The application of neural networks in active suspension

Andrew FAIRGRIEVE

2003

A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy at Loughborough University.

Acknowledgement

This thesis is based on work carried out at Loughborough University in the Department of Aeronautical and Automotive Engineering, with funding assistance from the Engineering and Physical Sciences Research Council, EPSRC. The Author would like to thank them for their much needed financial support during the project.

The research was performed under the supervision of Professor Tim J Gordon, and I would like to extend my thanks to him for his support and guidance. Particularly during his time as Head of Department and the Departmental move, when his time was at a premium.

I would also like to thank my office colleagues Ayao Komatsu, Mark N Howell, Ashley J Kells, and Simon Tuplin.

Finally, I would like to thank Tim G Reynolds for selflessly proof reading this thesis at considerable expense to his social life.

Summary

This thesis considers the application of neural networks to automotive suspension systems. In particular their ability to learn non-linear feedback control relationships. The speed of processing, once trained, means that neural networks open up new opportunities and allow increased complexity in the control strategies employed.

The suitability of neural networks for this task is demonstrated here using multi-layer perceptron, (MLP) feed forward neural networks applied to a quarter vehicle simulation model. Initially neural networks are trained from a training data set created using a non-linear optimal control strategy, the complexity of which prohibits its direct use. They are shown to be successful in learning the relationship between the current system states and the optimal control.

Integrated control strategies based around Hamiltonian minimisation and costate maps have significant benefits to offer. Their use simplifies the development process and gives increased flexibility during redevelopment, and a performance level comparable with global control approaches. However, for non-linear optimal control it is difficult to calculate the costate values. In the second part of this thesis neural networks are trained directly and indirectly to learn the state - costate relationship.

It is shown that a network of sufficient size is capable of learning the state - costate relationship indirectly from a data set. Creation of the training data set requires the use of a complex and time consuming numerical solution of a two-point boundary problem. For this reason the remainder of the thesis seeks to train the neural network directly. This is found to be a faster approach when successful but is less robust and can find incorrect solutions or local minima. A number of approaches are applied that seek to increase the reliability of the training technique. While the likelihood of success is much improved by these techniques, further work is still required. The work so far, proves that the principle of the training process is viable. With further investigation, it maybe possible to create a robust, viable alternative to the indirect learning approach.

Contents

Nomenclature

Variable	Description	Unit
x	generic function input	
y	generic function output	
$y = f(x)$	generic function	
State Space, Quarter Vehicle Parameters and Related Cost Functions		
g	Gravitational constant	m/s^2
t	Time	s
A	System matrix	
B	Input matrix	
G	Disturbance matrix	
C	State space output matrix	
D	State space output matrix	
x_1	Relative tyre deflection	m
x_2	Relative suspension deflection	m
x_3	Velocity of the unsprung mass	m/s
x_4	Velocity of sprung mass	m/s
$\mathbf{x}$	State vector	
x_{1Lo}	Tyre lift off point.	m
u	Control action	
u_{NN}	Neural Network Calculated Control Force	N
u_{LQR}	LQR Calculated Control Force	N
F_t	Dynamic Tyre force	N
k_t	Tyre spring stiffness	N/m
C_t	Tyre Damping	N/m/s [0]
F_s	Dynamic Suspension force	N
C_p	Passive Damping Rate	N/m/s
k_p	Passive Spring Stiffness	N/m
K_i	Gain Matrix Value	
K_{LQR}	LQR gain matrix	
m_b	Sprung mass	Kg
m_w	Unsprung mass	Kg
J	Dynamic cost	
Q	State cost matrix	
R	Control cost matrix	
S	Riccati matrix	
L	Dynamic cost	
α	Tyre Deflection Cost Coefficient	
β	Suspension Deflection Cost Coefficient	
n_i	Cost Function Powers	

p	Lagrange Multiple / Costates	
H	Hamiltonian Equation	
h_i	H_{norm} cost element	
g_{im}	H_{norm} Gradient Element	
G	H_{norm} Gradient	
ε	Small Number	
Z	Measure of Symmetry	
$q(x(t))$	General Equation	
r	Random Length (Point Generation)	$0 < r < 1$
θ, Φ, λ	Random Angles (Point Generation)	
Kalman Filters		
$v(t)$	Measurement Noise	
$\wp_v$	Sensor Error ($\times 10^{v}$)	
y_v	Measurement Output	
$\hat{x}$	Estimated States	
L	Kalman Filter Gain Matrix	
Road Models		
v_r	Vertical road velocity	m/s
Z	Vertical Displacement	
U	Vehicle Forward Velocity	
G	Road Roughness	
w	unit amplitude white noise	
f_o	Low Frequency Cut-off	
A_1, A_2	Stochastic Road Parameters	
γ	Integrated Suspension Deflection Cost Parameter	
λ	Low Frequency Cut-off	
v_L	Local Gradient Velocity	
v_c	Constant Vertical Road Velocity	
Neural Networks		
u	Neural Network Input	
y	Neural Network Output	
w	Neural Network Weights	
$y = f(x,w)$	Generic Neural Network	
$\mathscr{B}$	Neural Network Bias (Threshold)	
$f(N)$	Neural Network Transfer Function	
N	Transfer Function Input	
k	Layer Number	
L	Total Number of Network Layers	
j	Neuron Number	

E	Neural Network Error	
ϕ	Neural Network Error Derivative	
r	Reference Data	
m	Number of Training Points	
d	Time Delay	
e	Error	
Optimisation Functions		
λ	Levenberg Marquardt Step Size	
a	Step Size (Taylor Series)	
H_{LM}	Hessian Matrix (Levenberg Marquardt)	
$S(x)$	Residual Error	
E_c	Complexity Function Cost	
ψ	Regularisation Parameter	
$\mu(x)$	Weighting Function	
g	Point Gain	
γ	Cost scale	
Abbreviations		
PSF	Perfect State Feedback	
LGF	Local Gradient Feedback	
KSF	Kalman Filter State Feedback	
IFC	Integral Feedback Control	
MLP	Multi-Layer Perceptron	
NNF	Neural Network Force (Defined Page 57)	
NNC	Neural Network Costate (Defined Page 57)	
LQR	Linear Quadratic Regulation	

Chapter 1 Introduction

Chapter 1 Introduction

This thesis covers the application of Artificial Neural Networks, ANNs, in vehicle suspension control. The constrained workspace, non-linear components, kinematics and system interactions, make vehicle suspension an ideal application for non-linear control and a suitable environment to investigate neural networks for non-linear control when calculation speed is important.

In this chapter, the status of modern vehicle suspensions is considered, including evolving practical techniques, which allow significant improvements in vehicle ride and handling. A number of non-linear control techniques are discussed in the context of active suspension. Neural networks are highlighted as having a key role that warrants further investigation. The objectives and structure of the remainder of the thesis are defined at the end of the chapter.

1.1 Vehicle Suspension

The basic role of vehicle suspension is well understood - to isolate the vehicle body and passengers from road disturbances and to control the contact force at the road tyre interface, thereby stabilising and transmitting cornering, braking and traction forces through the links / strut. Both attributes are affected by the suspension kinematics, however the main focus of this thesis will be on the vertical (ride) dynamics.

The field of vehicle suspension design can be broken down into a number of distinct areas. Work has been both practical [1-3] and theoretical [4-6] in all fields. The practical implementation falls into three categories; Passive, Active and Semi-Active.

1.1.1 Passive Suspension

The majority of current production vehicles have passive suspension, as illustrated in Figure 1.1(a). This normally takes the form of a metal leaf or coil spring (torsion bars, rubber and pressurised gas have also been used) and a fluid damper (friction dampers have been used in the past). The spring elements isolate the vehicle body and maintain the tyre contact force. The dampers dissipate energy from the system, the viscous action of the oil passing through the small orifices changing mechanical energy to heat that is dissipated to the surrounding atmosphere.

1.1.2 Active Suspension

Active suspension as it is known today was pioneered practically by Group Lotus [7], in both passenger vehicles and motorsport, in which it was used to great effect in their Formula 1 cars. Other companies had worked in the field previously as early as the 1950s. These systems had been relatively simple and it was Lotus that for the first time successfully implemented the more advanced electronically controlled technology that is now considered to be Active suspension. They achieved a significant improvement over equivalent passively suspended vehicles, so much so, that the technology was subsequently banned in Formula 1. Active suspension is normally considered to refer to a vehicle suspension system that incorporates some form of Electro-hydraulic actuation. This is normally supplemented by a passive system either in parallel, to maintain vehicle elevation when stopped Figure 1.1(c) (Mercedes-Benz CL500 [8]) or in series, when the active system has only limited bandwidth Figure 1.1(d). A number of examples also exist where hydraulic rotary actuators [9] are used to augment passively suspended vehicle roll control (Land Rover Discovery, Series 2 [10]).

1.1.3 Semi-Active Suspension

Because of the expense of the components involved, and the high levels of power consumption concerned, active suspension is considered by most manufacturers to be unsuitable for mass production vehicles. A more practical approach is the use of semi-active suspension Figure 1.1(b). Semi-active suspension involves the control of energy dissipation from the suspension through the modulation of the damping rates. This requires only minimal external power. It can be implemented in two forms; an adjustable valve or valves in the damper controlled electronically to vary pressure levels in the damper fluid, or through the use of electro or magneto rheological fluids for which the viscosity can be changed by varying the applied electric or magnetic field passing through it respectively.

