distribution is the expected value From standard random variable theory, for any random variable x with associated probability distribution $p(x)$, the expected (or mean) value is defined as

$$E[x] = \int_{-\infty}^{\infty} x.p(x).dx \qquad (6.10)$$

The action α is the random variable in the CARLA, with associated probability distribution $p(\alpha)$ defined in the range $[\alpha_{min}, \alpha_{max}]$ Therefore the expected action is

$$E[\alpha] = \int_{\alpha_{min}}^{\alpha_{max}} \alpha.p(\alpha).d\alpha \qquad (6\ 11)$$

However, the probability distribution development shown in Figure 6 6 is one instance where this statistic gives seemingly non-intuitive values. The expected action value for this example is shown in Figure 6.7(b). Although two regions of good performance were clearly evident in Figure 6.6, the expected value wanders between these two regions as the favour of the CARLA alters. When probability is comparatively evenly distributed between two regions of the action space, it is often seen that the expected value will lie somewhere in-between, and therefore likely to lie in a region of low performance in terms of environment response

Modal value

An alternative statistic is to simply take the action value corresponding to the highest peak in probability density as the 'learnt value' of the automaton, i e. the modal value of the distribution

Figure 6.7(b) plots the modal value alongside the expected value for the learning results of Figure 6 6. It is seen that the modal value clearly picks out the learning trend in the two strong performance regions as the automaton switches its favour between -1.5 and 1.5.

However, the modal value only returns a single value at any iteration, and in examples where more than one strong region is evident, as above, other 'optima' are overlooked by this statistic alone.

6.3.3 CARLA results

Defining a dimensionless statistic C_m as a measure of progress of a CARLA, it has been shown that the implementation is self-limiting, although in practice such a conclusion to learning is unlikely to be achieved. The CARLA has no defined stopping/convergence criterion and it is therefore left to the user to decide when learning is 'complete'. In practice this will involve a combination of observation of C_m and visual inspection of probability distribution time-history plots to ascertain when strong learning trends are present. Two statistics, the expected value and modal value, have been identified as candidates for defining the 'learnt action' upon cessation of a learning run. It has been found that the expected value can give a false result where more than one strong region of performance has been located by the CARLA, and so from hereon the modal value is taken as the 'learnt action' result from a CARLA investigation. However, a visual inspection of probability distribution time-history plots should not be overlooked in identifying if other high performance regions are present that may require further investigation, e g. both regions identified in Figure 6 6 may be worthy of further investigation.

6.4 Implementation

Implementation of the above CARLA algorithm is straightforward apart from representation of the probability density function that defines the state of the automaton It is required that the density function representation will be a smooth, complete description of a continuous curve between α_{min} and α_{max} that is able to satisfy the constraint of equation (6.3) easily. Throughout development of the algorithms, particular thought has been given to maintaining an efficient implementation which will take up little computing resources for its storage and manipulation in an on-line scenario where memory and processing power may be constrained.

6.4.1 Probability density function representation

The probability density function is represented by a piece-wise linear approximation. The whole curve is simply described by a vector of (*action, probability density*) pairs

A basic representation is shown in Figure 6 8(a). Equally spaced action values have been selected at which the probability density function is recorded It can be seen that resolution of the representation at higher values of probability density is deficient. In this example the representation has acted to distribute reward away from the expected peak value at $\alpha = 5$ to values around it.

A 'smoother' representation with improved resolution at peak values is shown in Figure 6.8(b). Here, action values at which a corresponding density value is recorded are separated such that each 'segment' defines an equal fraction of the total area. At higher density values the curve is then defined by more closely packed segments. In lower regions, of lesser importance, and where the density function tends to be more 'flat', the curve only requires a few widely spaced values for its definition

Of course, improved curve definition could simply be maintained throughout by defining the curve at a larger number of equi-spaced points from the outset. However, the application of the above refinement allows a high level of definition of the curve around the important points, peak areas of probability density, whilst maintaining a much smaller number of points to define the curve overall; memory resources are conserved The additional processing required to implement the refinement can be offset against the processing that would be required to process an equi-spaced definition of high resolution.

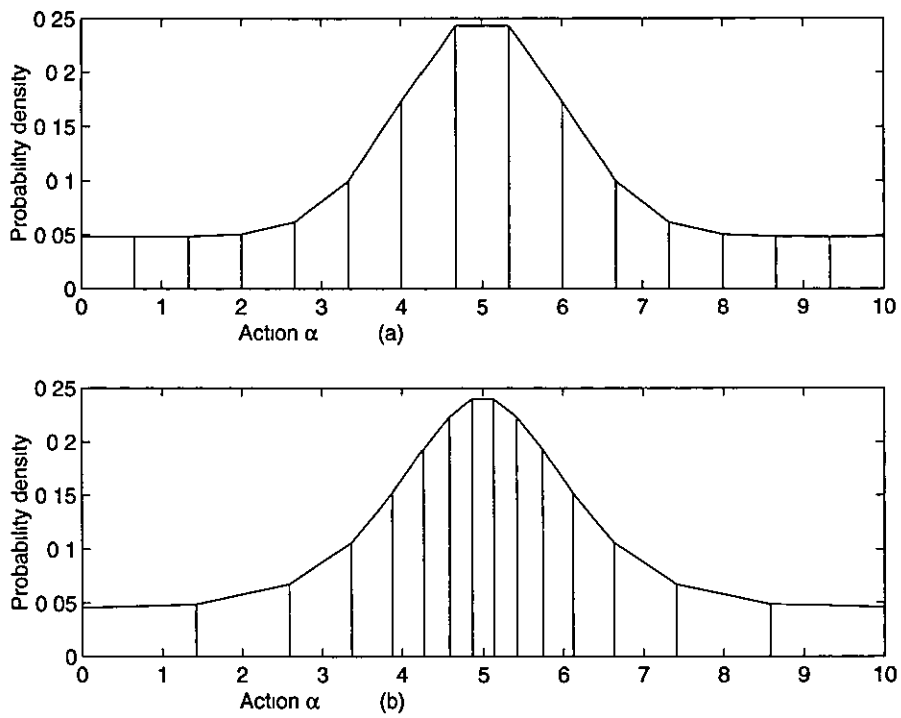


Figure 6.8 - Refinement of probability density function representation

The CARLA employs this method of curve representation for maintaining the probability density function. After any application of reinforcement the new curve is re-defined to maintain the resolution of representation. This is demonstrated further in Section 6 4 2.

The refinement algorithm is not described here, as it is largely independent from the successful operation of a CARLA. Appendix B outlines the implementation of the curve refinement algorithm.

6.4.2 Application of reinforcement

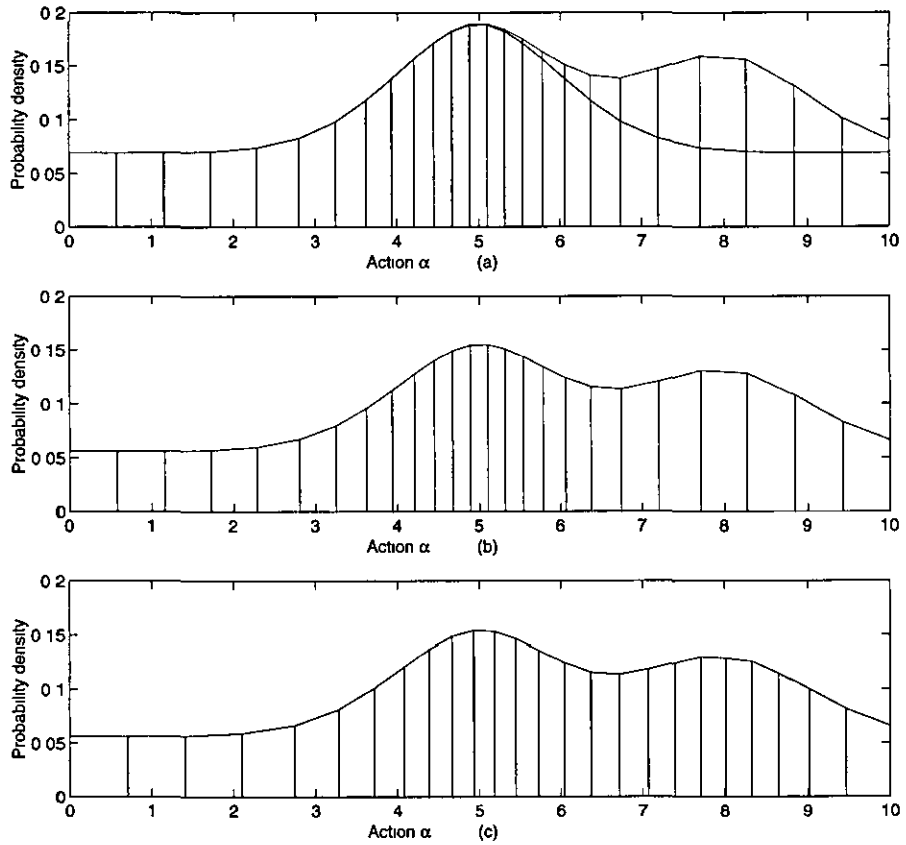


Figure 6.9 - Stages of reinforcement application

The reinforcement is applied via addition of the reward function of equation (6 6) to the probability density function. Using the piece-wise linear representation, equation (6 6) is applied at each action value used in defining the density function. For example, Figure 6.9(a) shows the effect on the density function of a reward at $\alpha = 8$. The initial probability density function is shown in red, onto which the reinforcement is applied, shown in blue.

Of course, the density function now does not satisfy the constraint of equation (6 3), and so the curve is normalised - Figure 6 9(b).

The final step is to redefine the curve, with the method outlined in Appendix B, to maintain a high resolution at peak values. The outcome of redefinition of the curve from Figure 6.9(b) is shown in Figure 6.9(c). Note that more action values now define the curve around $\alpha = 8$ where the new peak in the distribution is present.

6.5 Summary

A formulation of a learning automaton exhibiting a continuous action set has been introduced. The continuous action set is facilitated by maintaining the internal state representation of the automaton, a probability distribution, as a continuous function. The probability distribution is stored as a vector of (*action, probability density*) pairs defining the function in piece-wise linear fashion between user-defined limits. The underlying one-to-one mapping of probability distribution to action value limits the CARLA to single dimension action spaces. However, via an interconnected architecture of multiple CARLA it is possible to consider learning tasks of higher dimension.

The definition of the CARLA to consider a continuous action space immediately allows a complete search of the whole action space. This overcomes one of the major limitations of discrete automata, in an optimisation setting, where optima could easily lie between actions and not be identified by the automaton

The 'curse of dimensionality' is another major area of concern when using discrete automata; higher dimension tasks and/or high definition of actions quickly leads to large action sets resulting in slow or inconclusive learning. The CARLA formulation gives an effectively infinite action set covering the continuous action space, but the simple state representation and generalisation of reinforcement has made the learning process completely independent of action set size. The only dimensionality concern now arises where multiple CARLA are required for high dimension tasks, and interaction between automata is required to affect successful learning.

Another benefit of CARLA has been stated as full-adaptability in non-stationary environments The CARLA implementation and effective continuous learning allows the CARLA to respond to non-stationary environments. This phenomenon is investigated further in the following chapters where the CARLA is applied to various learning tasks to compare its performance with both discrete automata and the continuous action set automaton proposed by Santharam et al

Chapter 7 - CARLA Performance

Previous chapters have shown the classic discrete automaton to be a useful optimisation tool, especially in stochastic environments where many traditional optimisation techniques will fail. However, a number of limitations of discrete automata have also been noted. A new formulation of a learning automaton offering a continuous action set, the CARLA as described in the previous chapter, is able to overcome many of those limitations.

This chapter now investigates these claims via a comparison between CARLA and discrete automata on a simple stochastic optimisation task, seeking the global maximum of a noise-corrupted function. The optimisation task particularly tests the ability of the automata to distinguish between global and local optima.

An advantage of the CARLA previously only alluded to is its ability to adapt in non-stationary environments A demonstration of this feature with an environment exhibiting an abrupt change of response during learning is given.

A comparison is also made between CARLA and the continuous action set learning automaton (CALA) proposed by Santharam et al (1993). Their study used a penalised Shubert function to analyse the CALA performance and that function is thus applied here for the comparison.

7.1 Comparison with Discrete Learning Automata

A primary concern with applying a discrete learning automaton to an optimisation task has been in its ability to locate the optima accurately. A small action set giving efficient learning can easily miss optima that lie between actions. Taking a better defined, and hence larger action set may overcome this but is then likely to slow learning considerably. The automaton can even become inconclusive for very large action sets.

Here, a function of two variables is identified which exhibits two similar optima. A discrete automaton and CARLA are applied to identify values of the function variables that maximise the function. The addition of noise to this function forms a difficult stochastic optimisation task for comparison of the performance of the two types of automaton.

7.1.1 Optimisation task

The function of two variables given in equation (7.1) defines the underlying environment response for the automata

$$g(x,y)=\left|3.0(1-x^2)e^{\left(-x^2-(y+1)^2\right)}-10.0\left(\frac{x}{5}-x^3-x^5\right)e^{\left(-x^2-y^2\right)}\right| \quad (7.1)$$

where $-3\le x\le 3$, $-3\le y\le 3$. Equation (7.1) is derived from the 'peaks' function used in surface plot demonstrations with the application package, MATLAB. This function exhibits three maxima: A at (1.5,0), B at (-1.53,0.06), and C at (-0.48,-1.02) - see Figures 7.1 and 7.2. It is seen that the optima at A and B are of very similar magnitude, with the global optimum found at A. The optimisation task is made stochastic by corrupting $g(x,y)$ with zero mean, uniformly distributed noise in the range [-5, 5]. This noise signal overwhelms the difference between A and B, presenting the automata with a difficult task to distinguish between them.

7.1.2 Discrete automaton configuration

A single discrete S-model automaton, of the form introduced in Chapter 3, is applied with learning parameters $\eta=0.5, \lambda=0.4$ as before. The action set is defined to give an equally spaced square matrix of actions across the action space, e.g. choosing a quantisation level of 4 for each parameter gives an action set of 4^2, 16 actions - see Figure 7.3. It is seen that the near symmetry of the two main optima, at A and B, and the definition of a symmetrical action set gives no single action which could bias learning in favour of either optima in the primary stage of learning.

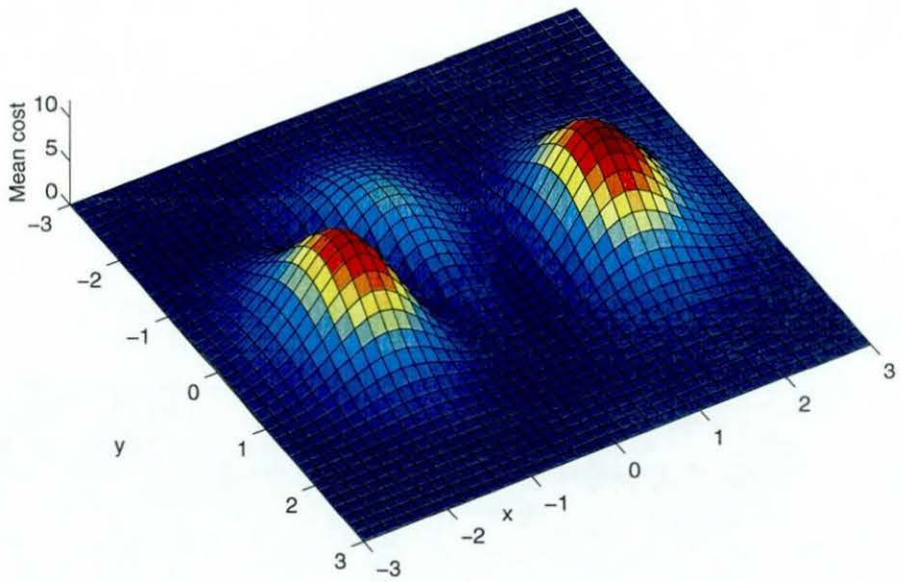


Figure 7.1 - $g(x, y)$

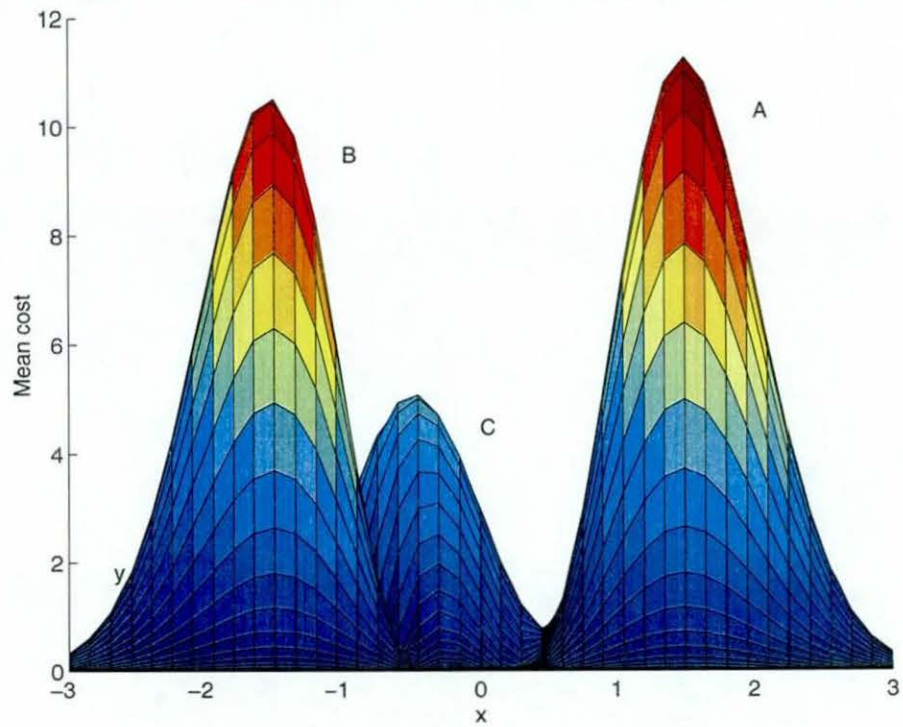


Figure 7.2 - $g(x, y)$ viewed along y-axis

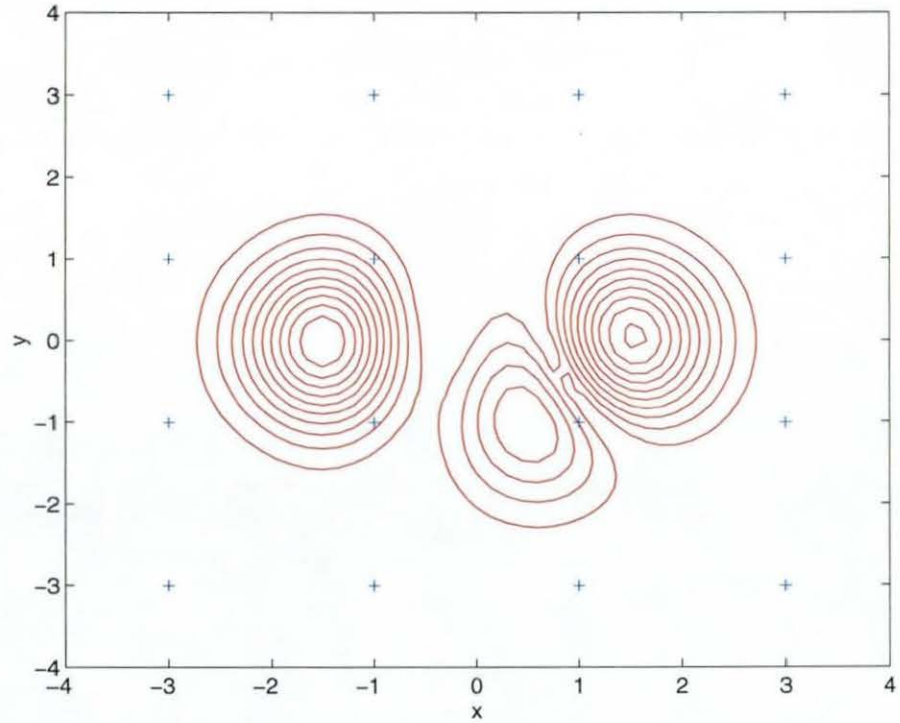


Figure 7.3 - Discrete automaton action set for initial learning stage

7.1.3 CARLA configuration

The implementation of CARLA necessitates that multiple automata are linked in interconnected fashion for multi-parameter optimisation tasks. Therefore, on the optimisation task defined above, one CARLA is assigned to each of x and y respectively. Figure 7.4 illustrates the configuration of CARLA for this task. Both automata define their action sets between $\alpha_{\min} = -3$, $\alpha_{\max} = 3$. The learning generalisation parameter, g_w, is maintained throughout this study at 0.02. The learning rate, g_h, is altered between 0.3 and 0.6 to analyse its effect on the automata results.

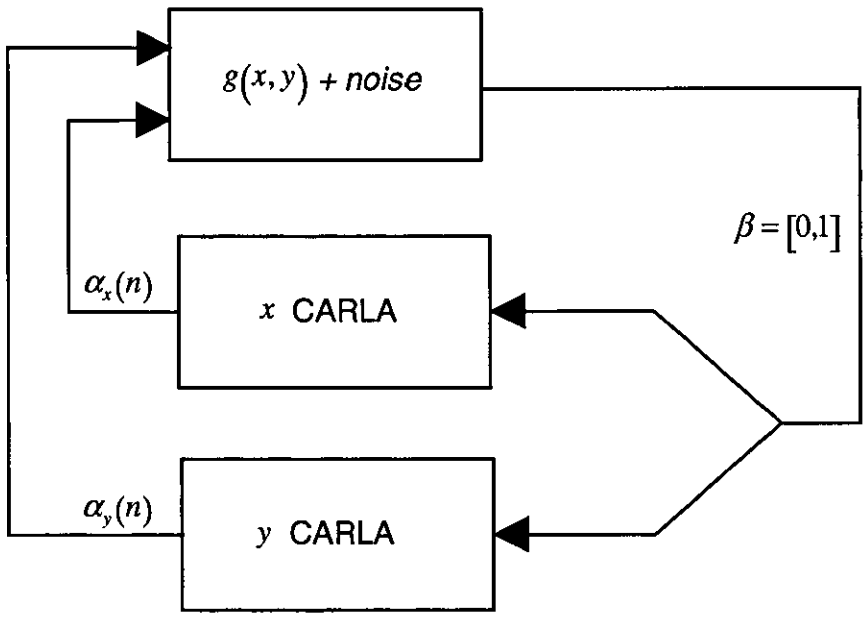


Figure 7.4 - Interconnected configuration

7.1.4 Performance index

As this task requires maximisation, the performance index calculation, previously described by equations (6.7) and (6.8) for minimisation, is amended to

$$\begin{aligned} J_{\max} &= \max\{J(1), J(2), \ldots, J(n)\} \\ J_{\text{med}} &= \text{median}\{J(1), J(2), \ldots, J(n)\} \end{aligned} \tag{7.2}$$

$$\beta(n) = \max\left\{0, \frac{J(n) - J_{\text{med}}}{J_{\max} - J_{\text{med}}}\right\} \tag{7.3}$$

for both automata configurations. $\beta = 1$, maximum reward, is now returned for a 'maximum encountered so far' environment response. $\beta = 0$ is returned for any response of J_{med} or below.

7.1.5 Optimisation results

Each analysis of a particular automaton configuration underwent a set of 100 trials to give an estimation of the average automaton performance and provide a measure of the frequency with which the global optimum is located. As this is a stochastic optimisation it is unlikely that the automata will exactly locate any optimum, so 'locate' is taken here to mean a result that lies acceptably close to an observed optimum.

A summary of the results for the discrete automaton, for various quantisation levels and learning rates are shown in Table 7.1. The initial learning run of DA1 implements

a learning automata similar to that used in the studies of previous chapters, with learning rate $\delta = 0.1$.

Discrete Automaton	Quantisation per parameter	δ	Convergence to peak A (%)	Convergence to peak B (%)	Iterations to convergence (+- std. dev)
DA1	4	0.1	52	48	825 (143)
DA2	7	0.1	56	44	1007 (189)
DA3	10	0.1	49	51	1017 (201)
DA4	4	0.05	73	27	2954 (685)
DA5	4	0 025	74	26	9464 (2062)

Table 7.1 - Discrete automaton results

It is seen that this automaton is unable to distinguish between the maxima at A and B, converging to each with similar frequency. This may have occurred because the learning rate is too high, or because greater action set definition is required to enable a distinction. However, increasing the quantisation level to 7 (49 actions, DA2) and then 10 (100 actions, DA3) has little effect. As would be expected for larger action sets, the average number of iterations to convergence increases as the automaton takes longer to decide between the increased number of options available to it, and yet the frequency of correct convergence remains around 50%. Therefore, it can be assumed that the learning rate is too high for this task, so the automaton is not gathering enough information in the first learning stage to make a valid decision on the area of the action space it should converge towards.

Returning to a quantisation level of 4, and halving the learning rate has an immediate effect - DA4. Now the automaton converges to the global optimum at A around 75% of the time The number of iterations to convergence has risen to a similar level where effective learning has been observed before, 3000 iterations. The automaton is seen to take longer for stage one learning (330 iterations on average for DA4 in comparison to 130 iterations for DA1) and is thus far more likely to make a correct decision at this early stage

However, considering any slower learning rate has little beneficial effect. Analysis with $\delta = 0025$ - DA5 - reveals a similar convergence result, around 75% success rate, but the iterations to convergence has risen significantly. The learning rate is now too slow, so indecision between actions begins to dominate with no improvement in

quality of learning; stage one learning takes 950 iterations on average. A 75% success rate appears to be the best performance a discrete automaton will provide on this task.

CARLA	Learning rate, g_h	Convergence to peak A (%)	Convergence to peak B (%)	Iterations to learning halt
C1	0 3	100	0	3000
C2	0 6	98	2	3000

Table 7.2 - CARLA results

No stopping criterion is defined for CARLA, so it is chosen here to simply stop the CARLA after 3000 iterations for the learning to be comparable with DA4, the discrete automata observed to take a similar number of iterations in its optimum configuration. Results for the CARLA configuration on this task are given in Table 7.2. A learning rate of $g_h = 0.3$ has been found to provide capable performance in other stochastic scenarios, such as those presented in later chapters, so this value is applied in the 'first try' of the CARLA - C1. It is observed that C1 is a complete success; all trials result in A being located.