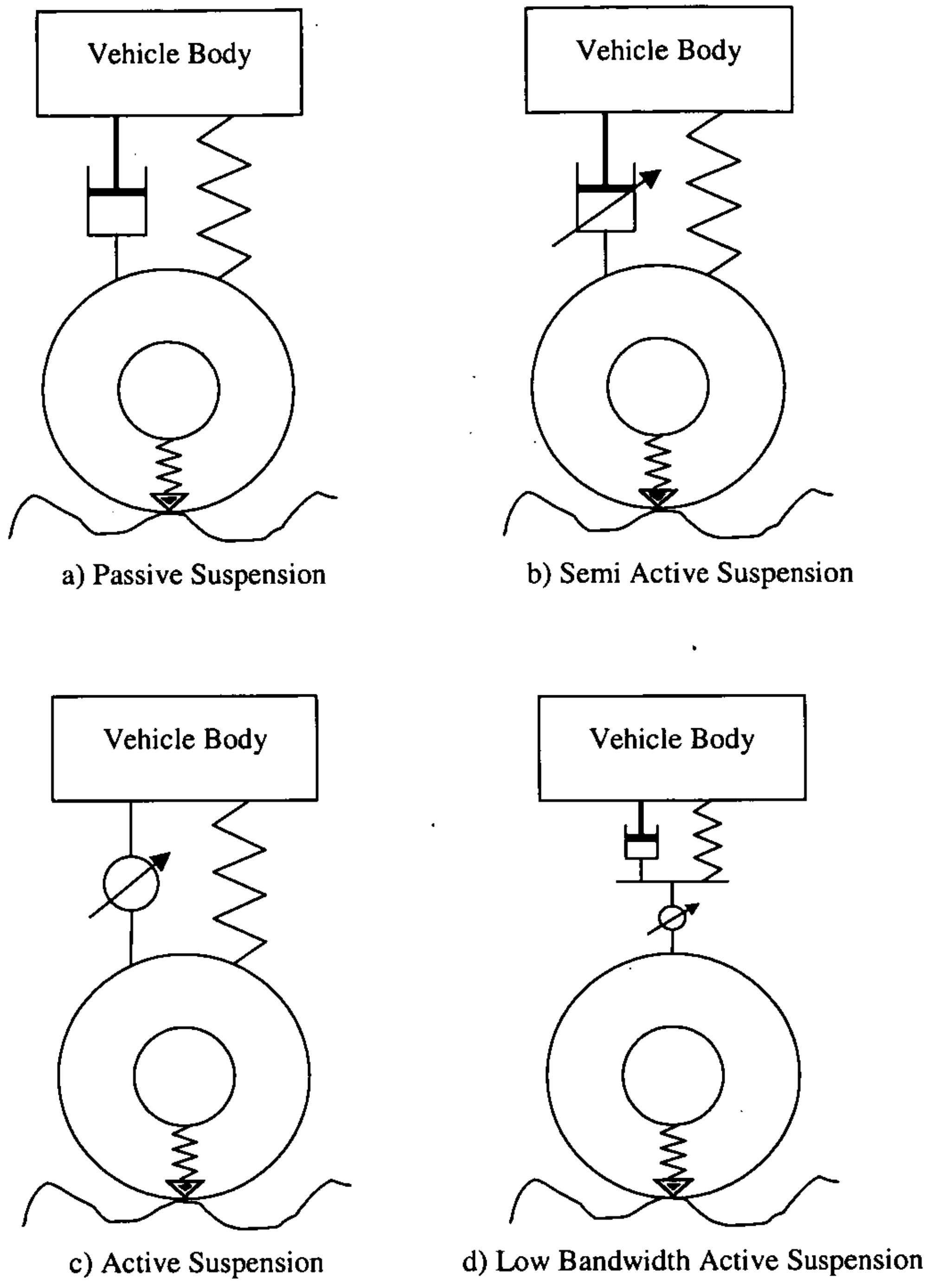


Figure 1.1- Suspension Types

1.2 Suspension Control

Both semi-active and active suspension requires some form of control, and it is to this control strategy that this thesis seeks to apply neural networks. Control algorithms for active suspension have been the subject of theoretical analysis since before its practical implementation was truly feasible. Bender [11] and Karnopp began investigations in this field in the mid-sixties. Karnopp later developed the now classical sky-hook-damping concept [12], where the vehicle body is controlled relative to a hypothetical inertial reference frame. Since then there have been many hundreds of papers on investigations in this field, most based only on theoretical analysis, with vehicle models of varying degrees of freedom and complexity [13]. The majority of work has been carried out using the quarter vehicle model [14]. This is normally a two degree of freedom model, representative of one corner of the vehicle and described in more detail in Chapter 2. More complex models exist in the form of the half car model [15], allowing pitch analysis, and the full vehicle model [16], allowing full dynamic analysis. The complexity of these models incorporate varying degrees of accuracy, depending on the features of the algorithm being developed. Active suspension has become a showcase for developing control techniques and hence there is a vast array of choices to be considered, and to which comparison can be made.

Early work centred around linear control techniques; Karnopp's sky-hook damper, Linear Quadratic Regulation (LQR), PID and other classical control techniques have been analysed and compared [17]. LQR in particular seems suited to the task of active suspension control when optimality is a key concern. However several papers have highlighted a number a weaknesses in its application [6, 18], in particular its inability to account for the constrained suspension deflection workspace and the limited range of tyre deflection. It is also incapable of accounting for non-linearities in the suspension kinematics and actuation. More recently this has been the motivation for investigation into the application of a number of non-linear control techniques: Sliding mode, Fuzzy logic, adaptive control, feedback linearization and neural networks. These are now reviewed in turn.

1.2.1 Sliding Mode

Sliding mode control is a technique used to drive the behaviour of one system to track that of another, normally a model. Therefore, its success is not just dependent on how well System *A* follows *B*, but also whether *B* is an appropriate system to follow. Given the right dynamics for the sliding surface it can be a valuable asset to the control of non-linear systems. The influence of un-modelled non-linearities on system performance can be significantly reduced.

Sliding mode control has been introduced successfully in both active [19, 20] and semi-active [21] suspension problems. Its introduction in active suspension is relatively easy although care must be taken to avoid control chatter. Several examples exist of its use in the control of a non-linear model to track a model with Linear Quadratic Regulation, (LQR). Semi-active application of sliding mode control is more difficult due to the lack of power to execute control. In order for the technique to be successful the reference model must be carefully formulated, so as to make it feasible to track using control of energy dissipation only.

1.2.2 Fuzzy logic

Fuzzy techniques have been applied successfully to both active and semi-active suspension control. Several methods exist for the development and optimisation of the fuzzy rule sets. Originally, fuzzy logic was developed as a tool to represent human system knowledge mathematically. This approach has been applied successfully by Cherry et al [22] to control semi-active suspension on a full vehicle model, using rules provided by vehicle dynamics engineers. However the resulting control system cannot be considered to be optimal and there is no guarantee of robustness when applied to active systems.

New techniques have evolved to automate, optimise and adapt the selection of the fuzzy rules set. These have also been applied to active suspension problems successfully [23]. In active applications, the fuzzy system has been used to blend between different control strategies rather than directly selecting the control action. Because the original control strategies are robust, this is not altered though the use of fuzzy logic in their selection. Fuzzy logic however, can still not be described as “optimal”, particularly when triangular membership functions are used as these are only capable of giving piecewise adaptation between control strategies.

1.2.3 Adaptive Control

Idealistically, adaptive control has a number of advantages over other non-linear control strategies. The control of active suspension is carried out in the knowledge that the plant, and the environment surrounding it, is in a constant state of flux.

Therefore, the idea of a control strategy capable of adapting to those changes seems well founded. However, the creation of a robust technique, that reacts fast enough to the changing environment to be of benefit, is more challenging. The majority of published techniques are based around linear control strategies that are adapted to deal with the changing environment [18, 24]. This helps ensure that robustness is imposed, but also means that fixed non-linear strategies often give comparable results [6]. Adaptive control is not always able to cope with sudden changes in the operational environment, such as a series of step inputs. The adaptation time to deal with such events must be fast and this can affect the resulting stability of the system in other areas. Adaptation clearly has a part to play in the control of active suspension, but it is feasible that it will be on a much more simplistic level within the context of a more complex non-linear control strategy than has so far been published.

1.2.4 Feedback Linearization

Although feedback linearization is not unusual in the field of non-linear control [25], there are relatively few examples of its application to active suspension. This is perhaps because the nature of the non-linearities found in vehicle suspensions do not suit feedback linearization. It is also difficult to quantify the robustness of the control, and some implementations of feedback linearization result in unreasonably large control input requests. It is hard to assess how close the resulting control is to optimal for the original non-linear system.

Buckner [26] recently applied what was termed intelligent feedback linearization to a real vehicle application using a radial basis network to identify vehicle parameters across the operating space of the suspension. A linear model was used, but selected parameters were varied depending on the current state of the system. The linear model was then updated every calculation cycle and the controller updated simultaneously according to linear control theory. Thus linear control techniques can be applied within the context of the current system state. Buckner has so far achieved a significant improvement in the control of suspension deflection for a quarter vehicle test rig. However, the control law used is limited to suspension deflection. It is not clear how the approach will cope as both the complexity of the system and the cost function are increased. It is, however known that radial basis function networks are limited to low dimensional problems and this will prevent extreme complexity being included. One might also surmise from the work of Gordon [6] and others, that adapting a linear system model to allow linear control of a non-linear system to take place is not always preferable.

1.2.5 Neural Networks

Neural networks are now commonly used for black box identification of non-linear systems [27]. It has been proved by a number of authors that a neural network of sufficient complexity is capable of replicating any non-linear relationship. As such, neural networks have been used for a wide variety of tasks, from the modelling of an automotive damper [28], to hand writing recognition [29] and predicting the outcome of litigation cases [30]. There have also been attempts to apply neural networks to problems that are inherently chaotic, such as horse racing and other gambling problems, for which they offer an alternative to classical probability analysis. However, it is important as with any other technique, that a systematic relationship does indeed exist in order for the neural network to find one.

There are a growing number of published examples on the successful application of neural networks to non-linear control problems [31-33]. Some of these include the application to active suspension [14, 34-36]. Early work in neuro-controlled active suspension was performed by Hampo and Marko [37, 38] at the Ford motor company research centre in Dearborn. They successfully trained neural networks using three different methods to control a quarter vehicle model.

The flexibility and speed of neural networks makes them an ideal tool for the control of non-linear systems. Often the relationship between the control action and the current system state cannot be defined using simplistic approaches, and more complex routines are too time consuming. One particular example is their application within the integrated control strategy outlined by Gordon [39]. So far, research into the strategy has been limited to linear systems due to the complexity of applying the scheme with non-linear control strategies. The approach is based around the costate vector, for which the vector differential equation is

$$\dot{p} = -\frac{\partial H}{\partial x} \tag{1.1}$$

For Linear Quadratic Regulation (LQR) problems, this is easily resolved as a function of the matrix solution S to the Riccati equation (1.2).

$$SA + A^T S - (SB)R^{-1}(B^T S) + Q = 0 \tag{1.2}$$

$$p = Sx \tag{1.3}$$

However for non-linear control problems the relationship is no longer a linear function of the states; the flexibility of neural networks should allow them to learn this relationship, thus making the approach viable for non-linear systems and control strategies.

The training of neural networks and their application within control schemes will be outlined in more detail in Chapter 3.

1.3 Thesis Objective

In light of the above the objective of this thesis is to investigate the utilisation of neural networks in the context of non-linear control for active suspension. Particular reference is given to the optimal solution for the minimisation of a non-quadratic cost function and the calculation of the related costates, with the aim of creating a training technique for the direct learning of the non-linear costate equations through gradient based learning without the need for system simulation. This will ultimately allow for the creation of the sub-elements required for use in the integrated control strategy outlined by Gordon [39] for advanced non-linear full vehicle control.

1.4 Outline of Thesis

Chapter 2 details the suspension system model used throughout the thesis and highlights the need for non-linear control in automotive suspension. A non-linear control strategy using a non-quadratic cost function with proven success is then described. This approach and its utilisation with neural networks forms the basis for the remainder of the thesis.

Chapter 3 describes the principles behind neural networks, and their structure in respect to a particular type know as Multi-Layer Perceptron Networks, (MLP). Back-propagation, the means by which MLP networks are optimised, is described in full. A number of methods for the use of neural networks in control are described in further detail to support the information already covered in Chapter 1.

Chapter 4 covers the learning algorithms used for the training of MLP networks to learn the force/state and costate/state relationships for the non-quadratic cost function described in Chapter 2 using a supervised learning approach.

Chapter 5 describes various adaptation to the techniques introduced in Chapter 4 that were required to achieve a successful control strategy.

Chapter 6 compares the performance of the neural controllers with the original control strategies described in Chapter 2. The control of suspension deflection with respect to low frequency disturbances, a problem which effects all 3 techniques, is highlighted.

Chapter 7 further investigates the problem of low frequency suspension drift. An approach using Kalman filter estimation of the underlying road profile is developed which successfully reduces drift, maintaining maximum workspace utilisation. The approach is investigated using linear analysis, but is also proven viable for use with neural networks.

Chapter 8 describes a means for direct learning of the costate/state relationship by a neural network. The method, H_{norm}, is different to other previously tried approaches as the costates, instead of the force/state relationship, are learnt and no simulation is required. The basic theory is proven to be valid for a linear example for which the solution is known using the Riccati equation.

Chapter 9 covers further developments of the H_{norm} approach for successful application to the non-linear quarter vehicle case. Initial development work is carried out using a single degree of freedom model on which the technique is proven to be successful. Further work is required and applied to achieve a working solution for the quarter vehicle case.

Chapter 10 concludes the thesis with a discussion of the work presented and opportunities for further research in the field.

Chapter 2 Non-Linear Optimal Control

Chapter 2 Non-Linear Optimal Control

In this chapter, the motivation for the use of non-linear control strategies is examined. A quarter vehicle model employing force feedback from a Linear Quadratic Regulator (LQR) [40], is used to highlight the weaknesses of linear control for this problem. In later chapters, a similar model will be used in the development of the non-linear neural network controllers. The performance of the LQR based control strategy will be used as a reference to compare the performance of the non-linear controllers developed.

The simplicity of the quarter vehicle model provides opportunity for clear, generalised results to be obtained, allowing meaningful comparison and evaluation between the different suspension control strategies to be employed. Through the addition of non-linear elements to the quarter vehicle model, assessment of the ride quality and handling of a vehicle working in a constrained environment can be made. The non-active components of the model and the passive suspension model to which it is compared are based on values typical of mid range family saloons.

2.1 Quarter Vehicle Model with Active Suspension Actuator

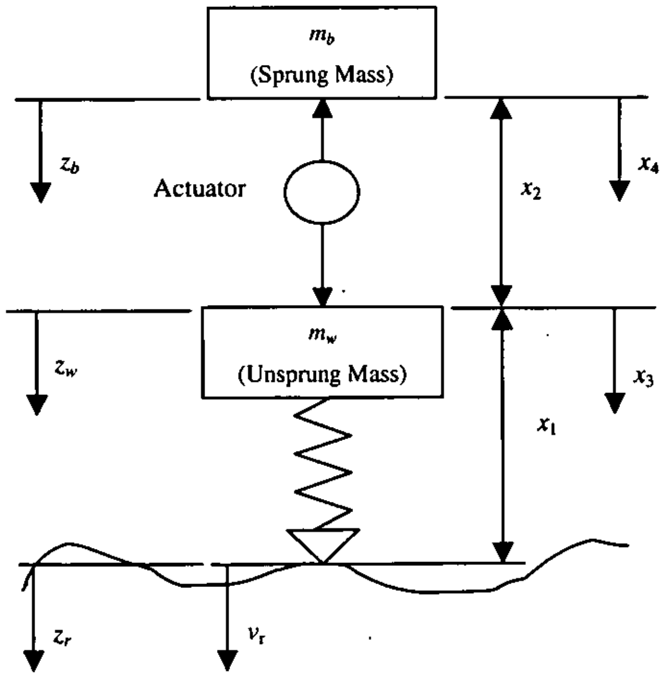


Figure 2.1 - Active Quarter Vehicle Model

The Quarter Vehicle Model is depicted in Figure 2.1. There are two masses, sprung and unsprung, representing the approximate body mass of one corner of the vehicle, m_b, and the wheel assembly, m_w, respectively. The unsprung mass is isolated from road inputs by the tyre, which is modelled initially as a simple linear spring with constant k_t. The body is isolated from the wheel by an active suspension actuator, which applies a force between the two. The passive spring shown in Figure 1.1.(c) is removed to simplify the mathematics, this does not compromise the results in anyway. The actuator is used instead of the spring and damper unit that typifies the passive suspension systems used on the majority of motor vehicles available currently.

The vehicle masses and tyre stiffness are.

$$m_b = 240\text{kg}$$

$$m_w = 38\text{kg}$$

$$k_t = 160000\text{N/m}$$

The system has two degrees of freedom whose dynamics are described by four state variables and defined as a function of the actual vertical displacements:

$x_1 = z_r - z_w$: Relative tyre deflection

$x_2 = z_w - z_b$: Relative suspension deflection

$x_3 = \dot{z}_w$: Velocity of unsprung mass

$x_4 = \dot{z}_b$: Velocity of sprung mass

(z_r : Actual vertical road displacement, z_w : Actual vertical wheel displacement, z_b : Actual vertical body displacement)

The tyre force is given by

$$F_t = K_t x_1 \tag{2.1}$$

The suspension force is given by

$$F_s = -K_{LQR}x \quad \text{for the active suspension and} \tag{2.2}$$

$$\begin{gathered} F_s = K_p x_2 + C_p(x_3 - x_4) \quad \text{for the passive suspension used for comparison} \\ K_p = 18000\,\text{N/m} \quad C_p = 1200\,\text{N/m/s} \end{gathered} \tag{2.3}$$

The optimal gain matrix K_{LQR} is given by the minimisation of a quadratic performance criterion in the following form

$$J(x,u) = \int_0^\infty \{x^T Q x + u^T R u\} dt \tag{2.4}$$

The weighting matrices Q and R are selected to trade off the regulation of the states x and the suspension actuation force u. For the quarter vehicle problem, the values of Q and R are chosen to limit tyre and suspension deflection without the use of large actuator forces that would result in high body accelerations. α and β are adjusted to meet reference criteria based on the passive system's response to the following initial conditions: (a) x =[0 0 0 1.25]^T(b) x = [0 0 2.5 0]^T. The passive suspension model following these two initial condition events gives peak deflection values of (a) suspension deflection = 0.0995m, (b) tyre deflection = 0.0278m

$$\begin{gathered} Q = diag([\alpha \quad \beta \quad 0 \quad 0]) \quad R = 1/m_b^2 \\ \alpha = 112800 \quad \beta = 1492.5 \end{gathered} \tag{2.5}$$

The minimising gain matrix K_{LQR} is found by solving the algebraic Riccati equation [41].

$$\begin{gathered} A^T S + SA - SBR^{-1}B^T S + Q = 0 \\ K_{LQR} = R^{-1}B^T S \end{gathered} \tag{2.6}$$

All four states and the control force act purely in the vertical plane. The sole disturbance input to the model is from the road, which for simplicity is described as the velocity of a point tyre contact patch v_r. Thus the following state differential equations for the model can be obtained:

State Matrices

$$\dot{x} = Ax + Bu + Gv_r$$

$$A = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ k_t / m_w & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ -1/m_w \\ 1/m_b \end{bmatrix} \quad G = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad (2.7)$$

2.2 Dynamic Analysis of Quarter Vehicle Model.

The quarter vehicle model can be analysed in a number of different ways to provide insight into the ride and handling performance of a vehicle. In the time domain, information can be obtained from initial condition responses and simulated responses to road inputs. In the frequency domain, information can be obtained using either modal analysis or frequency response analysis.

The time domain responses can be used to study the conflict between ride comfort and use of the suspension workspace. A suitable measure of ride comfort [42] is the r.m.s. value for vertical body acceleration and this can be contrasted against r.m.s. and peak suspension workspace usage.

Similarly, assessment of the handling capability of the vehicle can be made from the time history of tyre deformation. This, combined with knowledge about how tyres perform under vertical load, Figure 2.2 [43, 44], can be used to assess the potential effects of different suspension set-ups on vehicle handling, clearly fluctuations in tyre deflection will result in a reduced ability to generate lateral force.

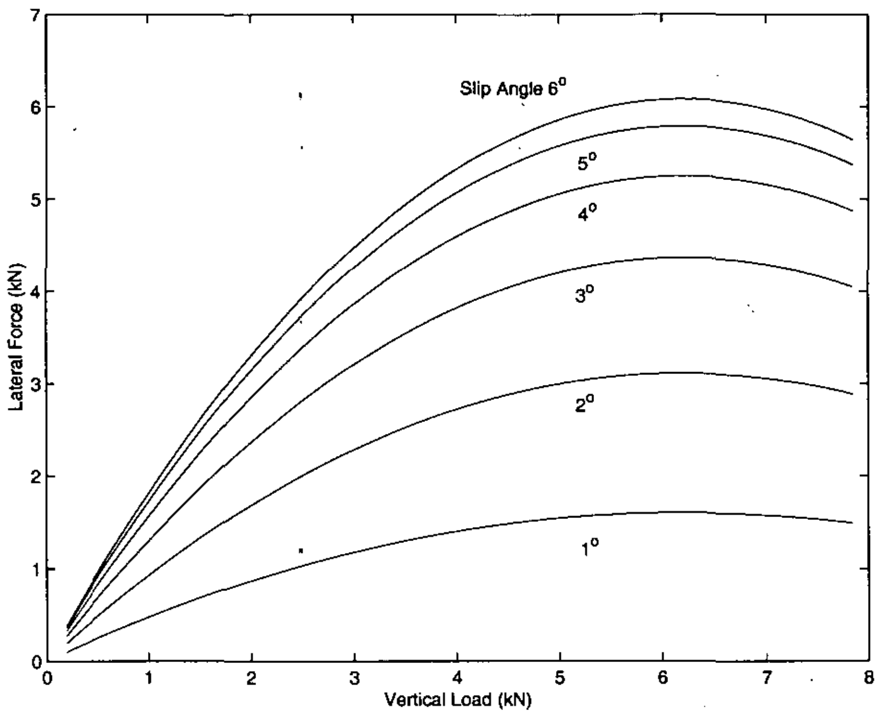


Figure 2.2- Lateral Tyre Force versus Vertical Load

In addition to the basic model, a number of non-linear properties of the real system can be simulated. A model that simulates the tyre lift off point can replace the linear tyre spring. The point at which the tyre will leave the road when using a simple linear spring model can be calculated using the following equation.

$$x_{1_{LF}} = \frac{(m_b + m_w)g}{k_t} \tag{2.8}$$

thus Equation 2.7 can be redefined.

$$\begin{aligned} \dot{x}_3 &= (k_t/m_w).x_1 - (1/m_w).u & x_1 &<= x_{1_{LF}} \\ \dot{x}_3 &= ((m_b + m_w)g/m_w) - (1/m_w).u & x_1 &> x_{1_{LF}} \end{aligned} \tag{2.9}$$

Most real vehicles use stiff rubber elements called 'bump-stops', which prevent sudden impact between the vehicle body and the suspension hardware. When contacted, they result in high suspension forces that lead to large body accelerations and/or changes in tyre load. Such events will negatively affect the overall ride comfort and road holding of the vehicle. The suspension workspace available to a vehicle of this type is typically of the order ±100mm. These limits can be considered when assessing the time history of the suspension deflection, or implemented more rigidly in the model with stiffer linear springs representing the bump-stops. These become active when the range of the suspension workspace is exceeded.

$$\begin{array}{ll} x_2 < -0.095\text{m} & F_{s_{Total}} = F_s + K_{\text{Bump\_stop}}(x_2 + 0.095) \\ -0.095\text{m} \le x_2 \le 0.095\text{m} & F_{s_{Total}} = F_s \\ x_2 > 0.095\text{m} & F_{s_{Total}} = F_s + -K_{\text{Bump\_stop}}(x_2 - 0.095) \\ \multicolumn{2}{c}{K_{\text{Bump\_stop}} = 200000\,\text{N/m}} \end{array} \tag{2.10}$$

2.2.1 Initial Condition Analysis

The two initial conditions to be considered having either an initial body or wheel velocity, promote responses that correspond to the body and wheel resonances respectively. These conditions are known as body bounce and wheel hop. Body bounce is the resonance at which the body oscillates. The frequency at which this takes place is typically in the range 1.2 – 2.5Hz depending on the body mass / suspension stiffness ratio. Wheel Hop is the resonance during which the wheel oscillates at its maximum, the body remaining nearly static. That is, the suspension deflection and tyre deflection act in equal and opposite directions to maintain a near constant body position while the wheel oscillates beneath. This happens typically at a frequency of around 10 - 13Hz [13]. Figures 2.3 and 2.4 show state settling responses following an initial body velocity of 1.25m/s and an initial hub velocity of 2.5m/s respectively. In both cases, all other states are initially zero. Therefore the initial state vectors can be expressed in the following format; $[\ 0\ 0\ 0\ 1.25]^T$ and $[0\ 0\ 2.5\ 0]^T$ respectively.