A typical result for the CARLA pair is shown in Figures 7.5 and 7.6. The x CARLA typically identifies strong performance around both major optima early in a learning run - Figure 7.5(a) and (c). A period is then apparent where C_m rises steadily as more information is gathered. There appears to be a point at which the CARLA 'decides' between the strong performance regions it has located, to converge such that its interest becomes concentrated on a single region. Figure 7.5 shows this happening between 1500 and 2000 iteration as C_m rises sharply during the shift of interest The y CARLA is seen in Figure 7 6 to quickly identify its optimum around $y = 0$, where both major optima lie close by; within 500 iterations the CARLA identifies the optimal region and the sharp rise in C_m is again evident from thereon.

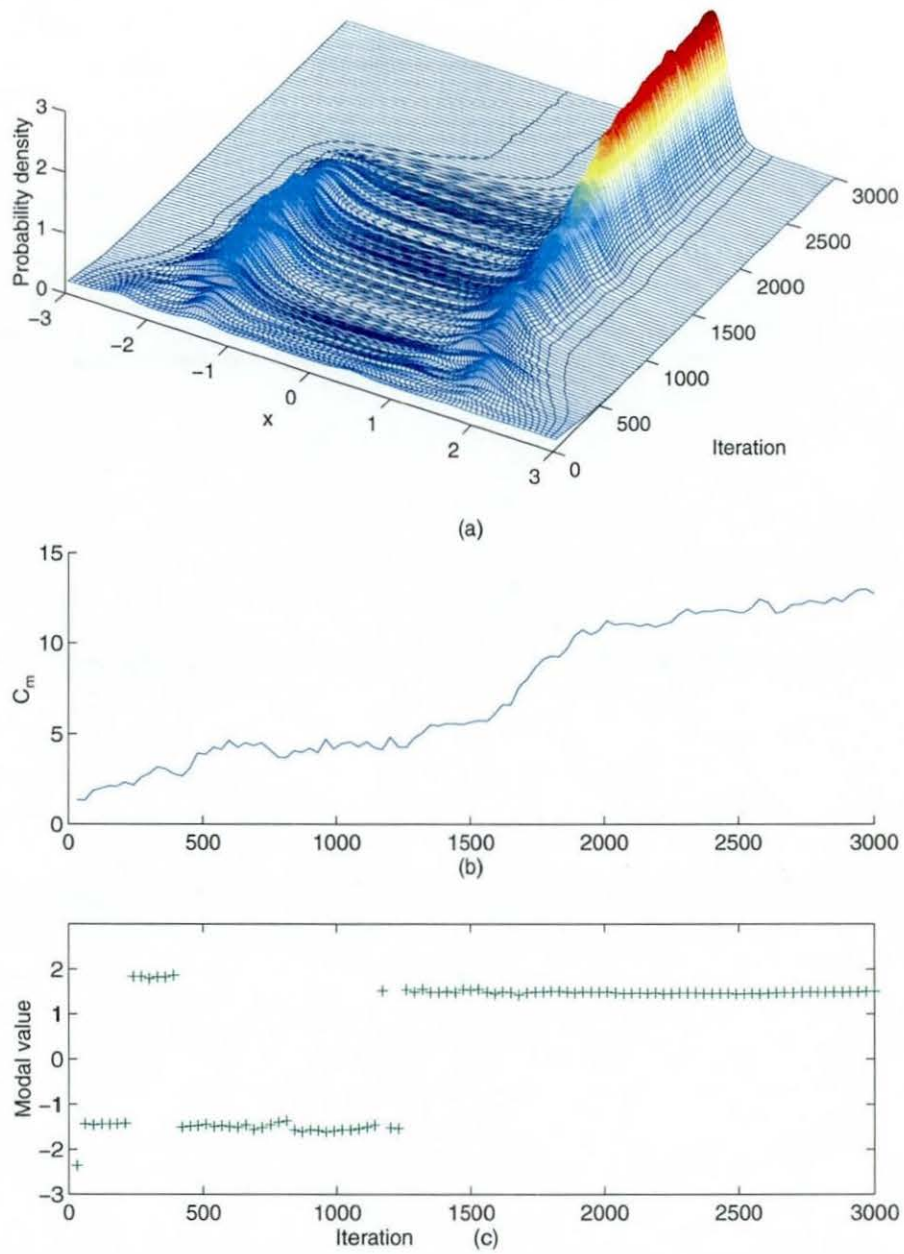


Figure 7.5 - Typical x CARLA time history: (a) probability distribution, (b) convergence measure, (c) 'learnt' value, modal

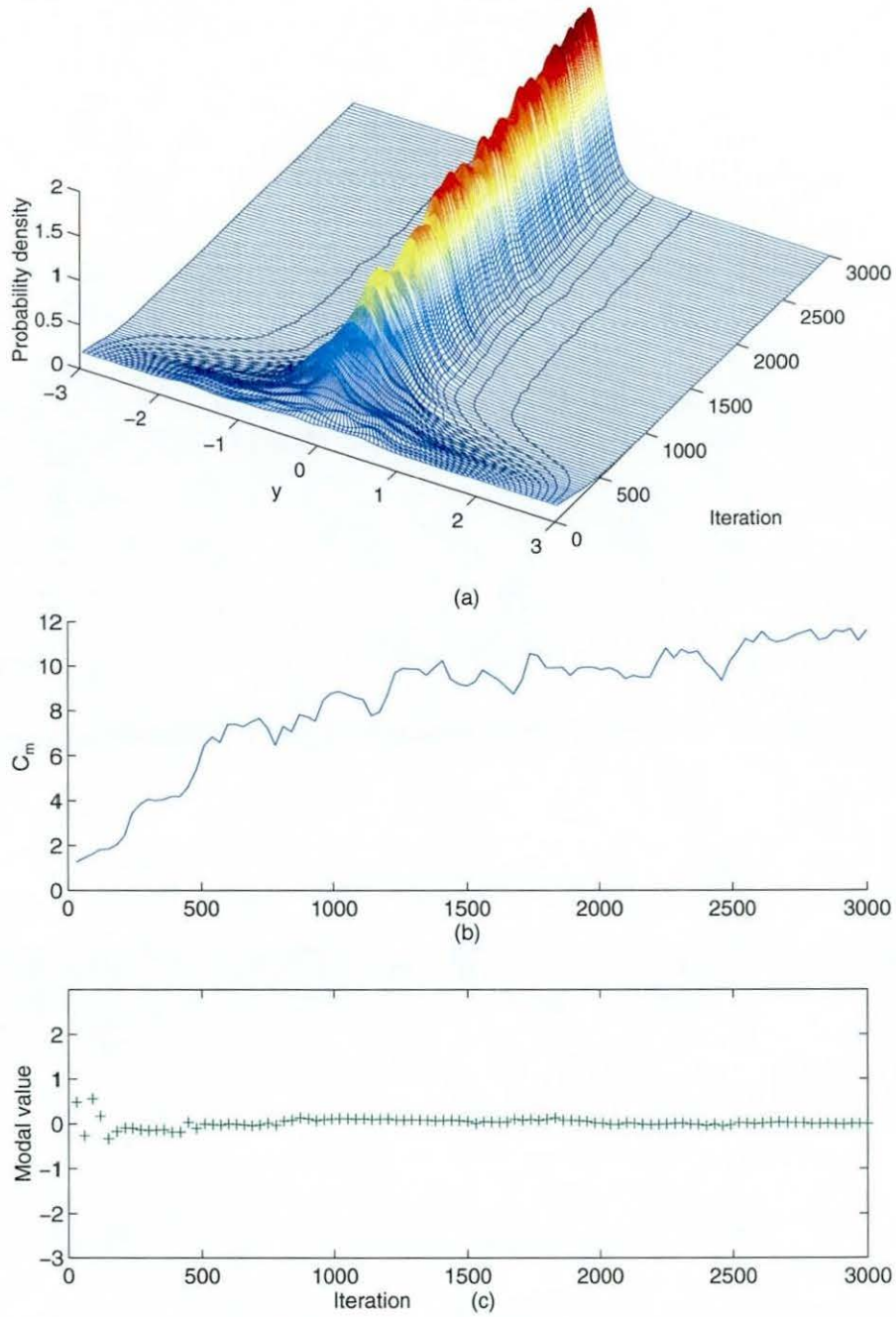


Figure 7.6 - Typical *y* CARLA time history: (a) probability distribution, (b) convergence measure, (c) 'learnt' value, modal

As the CARLA has faultlessly located the global optimum in C1, it is now interesting to try to instigate a 'failure'. Discrete automata have been seen to increasingly make mistakes if the learning rate is set too high so that 'hasty' decisions are made during learning. Raising the CARLA learning rate g_h to 0.6, two 'failures' from the set of 100 are observed - C2 in Table 7.2. The x CARLA results for one of these is shown in Figure 7 7 Again C_m exhibits a steady rise during a period of exploration where both regions of strong performance for x are identified. However, this period is now much shorter, in this case only around 500 iterations, before the CARLA opts for one region alone Once the CARLA is concentrating on the single region almost all chance of further exploration away from this region is removed The increased learning rate has indeed forced the CARLA into a 'hasty', erroneous decision

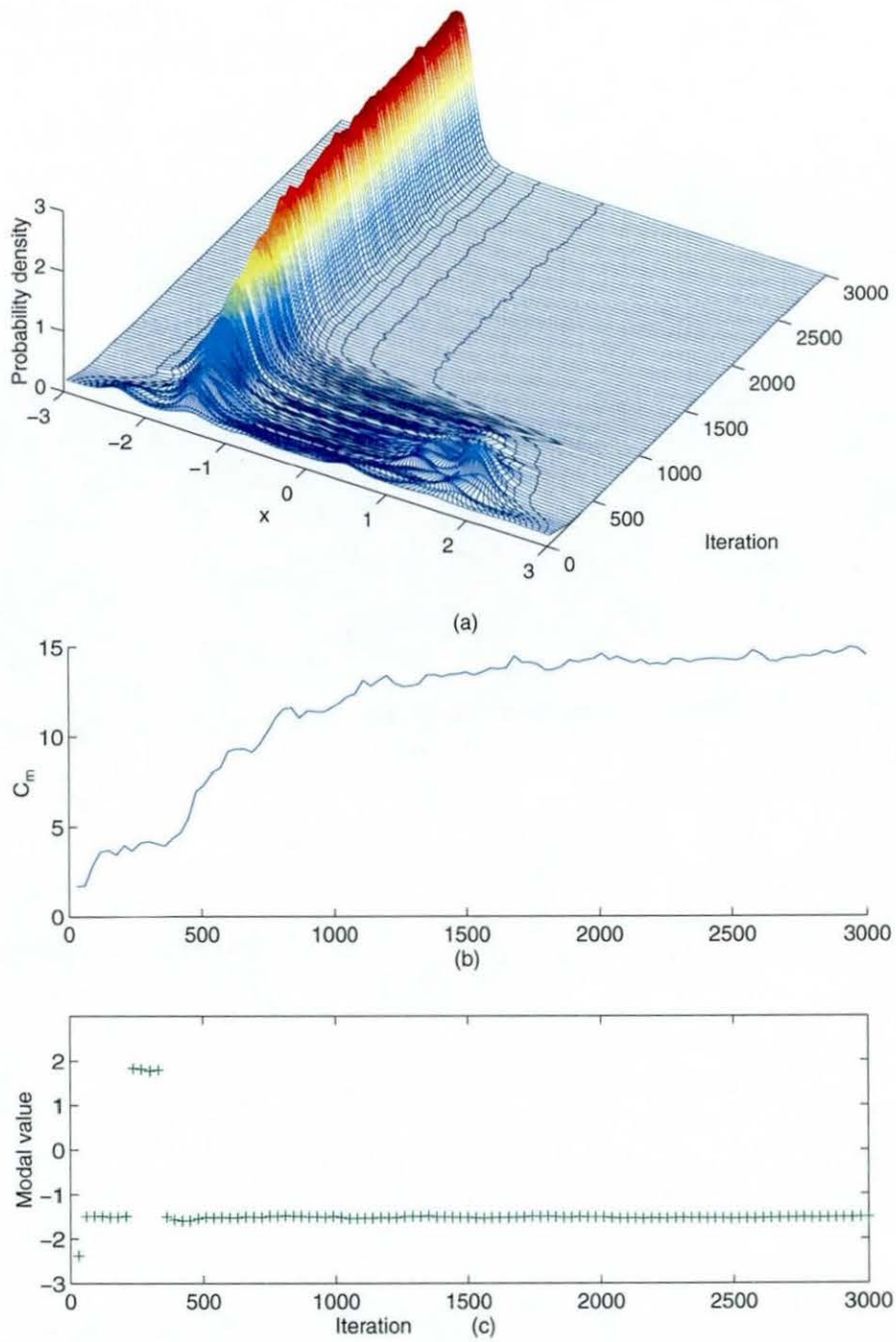


Figure 7.7 - Increased g_h leads to 'hasty' decision making on x: (a) probability distribution, (b) convergence measure, (c) 'learnt' value, modal

A comparison of the variability of automaton results is shown in Figure 7.8. The 'learnt' action values from both the discrete automaton (DA4) and CARLA (C1) around the global optimum are plotted. It is seen that the CARLA also locates the optimum more accurately, with less variability than the discrete automaton on this task.

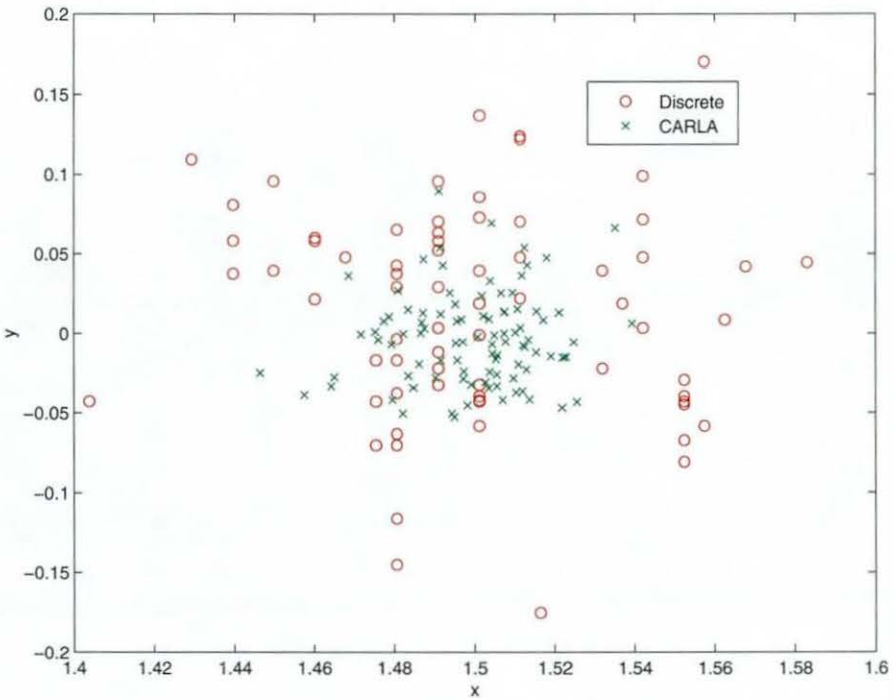


Figure 7.8 - Scatter plot of results around global optimum

7.2 Adaptive Nature of CARLA

Another major concern with discrete automata is their inability to operate successfully in non-stationary environments. Any automaton must maintain exploration of a sufficiently large area of the action space if it is to notice any shift in the environment response. Discrete automata would require a large action set to be able to maintain an adequate exploration of the whole action space throughout a learning run. A trade-off has to be made between enough actions to cover the action space, but not providing accurate location of optima, and too many actions which may improve optima location accuracy but which then lead to much longer learning times. The extension to the methodology proposed by Wu (1993) allows improved optima location without the requirement for a large action set. The drawback is that the technique relies upon a forced convergence of the automaton to a smaller action space in multiple learning stages. This removes continual coverage of the total action space throughout a learning run and hence any variation of an environment during this time can confuse the automaton, e.g. if the automaton is converging towards an optimum and then the

optimum moves to lie outside the current automaton action space, the prospect of the automaton recovering to relocate the optimum is limited Tuning of the convergence parameters may give the automaton more scope for recovery but this in turn increases the time taken to complete convergence.

It has already been demonstrated that a continuous action set allows a complete search of the action space it encloses throughout a learning process. This may allow the CARLA to be able to react to variations in a non-stationary environment, adapting its state in response to the change A test case is described below in which the CARLA is presented with an environment that exhibits an abrupt change of characteristic during learning

7.2.1 Optimisation task

The environment used here is again described by equation (7.1), but after 1000 iterations of learning the transformation $x \rightarrow y$ and $y \rightarrow x$ is made, giving

$$g(x,y) = \left| 30(1-y^2)e^{(-y^2-(x+1)^2)} - 10.0\left(\frac{y}{5} - y^3 - y^5\right)e^{(-x^2-y^2)} \right| \quad (7.2)$$

The two major optima are of course then located at (0,1.5) and (0.06,-1.53) of which the global optimum is found at the former. Function evaluations of equation (7.2) are corrupted as before to implement a highly stochastic environment.

7.2.2 The CARLA configuration

Two CARLA are configured as in the previous study, with one assigned to x, and one to y. The learning rate is maintained throughout at $g_h = 0.3$. The CARLA stopping criterion was raised here to 5000 iterations to capture the entire automata response to the change in the environment response at 1000 iterations

7.2.3 Adaptive results

A set of 100 independent trials was taken on this task. In every case the CARLA locates the shifted optimum successfully. The success of this trial can first be observed with reference to a plot of the mean (rolling average of 100 values) environment response to the CARLA actions during a typical learning run - shown in Figure 7.9. Strong learning is evident up to 1000 iterations as the observed cost rises sharply, indicating that the CARLA are already locating the optima well. At 1000 iterations the average environment response drops as the CARLA initially continues to

explore around the former optima which now exhibit the low response However, a recovery is apparent, although slow at first.

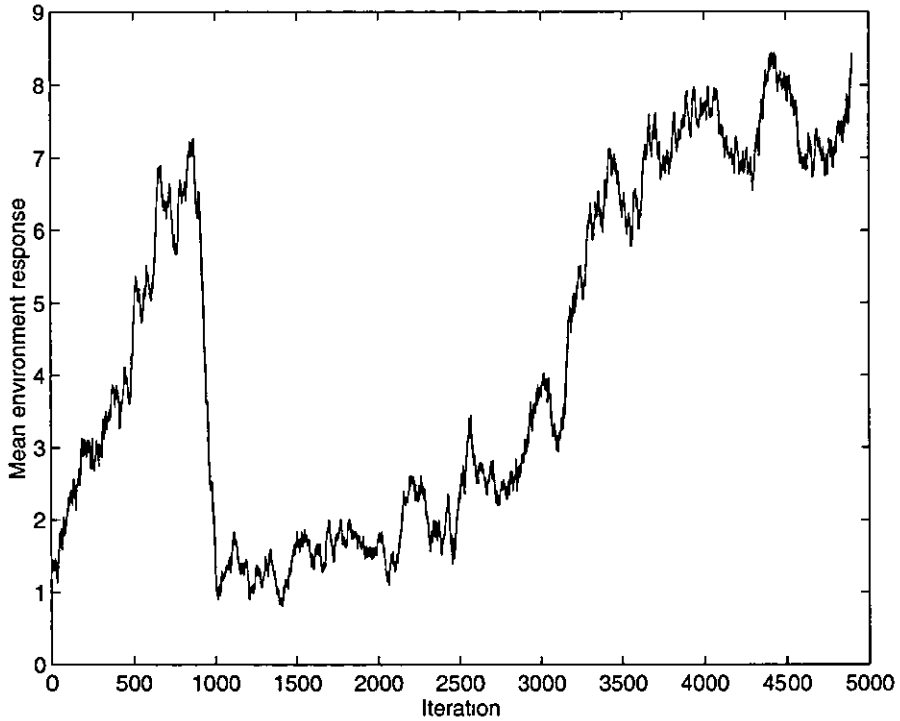


Figure 7.9- Mean environment response time history

Typical CARLA results for this run are given in Figures 7.10 and 7.11, clearly showing the CARLA adapting to relocate the new maximum, although, as suggested by the mean cost plot of Figure 7.9, the shift in emphasis of the automata is seen to take some time to materialise. Figure 7.11 shows this most clearly. As seen in previous y CARLA results, C_m rises sharply early on as $y = 0$ is quickly identified as the optimum, such that by 1000 iterations the CARLA has established a strong preference for that region. Now, on the environment translation, the CARLA undergoes a long period of 'unlearning' that preference before any other can emerge. C_m drops accordingly between 1000 and 3000 iterations.

In contrast the x CARLA - Figure 7.10 - is still in an exploration phase when the switch occurs; two strong regions are identified but no decision has been made between them so the CARLA still has its attention spread over a wide area of its action space Between 1000 and 3000 iterations the x CARLA has little to 'unlearn' but is tied to some extent by the y CARLA performance, and hence has to wait for that to spread its attention again before both automata can co-operate in locating new optima. This state occurs around 3000 iterations when learning clearly proceeds apace to locate the new global optimum at (0,1 5).

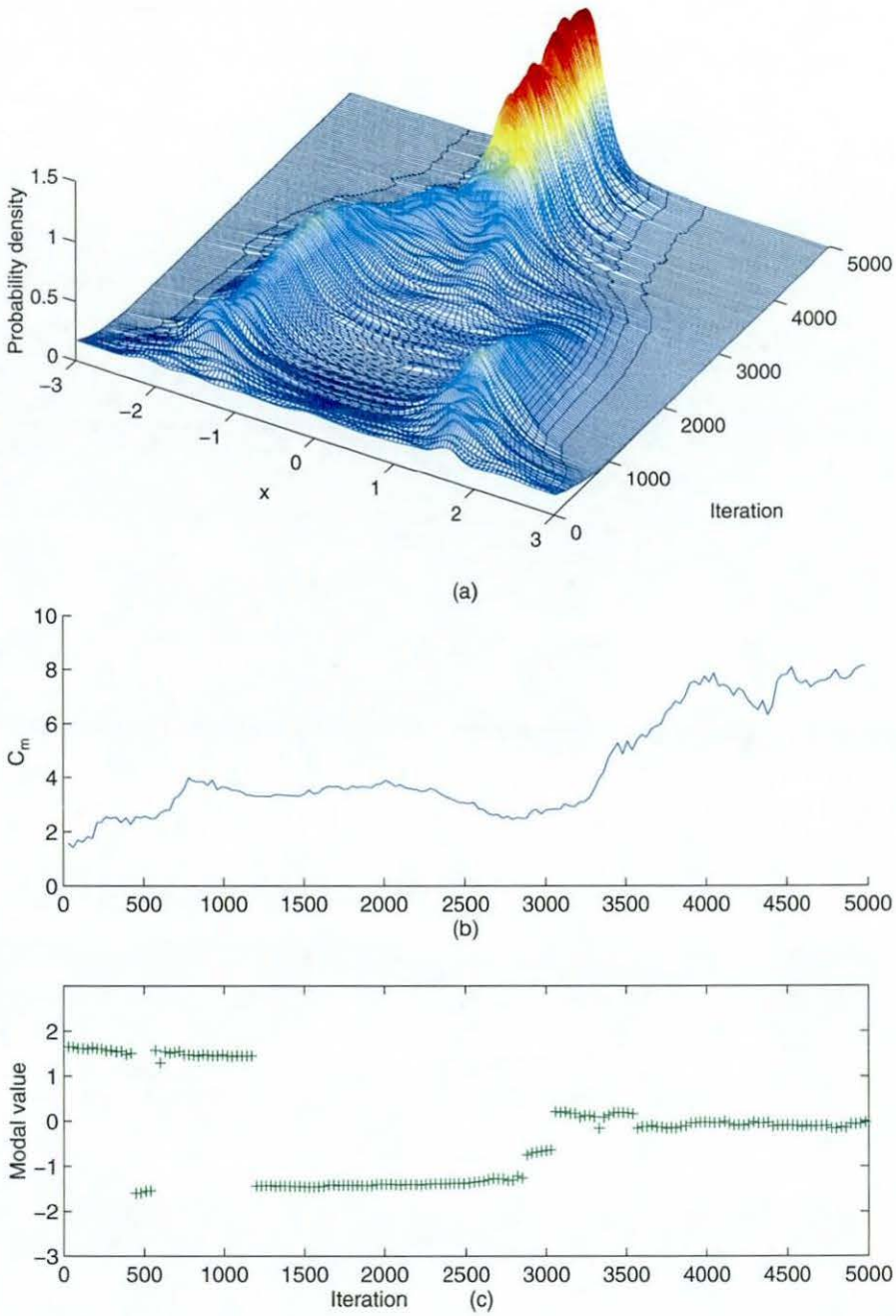


Figure 7.10 - Adaptive *x* CARLA time history: (a) probability distribution, (b) convergence measure, (c) 'learnt' value, modal

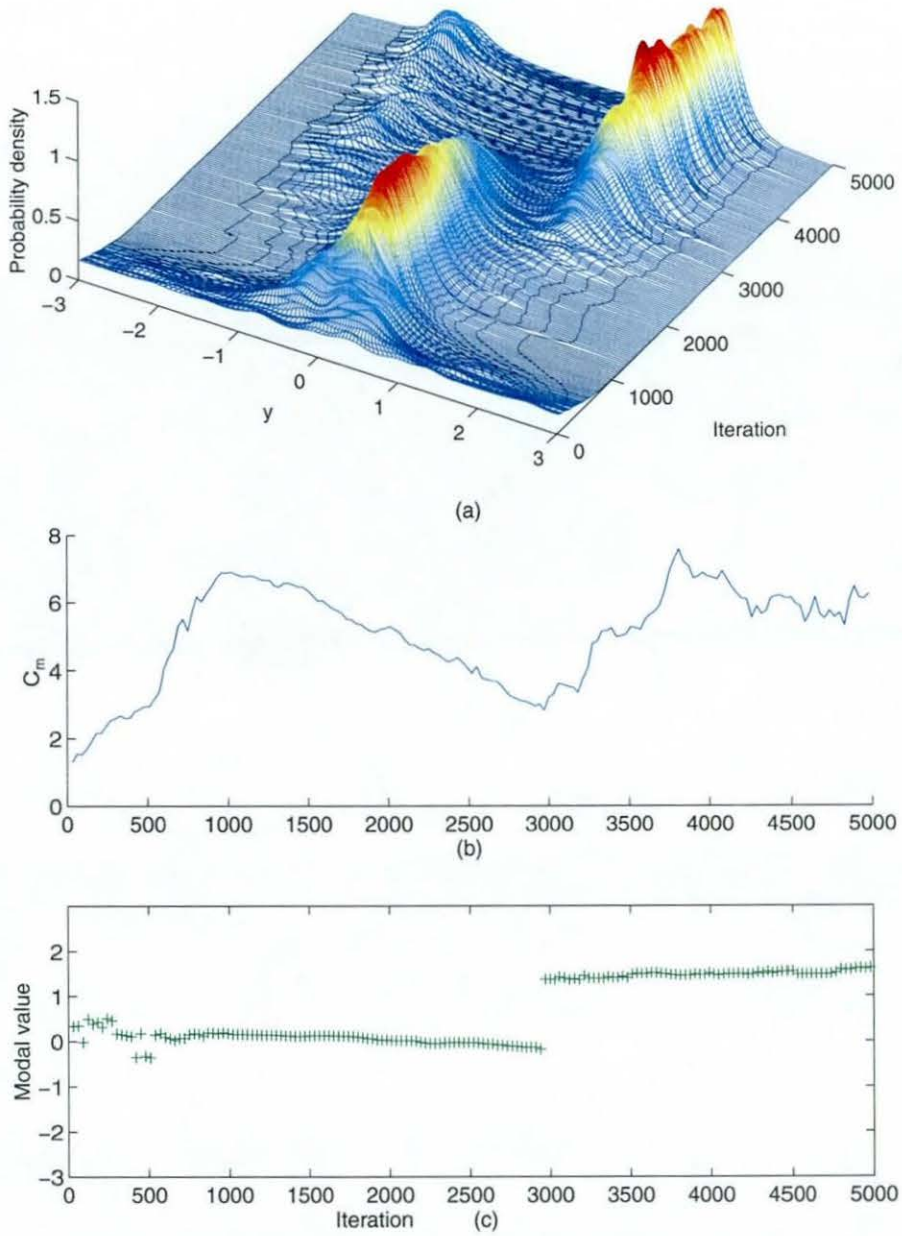


Figure 7.11 - Adaptive *y* CARLA time history: (a) probability distribution, (b) convergence measure, (c) 'learnt' value, modal

7.3 Comparison with Santharam's CALA

Santharam et al. (1994) also proposed a learning automaton formulation with a continuous action set, formed by recording the probability distribution as a continuous function. They described the whole probability distribution as a Gaussian curve, with the mean μ and standard deviation σ as parameters under control of the learning automaton via reinforcement. The probability distribution is mapped one-to-one with an action value and hence the CALA is also limited to an action space of a single dimension, so requiring an interconnected configuration of multiple automata to consider higher dimension tasks.