The results (Figures 2.3 & 2.4) clearly show the benefit of active over passive suspension, as while the initial hub velocity response remains similar, significant improvements are made in body control during the initial body velocity response. A significant reduction is made in R.M.S. body acceleration with only a minor increase in R.M.S suspension deflection.

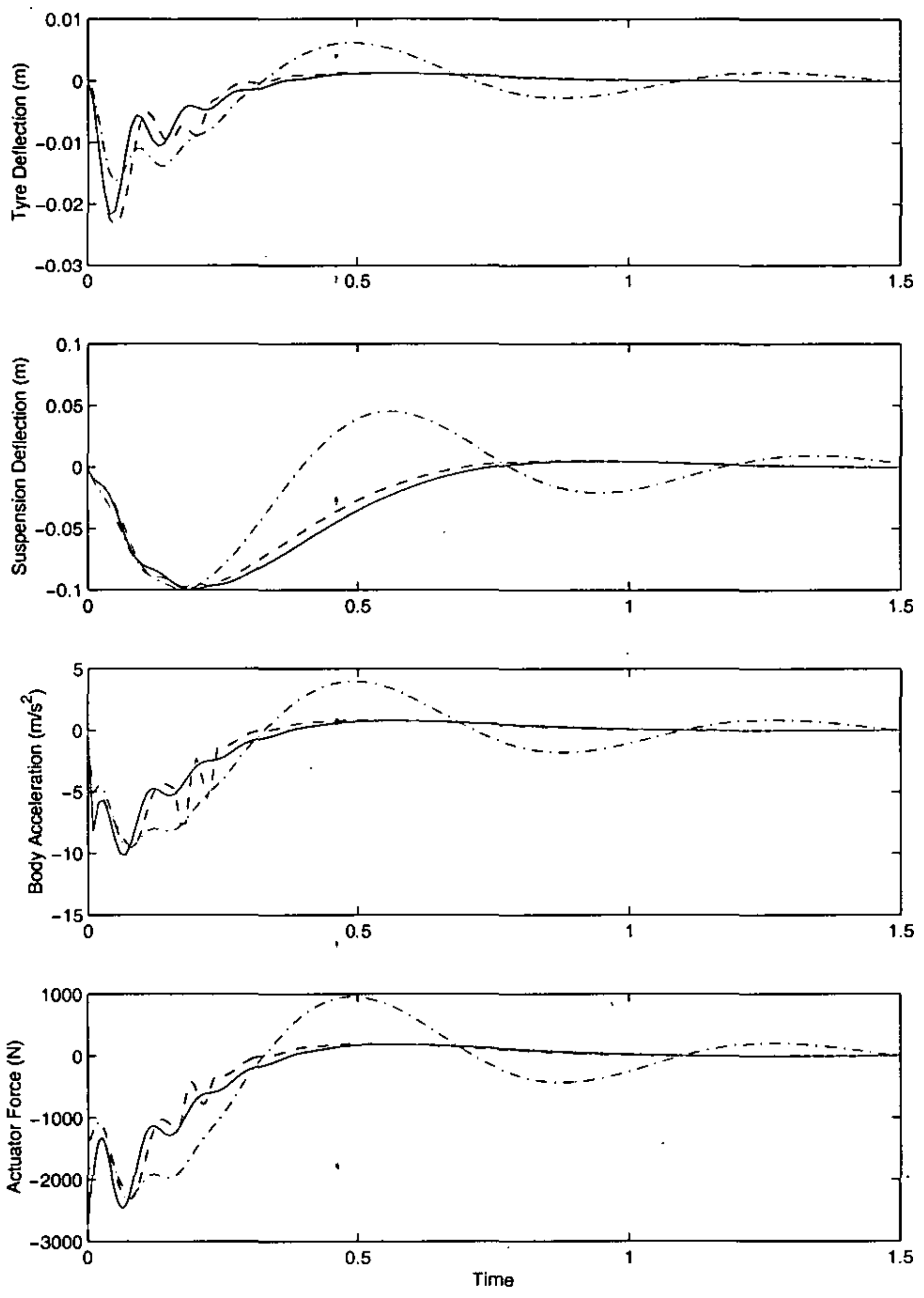
Tables 2.1 & 2.2 and Figures 2.3 & 2.4 also show the significance of the non-linearities to the resulting response. When developing the LQR controller, the constraints put on the operating range of the tyre and suspension workspace cannot really be considered and therefore the system response, does not act to prevent tyre lift off or bump-stop contact. In the wheel hop mode this results in higher levels of tyre deflection. In reality, the tyre leaves the road, which would significantly affect handling. This also results in slightly higher suspension deflection but would have no significant effect on ride comfort. The effect of the non-linearities in the model is more significant to the body bounce mode; due to the harshness of the bump-stop contacts, the ride comfort is seriously affected. This can be seen in figure 2.4 as two additional spikes in the body acceleration plot at approximately 0.2seconds which do not correspond with the actuator force. Higher loads are also put on both the occupants and the vehicle's tyres. The introduction of bump-stops to the model shows the significant effect this has on the suspension performance, when a linear system is assumed during controller development. If the controller was developed with awareness of the constraints put on the suspension workspace and of tyre lift [45], then it should be possible to significantly improve the overall suspension performance.

Table 2.1 – Initial Hub Velocity Response

Initial Hub Velocity	R.M.S Values for First Second			Peak Tyre Deflection (mm)
	Tyre Deformation (mm)	Suspension Deflection (mm)	Body Acceleration (m/s^2)	
Passive	4.9	5.4	1.5926	27.8
LQR Controlled with Linear Model	4.9	5.9	1.6152	27.8
LQR Control with Non-linear Model	5.4	6.9	1.5793	28.9

Table 2.2 - Initial Body Velocity Response

Initial Body Velocity Response	R.M.S Values for First 1.5 Seconds			Peak Suspension Deflection (mm)
	Tyre Deformation (mm)	Suspension Deflection (mm)	Body Acceleration (m/s^2)	
Passive	5.2	39.4	3.3619	99.5
LQR Controlled with Linear Model	4.2	42.9	2.47	99.5
LQR Control with Non-linear Model	4.7	43.1	2.471	101.1



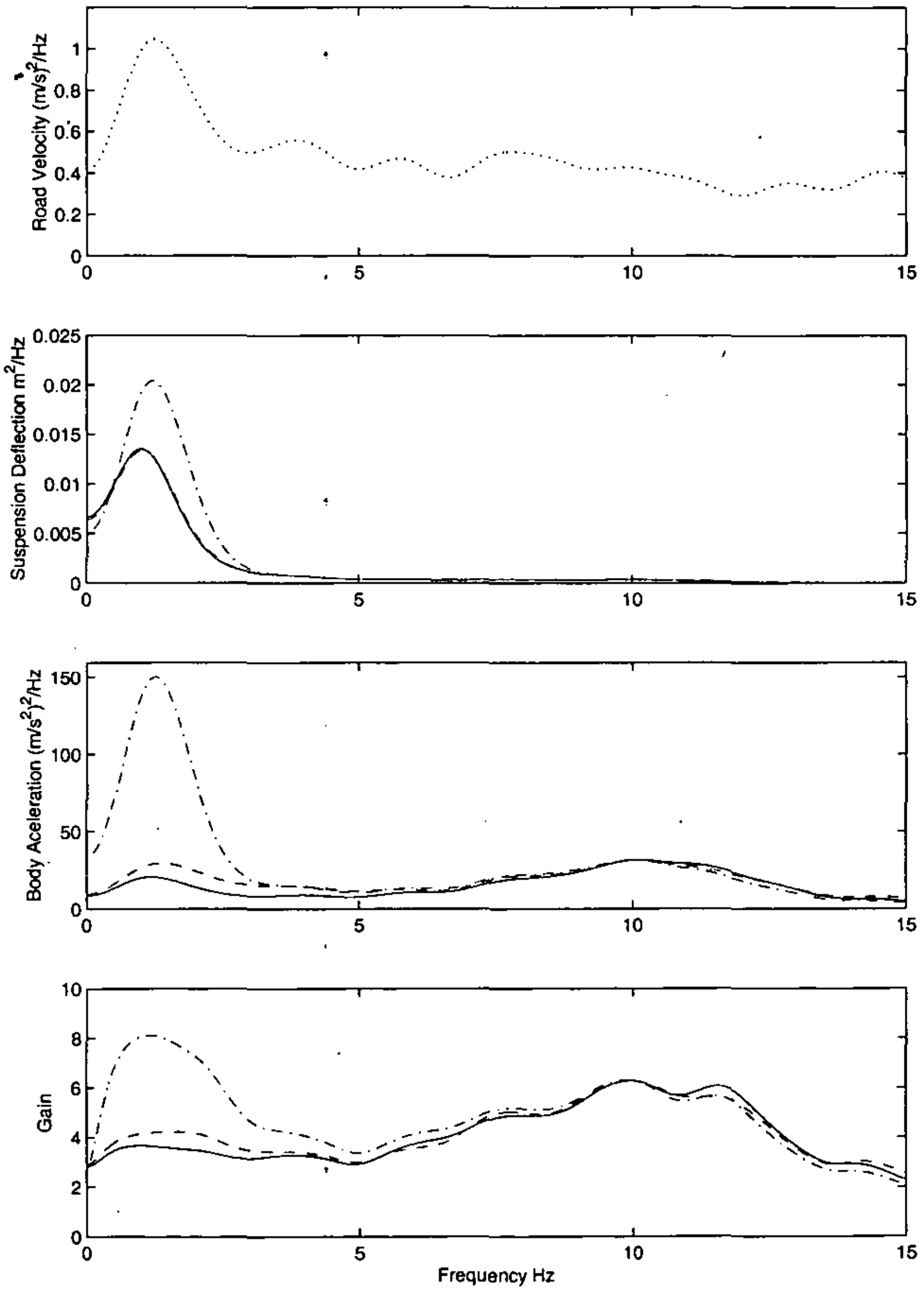
–· –Passive Model – Linear Quarter Vehicle Model, – – Non-linear Vehicle Model

Figure 2.4- LQR Controlled Response to Initial Body Velocity

2.2.2 Frequency Response Analysis

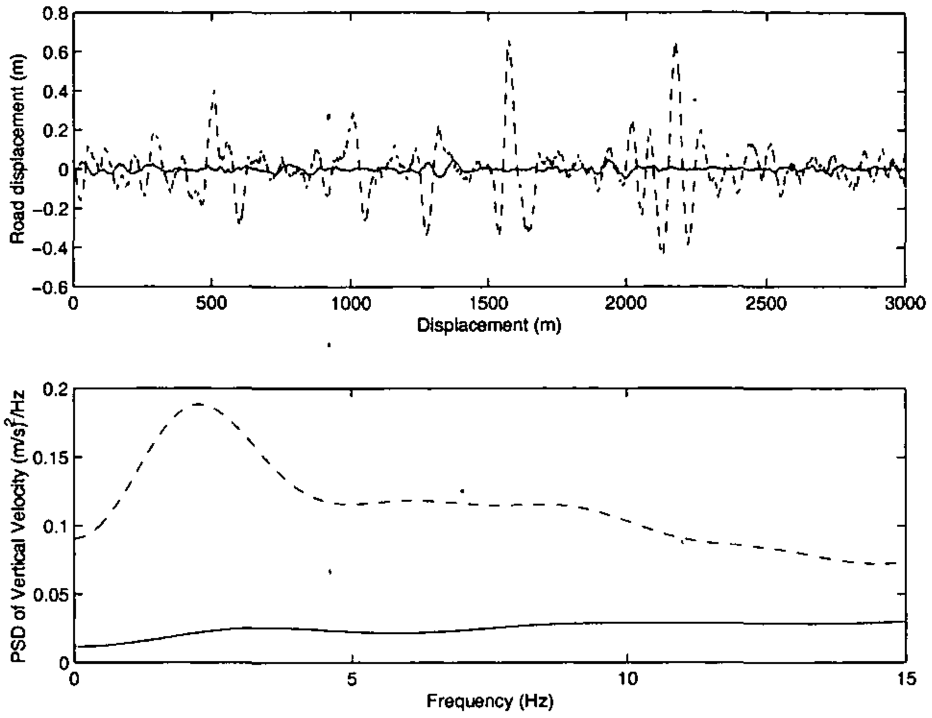
Frequency response analysis has the advantage that the full dynamic range of the suspension is visible, the resonances showing up clearly. The body bounce and wheel hop frequencies (~ 1.2 Hz & ~ 10.1 Hz respectively) can clearly be seen on the Power Spectral Density, PSD plots of suspension deflection and body acceleration for the quarter vehicle models. Again the benefit of active suspension over passive at the body resonance point is clear with significant reductions in the PSD of both suspension deflection and body acceleration. When working with linear models the gain does not change, irrespective of the magnitude of the input used. However, as can be seen from Figure 2.5, the addition of non-linear elements to the model changes the frequency response while the model has remained the same in the linear operational region. Different responses will be obtained depending on the relative magnitude of the road input used, and the *x*-axis position of the relevant peaks will move. For this reason, frequency analysis of the performance of non-linear systems in the context of the PSD is not considered a particularly appropriate analysis tool.