The CALA operation can be summarised as follows An action reinforcement acts to move the distribution mean towards that action. If the rewarded action lies close to μ already, then σ is reduced so the automaton converges around that region Otherwise, σ is increased in an attempt to encompass the rewarded action and hence broaden its scope to continue exploration across a wider area.

The behaviour of the CALA was examined on a stochastic optimisation task, locating the minimum of a penalised Shubert function defined as

$$f(x)=\sum_{i=1}^{r} i.\cos((i+1)x+1)+u(x,10,100,2) \qquad (7\ 3)$$

where

$$u(x,a,k,m)=\begin{cases} k(x-a)^m & \text{if } x>a \\ 0 & \text{if } |x|\le a \\ k(-x-a)^m & \text{if } x<a \end{cases} \qquad (7.4)$$

This function is shown in Figure 7 12. It has 19 minima in the region $[-10,10]$ of which three are global minima, located close to -5.86, 0.43 and 6 71 respectively A stochastic element was added by corrupting function evaluations of $f(x)$ with zero mean, uniformly distributed noise in the range [-0.5, 0.5].

The CALA was used to find a minimum of this function with various different initial values of μ and σ. The simulations were run for 8000 iterations each

It was found that the mean value, μ, of the CALA always converged close to a minimum of the function. It was seen that the initial values of μ and σ had the greatest bearing on which minimum was located. However, the overall success rate of the CALA was not made clear, and only a small number of simulation results were

presented - See Table 7.3. It is evident that the CALA was found susceptible to locating local minima.

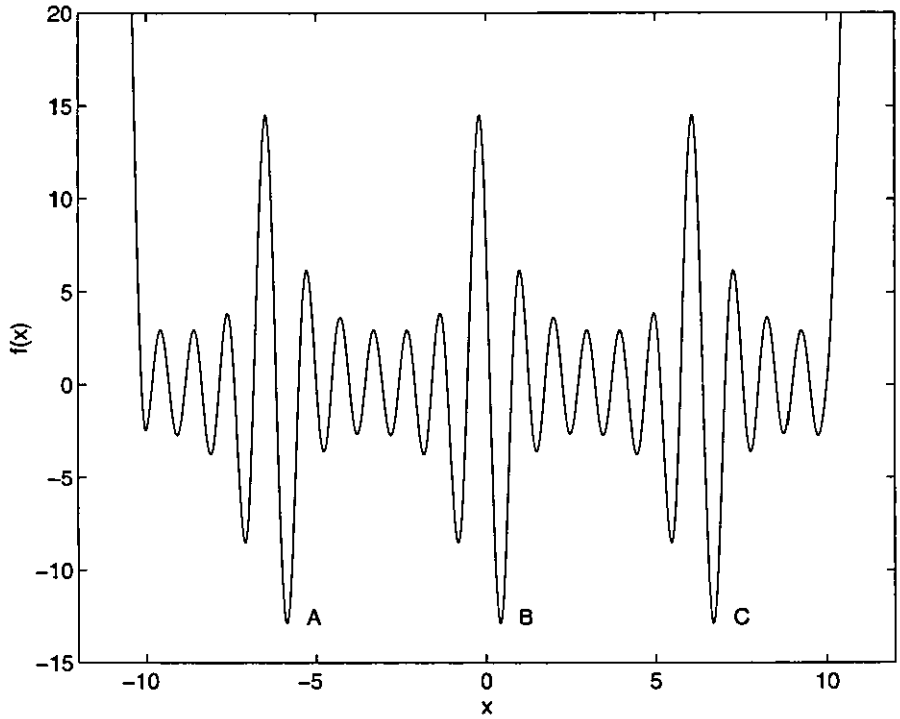


Figure 7.12 - The penalised Shubert function

Initial Values		After 8000 Iterations	Function Value
μ_0	σ_0	μ_{8000}	
4	6	2.534	-3 578
4	10	0 4038	-12 87
8	5	5.36	-8 5
8	3	6.72	-12.87
12	6	1.454	-3.58
-10	5	-7.1	-8.5
-10	6	-5.8	-12 87

Table 7.3 - CALA results (from Santharam et al , 1994)

A comparison between CALA and CARLA is made here by applying a CARLA to the above task. To this end the performance index is defined in the minimisation form of equations (3.11) and (3.12). The CARLA is defined with an initial action set of $\alpha_{min} = -12$, $\alpha_{max} = 12$ and learning parameters of $g_h = 03$ and $g_w = 002$ are again applied.

A set of 100 results of CARLA is obtained with learning halted after 1000 iterations. Despite running the CARLA for significantly less iterations than the CALA, it is found that the CARLA are able to locate a global minimum on every occasion, with an even spread of results between the three minima. A typical result is shown in Figure 7.13, where the CARLA is seen to have identified the regions of all three optima, evident in the distribution peak trends of Figure 7.13(a) and (c).

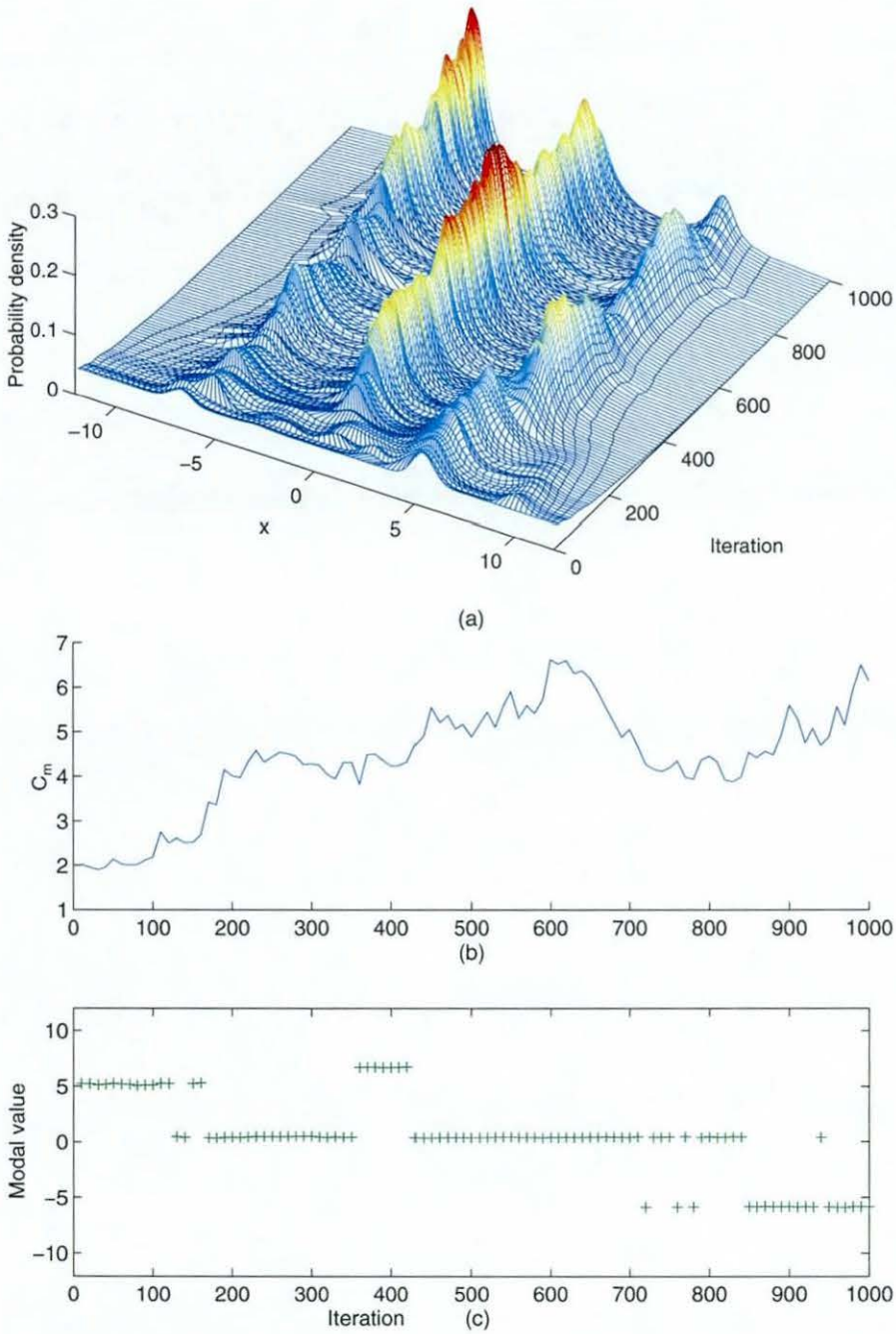


Figure 7.13 - CARLA time history on Shubert function: (a) probability distribution, (b) convergence measure, (c) 'learnt' value, modal

7.4 Summary

Throughout this chapter the CARLA has repeatedly demonstrated many benefits in comparison with discrete automata and CALA in stochastic optimisation simulation studies, including

- global optimum location/ local optima avoidance
- accurate optimum location
- faster learning
- a capability in non-stationary environments
- insensitive learning parameters, g_h and g_w

The next chapter returns to the vehicle suspension application to analyse the CARLA performance further.

Chapter 8 - CARLA on Vehicle Suspension Applications

The CARLA has thus far demonstrated some promising characteristics. Comparison between the CARLA and discrete automata on optimisation tasks with known optima has highlighted additional capabilities and increased performance offered by CARLA This chapter now returns to the vehicle suspension application to analyse the performance of the CARLA further. Learning tasks based on the quarter-vehicle suspension model are revisited to verify the CARLA capabilities in a practical application. The 'moderator' of Chapter 4 is used again to enable safe on-line learning, controlling excessive workspace usage, so allowing hardware testing of the CARLA on the test vehicle used in Chapter 5.

8.1 Learning with White Noise Road Spectra

Considering a linear quarter-vehicle model exhibiting ideal full-active suspension force actuation between sprung and unsprung masses, excited by white noise road velocity input, LQG theory provides an optimal linear state feedback controller for a pre-defined quadratic cost function. This has been derived in Section 2.1 for a representative vehicle model and is of the form

$$u(t) = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix} x \tag{8.1}$$

Previously a single discrete automaton has been applied to learn these four gain values, and is found to learn controllers with performance costs close to the optimal value for this system, $J_{opt} = 2.6621$ - see Chapter 3

Here the CARLA is employed on the same task, optimising the four gains subject to the LQG quadratic cost function of equations (3 14) and (3.15). A single CARLA is assigned to learn each k_i respectively ($g_h = 0\,3$, $g_w = 0\,02$), the automata interacting and co-operating on the overall task via their interconnected configuration. The range for each individual parameter, and hence each CARLA, is chosen as in the comparable discrete automaton implementation of this task in Section 3.6

$$\begin{aligned} k_1 &\in [-20000,0] \\ k_2 &\in [0,15000] \\ k_3 &\in [0,2000] \\ k_4 &\in [-4000,0] \end{aligned} \tag{8 2}$$

These ranges border on unstable regions of the action space, but the CARLA implementation assures these regions are not encroached upon during learning. As a minimisation task the performance index is calculated with equations (6 7) and (6 8).

A set of ten learning sessions is taken, referred to as Automaton F, with the stopping criterion set at 3000 iterations, comparable to the observed iterations to convergence average (2780 iterations) of the equivalent set of discrete automaton results, Automaton B. Figure 8.1 summarises the learnt parameter results of Automaton F where it is seen that the CARLA produces far more consistent values (c f. Figure 3.6)

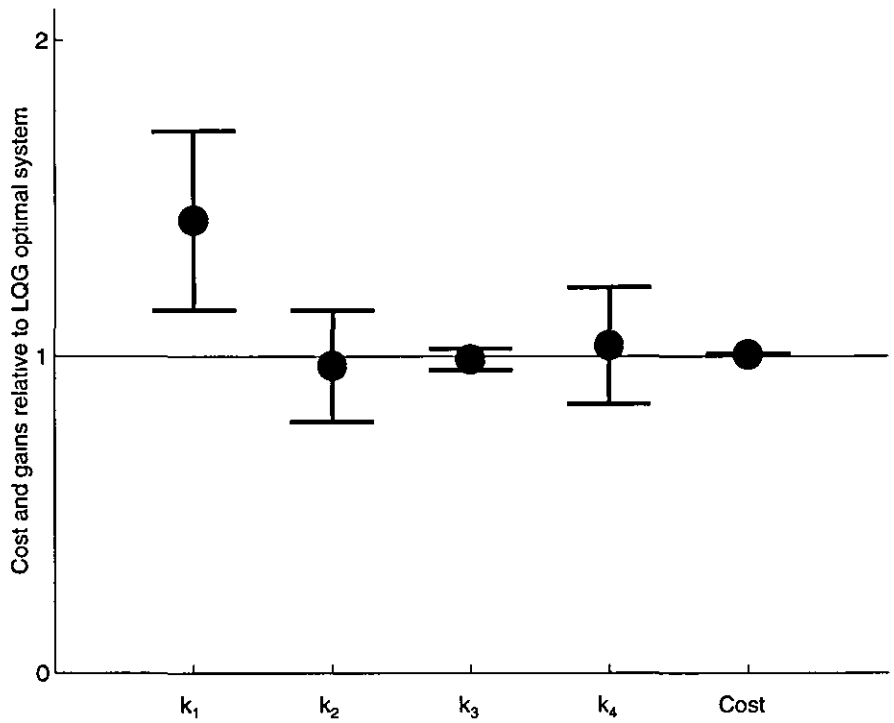


Figure 8.1 - Automaton F. learnt parameter results (mean ± one standard deviation)

Discrete automata were seen to learn more variable, but distinctly correlated, k_2, k_4 values Automaton F exhibits no such correlation at first glance. However, comparing the distributions of the resultant pairs, Figure 8.2, with those from a discrete set of results, Figure 4 3 (from the moderated Automaton C results, but with essentially similar learning characteristics as unmoderated Automaton B) it is seen that the CARLA results are grouped closely such that any correlation is not as evident.

The derived theoretical costs of Automaton F controllers are outlined in Table 8.1. Comparison with discrete results on this task, Table 3.2, again shows performance and consistency gains from CARLA learning.

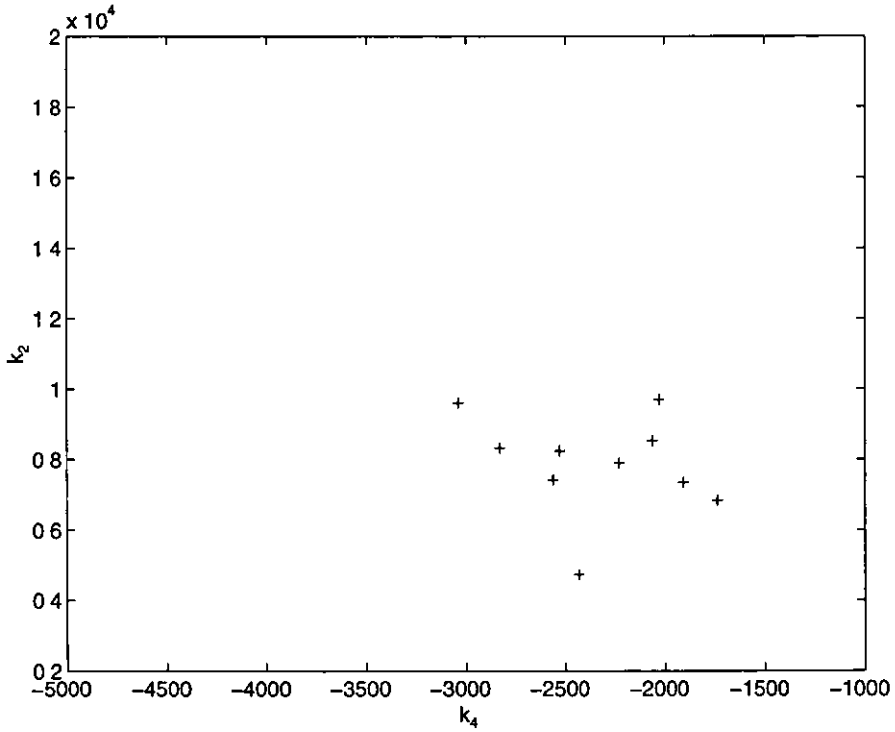


Figure 8.2 - Automaton F. k_4 vs. k_2

	Maximum	Minimum	Mean	Std. Dev.
Cost	2.6971	2.6664	2 6774	0.0097
Increase over J_{opt} (%)	1.3	0 2	0 6	-

Table 8.1 - Automaton F cost performance

In contrast to the discrete automata application on this task, one CARLA has been applied per parameter. Each CARLA is free to learn at its own rate, but any assimilated correlation it sees between the actions chosen and the environment response will be somewhat dependent upon the interaction between all four automata overall in selecting successful actions It can be expected that fast learning should be evident when an automaton detects a strong correlation between its own actions and the environment response; a weak correlation should result in little or no apparent learning occurring. The relative learning rates of the automata applied to the task then implies the relative impact of the respective parameters in producing an action with favourable response from the environment. Strong learning in an individual automaton from an interconnected team indicates that the parameter controlled by that automaton

has most effect in producing a successful action. Such effects are seen in studying the convergence measures of Automaton F.

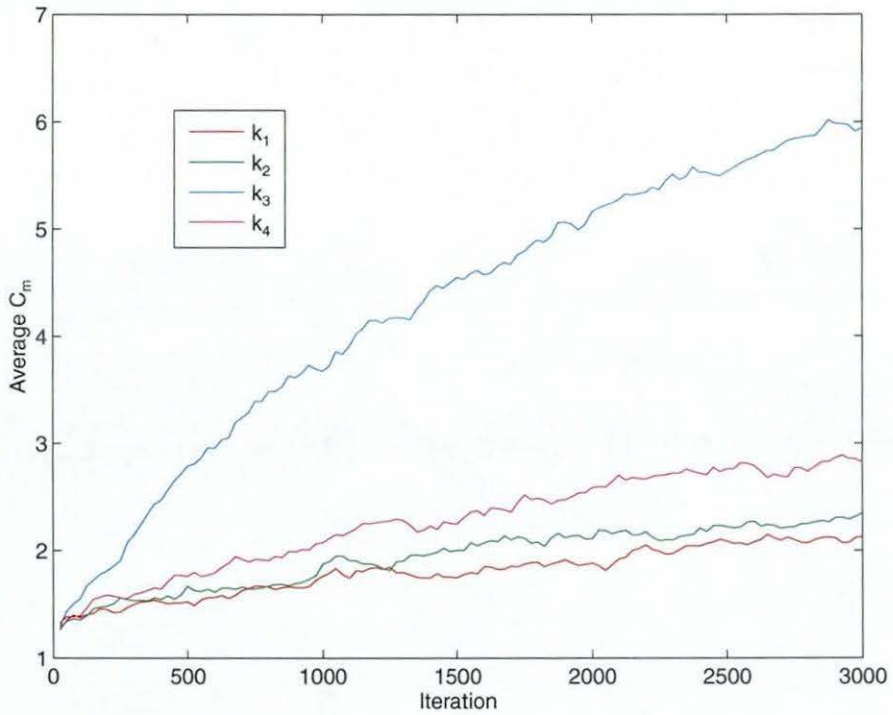


Figure 8.3 - Automaton F: average convergence measure evolution

Figure 8.3 shows the convergence measure for each parameter averaged over the ten learning runs of Automaton F. k_3 is evidently identified as central to controller performance on this task, showing the fastest learning rate. The evolution of the CARLA states during a learning run, Figures 8.4 and 8.5, show this clearly as modal values for k_3 appear around $k_3 = 1000$ very early on, and a single definite peak in the probability distribution is seen from thereon.

Conversely, the k_1 CARLA returns a weak response, with slow learning rates. The probability distribution for the k_1 CARLA appears comparatively 'flat', as no strong trends emerge, and the modal value varies widely throughout the learning period. The performance of such a controller has been noted earlier as being largely independent of k_1 and so the weak CARLA performance on this parameter can be appreciated.

The k_2 and k_4 CARLA return slower responses than that for k_3, similar to k_1, although they do tend to show steady trends in their probability distribution time-histories. For example, the particular learning run in Figure 8.4 shows a case where the k_2 and k_4 CARLA have each identified two values, seen as two peak trends, corresponding to two separate (k_2, k_4) pairs. Indecision between these correlated values is seen to continue throughout much of the learning period - see Figure 8.5.

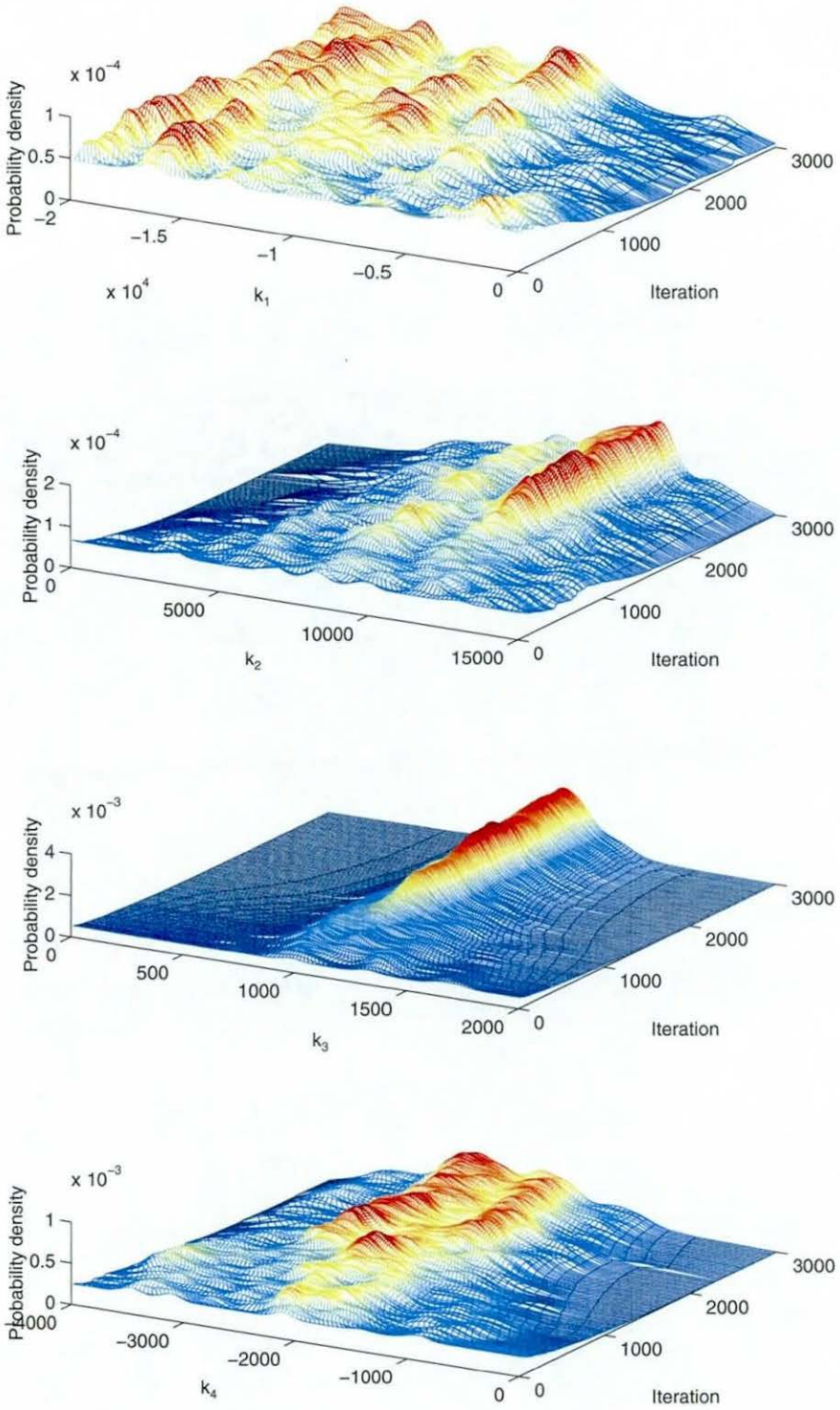


Figure 8.4 - Automaton F: typical probability distribution evolution

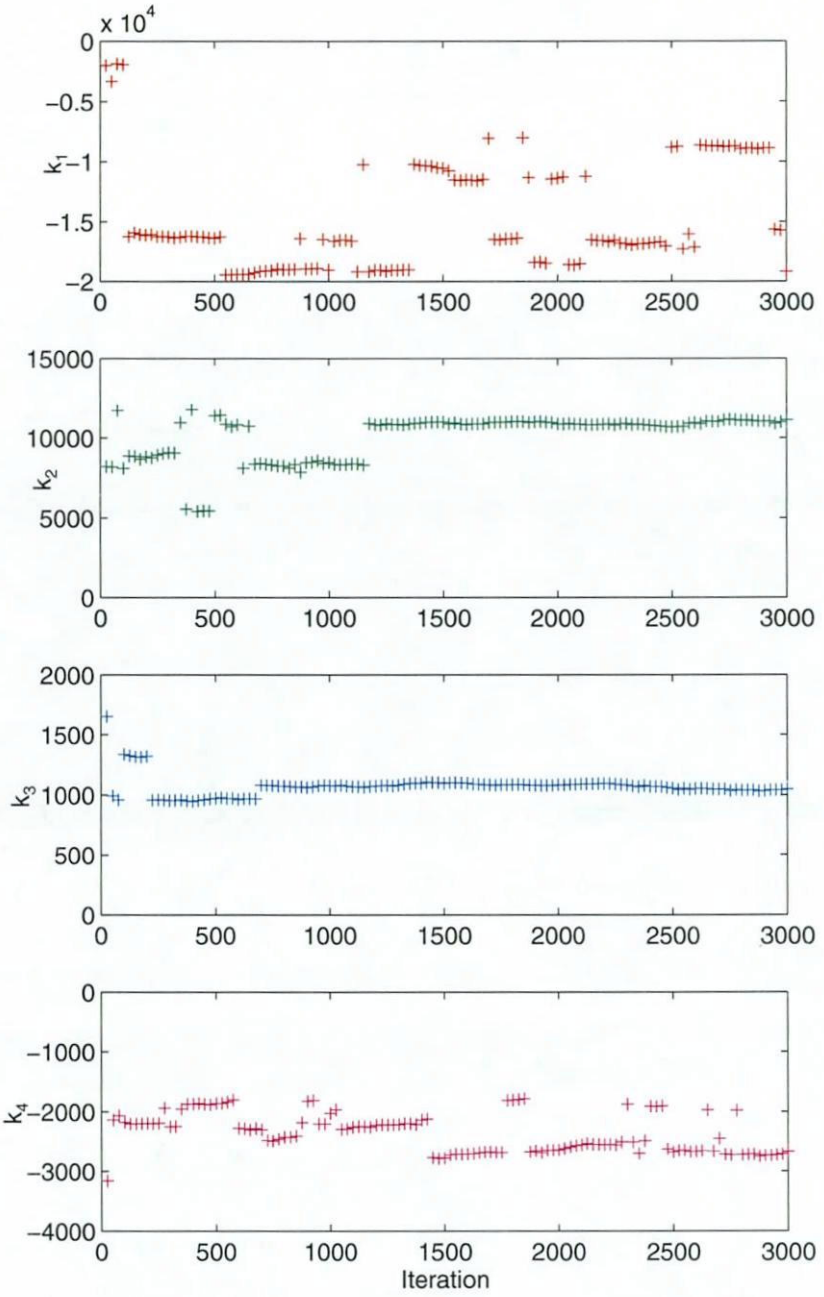


Figure 8.5 - Automaton F: modal value evolution

8.2 Learning with Realistic Road Spectra

Now that the CARLA has demonstrated its effectiveness in a known suspension system environment, it can be developed further with the aim of applying it as an on-line tool This route has already been traced for discrete automata, resulting in some useful modifications to the technique along the way. The primary modification there was inclusion of a 'moderator' that guards learning against excessive state perturbations such that any actions producing such deviant behaviour are rejected immediately. There is no reason to suppose that the moderator will not prove similarly useful when applied alongside CARLA. Consideration of a road spectra input, which can excite the modes of the suspension system more fully, will test this claim.

A suspension system and CARLA configuration, as used in the previous section, is now excited by a Breakback road profile input (Section 2.2). A moderator is applied with the acceptable state limits defined as in equation (4.1), and hence the environmental cost function is reduced to that of equation (4.4), removing terms to control workspace usage which the moderator now controls indirectly. Ten such learning runs are recorded, each concluding at 3000 iterations, referred to as Automaton G. The parameter results are summarised in Figure 8.6. Again, comparison with discrete automaton results on a similar task - Automaton E, Figure 4.8 - shows the CARLA return more consistent parameter results. This is seen most clearly when comparing the (k_2, k_4) pairs from Automata E and G - see Figure 8 7

Automata E and G return similar low variance k_3 distributions, and k_1 has been seen earlier as being comparatively unimportant to controller performance This suggests that the variance difference in (k_2, k_4) results may account for the marked difference in average RMS response seen in simulation - Table 8 2. The controllers of Automaton G are found to make less demand on suspension workspace usage, on average, whilst still achieving a reduced body acceleration over Automaton E.

RMS Response	Automaton E	Automaton G
x_1 (mm)	3 0	3.0
x_2 (mm)	28 6	22.2
x_4 (m/s^2)	0.45	0.43

Table 8.2 - Controller evaluation on Copt Oak road - Average RMS responses.

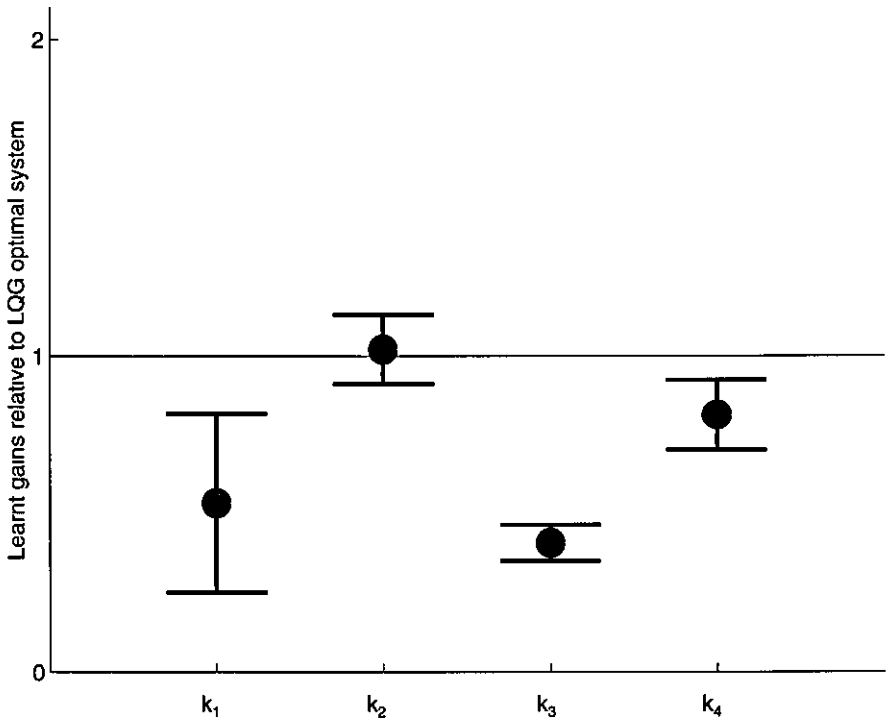


Figure 8.6 - Automaton G: learnt parameter results (mean ± one standard deviation)

Analysis of the convergence measures for Automaton G CARLA - Figure 8.7 - again shows k_3 to be learnt the fastest, implying this parameter is most important to controller performance, although not as prominent as for Automaton F However, k_2 and k_4 now seem to make a greater contribution to performance than before as their convergence measure levels have increased

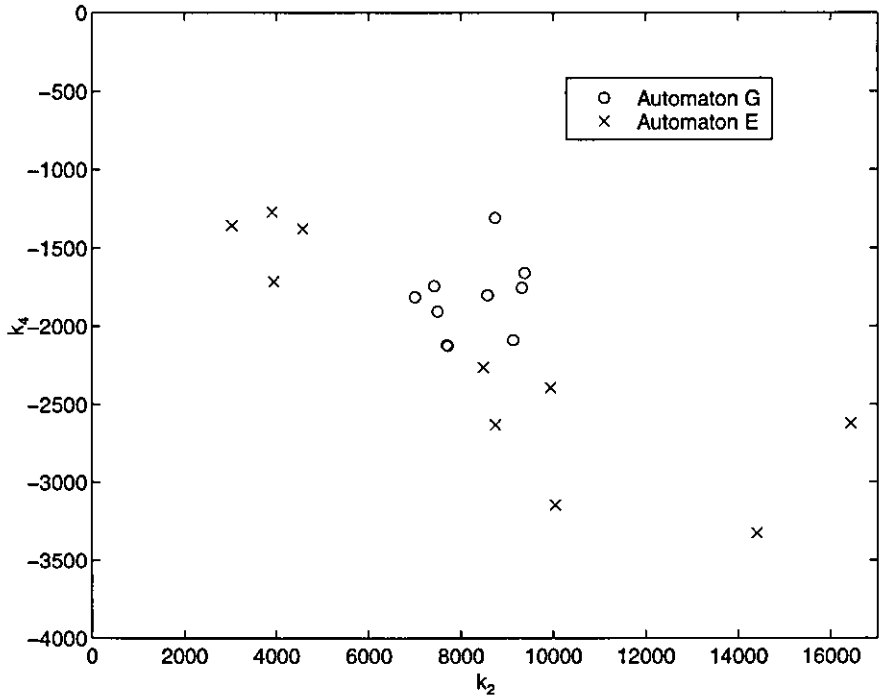


Figure 8.7 - Comparison of (k_2, k_4) values from Automata E and G

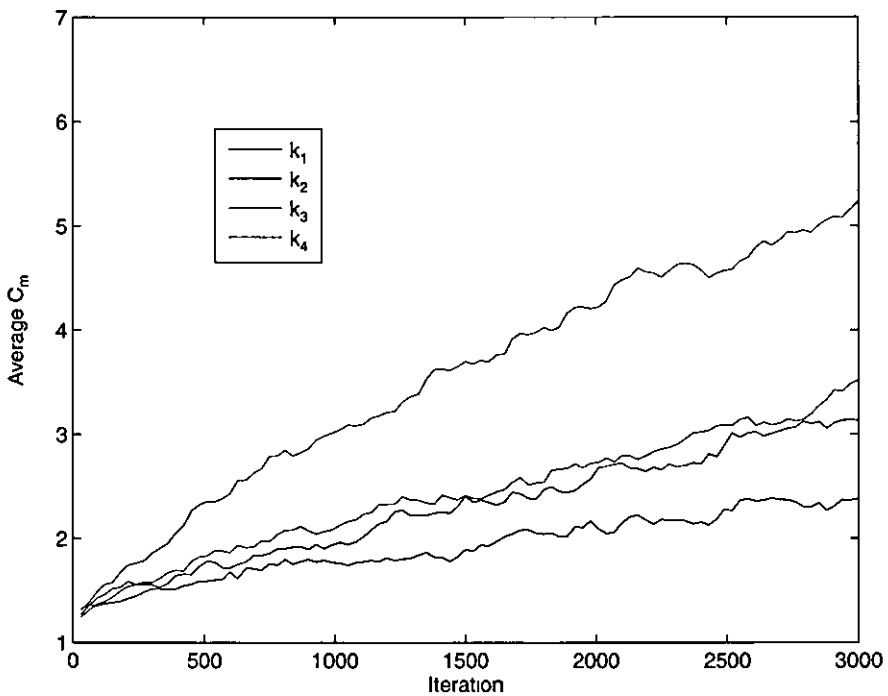


Figure 8.8 - Automaton G: average convergence measure evolution

8.3 On-line Application of CARLA

The previous study has illustrated that CARLA can perform at least as well as discrete automata on a realistic suspension task and, more importantly, is more likely to produce consistent results. This can be tested further in an on-line application. A hardware experiment, based on the system described in Chapter 5, is detailed here. This experiment is previously presented by Howell et al. (1997).

Using identical vehicle settings and input type as used in Chapter 5, three interconnected CARLA are applied at each corner of the test vehicle to learn the three free parameters of equation (5.1) The initial action space of each CARLA cover the parameter ranges

$$\begin{aligned} k_2 &\in [0,15000] \\ k_3 &\in [0,2000] \\ k_4 &\in [-6000,0] \end{aligned} \tag{8.3}$$

with the CARLA learning parameters set at $g_h = 0.3$ and $g_w = 0.02$.

Three independent examples of the learning system are run on the rig with each example using a different Robson road random driving input. No explicit stopping criterion is applied with maximal use of available rig time being the prime objective. This resulted in two tests run for around 2900 iterations, about 13 hours of rig time, with the third test able to be extended to 3300 iterations.

Figure 8.9 shows a typical probability evolution through a learning test where, in conjunction with the convergence measure analysis and modal value evolution, shown in Figures 8.10 and 8.11, it is seen that all automata respond similarly. Each automata is able to identify a strong region of its respective action space despite the non-linearities and sensor noise that are inherently present in a hardware environment.

The relative performance of each learnt controller is assessed by measuring body acceleration during a complete pass of the vehicle over an independent section of road with the same controller gains applied at all four corners. The test road applied for this purpose in Chapter 5 is used here Table 8.3 gives the percentage improvement in r.m.s body acceleration compared to the nominal passive setting - c.f Table 5.1 Generally the CARLA learnt controllers perform similarly to those learnt by discrete automata However, the controllers of Automaton 3, after the longer learning period of 3300 iterations, show a significant advantage. A comparison between the power spectral density body acceleration response of a vehicle with passive suspension and

that of a vehicle with a learnt controller applied is shown in Figure 8.12, where the learnt controller is seen to yield a reduced response across a wide frequency band

	Corner A	Corner B	Corner C	Corner D
Automaton 1	6.5	5 6	3.2	6 0
Automaton 2	6.0	5 6	0.8	7.0
Automaton 3	10 1	9.4	8 7	13.8

Table 8.3 - Percentage improvement in RMS body acceleration over nominal passive damper setting

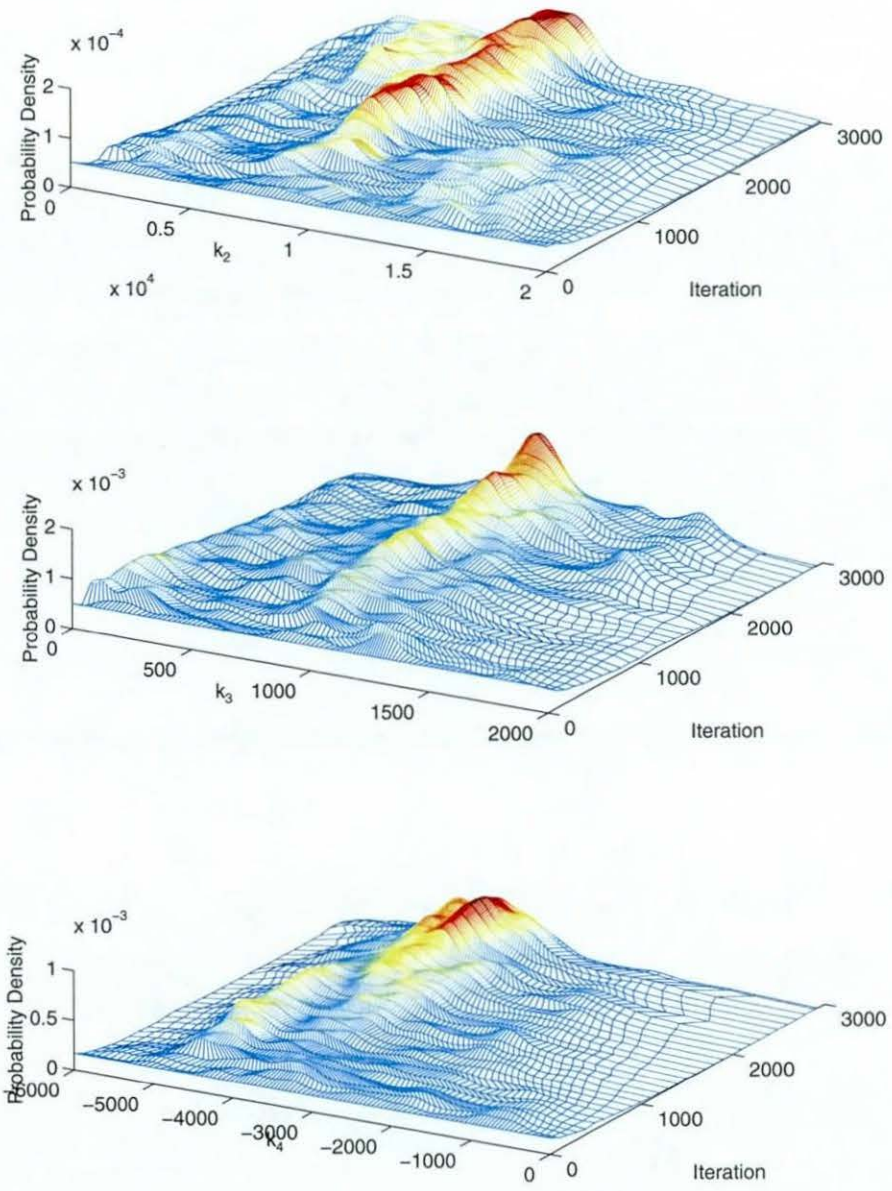


Figure 8.9 - On-line probability distribution evolution (typical)

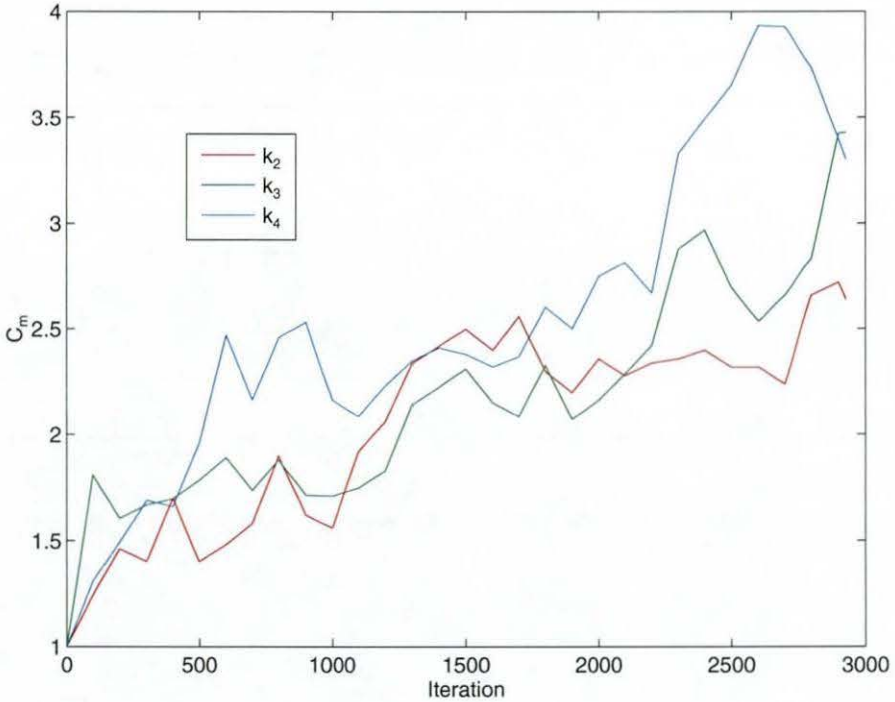


Figure 8.10 - Convergence measure evolution

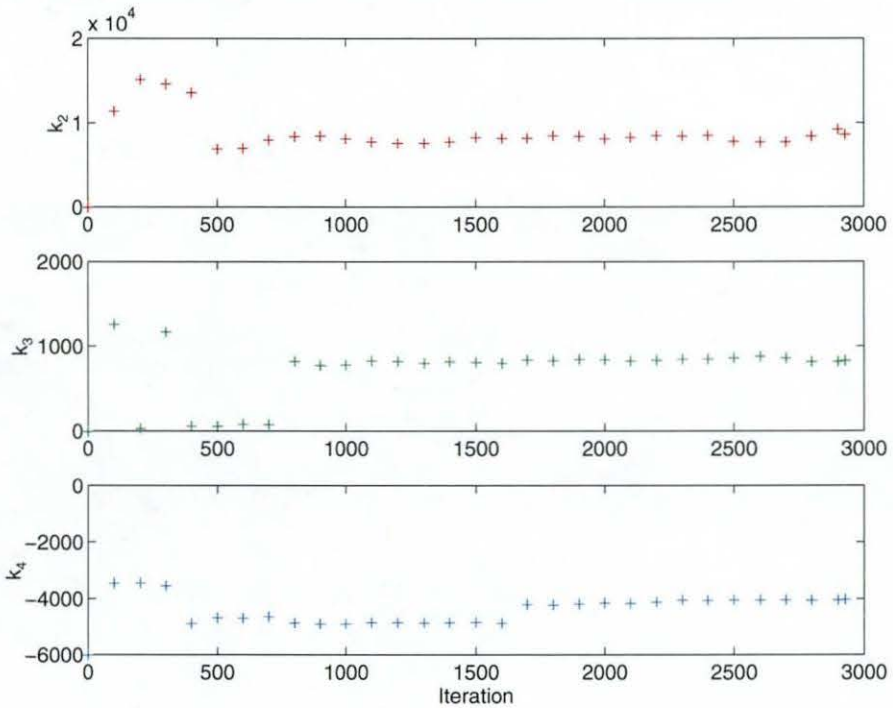


Figure 8.11 - Modal value evolution

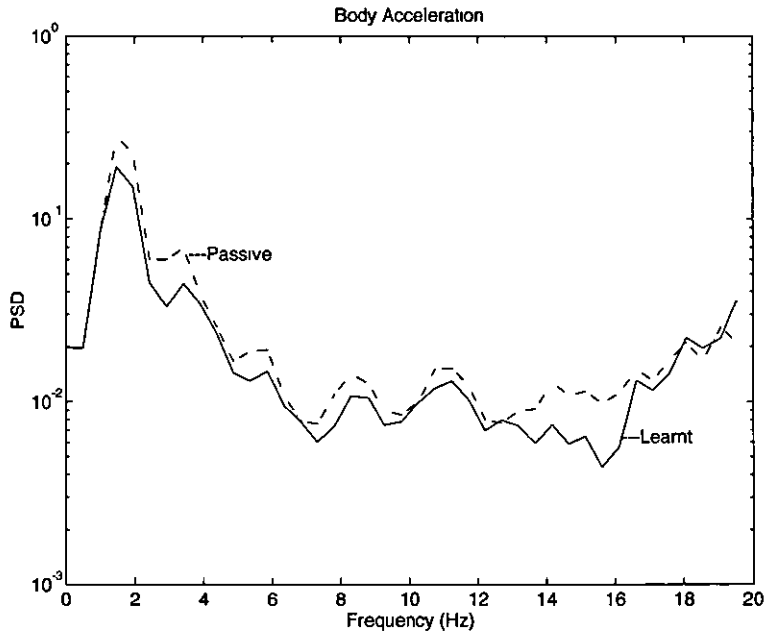


Figure 8.12 - A comparison of the power spectral density function for the best passive damper setting and a learnt controller.

8.4 Discussion

Many of the suspension tasks set for discrete automata have been reassessed in this chapter with CARLA applied. The CARLA has repeatedly shown itself to return more consistent learning with the resultant learnt controllers indicating improved performance.

Practical differences between discrete automata and CARLA were particularly highlighted in the on-line experiment Discrete automata, with the addition of Wu's action set requantisation technique, profit from their small number of actions to quickly locate, and requantise in, a strong region of the action space. However, they then struggle to progress any further with learning after the action set has been re-defined two or three times; the automata is unable to distinguish the relative responses of actions located so close in the overall environment action space. CARLA does not suffer from this as it can maintain an overview of a much larger action space throughout learning. Where discrete automata 'homes in' on a smaller region of the action space, and then attempts to distinguish between specific actions in that region by repeated trial, the CARLA maintains a full action space view and uses the generalisation of reinforcement such that the cumulative effect of trials around a region identifies an optimum point. One effect of this is apparent slower learning by CARLA; it suffers from its much larger action space by initially being unable to locate

strong action space regions as quickly as discrete automata, but profits later on with its ability to continue learning with its cumulative reinforcement generalisation to produce much improved and consistent results.

It was noted that, in multi-CARLA applications, comparison of the convergence measures of each CARLA could indicate the relative import of the respective parameters to the particular learning task. Strong learning of an automaton indicates that its associated parameter has a large impact on the success of any action Conversely, weak learning suggests that a parameter is having little effect, whatever value is chosen for trial by its automaton.

Chapter 9 - Dynamic Vehicle Roll Control

This chapter presents a simulated multi-goal learning task and investigates the ability of CARLA in such a scenario. Learning tasks of previous chapters have each had a single aim, defined via a single performance index feedback from the environment to all automata applied to the task

Here a roll control learning strategy is derived, based on an engineering analysis of actual vehicle hardware, as a precursor to hardware implementation. The strategy splits into two learning tasks with independent but complementary aims: learning a simple ideal feedback roll control law, and learning how the vehicle hardware, with its inherent limitations, can best attempt to achieve the demands of that roll control law. Two interconnected CARLA learning units are implemented on a full vehicle simulation to learn the free parameters identified in the derived control strategy, and investigate the efficacy of CARLA application. Frost et al. (1996) first presented this study

Section 9 1 details the derivation of a roll control strategy, based on the minimisation of a dynamic cost function, aiming to maximise the performance of a semi-active suspension system in attempting to achieve a full-active control law response. The derived control strategy has five free parameters, and Section 9 2 then describes how CARLA are applied, in two teams with independent aims, to learn values for these. The results of simulation studies are presented in Section 9 3, with a discussion of the conclusions that can be drawn from these in Section 9.4

9.1 Roll Control Strategy Derivation

A simple control law for the *desired* roll moment to stabilise a vehicle is

$$M(t) = W_1\phi(t) + W_2\dot{\phi}(t) \tag{9.1}$$

where ϕ is roll angle. To achieve such a roll control law, in practical application, would require ideal force actuation. The available semi-active suspension hardware fitted to the test vehicle is able to achieve some degree of roll moment via variation of damping rates, and thus suspension force actuation, at each wheel station independently. This supplied roll moment is

$$\begin{aligned}\tilde{M} &= \frac{(p+q)}{2}\left(-F_{s1}+F_{s2}+F_{s,3}-F_{s,4}\right) \\ &\equiv \frac{(p+q)}{2}\left(\sum_i \sigma_i F_{s,i}\right), \qquad \sigma=[-1,1,1,-1]\end{aligned} \tag{9.2}$$

where p and q are the lateral distances between the wheels and the vehicle's centre of gravity, and $F_{s,i}$ is the suspension force at a vehicle corner produced by a spring in parallel with a continuously variable damping rate actuator

$$F_{s,i} = K_i X_i + \mu(v_i, c_i) \tag{9.3}$$

Here, X is the suspension deflection, v is the relative velocity across the actuator, and c is the percentage damping request control signal ($0 \le c_i \le 100$) - see Section 2.1.3 for further details.