However, it is possible to use spectral density plots to compare the frequency components of different roads. Figure 2.6 shows the displacement and the PSD (frequency components) of two roads in the local area to Loughborough. Both roads have been sampled at 0.1m intervals using laser-measuring equipment. The displacement is then compared to a 1000 sample moving average, so very low frequency displacement information is lost. The first road used is a B-class road known as Breakback Road [LandRanger Map reference, Start: SK5017 Finish: SK5214]. The second is part of the A6, running between Loughborough and Leicester [LandRanger Map reference, Start: SK5517 Finish: SK5910]. As can be seen from the displacement plot, the Breakback Road is a lot rougher than the A6 with, proportionally, a much larger low frequency content in its PSD. Despite the speed difference at which the two roads are simulated in order to evaluate the PSD, the amplitude of Breakback Road is clearly much larger than the A6. It also has a very large low frequency component close to the body bounce frequency, which would cause serious disturbance to the ride and handling if the vehicle were to travel faster. The A6 has very little low frequency content and will principally excite the wheel resonance in the vehicle model.



• • Vertical Road Velocity, –· – Passive Model, – Linear Quarter Vehicle Model, – – Non-Linear Vehicle Model

Figure 2.5- PSD and Gain Responses to Simulated Road



-- Breakback Road at 20m/s, - A6 at 30m/s

Figure 2.6-Displacement Profiles & Vertical Velocity Spectra for the Breakback and A6 Roads

2.2.3 Simulated Road Responses

The quarter vehicle model was simulated for 60 seconds over the two road profiles, the results of which are summarised in Table 2.2. Breakback Road, as expected from the frequency analysis, excites the body bounce mode to a much higher degree than the A6. This causes large suspension deflections, which result in bump-stop contact and higher levels of body acceleration in the non-linear model. This also causes greater variation in the levels of tyre deflection and therefore would negatively affect the handling of the vehicle. The A6 by comparison causes very little suspension deflection even at the higher speed, and the degree of comfort on this road would be much greater for the vehicle passengers. Proportionally however, because of the greater high frequency content of the A6, the levels of tyre deflection are much larger. The differences between the behaviour of the suspension on these two roads highlights the need for non-linear control of active suspension. This is not only because of the constrained suspension workspace, but also the wide variety of working conditions experienced by the suspension. These two roads are less than 3 miles apart and the vehicle should be able to deal with both, with equal levels of satisfaction.

Table 2.3 - Breakback Road at 20 m/s

Suspension Model	R.M.S			Peak Values	
	Tyre Deflection (mm)	Suspension Deflection (mm)	Body Accele-ration (m/s^2)	Tyre Deflection (mm)	Suspension Deflection (mm)
Passive	6.7	26.1	2.954	32.2	106.8
Linear	6.4	42.2	2.248	33.6	167.1
Non-linear	6.5	38.5	2.444	40.1	116.5

Table 2.4 - A6 at 30 m/s

Suspension Model	R.M.S			Peak Values	
	Tyre Deflection (mm)	Suspension Deflection (mm)	Body Acceleration (m/s^2)	Tyre Deflection (mm)	Suspension Deflection (mm)
Passive	2.5	6	0.85	14.1	32.8
Linear	2.6	6.8	0.76	13.3	30.8
Non-linear	2.6	6.8	0.85	14.1	32.8

2.3 Non-linear Control

There are a number of approaches to achieving non-linear control, as described in the introduction. However, the principles of LQR control do seem to be the natural solution to the basic problem of controlling the quarter vehicle model [46, 47]. An approach that builds on these principles, to give a control strategy that takes account of the non-linearities of the system, would seem to be a sensible solution to the problem. Such an approach is proposed by Marsh [48]. Marsh puts forwards a method that builds on a secondary approach to calculating the LQR gains, based on the Hamiltonian equations. Having proved the method for the linear case, it is then applied to a non-quadratic cost function incorporating additional higher power terms to account for the workspace constraints, as well as for the non-linear elements of the system.

2.3.1 General Optimal Regulator Design

Marsh's approach to non-linear control uses the Pontryagin formulation [49] for the development of a general optimal controller. The resulting method calculates an open-loop series of controls that stabilise the system from a given initial condition. Assuming no further disturbances to the system occur, the state equations for any system can be described as a function of the system states and the control inputs.

$$\dot{x} = f(x,u) \tag{2.11}$$

In the case of the quarter vehicle, and for simplicity, only one control input is considered. The dynamic cost is, itself a function of the system states and control force. This is combined with a cost on the final state, $\varphi(x_{t_f})$ to form a scalar value as a performance index.

$$J = \varphi(x_{t_f}) + \int_0^{t_f} L(x,u)dt \tag{2.12}$$

A control signal $u(t)$ must be found that minimises the performance index J. The performance index is combined with Lagrange multipliers p and the system differential equations.

$$J = \varphi(x_{t_f}) + \int_{t_0}^{t_f} L(x,u) + p^T\{f(x,u) - \dot{x}\}dt \tag{2.13}$$

The Hamiltonian is formed from a subset of this equation

$$H = L + p^T.f \tag{2.14}$$

The last component of Equation 2.13 can be integrated by parts to give

$$\int_{t_0}^{t_f} -p^T\dot{x}dt = -p_{t_f}^T x_{t_f} + p_0^T + \int_{t_0}^{t_f} \dot{p}^T x dt \tag{2.15}$$

and combined with the Hamiltonian, J is now defined

$$J = \varphi(x, t_f) - p_{t_f}^T x_{t_f} + p_0^T + \int_{t_0}^{t_f} \{H(x,u) + \dot{p}^T x\} dt \tag{2.16}$$

u must then be adapted to minimise the cost function and hence the derivative of the cost function with respect to u must be calculated. A standard variational calculus approach gives [48]

$$\delta J = \left[\left(\frac{\partial \varphi}{\partial x} - p^T\right)\delta x\right]_{t_f} + \left[p^T \delta x\right]_{t=t_0} + \int_{t_0}^{t_f}\left[\left(\frac{\partial H}{\partial x} + \dot{p}^T\right)\delta x + \frac{\partial H}{\partial u}\delta u\right] dt \tag{2.17}$$

It would be computationally tedious to calculate the variation of δJ caused by δx due to a particular δu, so the Lagrange multipliers are set to nullify the coefficients of δx.

$$\dot{p}^T = -\frac{\partial H}{\partial x} = -\frac{\partial L}{\partial x} - p^T \frac{\partial f}{\partial x} \tag{2.18}$$

and they have the following boundary condition at t_f.

$$p(t_f) = \frac{\partial \varphi}{\partial x_{t_f}} \tag{2.19}$$

Equation 2.17 now becomes

$$\delta J = p_0^T \delta x_0 + \int_{t_0}^{t_f} \frac{\partial H}{\partial u} \delta u dt \tag{2.20}$$

Assuming u is held constant and the equations of motion maintained, p is hence $\delta J/\delta x_0$ and, as such, often referred to as the costates or influence functions. For time independent systems, a given p will correspond with the current state irrespective of time.

The impulse response function $\partial H/\partial u$ is so called because it gives the change in J due to a unit impulse from a given control input at time t. To find the unconstrained minimum value for the performance index, δJ must be 0 for all δu. This can only occur if

$$\frac{\partial H}{\partial u} = 0 \quad t_0 \le t \le t_f \tag{2.21}$$

Confirmation of minima can be established by calculating the second order derivatives of *J*. There are a number of methods for finding the minima, not all of which can be successfully applied to the quarter vehicle problem; Marsh successfully applied a discrete control method.

2.3.2 Non-Linear Cost Function and Costate Calculations

Marsh considered a non-quadratic cost function of the form

$$L = \tfrac{1}{2}\left[x^T Q x + u^T R u\right] + N(x,u) \tag{2.22}$$

where *N* contains the non-quadratic terms. The following form was chosen.

$$\begin{gathered} L = 0.5\left(\alpha_1 x_1^2 + \beta_1 x_2^2 + \alpha_2 x_1^{n_1} + \beta_2 x_2^{n_2} + \left(\frac{u}{m_b}\right)^2\right) \\ \alpha_1 = 16000 \qquad \beta_1 = 500 \\ \alpha_2 = 2\times10^{12} \quad \beta_2 = 5\times10^{11} \\ n_1 = 6 \qquad n_2 = 10 \end{gathered} \tag{2.23}$$

n$_1$ & n$_2$ must be integer valued, and a multiple of two, their magnitude dictates how quickly the additional constraint in the penalty function rises and hence the x_2 is more strongly constrained than x_1.

The Hamiltonian Function *H* can be derived from the cost function *L* and the state equations that describe system dynamics.

$$\begin{gathered} H(t) = L(x(t), u(t) + p(t) f(x(t), u(t))) \\ H(t) = 0.5\left(\alpha_1 x_1^2 + \beta_1 x_1^{n_1} + \alpha_2 x_2^2 + \beta_1 x_1^{n_2} + \left(u/m_b\right)^2\right) \\ + p_1 \dot{x}_1 + p_2 \dot{x}_2 + p_3 \dot{x}_3 + p_4 \dot{x}_4 \end{gathered} \tag{2.24}$$

The Lagrange multiplier, costate elements of equations 2.24 can then be redefined to include the system dynamics.

$$
\begin{aligned}
p_1\dot{x}_1 &= -p_1x_3 \\
p_2\dot{x}_2 &= p_2(x_3 - x_4) \\
p_3\dot{x}_3 &= \frac{p_3}{m_w}(F_t - u) \\
p_4\dot{x}_4 &= \frac{p_4u}{m_b}
\end{aligned}
\tag{2.25}
$$

The derivative of the Hamiltonian function can then be written down with respect to both the systems states and control force *u*.

$$
\begin{aligned}
\dot{p}_1 &= -\frac{\partial H}{\partial x_1} = -\alpha_1 x_1 - \frac{n_1}{2}\beta_1 x_1^{n_1-1} - p_3\frac{K_t}{M_w} \\
\dot{p}_2 &= -\frac{\partial H}{\partial x_2} = -\alpha_2 x_2 - \frac{n_2}{2}\beta_2 x_2^{n_2-1} \\
\dot{p}_3 &= -\frac{\partial H}{\partial x_3} = p_1 - p_2 \\
\dot{p}_4 &= -\frac{\partial H}{\partial x_4} = p_2 \\
\frac{\partial H}{\partial u} &= \frac{u}{m_b^2} - \frac{p_3}{m_w} + \frac{p_4}{m_b}
\end{aligned}
\tag{2.26}
$$

This allows the optimal control force to be defined as a function of the costate values *p*.

$$
\frac{\partial L}{\partial u} + p\frac{\partial f}{\partial u} = 0 \tag{2.27}
$$

Since *u* is a scalar function in the present case

$$
\begin{aligned}
&\frac{u}{m_b{}^2} - \frac{p_3}{m_w} + \frac{p_4}{m_b} = 0 \\
&u = \frac{m_b{}^2 p_3}{m_w} - p_4 m_b \\
&\frac{\partial u}{\partial p} = \frac{m_b{}^2}{m_w} - m_b = -m_b{}^2 B
\end{aligned}
\tag{2.28}
$$

2.3.3 Parameter Optimisation for Discrete Optimal Controls

Marsh's implementation was based on discrete controls with zero order held (ZOH) intervals T seconds; as T tends to zero, so the resulting controls tend to that of a continuous time solution. As the optimal controls form a finite set of parameters, $u_0,u_1,u_2....u_{n-1}$, these can be optimised to minimise the cost, J, using standard parameter optimisation techniques such as gradient descent.