The natural aim of this control strategy is to minimise the difference between M and $\tilde{M}$ at any point in time. A further aim, related to consideration of the actual hardware operation, is to achieve M via a smooth application of control, i.e. prevent the actuator valves from extensive operation at their limits, and avoid harsh 'bang-bang' control by limiting the rate of change of control current.

These aims are encapsulated in a dynamic 'cost function' L as

$$L(t) = \frac{1}{2}\left(\tilde{M}(t) - M(t)\right)^2 + \sum_i \alpha_i \Psi(c_i) \tag{9.4}$$

where the second term costs deviation of the control signal from the central band $25 < c_i < 75$ with a quadratic cost outside these limits

$$\Psi(c_i) = \begin{cases} 0 & \text{if } |c_i - 50| \le 25 \\ \frac{1}{2}(c_i - 50)^2 - 312.5 & \text{otherwise} \end{cases} \tag{9.5}$$

To avoid rapid switching between upper and lower limits of the actuator valve, a limit is imposed on the magnitude of the change in control during the zero-order hold interval of the controller

$$\begin{aligned}\delta c_i &= c_i(t) - c_i(t-\delta t) \\ |\delta c_i| &\le b\end{aligned} \tag{9.6}$$

Then the control vector is chosen to minimise an estimate of $L(t)$ at each time step

$$\underline{c}(t) = \arg\min \hat{L}(t) \tag{9.7}$$

Subject to the constraint, (9.6), the change in L during the time interval $(t-\delta t, t)$ is given by

$$\begin{aligned}\delta L \approx & \left(\tilde{M} - M\right)\left(\delta\tilde{M} - \delta M\right) \\ & + \sum_i \alpha_i \frac{d\Psi}{dc_i}(c_i)\delta c_i\end{aligned} \tag{9.8}$$

and, from (9.2) and (9 3)

$$\begin{aligned}\delta\tilde{M} &= \frac{(p+q)}{2}\sum_i \sigma_i \delta\tilde{F}_{s,i} \\ &= \frac{(p+q)}{2}\sum_i \sigma_i\left(K_i\delta X_i + \frac{\partial\mu_i}{\partial v_i}\delta v_i + \frac{\partial\mu_i}{\partial c_i}\delta c_i\right)\end{aligned} \tag{9.9}$$

Here δX_i and δv_i are only weakly influenced by any (bounded and small) change in control; hence the first two terms can be ignored. Similarly

$$\delta M = k_1\delta\phi + k_2\delta\phi \approx 0 \tag{9 10}$$

Therefore, from equations (9 8), (9.9) and (9.10)

$$\delta L = \left(\tilde{M} - M\right)\frac{(p+q)}{2}\sum_i \sigma_i \frac{\partial\mu_i}{\partial c_i}\delta c_i + \sum_i \alpha_i \frac{d\Psi}{dc_i}(c_i)\delta c_i \tag{9.11}$$

where

$$\frac{d\Psi}{dc_i}(c_i) = \begin{cases} 0 & \text{if } |c_i - 50| \le 25 \\ c_i - 50 & \text{otherwise} \end{cases} \tag{9 12}$$

For reasonably similar actuators it can be assumed that $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha$, and so (9 11) becomes

$$\begin{aligned}\delta L = & \left(\tilde{M} - M\right)\frac{(p+q)}{2}\sum_i \sigma_i \frac{\partial\mu_i}{\partial c_i}\delta c_i \\ & + \sum_i \bar{\alpha}(c_i).(c_i - 50)\delta c_i \\ \equiv & \sum_i G_i \delta c_i\end{aligned} \tag{9.13}$$

where $\overline{\alpha}$ is defined as

$$\overline{\alpha}(c_i)=\begin{cases}0 & \text{if } |c_i-50|\le 25\\ \alpha & \text{otherwise}\end{cases} \tag{9 14}$$

and G_i is the effective gradient of L w.r t c_i

$$G_i=\left(\tilde{M}-M\right)\frac{(p+q)}{2}\sigma_i\frac{\partial\mu_i}{\partial c_i} + \overline{\alpha}(c_i)\,(c_i-50) \tag{9.15}$$

The required change in control at each time step, to minimise equation (9.4), can be deduced from inspection of equations (9.4) and (9 13), whilst recalling the constraint of (9.6). $L(t)$ is defined as a positive quadratic function, which for minimisation implies that δL should be a maximum negative value to head towards the minimum of $L(t)$ whenever possible

Equation (9.6) limits the change in control to $\pm b$, so $L(t)$ can be best minimised with

$$\delta c_i=-b\,\mathrm{sgn}(G_i) \tag{9 16}$$

giving

$$\delta L=-b\left(\sum_i|G_i|\right) \tag{9 17}$$

Thus c_i always changes by $\pm b$ (maximum change) unless $G_i=0$. More realistically, for 'small' G_i there will be uncertainty in its sign and a deadband in G, $\pm a$, will be included.

The above equations therefore represent a non-linear control law, whose structure has been defined by conventional engineering analysis. The free parameters in this structure are $W_1, W_2, a, b,$ and α.

9.2 Multi-goal Learning Implementation

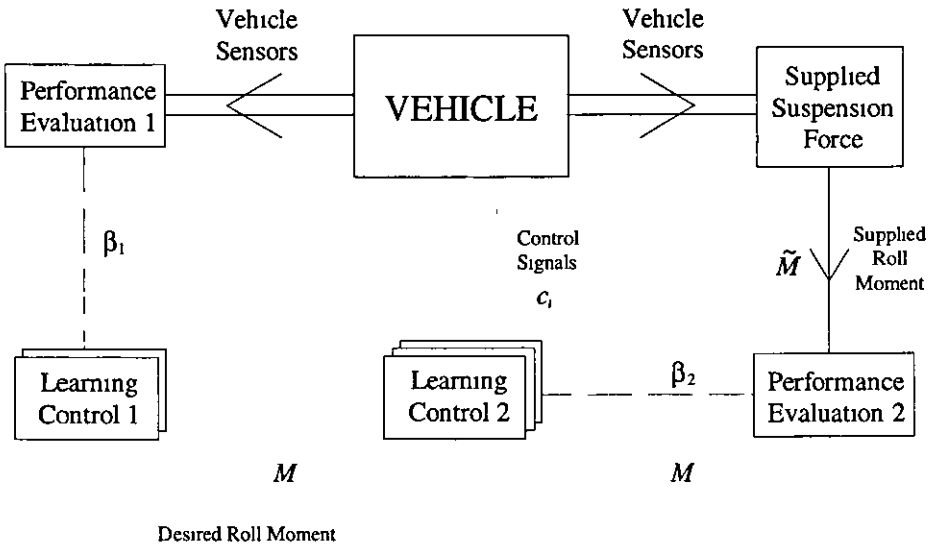


Figure 9.1 - Learning roll control structure

Figure 9.1 illustrates a control structure, including two learning control modules (LC1 and LC2) devised to implement on-line optimisation of the control strategy defined above, via CARLA learning.

LC1 is set to learn W_1 and W_2 of equation (9 1) The performance measure for LC1 is taken from the sum of squared roll angle over a test period of 8 seconds

$$J_1 = \sum_t \phi(t)^2 \tag{9.18}$$

This is derived from vehicle sensor information and is passed to the Performance Evaluation 1 (PE1) module PE1 provides the learning critic, defined for minimisation by equations (6 7) and (6.8) to return a performance index, $\beta_1 \in [0,1]$, to LC1.

LC2 then takes on the task of learning how best to achieve the desired roll moment of LC1, i e. learn operational values of a, b and α. Its aim, to minimise the difference between M and $\tilde{M}$, leads to the performance measure

$$J_2 = \sum_t \frac{1}{2}\left(\tilde{M} - M\right)^2 \tag{9.19}$$

passed to Performance Evaluation 2 (PE2) PE2 is defined similarly to PE1 to provide a minimisation critic, and hence β_2, for LC2.

LC1 and LC2 are thus separate learning units with independent aims, co-operating only via the interaction of their respective actions. Each unit consists of

interconnected CARLA ($g_h = 0.3$, $g_w = 0.02$) with a single CARLA for each free parameter to be learnt by the unit An initial range is identified for each parameter, within which the respective CARLA may search

$$\begin{aligned} W_1 &\in [-1000000, 0] \\ W_2 &\in [-40000, 0] \\ a &\in [0, 5000] \\ b &\in [0, 50] \\ \alpha &\in [0, 100] \end{aligned} \tag{9.20}$$

These values are somewhat arbitrary in choice, although the sign, and general size of the range, has been deduced from considerations of the effect of each parameter individually.

Input to the vehicle model is an artificial dual-track road as described in Section 2.3. A continuous track is formed from 150 seconds of such a road, assuming a vehicle forward velocity of 20m/s.

9.3 Results

Ten independent trials of the above learning strategy are simulated, and the parameter results summarised in Table 9.1. A coefficient of variation is defined as the standard deviation of the ten results for a parameter divided by the parameter's initial range, and hence gives some measure of the variation of the learnt parameters across the ten results, with a low value signifying a consistent learnt value.

Parameter	Mean	Std. Deviation	Coeff. of Variation
W_1	-303072	9091	0.9%
W_2	-12075	1093	2.7%
a	1793	899	18.0%
b	44.1	1.70	3.4%
α	66.4	15 3	15.3%

Table 9.1 - Parameter results

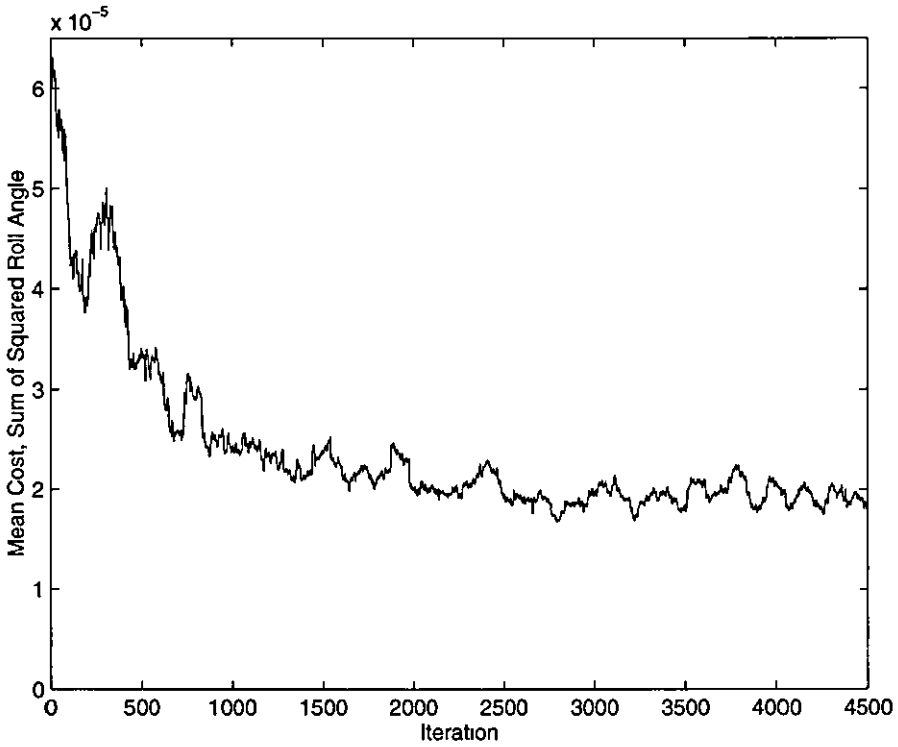


Figure 9.2 - Mean cost seen by LC1

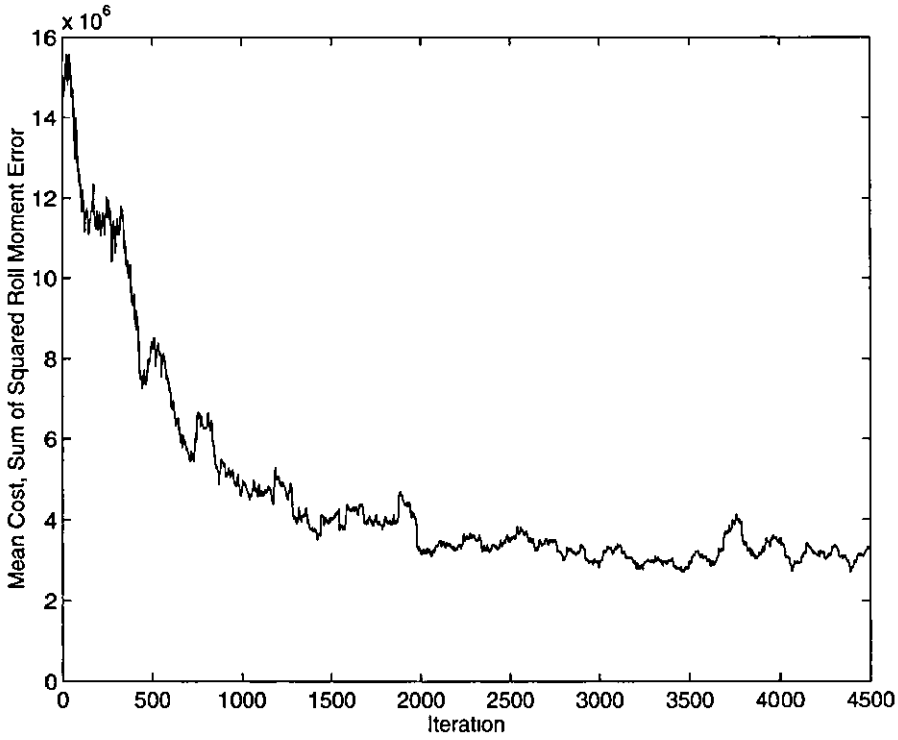


Figure 9.3 - Mean cost seen by LC2

Figures 9.2 and 9 3 show the mean cost (a rolling average of 100 values) returned to LC1 and LC2 respectively during one typical learning run. It is seen that the average cost seen by both units decreases significantly over time. Evaluation of a nominal passive suspension over a typical iteration of the road input (c_i = constant) returns a J_1 value of 14.7×10^{-5}. By applying the derived control structure, with learning of the free parameters, the system easily surpasses this passive suspension performance straightaway, and continues to improve for some time.

After around eight hours of learning (equivalent to 3600 iterations) no further improvement in the mean cost curves is apparent. Taking the parameter values corresponding to the modal value of the respective CARLA probability distribution as the learnt values from one typical learning run gives

$$\begin{aligned} W_1 &= -315300 \\ W_2 &= -11457 \\ a &= 11102 \\ b &= 43 \\ \alpha &= 48 \end{aligned} \tag{9.21}$$

Relative performance of the roll control system with the above parameter values is assessed by comparison with a passive suspension system in simulation over a 1000 metre section of the road on which learning took place. The roll performance of each system is given in Table 9 2. The vertical body displacement has changed little with application of the roll control, but the r.m s. roll angle has improved significantly. Figure 9.4 shows a short time history of the roll angle for the two controllers, and it is clearly seen that the learnt controller consistently returns a roll angle below that of the nominal passive system.

	r m.s. roll angle (rad)	r.m s. vertical body displacement (mm)
Learnt Controller	0.0105	12 6
Nominal Passive	0.0193	12.6

Table 9.2 - Comparison of controller performance

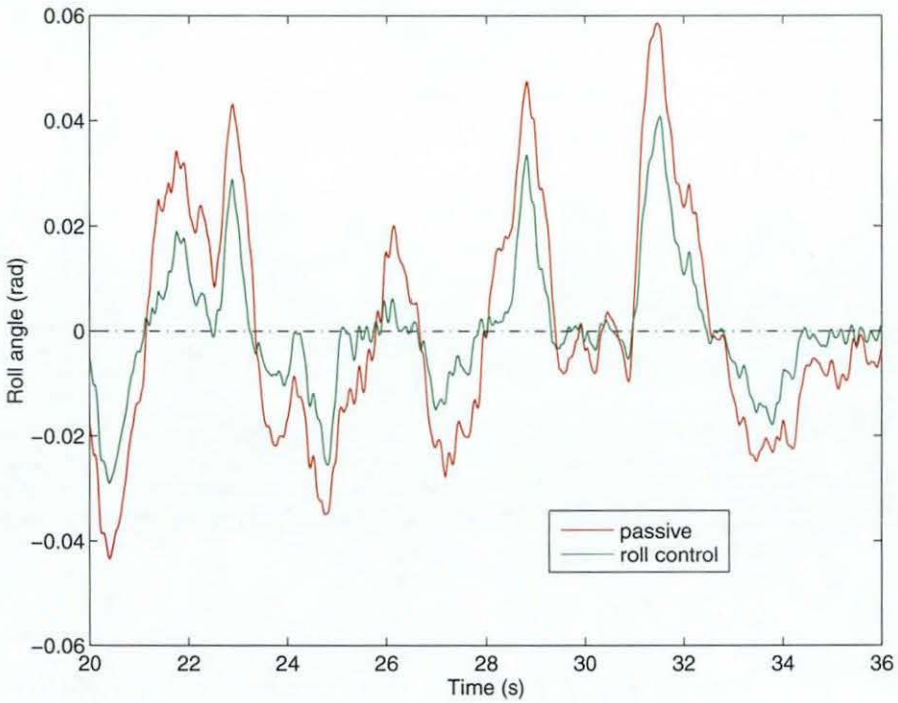


Figure 9.4 - Improvement in roll angle

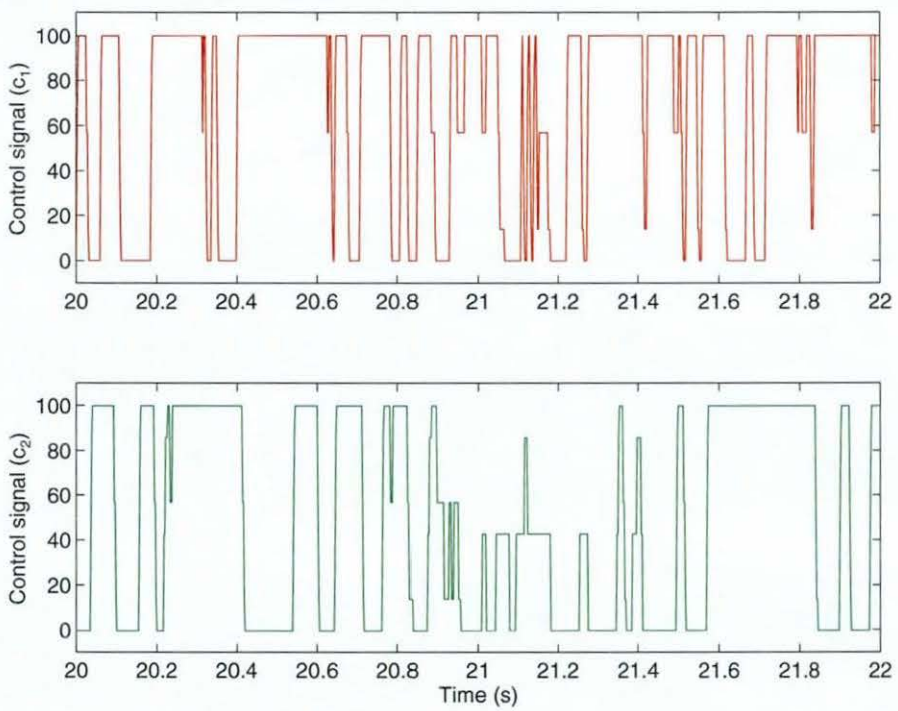


Figure 9.5 - Sample control signal

Considering a brief time history of the control signals to left and right actuators - Figure 9.5 - it is also seen that the control often opposes left to right, as would be expected, to counteract roll velocity in one direction.

The above simulation analysis has shown the roll control strategy, with learnt values of the free parameters, has successfully outperformed a passive suspension system, but how has the learning strategy itself performed? Chapter 7 introduced C_m, a convergence measure statistic which can indicate comparative importance of parameters to a learning task; the CARLA associated with parameters that have greater effect on system performance are likely to show stronger convergence.

C_m is measured throughout the learning periods of all ten simulation runs and the parameter averages are presented in Figure 9.6. Clearly LC1 is able to learn strong values for W_1 and W_2. This is supported by the low coefficients of variation identified in Table 9.1 for these parameters, signifying consistent learning of distinct values for W_1 and W_2. LC2, however, is seemingly somewhat less successful. The C_m plots for a and α barely rise at all, indicative of very weak learning of these parameters; the coefficients of variation are also considerably larger. A high value of b is consistently learnt, however, and the mean cost shown in Figure 9.3 fell considerably so LC2 has achieved its aim in some form.

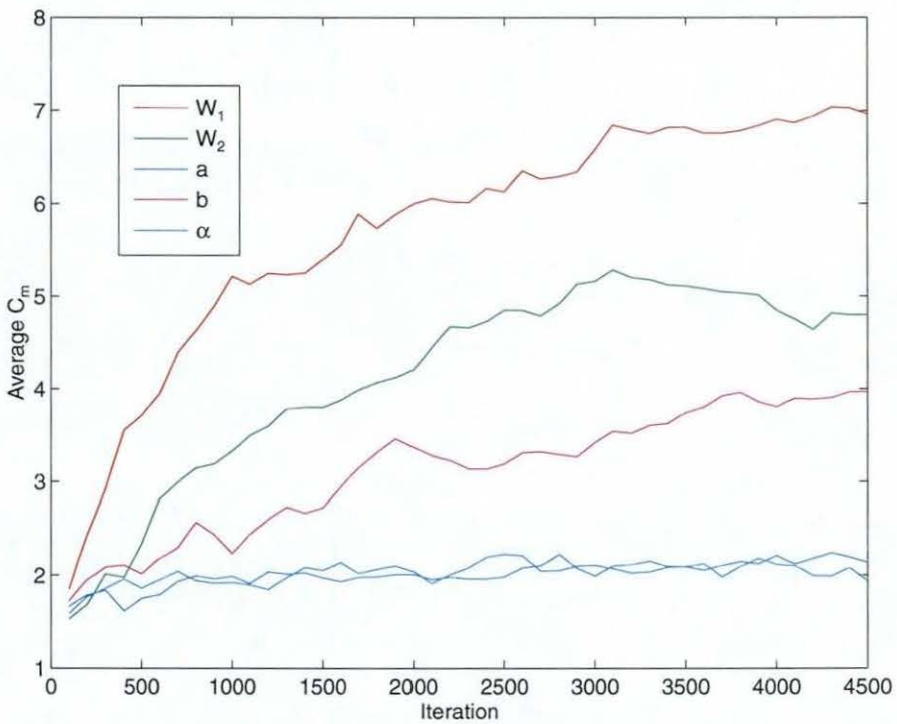


Figure 9.6 - Average C_m from ten experiments

9.4 Discussion

A complex non-linear control problem has been introduced where formal control theory is not well suited to obtaining an optimal design. Although numerical optimisation might be used in simulation, the CARLA methodology can be applied on-line using vehicle hardware and with no explicit detailed modelling. Here, prior engineering knowledge of the basic mechanics of the vehicle system have been combined with the capabilities of CARLA to learn parameters for a full vehicle roll control law.

Analysis of the convergence measure for each parameter highlights a number of points. W_1 and W_2 are quickly and consistently attained for a control law for desired roll moment. As the suspension system is semi-active it is unable to produce the high forces demanded in trying to match the desired roll moment. In trying to do this the system (LC2) has learnt that a 'bang-bang' control is the best compromise; b is learnt at the upper end of its range so the control switches rapidly between its upper and lower limits The parameter α in the control structure employed to avoid this is consequently ineffective, and no sharp value is obtained.

The results presented illustrate that the application of CARLA has been successful in reducing roll angles over a quite severe input This simulation study thus indicates that application in hardware is both feasible and worthwhile.

Chapter 10 - Speculative CARLA Extensions

Two possible modifications to CARLA have been identified during the development of the methodology, namely adaptive action space sizing and non-linear function learning. This chapter documents the preliminary studies of these extensions that demonstrate their viability in simulation. However, in both cases a number of issues remain which will require further research to turn them into truly useful techniques for practical application.

10.1 Adaptive Action Space

The CARLA methodology allows the application of one CARLA per free parameter of any learning task. As it has been defined thus far, the user is required to define a parameter range in which the CARLA can operate; a fixed size action space. The successful operation of the CARLA thus depends upon the user specifying an appropriate action space for each parameter. Normally a prior analysis of the task to which CARLA is being applied should indicate such parameter ranges, although there will always be some cases where a task may be too complex to analyse thoroughly. In those cases is would be preferable if CARLA could be given some freedom to adapt its own action space, effectively introducing an exploration of the infinite action space. An example from the roll control task of Chapter 9 raises the motivation for such ability

Simple analysis of the vehicle system to deduce a suitable range for the parameter W_1 of the feedback roll control law of (9.1) indicates a negative value is required to oppose an induced roll angle. However, an appropriate magnitude is not obvious as it is dependent upon numerous other factors such as the physical characteristics and geometry of the vehicle and the non-linear characteristics of the suspension actuators for instance In fact, the range of $[-1000000,0]$ for W_1 was selected after two iterations of CARLA application. Ranges of [-100000,0] and [-200000,0] were tried previously with the resultant probability distribution for W_1 heavily skewed showing a preference for larger negative values for W_1 Figure 10.1 shows a similar result for the b parameter of the roll control task where large values of b are repeatedly preferred for the most rapid rate of change of the control signal to be available. Clearly significant time expenditure could have been saved if an automata could have noticed such a heavy skew and acted upon it to search further for a more favourable action space itself, rather than requiring iterations by the user to find such an action space range by informed trial and error.