The Cost Gradient for each control parameter can be calculated from the derivative of the Hamiltonian equation with respect to that parameter. Because the control parameter is held constant for the control period, the cost gradient calculation is relatively simple.

$$\frac{\partial J}{\partial u_i} = \int_{t_i}^{t_i+1} \frac{\partial H}{\partial u_i} dt, \quad (i = 0,1,2,...,n-1) \tag{2.29}$$

The optimisation of the parameters is achieved using the following routine:

1. Make an initial guess at the control parameter set, $u_0,u_1,u_2,...u_{n-1}$. This may be based on either a passive or LQR based simulation of the problem from the initial condition x_0, sampling the controls at intervals T.

2. Integrate the state equations forward in time, from initial condition x_0 for simulation period $t_{\max}$, recording the dynamic cost and final state values for each control period T. $t_{\max}$ should be large enough to allow the states to settle to near zero values. Linear costate values can be used to estimate the costates and the remaining cost of settling over an infinite time period in order to reduce the value of $t_{\max}$.

3. Perform backwards integration from $t_{\max}$ to t = 0 of the system costates, performing the calculation of $\partial J/\partial u_i$ simultaneously. The Final states at $t_{\max}$, and calculated costates from step 2, are used as the initial starting position for the backwards optimisation.

4. Adapt the discrete control parameters, u_i, via a line search based on gradient information to minimise the overall dynamic cost calculated though

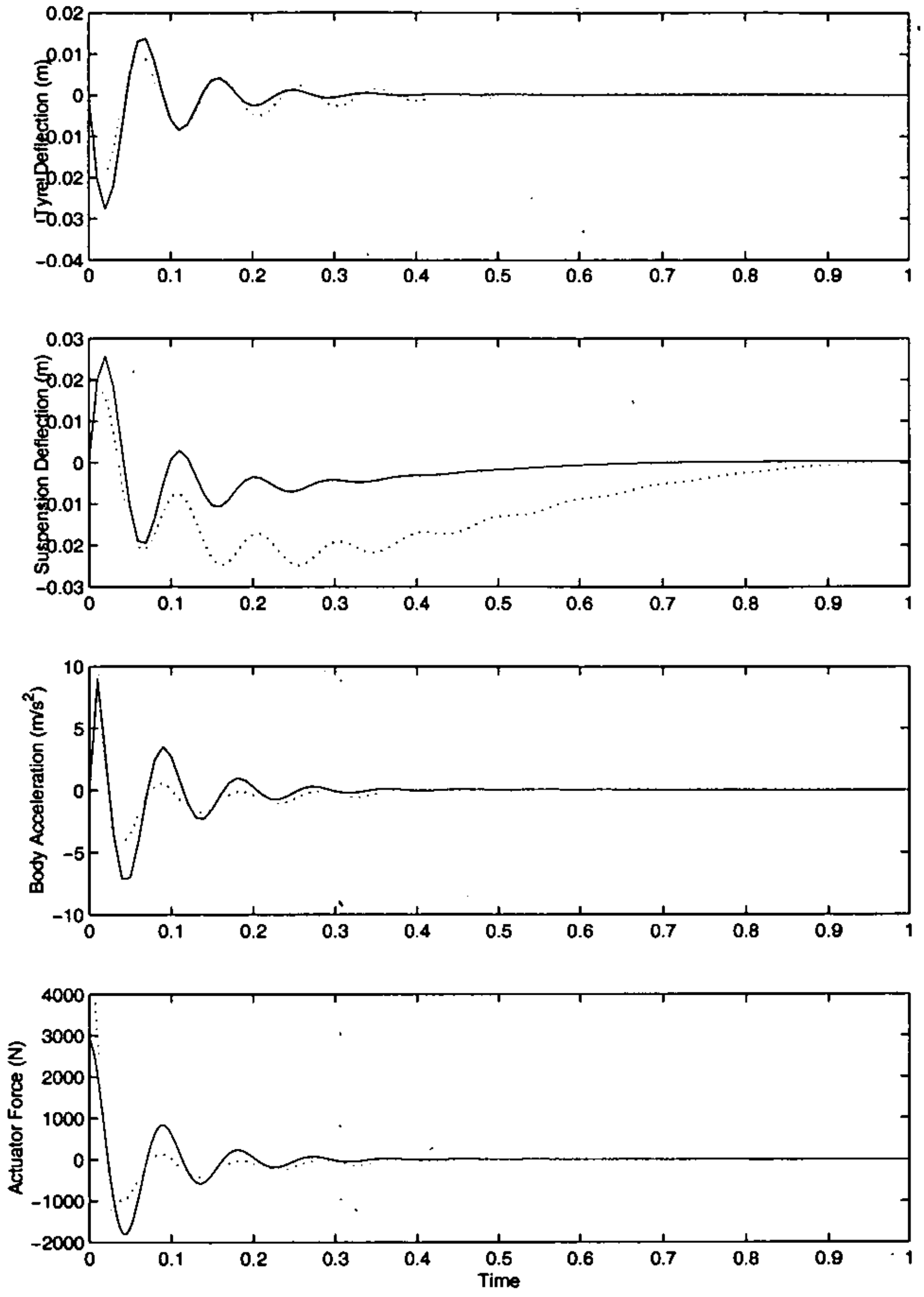
simulation, until the minimal cost in the current set of gradient directions is found.

5. Return to step 2, unless the convergence criteria are met.

2.3.4 Initial Condition Response Comparison.

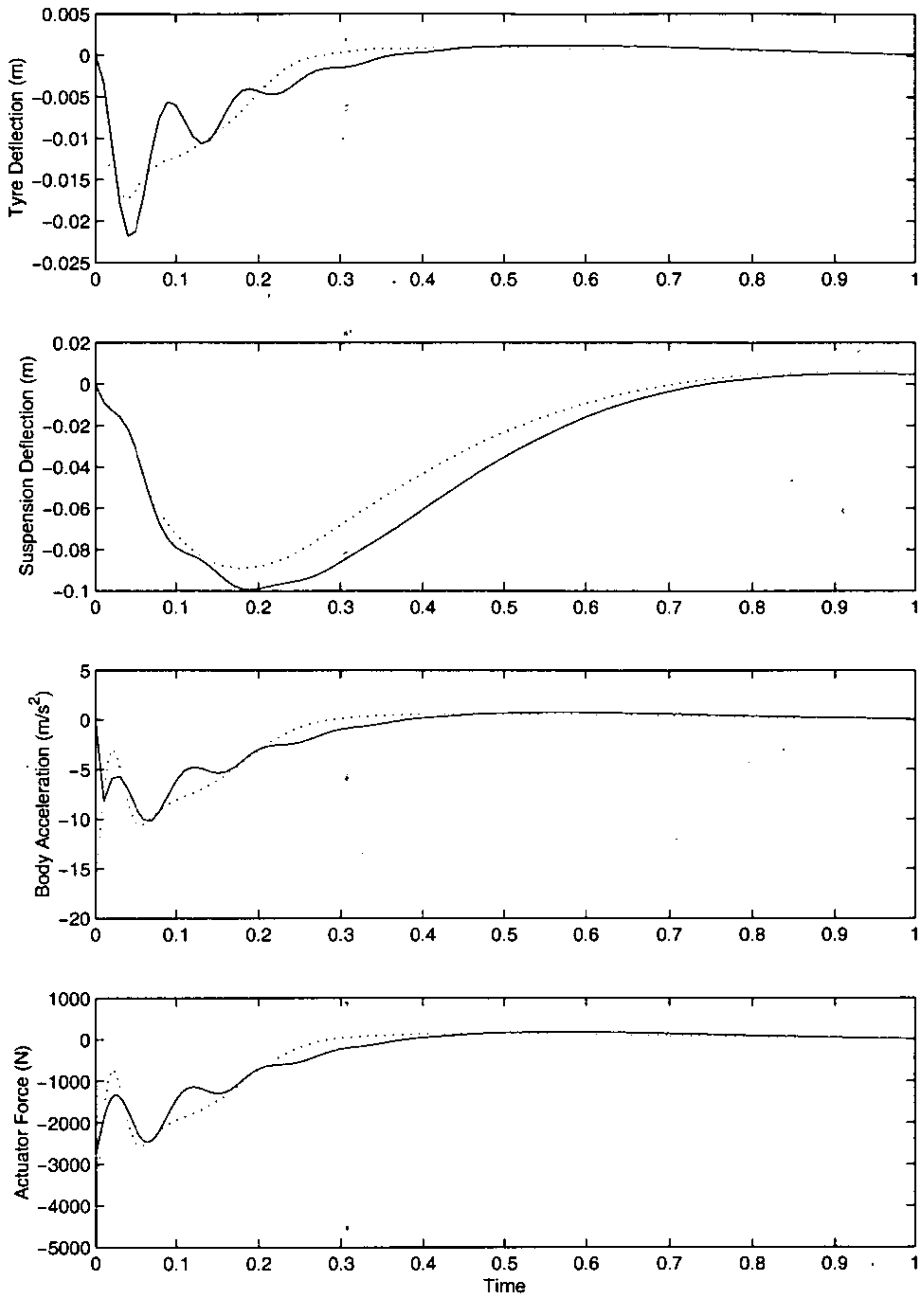
Figures 2.7 and 2.8 show the response of the linear controller in comparison with Marsh's non-linear control strategy for the two initial conditions used in section 2.2.1. The benefits of using a non-quadratic cost function are clearly apparent in both figures. For both conditions, the non-linear controller exhibits comparable or lower levels of body acceleration while operating within the limits of suspension and tyre deflection. For example during the initial hub velocity condition, the non-linear controller uses more suspension deflection but remains well within the acceptable limit. By doing this, tyre - road contact is maintained and body acceleration is significantly reduced.

During the initial body velocity response, the non-linear controller ensures that the bump stops would not be contacted without significantly increasing body acceleration. Peak tyre deflection is also significantly reduced.



– Linear Quarter Vehicle Model, ··· Open loop Non-linear Control

Figure 2.7 - A comparison of LQR and Marsh's Open loop Non-linear Control for an initial hub velocity.