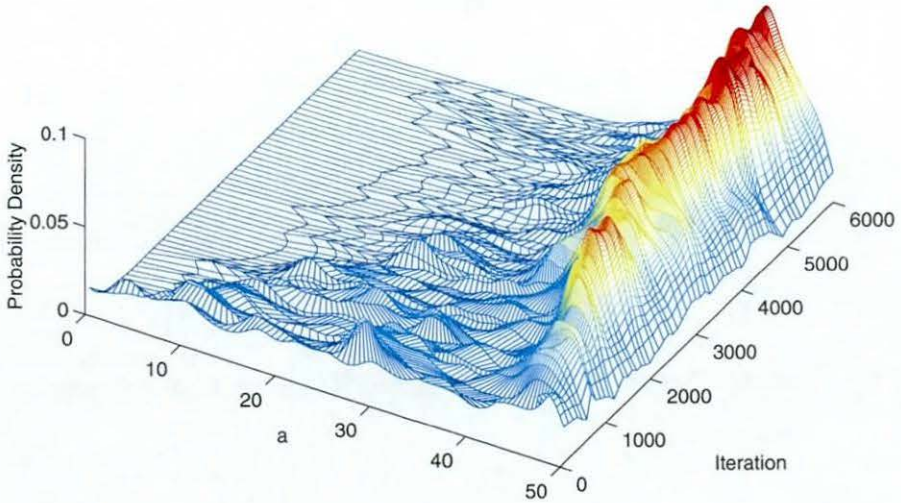


Figure 10.1 - Heavily skewed probability distribution of parameter *b*

The aim of this study is then to investigate addition of an adaptive action space capability to the CARLA methodology, whereby the CARLA is able to vary its own action space where learning shows clear trends.

10.1.1 Concept and implementation

The action space of a CARLA, as described in Chapter 6, is defined between two fixed limit values, α_{min} and α_{max}. Adaptive action sizing allows these values to vary during learning, taking reference to a measure of skewness of the probability distribution in deciding in what manner they should be shifted.

Skewness is ascertained by observing the position of two percentile points of the probability distribution, at 25% and 75% respectively. As the 25th percentile tends lower in the range then the probability distribution is skewing left and α_{min} should be lowered (shifted left) to compensate and allow the CARLA to search more widely into this preferred area of the action space. Similarly, motion of the 75th percentile to the right should raise α_{max}. Such expansion of the action set then allows the CARLA to explore new regions of the action space.

Conversely, there is no reason why α_{min} and α_{max} should not have the capability of moving in such a way as to reduce the action space size. This will be particularly applicable where the probability distribution continuously skews to one side for a prolonged period. Consider a skew to the right where α_{max} is repeatedly increased.

Evidently there is little or no reward for low α and hence no benefit in maintaining $\alpha_{\min}$ at a low value. Allowing $\alpha_{\min}$ to also move to the right, and thereby reducing the action size away from low α, enables the whole automaton action space to track up the real line; vice-versa for a skew to the left.

Motion of an action space boundary is controlled by comparison of a percentile point from the probability distribution against two user-defined limits - see Figure 10.2. A contraction limit (*con_lim*) and an expansion limit (*exp_lim*) are defined in terms of percentages of the current parameter range. If the percentile remains between these limits then the local boundary remains unchanged. However, if the skew of the probability distribution becomes such that the percentile exceeds the expansion limit then the local boundary is moved to expand the action space of the CARLA. Similarly, if the percentile falls within the contract limit then the local boundary is moved to contract the action space. Any expansion or contraction therefore acts to move the percentile back within the limits. The 25th and 75th percentiles are referenced for control of the local boundary. The expansion and contraction limits are similarly defined, symmetrically about the mid-point of the current parameter range.

For example, in Figure 10.2, the probability distribution skewness is such that the 75th percentile lies inside the contraction limit and the right boundary of the action space should therefore be moved to the left.

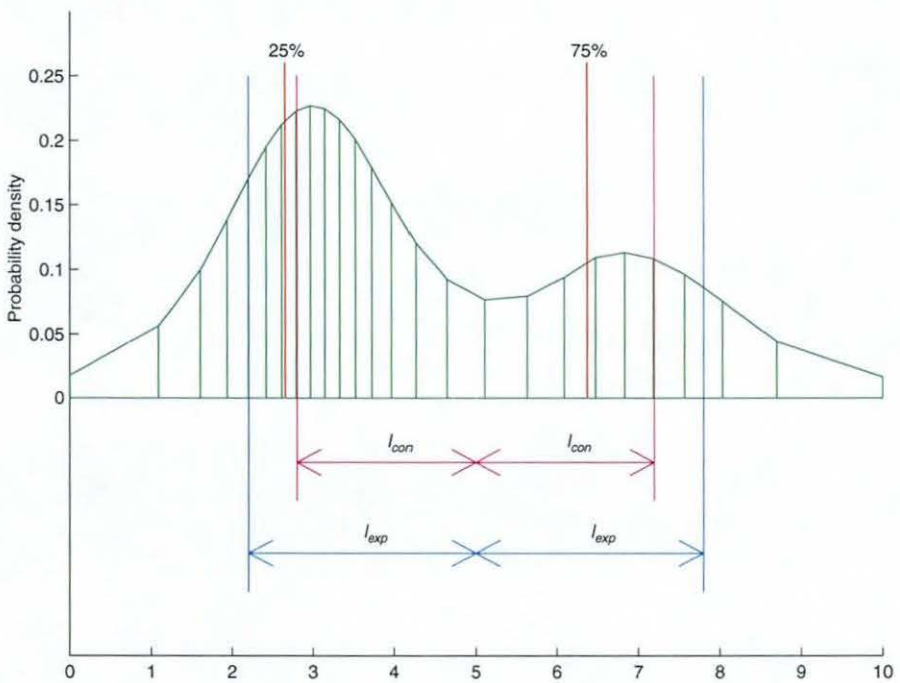


Figure 10.2 - Percentile limits for deducing distribution skewness

When the percentile exceeds one of the limits then the appropriate expansion or contraction of the probability distribution is achieved with

$$\alpha_{\text{limit}}(k+1) = \alpha_{\text{limit}}(k) \mp \theta_{\text{exp}} . \left(\alpha_{\max}(k) - \alpha_{\min}(k)\right) \qquad (10.1)$$

for expansion and

$$\alpha_{\text{limit}}(k+1) = \alpha_{\text{limit}}(k) \pm \theta_{\text{con}} . \left(\alpha_{\max}(k) - \alpha_{\min}(k)\right) \qquad (10.2)$$

for contraction In expansion the boundary point on the probability distribution curve definition is stretched out to a new boundary value, whilst in contraction the probability distribution is truncated back to the new boundary value. Figure 10.3 shows an example of this.

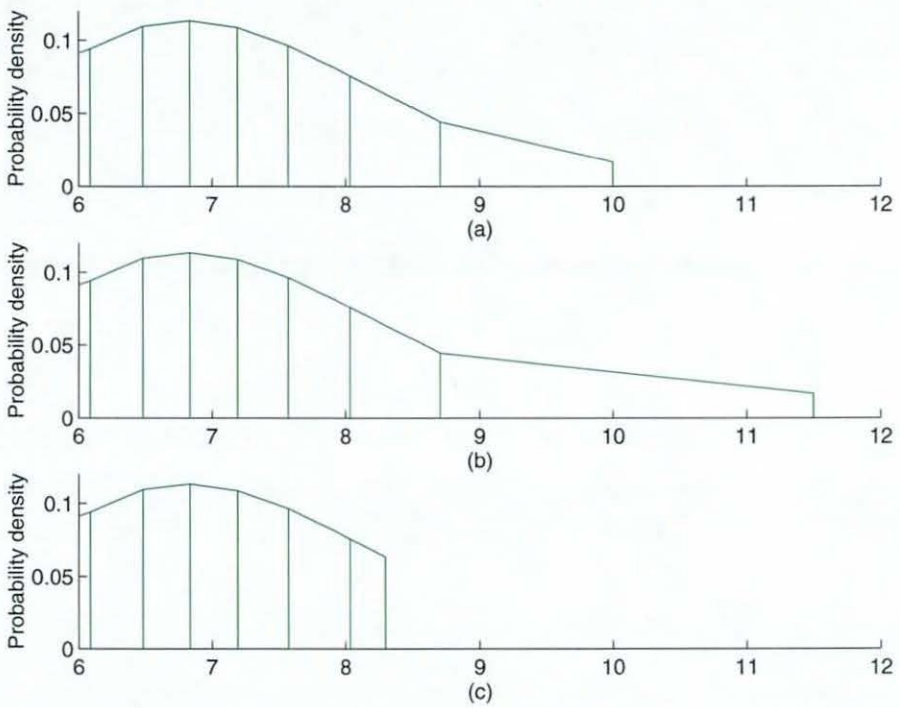


Figure 10.3 - Expansion and contraction of the action space, (a) original distribution, (b) after an expansion, (c) after a contraction

Any such movement of a boundary point will violate the constraint of $\int p(\alpha) = 1$, but in practice simply normalising the curve after any boundary motion enforces the constraint.

10.1.2 Demonstration task

The feasibility of the above algorithm is demonstrated here on a function maximisation learning task. This task is constructed so the modified CARLA has to recognise, and adapt to accurately locate significant changes in the environment response.

An environment response is modelled as the function of two variables

$$z = 30(1-x)^2.e^{-x^2-(y+1)^2} - 100\left(\frac{x}{5}-x^3-x^5\right)e^{-x^2-y^2} - x^2 - y^2 + n \qquad (10.3)$$

where n is uniformly distributed white noise signal, band-limited to ± 5. This function is derived from equation (7.1) used in the comparison of discrete automata with CARLA, with the addition of two quadratic terms that shape the overall surface - see Figure 10.3(a). To model a change in environment response during learning, two variants of (10.3) are taken, derived from the two linear transformations

$$\begin{aligned} x &\mapsto y-3 \\ y &\mapsto x-3 \end{aligned} \qquad (10.4)$$

and

$$\begin{aligned} x &\mapsto x+3 \\ y &\mapsto y+3 \end{aligned} \qquad (10.5)$$

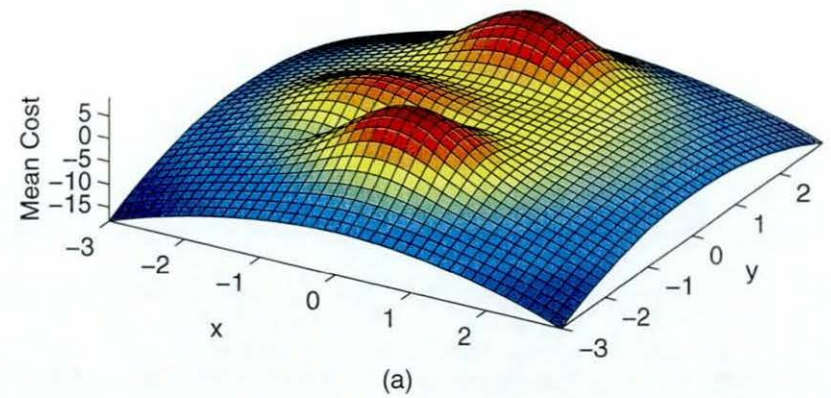
These surfaces (mean value) are shown in Figure 10.4, (b) and (c) respectively. For the initial 5000 CARLA iterations the environment response conforms to (10.4), and thereafter (10.5) is applied.

Two modified CARLA are applied in an interconnected format to learn maximising values of x and y respectively. The parameters of these CARLA are $g_h = 0.3$ and $g_w = 0.02$ as before, with the additional parameters for action space boundary control defined as, $l_{exp} = 0.018$, $l_{con} = 0.01775$, and $\theta_{exp} = \theta_{con} = 0.001$.

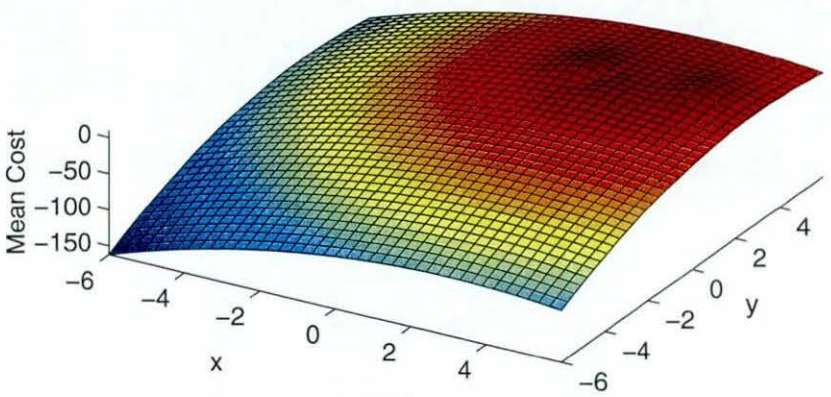
The initial action space for each CARLA is taken as [-10,0] This choice simulates an erroneous selection of initial action space by the user as the maxima of the first environment response function, (10.4), lie outside these limits.

The CARLA are applied for a total of 10000 iterations, and the probability distribution time histories are shown in Figures 10.5 and 10.6. These show the CARLA successfully altering their action spaces and locating maxima. They both manage to

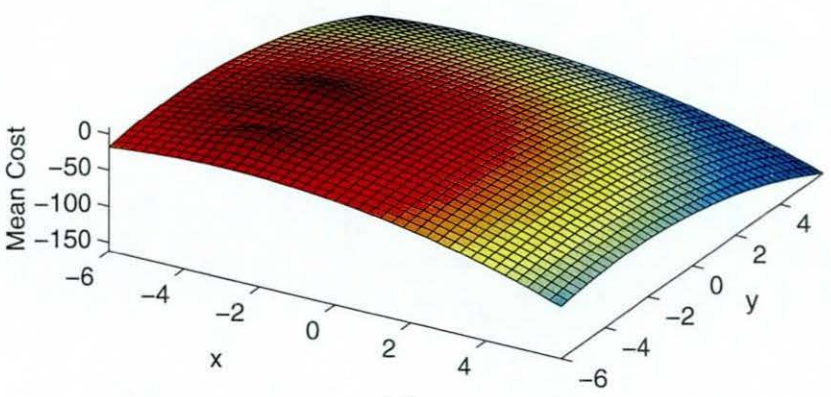
- alter the initial action space to contain the global maximum of (10.4)
- locate the global maximum accurately, concentrating attention around the maximum upon repeated reinforcement in that area
- react to the change in environment after 5000 iterations by 'unlearning' the prior knowledge
- relocate the action space to contain the new global maximum of environmental response, the maximum of (10.5)
- locate and concentrate attention around that maximum



(a)



(b)



(c)

Figure 10.4 - Mean value environmental response surfaces

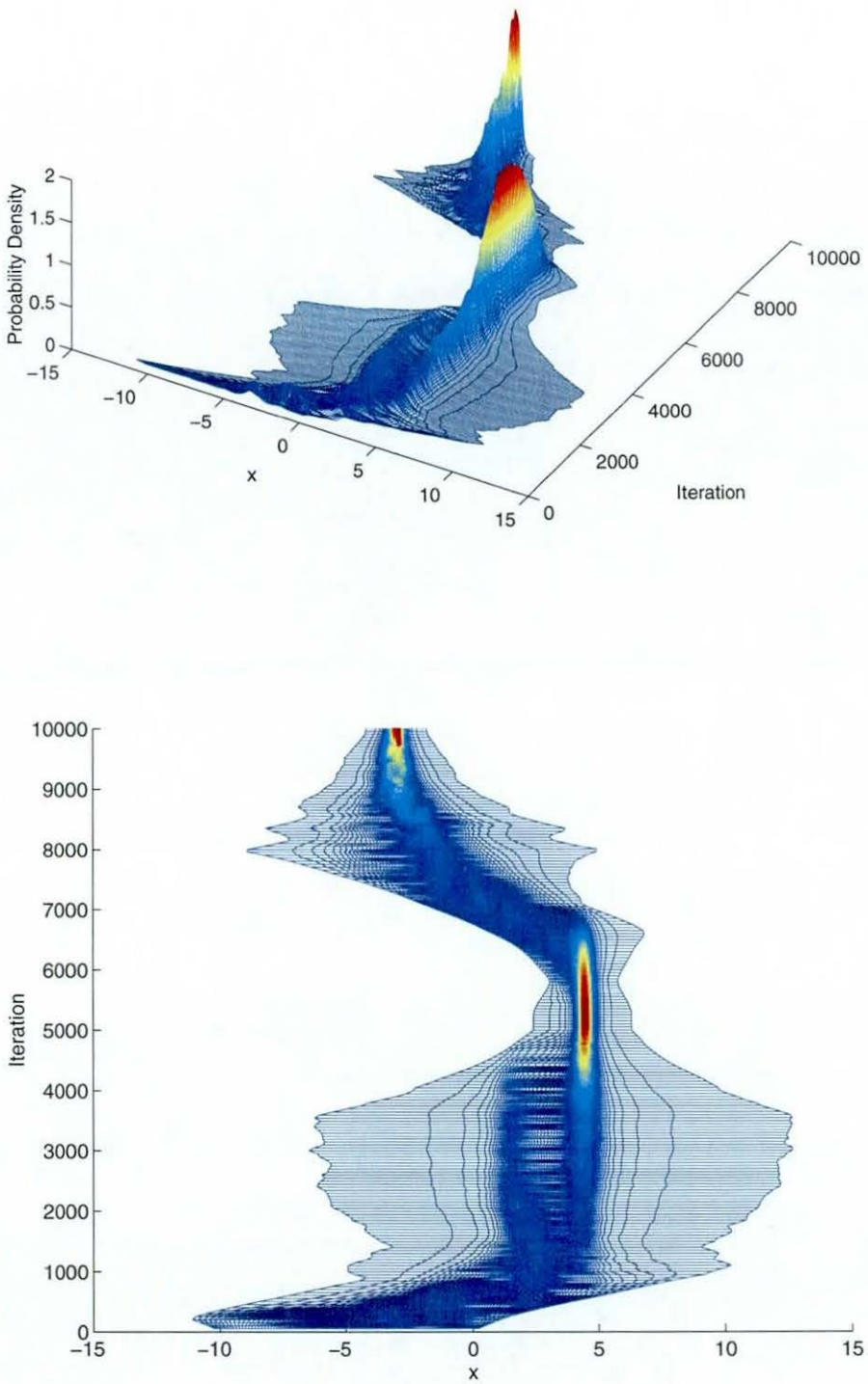


Figure 10.5 - Probability distribution time history for x with adaptive action space

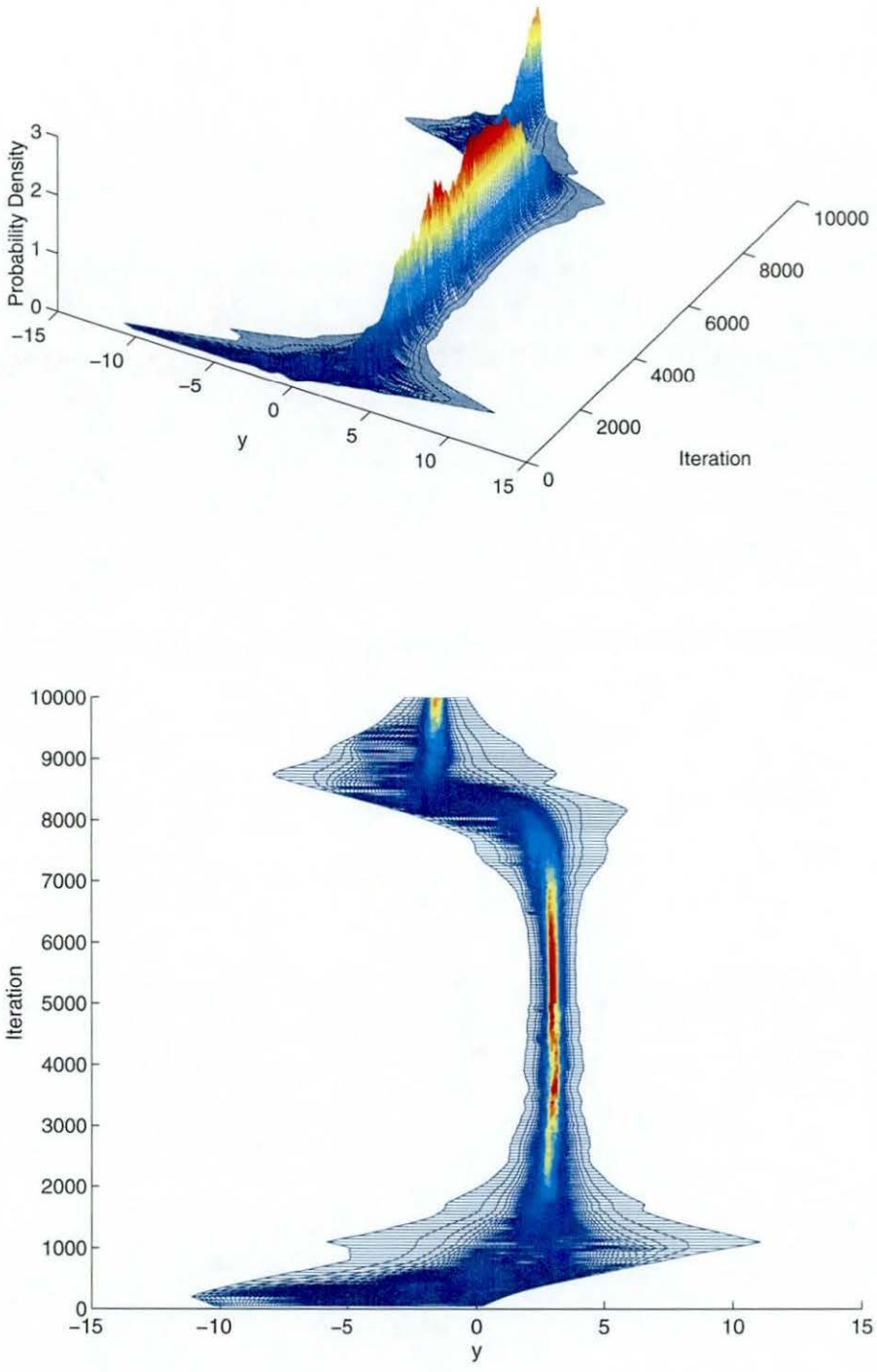


Figure 10.6 - Probability distribution time history for y with adaptive action space

10.1.3 Discussion

The above application of the modified CARLA with adaptive action space control clearly demonstrates the technique is viable and could be developed into a useful tool. However, the current methodology suffers from a few weaknesses resulting in the adaptive action space CARLA performance being very sensitive to small variations of the boundary control parameters for a given task; an inappropriate number of trial runs were required to set working boundary control parameters for the demonstration task. These parameters are seen, in practice, to define a fine balance between the exploration and exploitation properties of the automata

Over-exploration is seen to occur as the CARLA excessively extends its action space; the rate of increase of the CARLA action space size overwhelms any reinforcement process and hence recovery to a viable action space is unlikely. If applied in a control optimisation setting, such expansive behaviour could easily encroach on unstable regions of the control space.

Conversely, excessive exploitation can occur if the CARLA reduces its action space around a strong action to such an extent that, should the environment response change, the CARLA is unable to extract itself from this region of the total action space within a reasonable time scale.

The extra quadratic terms of equation (10 3) over (7.1) have been added as an aid to the boundary control on this task These additional terms introduce a significant gradient to the environment response away from the maxima. If the current CARLA action space is focussed away from the maxima this gradient helps to introduce a skew in action reinforcements such that the boundary control will move the action space towards the area of maximum reinforcement. Without these terms, the environment response is 'flat' away from the maxima and the CARLA tends to over-explore without limit

The introduction of adaptive action space sizing also raises a concern regarding local optima With the fixed action space CARLA, it has been seen that the CARLA frequently locates the global optimum. Considering an adaptive action space CARLA, location of a global optimum within the initial action set can be reasonably expected and the CARLA will reduce its action space around the region of that optimum If, say, the global optimum then shifted suddenly away from that region, the CARLA is required to expand its action space again. In this case it is possible that the CARLA will locate and subsequently settle around the first optimum it finds, which is not necessarily the global optimum

10.2 Non-linear Control

Up until now only linear controllers have been considered in the application of CARLA to optimising vehicle suspension characteristics. Linear controllers were initially chosen as they provided, for some specific cases, a theoretical solution against which the ability of various learning automata could be judged during their development. However, where non-linear systems are considered, especially practical studies in hardware, it is likely that improved system performance can be found from applying non-linear control. This section investigates the application of CARLA to learn a non-linear ride controller for an ideal fullactive vehicle suspension where some significant non-linearities are present, in the form of bump-stops applied for suspension deflection above a pre-defined limit. The implemented technique is based on simple function approximators, where interconnected CARLA are applied to learn values for the free parameters of the approximators.

10.2.1 Concept and implementation

Previous linear feedback control laws for the quarter-vehicle suspension model have been of the form

$$u(t) = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix} x \tag{10.6}$$

The approach adopted here is to replace each static gain value of (10 6) with a gain function, so that the control law becomes non-linear in its parameters thus

$$u(t) = \begin{bmatrix} 0 & k_2(x_2) & k_3(x_3) & k_4(x_4) \end{bmatrix} x \tag{10.7}$$

Note that to reduce the learning task slightly, the relatively ineffective feedback of k_1 is again dropped

Each gain function can now be learnt by applying a family of interconnected CARLA to learn the free parameters, w_j, of a piecewise-linear function approximator construct of the form

$$k_i(x_i) = \sum_{j=1}^{5} r_j(x_i)\, w_{i,j} \tag{10 8}$$

where r_j are 'roof' basis functions. Figure 10.7 illustrates r_j for the feedback gain function on x_2. Here the nodal points of the r_j are distributed evenly between the state limits of x_2 defined for the moderator. r_1 and r_5 are shaped beyond these limits for a reasonable continuity of the gain functions, should these limits be exceeded.

Function approximators are similarly defined for $k_3(x_3)$ and $k_4(x_4)$ with roof functions distributed between the respective moderator state limits imposed on x_3 and x_4. Fifteen CARLA are thus required on this task to learn the $w_{i,j}$ and define three gain functions for state-feedback controller of (10.7).

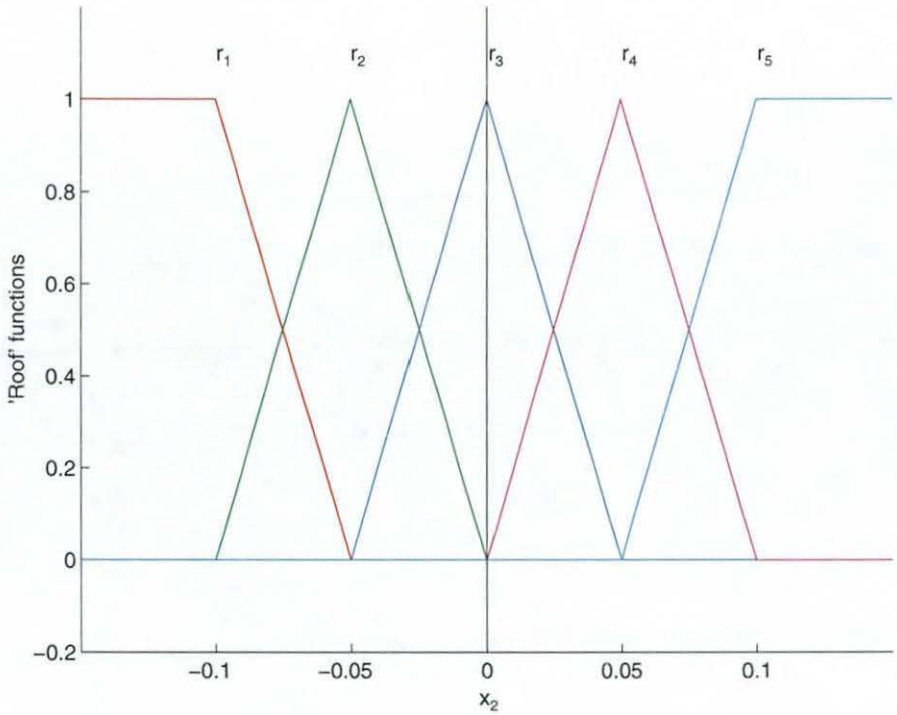


Figure 10.7 - Piece-wise linear function approximator, $k_2(x_2)$

10.2.2 Demonstration task

An active quarter-vehicle suspension system is considered, of the form described in Section 2.1.2. The state feedback control law of (10.7) is to be optimised by learning to minimise vertical body acceleration. A significant non-linearity is introduced into the system, in the form of bump-stops that the CARLA will need to make allowances for; a smooth ride response is best achieved by avoidance of any interactions with bump-stops. In hardware a bump-stop is often comprised of rubber cones which are used to limit excessive suspension deflections and protect surrounding hardware from the impacts caused by such deflections. In simulation they are effectively an additional strong spring force applied between the sprung and unsprung mass on excessive suspension deflections. Here bump-stops are applied for suspension deflections of $|x_2| > 50$mm with a spring rate of 60kN/m.

The fifteen CARLA are defined in standard fixed action space form, with learning parameters $g_w = 0.02$ and $g_h = 0.3$, and minimisation performance evaluation defined by equations (6.7) and (6.8). Reasonable action space ranges for the respective gain

functions can be taken as for single gain learning of previous studies. Each set of five CARLA are defined over the common action space

$$\begin{aligned} w_{2,j} &\in [0,15000] \\ w_{3,j} &\in [0,2000] \quad j = 1,2,\ldots,5 \\ w_{4,j} &\in [-4000,0] \end{aligned} \tag{10.9}$$

A continuous track is formed from the Breakback road profile traversed at 20m/s. This provides a harsh input to the system that can excite the system modes significantly and will bring the bump-stops into use during learning.

A moderator with the limits of (4.1) is applied, and the performance index is defined as the squared body acceleration over 16 second iterations, (4.4), for all CARLA.

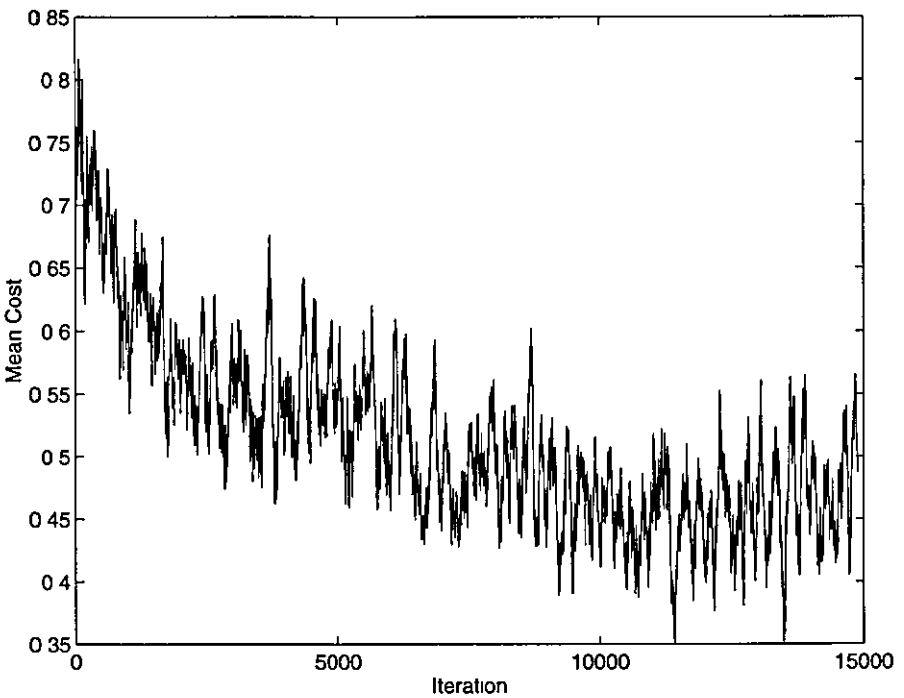


Figure 10.8 - Mean cost time history

The mean cost result of a typical learning run is shown in Figure 10.8. The CARLA are seen to exhibit improving performance to around 15000 iterations, considerably longer than the three thousand iterations required on the four gain learning task of linear feedback control. The mean cost is also 'noisier' than seen before. Both effects can be attributed to the increased dimensionality of the task, fifteen CARLA are required to co-operate in producing successful control here, where only four were used previously. The probability of selecting a successful action for trial is reduced as the dimension of the task increases, and hence the possibility for wide variations in action performance is increased.

A set of ten automata are simulated, with the stopping criterion set at 15000 iterations, referred to as Automaton H The average convergence measures at the end of each run for the fifteen automata are given in Table 10 1, showing a reasonable level of learning has taken place on all automata. In particular, $k_3(x_3)$ has shown the strongest learning. A similar result has been noted in previous studies where k_3 learning was often dominant as this parameter is found to be most important to achieving a favourable environmental response.

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$w_{i,2}$	2 7	3 1	3.1	3.5	2.4
$w_{i,3}$	2 2	3 6	5 5	4.0	2.5
$w_{i,4}$	2.2	3.0	3.2	2.9	2 4

Table 10.1 - Convergence measures of Automaton H

The average resultant gain functions are illustrated in Figure 10 9 There is a clear characteristic apparent in these plots. It is evident that the CARLA has learnt 'soft' suspension settings - low gain values - for small state deflections, whereas, for more excessive deflections, a larger magnitude control is preferred. Such settings seem entirely reasonable in attempting to avoid the harsh effects of impacting on the bump-stops.

Comparison simulations of an Automaton H controller against a controller from Automaton G confirms the above observation - Automaton G is similar to the task considered here, except that a four gain linear state feedback controller is optimised without bump-stops. The two Automata are each simulated over a 3000 metre section of the Breakback road, with bump-stops applied in both cases. In the first simulation the input magnitude is scaled up by 50% to produce a severe input which will cause the bump-stops to be applied. The magnitude is reduced by 50% in the second simulation to analyse the system response on a smoother input. Tables 10.2 and 10.3 record the RMS responses. Over the harsh version of Breakback Road the Automata are seen to respond similarly. However, across the reduced road profile Automata H profits slightly from applying reduced control force, resulting in marginally reduced body accelerations as greater use is made of the available tyre and suspension workspace.

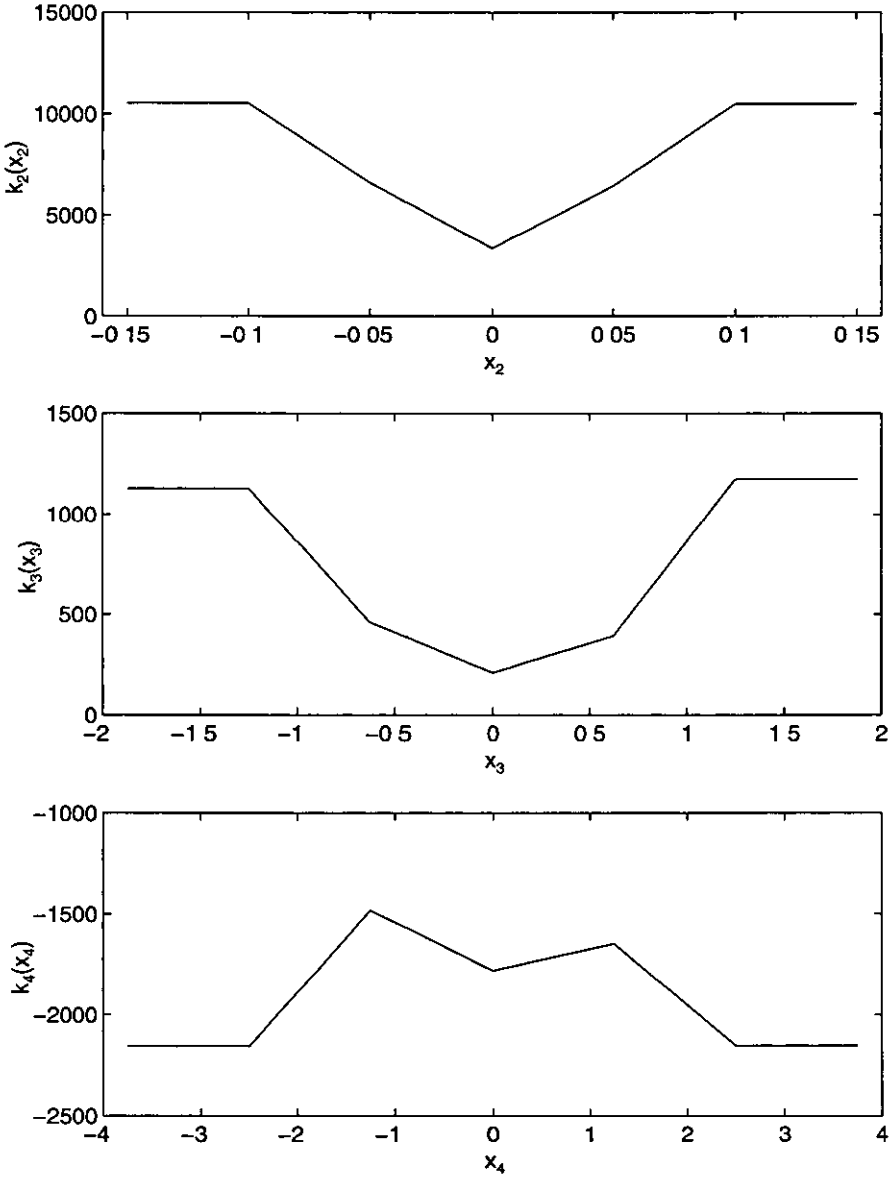


Figure 10.9 - Learnt controller gain functions

RMS Response	Automaton G	Automaton H
x_1 (mm)	6.7	6 8
x_2 (mm)	40.5	40.7
x_4 (m/s^2)	1.45	1.46

Table 10.2 - Controller evaluation on a 'harsh' Breakback road - RMS responses

RMS Response	Automaton G	Automaton H
x_1 (mm)	2.3	2 7
x_2 (mm)	19.3	23.1
$\dot{x}_4$ (m/s^2)	0.35	0.33

Table 10.3 - Controller evaluation on a 'soft' Breakback road - RMS responses

10.2.3 Discussion

The synthesis of a non-linear control law by reinforcement has been successfully demonstrated, in principle, even though there was only a marginal improvement in results on the particular task chosen in comparison with learning a linear control law. The major drawback seen in this CARLA application is plainly related to the increased dimension of the task; by introducing many more free parameters in the controller structure then significantly longer learning times are apparent. However, it should be noted that, had discrete automata been applied to this task with each free parameter quantised to just three values, say, then the action set, 3^{15} = 14,348,907 actions, would inevitably have resulted in no effective learning whatsoever. The generalisation of CARLA, and its application in an interconnected structure, has at least allowed such a task to be attempted with some success.

Chapter 11 - Summary and Conclusions

This dissertation has considered the application of a learning automaton technique to perform on-line parameter optimisation tasks in complex dynamic and stochastic environments Limitations of classical learning automata encountered in initial feasibility studies have led to the development of a new automaton formulation, the Continuous Action-set Reinforcement Learning Automata (CARLA). Subsequent study of the CARLA's properties, predominantly in the form of empirical investigations in simulation, has demonstrated that CARLA exhibits many beneficial properties. Successful on-line application of CARLA on vehicle hardware supports these findings. A discussion of the main points of interest arising during this study is now given, along with suggestions for further research.

The optimisation of a vehicle suspension system, with its complex dynamics and naturally stochastic driving input, presents a difficult task. Suspension tuning traditionally involves considerable modelling effort to facilitate standard optimisation techniques, and even then lengthy subjective tuning is often used to overcome the shortcomings of the modelling process when compared with real hardware characteristics. Classical discrete learning automata have been identified as an approach to such an optimisation task that does not require detailed system modelling. Learning automata apply a reinforcement learning method to learn an 'optimal' action via direct unsupervised interaction between the automaton and the target environment. Feasibility studies with linear, reward-inaction, discrete learning automata in simulation support this; a ride optimisation of a quarter vehicle full-active suspension controller, where a solution can be found *a priori* from LQG theory, demonstrates the ability of a learning automaton to locate near-optimal solutions.

It was noted, however, that learning automata in their standard form would not be suitable for immediate on-line application, as it is possible that unstable control actions can be selected for trial during learning To overcome this the concept of a moderator has been introduced. The moderator acts as an overseer to the learning process, monitoring the environment state for excessive, possibly unstable deviations. The learning automaton operates normally, still able to select unstable actions. However, if the moderator perceives an excessive deviation in the environment then the learning automata is flagged to immediately fail the current action under trial To return the environment to an acceptable state a known stable action is applied for the remainder of the trial period. Failure of any action makes the automaton treat that action as if it had returned a minimum performance index and hence receive no reinforcement. In

the long term, actions which are repeatedly 'failed' by the moderator become less likely because of increased probabilities elsewhere in the action space.

The moderator allows a learning optimisation task to be considered in two parts; hard limits can be monitored by a moderator whilst the automaton can concentrate its effort on the main optimisation task For instance, on the quarter vehicle ride optimisation, two terms of the original cost function were used to cost suspension workspace usage, which is not inherently part of a ride optimisation but is required to constrain learning to locate 'sensible' control levels within the physical limits of the system With a moderator in place to watch for excessive workspace deviations the workspace costing terms can be removed from the environment response. The learning automaton can thus optimise vertical body acceleration alone and the moderator limits the success of control actions to only those which can maintain reasonable workspace usage during their trial.

Applying the moderator in conjunction with the learning automaton now permits the usage of the methodology on-line, removing concerns arising from possible unstable action selection. A first hardware trial of discrete learning automata was subsequently implemented on a test vehicle, excited on a four-post hydraulic road simulator. Discrete learning automata were applied independently at each vehicle corner to learn the four gains of a linear state-feedback controller, as previously analysed in simulation. However, the vehicle suspension consisted of limited bandwidth semi-active suspension, unable to provide the control level of a fullactive system, and so included much more dynamic complexity than experienced by the automaton in simulation Despite this it was seen that discrete learning automata could learn a level of control which surpasses the ride performance of standard passive suspension without any modelling of the complex system being considered.

Two shortcomings of discrete learning automata had become evident in these studies, both related to the discretisation of the action space itself. Firstly, by quantising an action space into a finite number of actions, the automaton cannot investigate performance at intermediate actions. It may therefore easily miss features of the action space in those regions. Of course the likelihood of missing optima may be reduced with finer quantising of the action space, but this quickly leads to a large action set resulting in very slow or inconclusive learning. This effect is especially magnified in higher dimension tasks. Secondly, an effective method implemented to attend to the above weakness is seen to introduce another limitation. A multi-stage learning method that enabled increased resolution of search as learning progressed, excessively forces convergence; at each learning stage the action set is shrunk around the successful

action of the previous stage. By forcing a convergence the automaton is unlikely to recover if a misguided 'successful' action selection is made in an early stage of learning, leading to an increased likelihood of local optima location It is similarly unlikely to recover if there is a shift in environment response during learning

The CARLA has been developed as motivated by the above points. Its concept arises from replacing the discrete action set of traditional learning automata formulations with a continuous action region. Instead of quantising an action space into a discrete action set and then assigning a discrete probability distribution to the set, a continuous probability distribution function can be described across the region that gives a continuous action set with an infinite choice of possible actions. Point application of reinforcement in an infinite action set is not viable so a reasonable assumption is made, that actions in the immediate vicinity of a tested action will return similar results, thus allowing a 'spread' of reinforcement to be applied around an action subject to its performance.

The continuous probability distribution function is defined, for CARLA, by recording the function magnitude at a finite set of points across the action space and then assuming linear interpolation between those points in subsequent computation. An algorithm has been defined to maintain a high function definition resolution around areas of high probability density. In effect the function points are spaced to give an equal probability between them. A Gaussian distribution function is applied as the continuous reinforcement function with its shape defined by two parameters; g_h represents a learning rate parameter, g_w defines the width of the reward function for generalisation of reinforcement around the tested action.

The CARLA has been defined for a single dimension action space only, because of the problems of representing an N-dimension probability distribution function in the general case. This is hardly of any consequence, however, as CARLA are shown to co-operate successfully in interconnected formation on higher dimension tasks.

The formulation of CARLA now gives complete coverage of an action space throughout a learning period. As repeated reinforcement occurs in some areas, so the probability of action selection in those areas grows to the natural detriment of surrounding areas. However, there always remains some non-zero probability of selection of any action This gives the CARLA the opportunity to reassess and adapt its response in light of any non-stationary environmental response.

Simulation studies on basic learning tasks demonstrate these beneficial properties of CARLA over discrete learning automata. One task was posed to test the ability of

automata to distinguish the global optimum from a very similar local optimum. Where the discrete automata can at best manage around a 75% success rate, the CARLA almost always locates the global optimum Observation of the convergence measure on this task shows the CARLA assesses both optima initially. At some point the CARLA chooses in favour of one optima alone and is seen to accelerate its convergence towards that. To make the CARLA fail its learning rate, g_h, had to be increased considerably, so forcing the CARLA to rush its decision.

CARLA was also analysed on a non-stationary environment that abruptly changes its response after 1000 iterations. Discrete automata were unable to handle this case at all. CARLA, in maintaining its complete action space, can respond to the change and relocate the new optimum, although it was seen to take a considerable time to 'unlearn' the original optimum.

Returning to the vehicle suspension application, CARLA was again seen to outperform discrete automata. It was noticed in these multi-automata tasks that the individual CARLA learn at differing rates; the more important a parameter is to producing a 'successful' action, so the faster the related automaton is able to distinguish this and learn a precise value.

To test the CARLA further a multi-goal task was devised from formulation of a vehicle roll control strategy. The strategy assumes semi-active suspension as fitted to the test vehicle and the simulation study then forms a feasibility study prior to a hardware trial Two teams of interconnected CARLA with independent aims were required to co-operate to optimise the roll control strategy One aim of the strategy is to maintain required roll moments via smooth control application. The CARLA actually found this to be unachievable yet returned a solution of 'bang-bang' control which significantly improves upon the roll control afforded by a passive suspension system. This also highlights the capability of a CARLA based study to provide information and insight to the human investigators.

Recommendations for Further Research

This study has been very much an empirical investigation into the possibilities afforded by applying learning automata to parameter optimisation tasks in a highly complex stochastic environment. The investigation has led to the formulation of CARLA as a new form of learning automata. The properties of CARLA have only been investigated here with experimental studies. It has been shown that CARLA exhibits many beneficial properties in practice and is a promising optimisation

technique that requires further investigation. In particular it should be compared with other optimisation techniques to properly assess its potential

In investigating the CARLA itself it is likely that the convergence properties may be better defined via an analytical study Here the convergence measure, C_m, has shown the convergence history of an automaton and it is apparent that a CARLA can opt for a strong region of the action space to the exclusion of all other regions. Prior to reaching this state an automaton may have a better view of the overall action space whilst still locating the global optimum. After opting for a single region the CARLA is able to recover if the environment response changes, but there is an increased period of 'unlearning' required for the CARLA to spread its view before any effective relearning occurs. How might a balance of CARLA convergence be better achieved?

The reinforcement scheme for CARLA is of reward-inaction form with a Gaussian distribution used as the reinforcement function, defined with the parameters g_w and g_h. Values for these parameters have been set at constant yet fairly arbitrary values throughout the various applications of CARLA in this report. Learning appears relatively insensitive to these values, but further research is required to ascertain how they affect learning performance. It is possible that these parameters could be varied in some manner during learning, and could help control the convergent nature of the CARLA described above.

The CARLA requires the user to define the action space prior to learning; the user needs enough knowledge of the action-environment interaction to be able to set a useful working range for the CARLA For complex tasks this may not be the case. For example, in the roll control study, the initial range of one parameter required three iterations to enlarge it to a range with which the CARLA did not simply respond by learning a value at the limit of the defined action space. One possible extension to the CARLA methodology to tackle this problem has been suggested in Chapter 10. Implementing an adaptive action space may enable CARLA to search further afield if necessary. The preliminary study demonstrates such a technique, varying its action set to explore in a non-stationary environment However, the technique as implemented is particularly sensitive to its defining parameters, the learning task, and initial conditions of the action space.

Throughout this study the CARLA has been applied to learn one overall action for all situations. It may be worthwhile to consider how the CARLA could be included in an associative learning setting, enabling different actions to be learnt dependent upon various states of the environment For instance, to optimise ride performance it is

clear that a vehicle will benefit from a different control regime for a rough undulating road in comparison with a smoother one A tentative step towards this has been suggested by attempting to apply CARLA to learn a non-linear controller in Chapter 10

From the vehicle suspension perspective, ride and roll control has been attempted in independent studies It has clearly been demonstrated that CARLA could be applied on-line to reduce the time spent in optimising a production suspension system, especially as no accurate system modelling is required by the technique. Further studies are now required to investigate the feasibility of learning a more complete suspension control system combining ride and roll control strategies, together with other suspension control strategies, to learn the complete ride and handling characteristics required of a modern production vehicle.

References

Atkinson, R.C. Bower, G.H. and Crothers, E.J., 1965
An Introduction to Mathematical Learning Theory
Wiley, NY, 1965

Baba, N., 1984
New Topics in Learning Automata Theory and Applications Lecture Notes in Control and Information Sciences
Springer-Verlag, NY, 1984

Barto, A.G. and Anandan, P., 1985
Pattern Recognizing Stochastic Learning Automata
IEEE Trans. Syst. Man. Cybern., vol 15, no. 3, May/Jun. 1985

Barto, A.G. Sutton, R.S. and Anderson, C.W., 1983
Neuronlike Adaptive Elements that can Solve Difficult Learning Control Problems
IEEE Trans. Syst. Man Cybern., vol. 13, no. 5, pp834-846, Sept/Oct 1983

Best, M.C., 1995
On the Modelling Requirements for the Practical Implementation of Advanced Vehicle Suspension Control
Ph D Dissertation, Dept. AAE&TS, Loughborough University

Best, M.C. Gordon, T.J. Crawford, I.L. and Hand, P., 1994
Real-time Estimation of Vehicle Suspension Characteristics using Kalman Filtering
Proc FISITA Congress (paper 945063), Beijing, China, Oct. 1994

Bush, R.R. and Mosteller, F., 1958
Stochastic Models for Learning
Wiley, NY, 1958

Chandrasekaran, B. and Shen, D.W.C., 1968
On Expediency and Convergence in Variable-Structure Automata
IEEE Trans. Syst Sci Cybern., vol. 4, no. 1, Mar. 1968

Chidambaram, M., 1994
Control of a Non-Minimum Phase, Non-Linear System by Learning Automata
Hungarian J. Industrial Chemistry, vol. 22, no. 4, pp261-265, 1994

Franklin, G.F. Powell, J.D. and Workman, M.L., 1990
Digital Control of Dynamic Systems
Addison-Wesley, NY, 1990

Frost, G.P. Gordon, T.J. Howell, M.N. and Wu, Q.H., 1996
Moderated Reinforcement Learning of Active and Semi-active Vehicle Suspension Control Laws
Proc I Mech.E. Part I, vol 210, pp249-257, 1996

Frost, G.P. Gordon, T.J. and Wu, Q.H., 1994
The Application of Learning Automata to Advanced Vehicle Suspension Control
Proc of IUTAM Symposium, *The Active Control of Vibration*, Bath, pp153-159, 1994

Frost, G.P. Howell, M.N. Gordon, T.J. and Wu, Q.H., 1996
Dynamic Vehicle Roll Control using Reinforcement Learning
Proc. UKACC Int. Conf. Control '96, vol 2, pp1107-1112, 1996

Garcia, H.E. Ray, A. and Edwards, R.M., 1991
Reconfigurable Control of Power Plants using Learning Automata
IEEE Cont Syst. Magazine, pp85-92, Jan 1991

Gillespie, T.D., 1992
Fundamentals of Vehicle Dynamics
SAE Inc, 1992

Gordon, T.J. and Best M.C., 1994
Dynamic Optimization of Nonlinear Semi-active Suspension Controllers
Proc. IEE Control '94 Conf., pp332 - 337, 1994

Gordon, T.J. Marsh, C. and Wu, Q.H., 1993
Stochastic Optimal Control of Active Vehicle Suspensions using Learning Automata
Proc. I Mech E. Part I, vol. 207, pp143-152, 1993

Howell, M.N. Frost, G.P. Gordon, T.J. and Wu, Q.H., 1997
The Application of Interconnected Stochastic Learning Automata to Controller Design for Semi-Active Automobile Suspensions
Int. Symp. on Advanced Vehicle Control, AVEC-96, 1996

Howell, M.N. Frost, G.P. Gordon, T.J. and Wu, Q.H., 1997
Continuous Action Reinforcement Learning Applied Vehicle Suspension Control
J. of Mechatronics, vol 7, no. 3, pp263--276, 1997

Howell, M.N. Frost, G.P. Gordon, T.J. and Wu, Q.H., 1997
Real-time Learning of Vehicle Suspension Control Laws
Workshop on Modelling in Reinforcement Learning', Fourteenth Int Conf. on Machine Learning, ICML-97, 1997

Iosifescu, M. and Theodorescu, R., 1969
Random Processes and Learning
Springer, NY, 1969

Jarvis, R.A., 1970
Adaptive Global Search in a Time-Variant Environment
IEEE Trans Syst. Sci. Cybern., vol. 6, no. 3, Jul. 1970

Karnopp, D.C., 1983
Active Damping in Road Vehicle Suspension Systems
Veh. Syst. Dynamics, vol. 12, pp317-330, 1983

Karnopp, D.C., 1990
Design Principles for Vibration Control Systems using Semi-Active Dampers
Journal of Dynamic Systems, Measurement, and Control, vol. 112, pp448-455, Sept 1990