– Linear Quarter Vehicle Model, · · · Open loop Non-linear Control

Figure 2.8 - A comparison of LQR and Marsh's Open loop Non-linear Control for an initial body velocity

2.3.5 Implementation of the Non-linear Control Strategy.

Available computing hardware has improved significantly since Marsh did his work, but the time taken to perform the optimisation process outlined in section 2.3.3 still negates its use in real-time. It is not possible to complete the calculation of the optimal control input within the discrete control period, *T*. Therefore the process cannot be used directly on-line. The success of LQR as a control strategy is in some part due to the existence of a linear relationship between the current states of the system and the optimal control to be applied at that point. However, for the non-linear control strategy, the relationship is much more complex and hence cannot be replicated with simple linear equations. Because of this, Marsh chose to create a polynomial feedback function to relate the optimal control forces to the corresponding states.

Although Marsh found the performance of the polynomial feedback function satisfactory, the level of fitting was not sufficient to offer truly optimal non-linear feedback control. The time to achieve a suitable level of fit was also found to be unsatisfactory. The number of parameters to be set, order of polynomial and the distribution of the basis functions all add to the complexity of the problems Marsh experienced in the development of the polynomial feedback controller.

At the time of Marsh's work, neural networks were still in their infancy. Computational power and optimisation algorithms had yet to evolve sufficiently to warrant investigation by Marsh as a suitable solution to the problem. Since then, the desktop PC has evolved considerably; the power and memory capacity available, plus an increased knowledge and acceptance of neural networks as powerful data mapping tools, make it a prudent time to investigate how applicable they are to this problem.

Chapter 3 Principles of Neural Networks

Chapter 3 Principles of Neural Networks

In this chapter the principles of feed forward neural networks [29, 50, 51] are explained, including: the forward processing of data through the network to the output layer, the standard transfer functions used in each neuron, and back-propagation, the algorithm used to optimise the neural network coefficients to improve data mapping. A number of control applications of neural networks are reviewed and the viability of the methods employed in the rest of the thesis is discussed.

Neural networks are principally used to map an input data set to an output set. Where a relationship exists, neural networks have proven successful in finding it. A number of different training methods exist to adapt the network coefficients (weights), to improve the level of fit achieved by the network. The method used depending on the problem and type of network. The input data is passed through the network layer by layer, in a highly parallel fashion. In feed forward networks, the data only advances through the layers from the input to the output layer. In recurrent networks, the data is passed both forwards and backwards between layers. The feed forward network is most commonly trained to map data using back-propagation. The error gradient between the network output and the true values is propagated backwards through the layers to allow gradient descent type optimisation of the weights.

3.1 Forward Calculation of Neural Network

Figure 3.1 shows the basic format of a feed forward network, generally consisting of three or more layers; the input layer through which all data enters the network, the hidden layer(s) which processes the data internally and are not in direct contact with the external world and the output layer, through which all data is passed to the outside world. The input layer performs no calculations on the data and in many publications is not considered an actual layer. It is simply a set of nodes to which the hidden layer(s) are connected, through which data enters the network. The hidden layer(s) performs the majority of the calculation processes that take place in the network. The output layer scales the outputs and brings together the outputs of the preceeding hidden layer neurons.

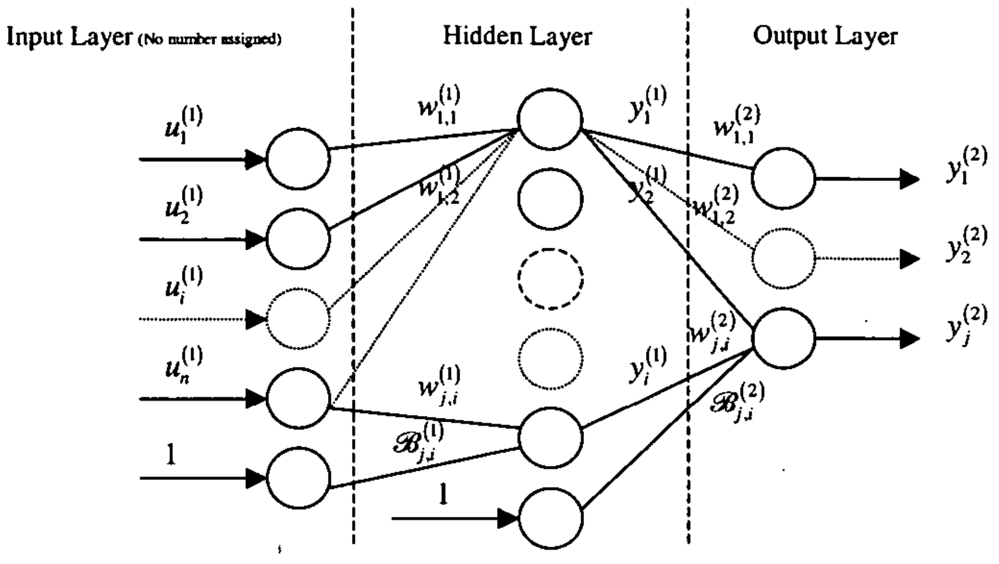


Figure 3.1-Basic Neural Network Architecture

The hidden layer(s) and output layer are formed from neurons. Each neuron consists of: a set of weights that are applied to the output of the preceeding layer, a bias (theshold) which is applied to the sum of the weighted inputs (the bias is commonly represented as a weighted unit input) and a transfer function, through which the result is passed to give a scalar output. Each neuron in the layer performs the same process simultaneously on the output of the preceeding layer to create a vector output.

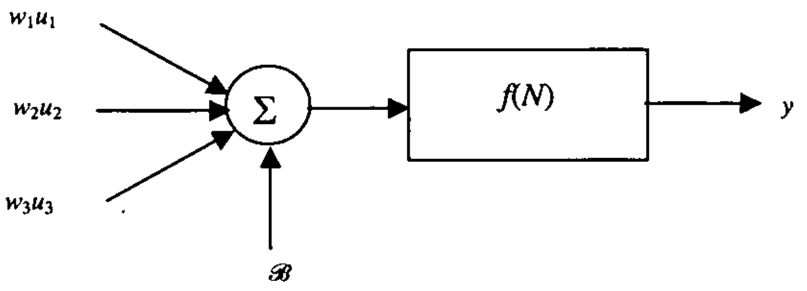


Figure 3.2 - Neuron Structure

The feed forward calculation process is easily represented algebraically, but can become prohibitively large due to repetitive nature of the network calculations. The output of the layer can be represented as a function of the layer inputs

$$y_j^{(k)} = f_j^{(k)}\left(\sum_i w_{j,i}^{(k)} u_i^{(k)} + \mathscr{B}_j^{(k)}\right) = f_j^{(k)}\left(N_j^{(k)}\right), \quad N_j^{(k)} = \sum_i w_{j,i}^{(k)} u_i^{(k)} + \mathscr{B}_j^{(k)} \quad (3.1)$$

$f_j^{(k)}\left(N_j^{(k)}\right)$ is the transfer function of the j^{th} neuron in the k^{th} layer.

The output of any given layer becomes the input for the next layer in the sequence.

$$u_i^{(k+1)} = y_j^{(k)} : \text{output of the } j^{\text{th}} \text{ neuron of } k^{\text{th}} \text{ layer,} \tag{3.2}$$

$\max(k) = L$: number of layers.

Because of the simple repetitive nature of the network calculations, the output of the network can be expressed as a number of repeated and subsumed calculations for each layer. Equation 3.3 shows the equation for the output of a two-layer network as a function of the input, where the biases have been ignored.

$$y_j^{output} = f_j^{(2)}\left(\sum_i w_{j,i}^{(2)} f_i^{(1)}\left(\sum_i w_{j,i}^{(1)} u_i^{(1)}\right)\right) \tag{3.3}$$

3.2 Back-propagation Calculation for a Neural Network

Back-propagation, also known as the generalised delta rule, is the most commonly used method for modifying the network weights. It is a simple gradient descent method to minimise the sum-squared error of the network output. It is most commonly used to minimise the error between a set of target data and the network output, but the error can be any differentiable function of the network's output.

The back-propagation algorithm is a three part process;

1. The forward calculation of the input data through the network layers and storing of the transfer function derivatives and layer inputs. It is because of the storage of the transfer function derivatives that the training of neural networks requires a large memory capacity. The programming of this is simple as the same actions are repeated for each layer.

2. The back-propagation of the error gradient through the network to the adapted weights.

3. The adjustment of the weights along the gradient to minimise the cost.

The back-propagation process starts from the error calculation. Taking the example of a least squared error optimisation to match a set of reference data r_j, the derivative of the error is taken with respect to the network output y_j:

$$E = \frac{1}{2}\sum_m \left(r_{mj}^{(L)} - y_{mj}^{(L)}\right)^2 \tag{3.4}$$

$$\frac{\partial E}{\partial y_j} = \frac{\partial}{\partial y_j}\left(\frac{1}{2}\sum_m \left(r_{mj} - y_{mj}\right)^2\right) = -\left(r_j - y_j\right) \tag{3.5}$$

The error gradient must then be processed backwards through the network using the chainrule.

3.2.1 Output Layers

The gradient calculation for the output layer is relatively simple; the error gradient is firstly calculated as a function of the transfer function input N, where E is the error function of the network output.

$$\phi_j = \frac{\partial E}{\partial N_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial N_j} \tag{3.6}$$

$$\therefore \phi_j = -(r_j - y_j)f'\left(N_j\right) \tag{3.7}$$

The error gradient must then be made a function of the layer weights.

$$\begin{gathered}
\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial N_j}\frac{\partial N_j}{\partial w_{ji}} \\
\frac{\partial N_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\sum_m w_{jm}u_m \\
\frac{\partial N_j}{\partial w_{ji}} = \frac{\partial w_{mi}}{\partial w_{ji}}\left(\sum_i w_{mi}u_i\right) = u_i \\
\therefore \frac{\partial E}{\partial w_{ji}} = u_i \frac{\partial E}{\partial\left(N_j\right)} = \phi_j u_i
\end{gathered} \tag{3.8}$$

3.2.2 Hidden Layers

The back-propagation of the error to the hidden layer weights is more complex, but still builds on the simplicity of the chain rule process. In order to propagate the error gradient through a layer, it is necessary to calculate the derivative of the layer output with respect to the layer input.

$$\frac{\partial f_j^{(k)}}{\partial u_i^{(k)}} = \sum_m \frac{\partial f_j^{(k)}}{\partial N_m^{(k)}} \frac{\partial N_m^{(k)}}{\partial u_i^{(k)}} \tag{3.9}$$

The derivative of the transfer function input with respect to the layer input is;

$$\begin{aligned} \frac{\partial N_m^{(k)}}{\partial u_i^{(k)}} &= \frac{\partial}{\partial u_j^{(k)}} \left(\sum_{mn} w_{mn}^{(k)} u_n^{(k)} \right) \\ &= \sum_n w_{mn}^{(k)} \delta_{nj}^{(k)} \\ &= w_{mj}^{(k)} \end{aligned} \tag{3.10}$$

and therefore the derivative of the error function with respect to the layer inputs is;

$$\frac{\partial E}{\partial u_j^{(k)}} = \sum_m \frac{\partial E}{\partial N_m^{(k)}} w_{mj}^{(k)} = \sum_m \phi_m^{(k)} w_{mj}^{(k)} \tag{3.11}$$

Therefore, the hidden layer derivative becomes a function of the above for the k+1 layer and the differential of it's own output $y^{(k)}$ with respect to the transfer function input $N^{(k)}$

$$\begin{aligned} \phi_j &= \frac{\partial E}{\partial N_j} = \frac{\partial E}{\partial u_j^{(k+1)}} \frac{\partial y_j^{(k)}}{\partial N_j^{(k)}} \\ \therefore \phi_j &= f'(N_j) \sum_m \phi_m^{(k+1)} w_{mj}^{(k+1)} \end{aligned} \tag{3.12}$$

Therefore, by applying the chain rule the error gradient can be calculated with respect to the weights for any layer.

$$\frac{\partial E}{\partial w_{j,i}^{(k)}} = \sum_{\substack{i_L \\ i_{L-1} \\ i_{L-2} \\ \dots \\ i_{k+1}}} \frac{\partial E}{\partial u_{i_L}^{(L)}} \frac{\partial y_{j_L}^{(L-1)}}{\partial u_{i_{L-1}}^{(L-1)}} \frac{\partial y_{j_{L-1}}^{(L-2)}}{\partial u_{i_{L-2}}^{(L-2)}} \cdots\cdots \frac{\partial y_{j_{k+1}}^{(k)}}{\partial w_{j,i}^{(k)}} \tag{3.13}$$

Having completed the back-propagation, the weights can be adapted to minimise the error, the simplest method for which is a fixed step gradient descent [52], equation 3.14, where η is the learning rate.

$$\Delta w_{j,i}^{(k)} = \eta \phi_j^{(k)} y_i^{(k-1)} \tag{3.14}$$

More complex methods will be explored in later chapters.