Karnopp, D.C. Crosby, M.J. and Harwood, R.A., 1974
Vibration Control using Semi-active Force Generators
Trans. ASME J. Eng. for Ind , vol. 92, no. 2, pp 619-626, 1974

Kohonen. T., 1982
Self-organized Formation of Topologically Correct Feature Maps
Biological Cybernetics, vol. 69, pp43-59, 1982

Kokar, M.M. and Reveliotis, S.A., 1993
Reinforcement Learning: Architectures and Algorithms
Int. J. Intelligent Systems, vol. 8, no. 8, pp875-894, 1993

Kwakernaak, H. and Sivan, R., 1972
Linear Optimal Control Systems
Wiley, NY, 1972

Lakshmivarahan, S., 1981
Learning Algorithms Theory and Applications
Springer-Verlag NY, 1981

Lakshmivarahan, S. and Thathachar, M.A.L., 1973
Absolutely Expedient Learning Algorithms for Stochastic Automata
IEEE Trans. Syst. Man Cybern., vol. 3, pp281-286, May 1973

McLaren, R.W., 1966
A Stochastic Automaton Model for the Synthesis of Learning Systems
IEEE Trans. Syst. Sci. Cybern., vol. 2, no. 2, pp109-114, Dec. 1966

McMurtry, G.J. and Fu, K.S., 1966
A Variable Structure Automaton used as a Multi-modal Search Technique
IEEE Trans. Aut. Cont , vol 11, no. 3, pp379-387, July 1966

Marsh, C. Gordon, T.J. and Wu, Q.H., 1995
The Application of Learning Automata to Controller Design in Slow-Active Automobile Suspensions
Veh Syst Dynamics, vol. 24, pp597-616, 1995

Mitchell, B.T. and Kountanis, D.I., 1984
A Reorganisation Scheme for a Hierarchical System of Learning Automata
IEEE Trans. Syst. Man Cybern., vol. 14, no. 2, pp328-334, Mar/Apr 1984

Najim, K., 1991
Modelling and Self-Adjusting Control of an Absorption Column
Int. J. Adaptive Control and Signal Processing, vol. 5, pp335-345, 1991

Najim, K. Pibouleau, L. and LeLann, M.V., 1990
Optimisation Technique based on Learning Automata
J. Optimisation Theory and Applications, vol. 64, no. 2, pp331-347, Feb 1990

Najim, K. and Poznyak, A.S., 1994
Learning Automata: Theory and Applications
Pergamon, 1994

Narendra, K.S. and Lakshmivarahan, S., 1977
Learning Automata - A Critique
J Cybern. and Info. Sci , vol 1, pp53-65, 1977

Narendra, K.S. and Mars, P., 1983
The Use of Learning Algorithms in Telephone Traffic Routing - A Methodology
Automatica, vol 19, no 5, pp495-502, 1983

Narendra, K.S. and Thathachar, M.A.L., 1974
Learning Automata - A Survey
IEEE Trans. Syst Man Cybern , vol. 4, no. 4, pp323-334, July 1974

Narendra, K.S. and Thathachar, M.A.L., 1989
Learning Automata: an Introduction
Prentice-Hall, London, 1989

Narendra, K.S. and Viswanathan, R., 1972
A Two-Level System of Stochastic Automata for Periodic Random Environments
IEEE Trans. Syst. Man Cybern , vol 2, pp285-289, Apr. 1972

Norman, M.F., 1972
Markov Processes and Learning Models
Academic Press, NY, 1972

Phansalkar, V.V. and Thathachar, M.A.L., 1996
Learning Automata in Feedforward Connectionist Systems
Int. J. Syst. Sci, vol. 27, no. 2, pp145-150, 1996

Press, W.H. Teukolsky, S.A. Vetterling, W.T. and Flannery, B.P., 1988
Numerical Recipes in C· The Art of Scientific Computing
Cambridge University Press, 1988

Robson, J.D., 1979
Road Surface Description and Vehicle Response
Int. J. of Vehicle Design, vol. 1, pp25-35, 1979

Santharam, G. Sastry, P.S. and Thathachar, M.A.L., 1994
Continuous Action-set Learning Automata for Stochastic Optimisation
Journal of the Franklin Institute, vol. 331B, no. 5, pp607-628, 1994

Shapiro, I.J., and Narendra, K.S., 1969
Use of Stochastic Automata for Parameter Self-Optimization with Multi-Modal Performance Criteria
IEEE Trans. Syst Sci. Cybern., vol. 5, no 4, pp352-360, Oct. 1969

Sharp, R.S. and Crolla, D.A., 1987
Road Vehicle Suspension System Design - A Review
Veh. Syst. Dynamics, vol. 16, pp164-192, 1987

Sharp, R.S. and Hassan, S.A., 1984
The Fundamentals of Passive Automotive Suspension System Design
Society of Environmental Engineers Conf. 'Dynamics in Automotive Engineering', pp 104-115, 1984

Sharp, R.S. and Hassan, S.A., 1986
The Relative Performance Capabilities of Passive, Active and Semi-Active Car Suspension Systems
Proc I.Mech E., vol. 200, no. D3, pp219-228, 1986

Tang, C.K.K. and Mars, P., 1991
Stochastic Learning Automata and Adaptive IIR Filters
IEE Proc.-F, vol. 138, no 4, Aug. 1991

Tang, C.K.K. and Mars, P., 1993
Games of Stochastic Learning Automata and Adaptive Signal Processing
IEEE Trans. Syst. Man Cybern., vol. 23, no 3, pp851-856, May/Jun. 1993

Thathachar, M.A.L. and Ramakrishnan, K.R., 1981
A Hierarchical System of Learning Automata
IEEE Trans. Syst. Man Cybern., vol. 11, no. 3, Mar. 1981

Thathachar, M.A.L. and Ramakrishnan, K.R., 1982
On-line Optimisation with a Team of Learning Automata
IFAC Theory and Application of Digital Control, pp297-302, 1982

Thompson, A.G., 1984
Optimal and Sub-optimal Linear Active Suspensions for Road Vehicles
Vehicle System Dynamics, vol. 13, pp61-72, 1984

Tseng, H.E. and Hedrick, J.K., 1994
Semi-Active Control Laws - Optimal and Sub-Optimal
Vehicle System Dynamics, vol. 23, pp545-569, 1994

Tsetlin, M.L., 1961
On the Behaviour of Finite Automata in Random Media
Automat. Telemekh., vol. 22, pp1345-1354, Oct. 1961

Tsetlin, M.L., 1973
Automaton Theory and Modelling of Biological Systems
Academic Press, NY, 1973

Unsal, C. Bay, J.S. and Kachroo, P., 1995
Intelligent Control of Vehicles: Preliminary Results on the Application of Learning Automata Techniques to Automated Highway System
IEEE Proc Int. Cont. on Tools with AI, pp216-223, 1995

Valenzuela, J. Najim, K. Villar, R. and Bourassa, M., 1993
Learning Control of an Autogeneous Grinding Circuit
Int. J. of Mineral Processing, vol. 40, pp45-56, 1993

Varshavskii, V.I. and Vorontsova, I.P., 1963
On the Behaviour of Stochastic Automata with Variable Structure
Automat. Telemekh, vol 24, pp353-360, 1963

Viswanathan, R. and Narendra, K.S., 1972
Comparison of Expedient and Optimal Reinforcement Schemes for Learning Systems
J. of Cybernetics, vol. 2, no. 1, pp21-37, 1972

Viswanathan, R. and Narendra, K.S., 1973
Stochastic Automata Models with Applications to Learning Systems
IEEE Trans. Syst. Man Cybern., vol. 3, pp107-111, Jan. 1973

Wilson, D.A. Sharp, R.S. and Hassan, S.A., 1986
Application of Linear Optimal Control Theory to the Design of Automobile Suspensions
Vehicle System Dynamics, vol. 15, pp103-118, 1986

Wu, Q.H., 1993
Learning Co-ordinated Control of Synchronous Machines in Multi-Machine Power Systems
Proc IEEE Syst. Man Cybern. Conf., vol. 3, pp728-733, France, 1993

Wu, Q.H. and Pugh, A.C., 1993
Reinforcement Learning Control of Unknown Dynamic Systems
Proc. IEE Part D, vol. 140, no 5, pp313-322, 1993

Appendix A - Full Vehicle Model

A rigid body vehicle fitted with semi-active suspension actuation is simulated in Chapter 9. An outline of the model applied was given in Chapter 2. The full derivation of the state-space model is detailed here

Figure A 1 defines some layout nomenclature for the vehicle body. Suspension actuation is applied at each corner of the body rectangle, and each corner is numbered for later reference

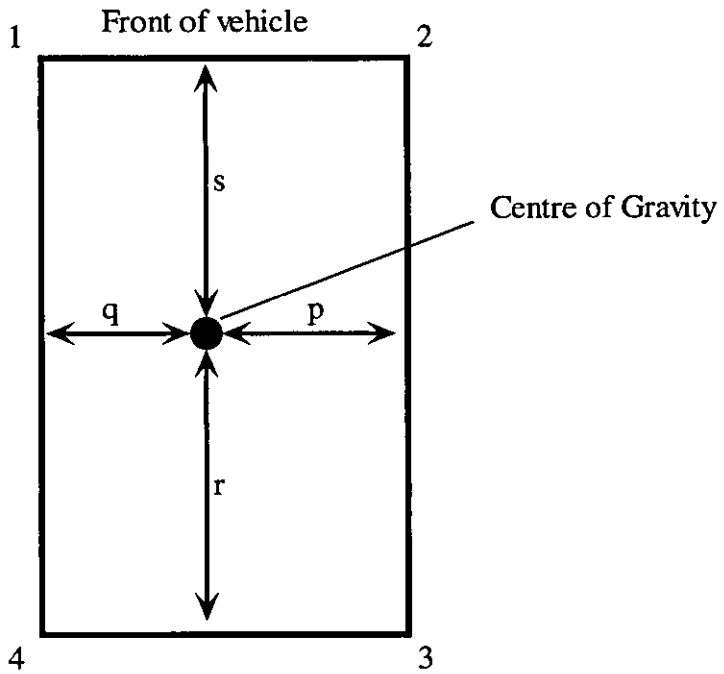


Figure A.1 - Plan view of vehicle

The dynamics of the unsprung mass at each corner, defined by equation (2.18), lead to the following state equations for vertical wheel displacement (x_1 to x_4) and velocity (x_5 to x_8)

$$
\begin{aligned}
\dot{x}_1 &= x_5 \\
\dot{x}_2 &= x_6 \\
x_3 &= x_7 \\
\dot{x}_4 &= x_8 \\
x_5 &= \left(F_{t,1} - F_{s,1}\right)/m_{w,1} \\
x_6 &= \left(F_{t\,2} - F_{s,2}\right)/m_{w\,2} \\
\dot{x}_7 &= \left(F_{t\,3} - F_{s,3}\right)/m_{w,3} \\
\dot{x}_8 &= \left(F_{t,4} - F_{s,4}\right)/m_{w,4}
\end{aligned}
\qquad \text{(A 1)}
$$

where $F_{t,i}$ and $F_{s,i}$ are the forces induced by the tyre and suspension components respectively, and $m_{w,i}$ is the unsprung corner mass. Tyre force, F_t, is modelled as a simple linear spring of stiffness k_t, so

$$F_{t,i} = k_t.\left(h_{\text{road},i} - x_i\right) \tag{A.2}$$

where $h_{\text{road},i}$ is the vertical displacement of the road at corner i

The dynamics of the sprung mass at each corner, defined by the equations of (2.17), lead to the following state equations for vehicle body displacement (x_9), roll angle (x_{10}) and pitch angle (x_{11})

$$\begin{aligned}
\dot{x}_9 &= x_{12} \\
\dot{x}_{10} &= x_{13} \\
\dot{x}_{11} &= x_{14} \\
\dot{x}_{12} &= \left(F_{s,1} + F_{s,2} + F_{s,3} + F_{s\,4}\right)/m_b \\
\dot{x}_{13} &= \left(p.\left(F_{s,2} + F_{s,3}\right) - q.\left(F_{s,1} + F_{s,4}\right)\right)/I_r \\
\dot{x}_{14} &= \left(r.\left(F_{s,3} + F_{s,4}\right) - s.\left(F_{s,1} + F_{s,2}\right)\right)/I_p
\end{aligned} \tag{A.3}$$

The suspension force, $F_{s,i}$, is calculated based on application of a semi-active actuator in parallel with a linear spring of stiffness $k_{s,i}$ at each corner

$$F_{s,i} = k_{s,i}.\left(x_i - b_i\right) + F_{d,i} \tag{A.4}$$

where b_i is the vertical corner displacement of the vehicle body, and $F_{d,i}$ is the actuator damping force.

Derivation of the state equations describing the transient operation of a realistic single actuator was given in Section 2.1.3. Generalising these equations to apply an actuator at each corner of the vehicle gives the additional states

$$
\begin{aligned}
x_{15} &= \left\{\frac{\partial \mu}{\partial x_{15}}\right\}^{-1}\left\{k_{b,1}\left(x_5 - b_1 - x_{15}\right) - x_{23}\frac{\partial \mu}{\partial x_{19}}\right\} \\
\dot{x}_{16} &= \left\{\frac{\partial \mu}{\partial x_{16}}\right\}^{-1}\left\{k_{b,2}\left(x_6 - b_2 - x_{16}\right) - x_{24}\frac{\partial \mu}{\partial x_{20}}\right\} \\
\dot{x}_{17} &= \left\{\frac{\partial \mu}{\partial x_{17}}\right\}^{-1}\left\{k_{b,3}\left(x_7 - \dot{b}_3 - x_{17}\right) - x_{25}\frac{\partial \mu}{\partial x_{21}}\right\} \\
\dot{x}_{18} &= \left\{\frac{\partial \mu}{\partial x_{18}}\right\}^{-1}\left\{k_{b\,4}\left(x_8 - b_4 - x_{18}\right) - x_{26}\frac{\partial \mu}{\partial x_{22}}\right\} \\
\dot{x}_{19} &= x_{23} \\
x_{20} &= x_{24} \\
x_{21} &= x_{25} \\
x_{22} &= x_{26} \\
\dot{x}_{23} &= -2\zeta_1\omega_{n\,1}x_{23} + \omega_{n,1}^2\left(u(t) - x_{19}\right) \\
\dot{x}_{24} &= -2\zeta_2\omega_{n,2}x_{24} + \omega_{n,2}^2\left(u(t) - x_{20}\right) \\
\dot{x}_{25} &= -2\zeta_3\omega_{n,3}x_{25} + \omega_{n,3}^2\left(u(t) - x_{21}\right) \\
x_{26} &= -2\zeta_4\omega_{n,4}x_{26} + \omega_{n,4}^2\left(u(t) - x_{22}\right)
\end{aligned}
\qquad \text{(A.5)}
$$

The vehicle corner height displacements, b_i, and velocities, $\dot{b}_i$, are found from superposition of the body bounce, roll and pitch characteristics. The displacements and velocities induced by roll motion of the vehicle body are defined with reference to Figure A.2. Similarly, pitch induced displacements and velocities are defined with reference to Figure A.3.

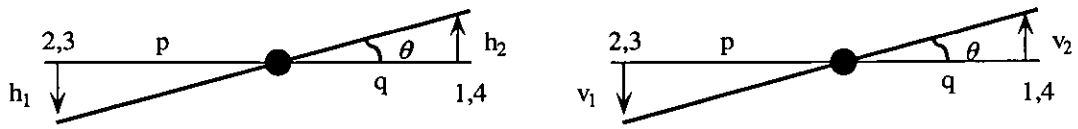


Figure A.2 - Roll induced corner displacements and velocities

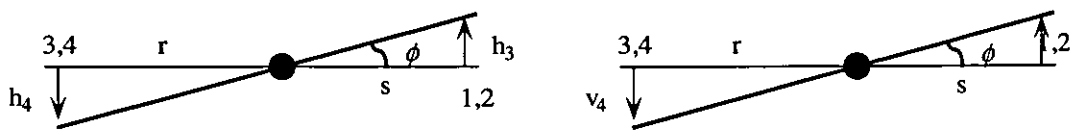


Figure A.3 - Pitch induced corner displacements and velocities

The height displacements are then

$$
\begin{aligned}
h_1 &= p.\tan\theta \\
h_2 &= -q.\tan\theta \\
h_3 &= -s.\tan\phi \\
h_4 &= r.\tan\phi
\end{aligned}
\qquad \text{(A.6)}
$$

and velocities are

$$
\begin{aligned}
v_1 &= p.\dot{\theta} \\
v_2 &= -q.\theta \\
v_3 &= -s\,\dot{\phi} \\
v_4 &= r.\dot{\phi}
\end{aligned}
\qquad \text{(A.2)}
$$

Applying superposition to deduce vertical corner displacements gives

$$
\begin{aligned}
b_1 &= x_9 + h_2 + h_3 \\
b_2 &= x_9 + h_1 + h_3 \\
b_3 &= x_9 + h_1 + h_4 \\
b_4 &= x_9 + h_2 + h_4
\end{aligned}
\qquad \text{(A 3)}
$$

and similarly for vertical corner velocities

$$
\begin{aligned}
\dot{b}_1 &= x_{12} + v_2 + v_3 \\
\dot{b}_2 &= x_{12} + v_1 + v_3 \\
b_3 &= x_{12} + v_1 + v_4 \\
b_4 &= x_{12} + v_2 + v_4
\end{aligned}
\qquad \text{(A 4)}
$$

Appendix B - Probability Distribution Representation Refinement

The CARLA represents its internal state via a piece-wise linear approximation of a probability distribution. In implementation on a microprocessor, a refinement algorithm is applied to efficiently use available memory resources whilst maintaining resolution of curve representation in regions of high probability density. This appendix explains the operation of this algorithm by way of an example.

The refinement algorithm is based around maintaining each successive pair of curve vertices at a distance apart such that the probability area swept out between each pair of vertices is roughly constant across the whole representation. For instance, in Figure B.1, the probability distribution is heavily skewed towards low values of α. Representing this curve with just 6 vertices, there are 5 segments to the probability distribution, each of an area close to 0.2. The reason for each area not being exactly 0.2 will be explained below.

Suppose there is now a large reinforcement to a high action value. Applying the reinforcement alone, the resultant normalised curve may be of the form seen in Figure B.1(b). As indicated on this plot, the areas between curve vertices are now far from equal.

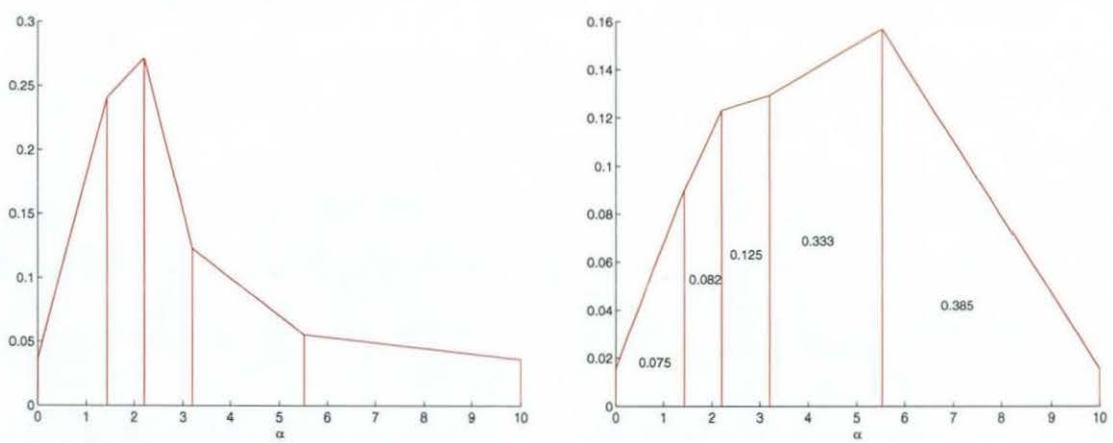


Figure B.1 - Probability distribution before (left) and immediately after (right) a reinforcement application

The steps taken to move the inner vertices are shown in Figure B.2. It is known that the ideal area between vertices should be 1/(number of vertices - 1), 0.2 in this example. Starting from $\alpha_{\min}$, an action value is calculated whereby the integral of the old curve between $\alpha_{\min}$ and this point is 0.2. The first inner vertex is placed here, with it's height taken from the old curve - see Figure B.2(a). This process is repeated, integrating along the old curve in steps of 0.2, placing new vertices at these boundary points until the whole of the old curve is traversed - see Figure B.2(b),(c),(d).

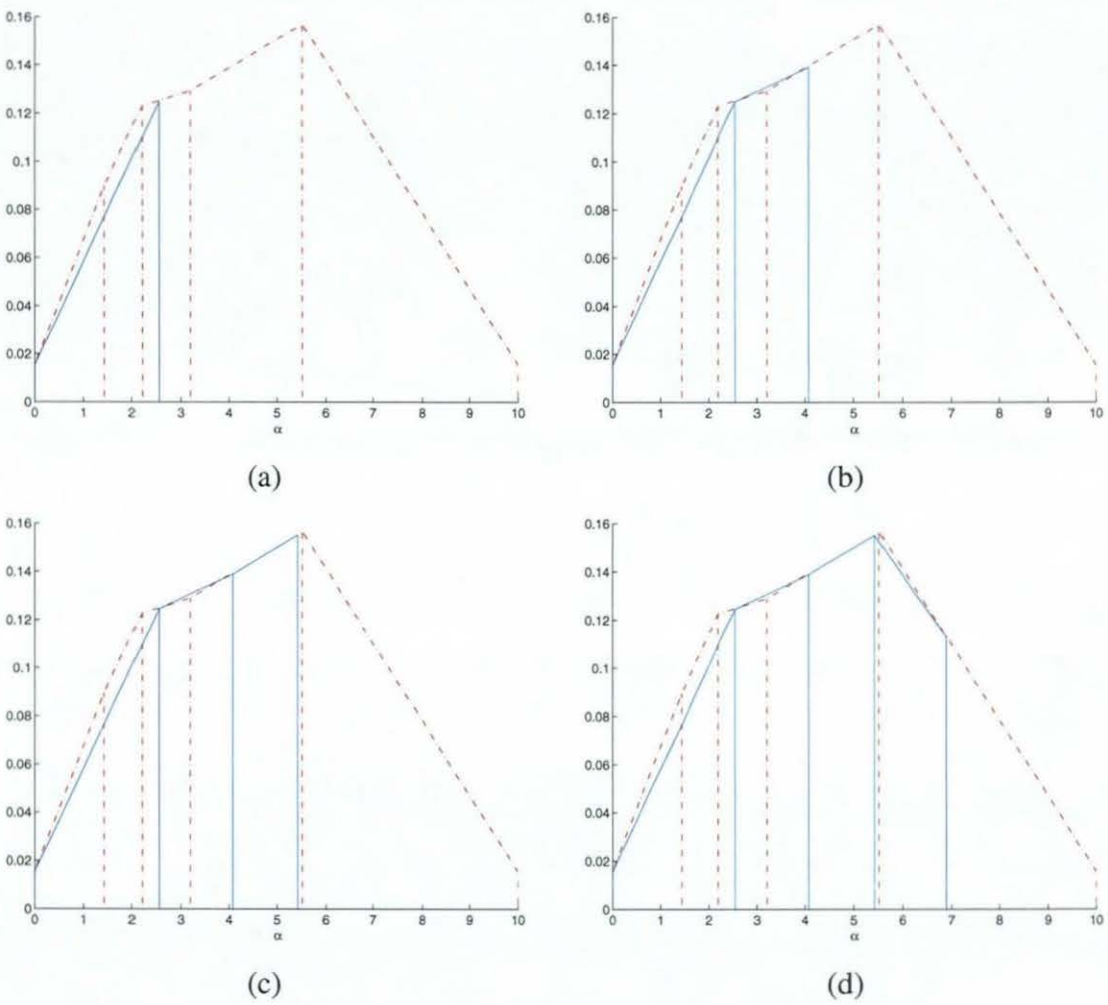


Figure B.2 - Step-by-step realignment of curve vertices

Note that, by taking vertex height from the old curve, the area between each pair of vertices on the new curve description will not necessarily be the target value. Normalising the new curve to a total area of 1, the areas between each vertex on the new curve are shown in Figure B.3. It is not a major concern that the areas are not all equal, only that the total is consistently 1 to meet the total probability constraint. It should be noted that attempts by the author to formulate an algorithm that maintain equal areas between vertices all led to a numerical unstable solution!

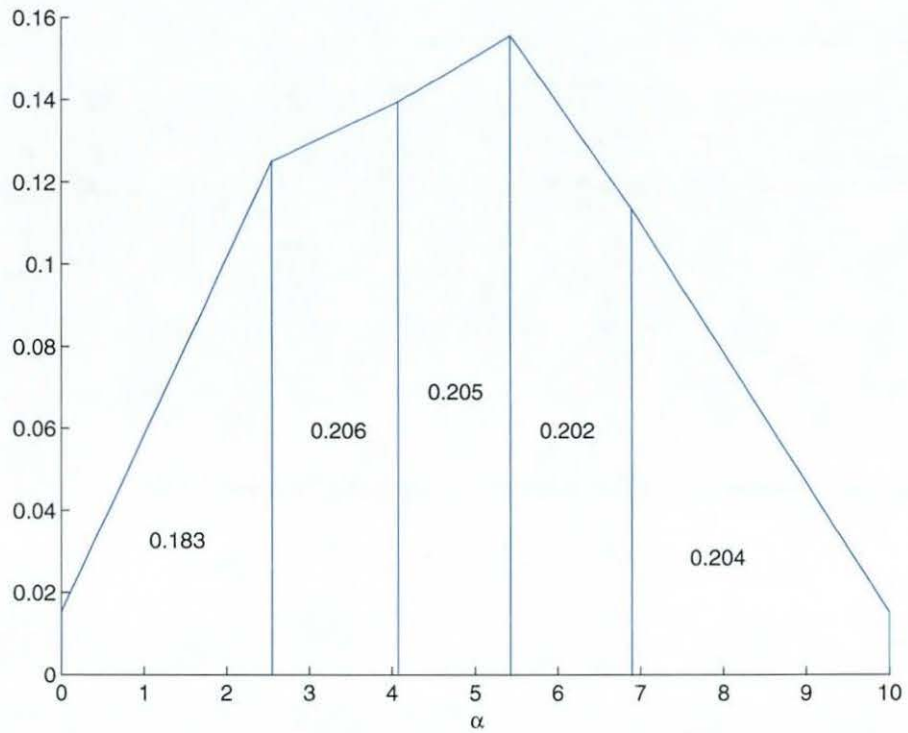


Figure B.3 - Redefined curve representation