3.2.3 Transfer Functions

Transfer functions, also known as activation functions, are perhaps the most important part of the neural network architecture. Each neuron contains one transfer function and typically, all neurons in a layer have the same type of transfer function. This is done mainly to simplify the calculation process. Table 3.1 and Figure 3.2 show the three transfer functions that are generally used with the back-propagation algorithm and are applicable to the work in later chapters. However, any function that is singular in its solution and continuously differentiable can be used. As a general "rule of thumb" the function output should also be positively proportional to the function input.

Table 3.1 - Standard Transfer Function Equations and Derivatives

	Tansig	Logsig	Purelin
Transfer Function	$y=\frac{2}{\left(1+e^{-2N}\right)}-1$	$y=\frac{1}{1+e^{-N}}$	$y=N$
Transfer Function Derivative.	$\frac{\partial y}{\partial N}=1-y^2$	$\frac{\partial y}{\partial N}=y(1-y)$	$\frac{\partial y}{\partial N}=1$

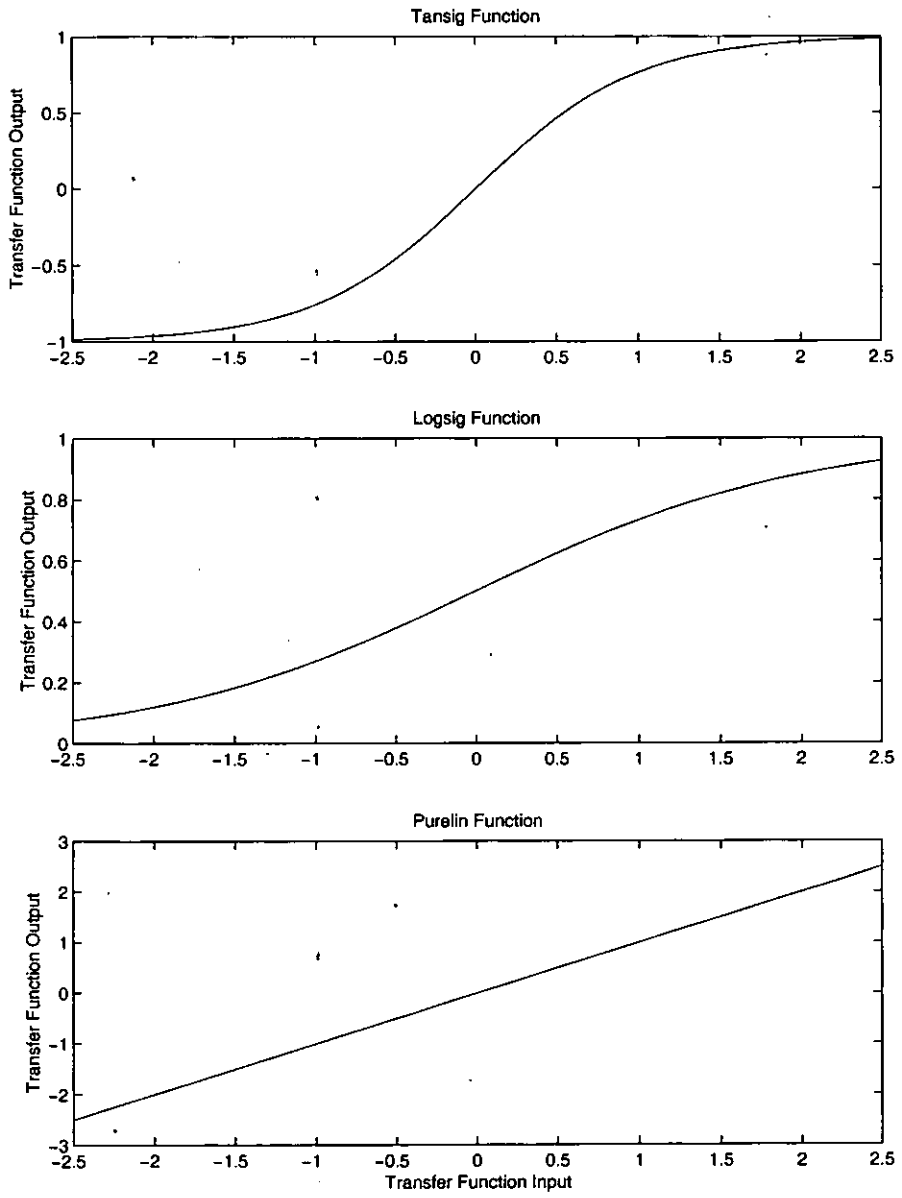


Figure 3.3 - Basic Transfer Function Forms

3.3 Applications of Neural Networks in Control

There are a number of methods for the utilisation of neural networks in control, some of which have already been covered in less detail in Chapter 1. Five of the basic methods for the training and use of neural networks in control are outlined in the remainder of this chapter. Although the techniques have been defined in five separate sections, there are a number of approaches that combine features of several of the procedures.

3.3.1 Direct Inverse Control

Based on the principles of dead beat controllers, and generally used in discrete time, the neural network learns the reverse system, relating the output $y(t+1)$ to the causal input $u(t)$. If the exact inverse is learnt the network can then be fed a required reference trajectory, $r(t+1)$, and the network will generate the control $u(t)$ that will drive the system output $y(t+1)$ to equal the reference signal. Figure 3.4

If the inverse is correct, the system should follow the reference exactly, except for a delay of one discrete sample period. Any time delay in the system makes the implementation of direct inverse based control more complex. The network must be able to predict, or have access to a prediction of, the interim response of the system between input $u(t)$ and $y(t + d)$ where d is the delay [53].

Practical applications [54, 55] for direct inverse based control are limited due to problems of robustness, particularly when system feedback is used. In such cases, system noise and other high frequency disturbances destabilise the system. Other problems occur when the system has multiple input solutions to a given desired response. Direct Inverse controllers can also generate large control requests that may not feasibly be matched by the system actuation.

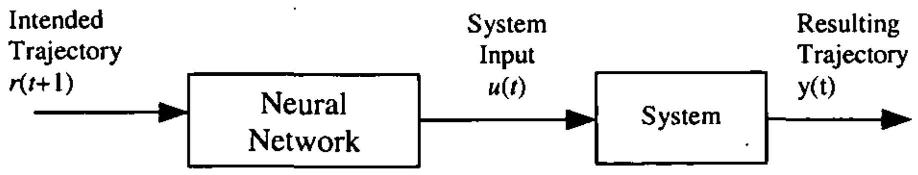


Figure 3.4-Direct inverse based control

3.3.2 Neural Adaptive Control

Where direct inverse control ends and adaptive control begins is not always apparent; there are many similarities between the two, adaptive control in many cases advancing the principles of direct inverse based control. However, the use of neural networks in adaptive control can be two fold; neural networks can be used both to form a reference model, and as the control or regulator system.

There are many papers to advocate the use of neural networks in the field of system identification [27, 28, 56], and because of this, they make an ideal technique for the creation of a reference model. In particular, they are able to capture the non-linearities of the plant. When used for model reference based control, this has the result that only disturbances and time variant changes in the plant are corrected. The controller does not try to force the non-linear plant to adopt the trajectory of a linear or more simplistic model. The network can also be continually updated on-line to improve the accuracy of the system model allowing adaptation to time based variations in the real plant. The control of the plant is then adjusted to ensure the plant follows the trajectory set by the model, this is know as Model-Reference Adaptive Control. This is one of the most common uses for neural networks in control.

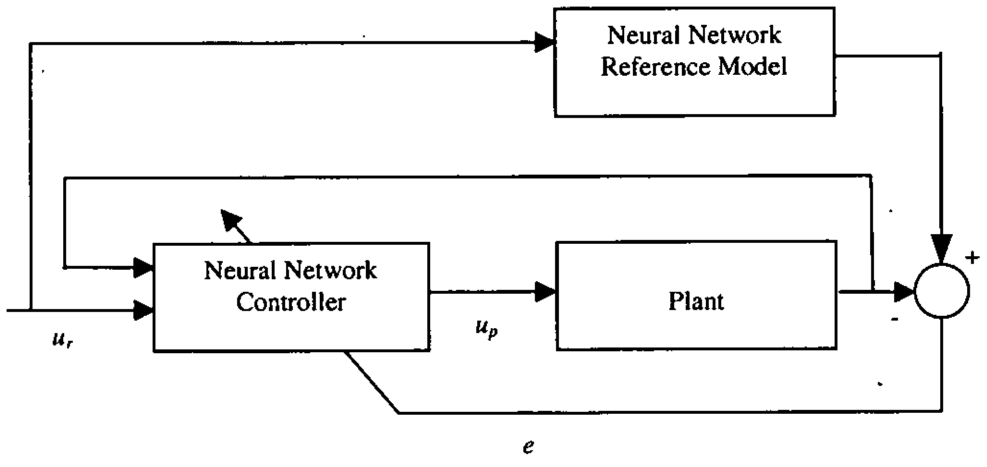


Figure 3.5 - Implementation of Neural Adaptive Control

Neural networks have been advocated for use as the regulator or controller, and a number of algorithms exist for tuning neural networks to give viable control. This allows the regulator to also take account of non-linearities in the plant. It is important, if neural networks are to be applied as a controller, that they are sufficiently trained in advance of their implementation. Insufficient pre-training can lead to a lack of robustness, resulting in poor and sometimes catastrophic control actions. The application of adaptive neural network controllers to time varying systems is considered by some to lack robustness in practice, the risk of instability is certainly heightened [57]. This is partially true when using multi-layer perceptron networks. They are best applied to simple repetitive tasks where the network's performance can be honed for a particular control scenario such as the control of robot manipulators on a production line.

The resulting control is not necessarily optimal, the strategy being based around obtaining similar behaviour from the plant as the reference model. The reference model may be controlled by a strategy that is optimal for the reference model, but not necessarily optimal for the plant.

3.3.3 Back-propagation of Utility

This approach compares well with the work described by Bryson and Ho, and the discrete time method applied by Marsh. The back-propagation of the utility function finds the optimal set of neural network control weights. However, it can also be applied to find the optimal schedule of control actions. Either method can be implemented using neural networks. Back-propagation through time [58] is one of the most publicised methods for the utilisation of neural networks within a non-linear control strategy. The most commonly cited application of this technique is Nguyen and Widrow's Truck Backer-up [59] in which a 26 neuron / 2 layer feed forward adaline is used to dock a computer simulated articulated vehicle. The inputs to the network in this example were the x, y position of the dock, trailer and cab, and the angle of the tractor and trailer unit. The network output is the steering angle of the front wheels.

The training process, performed using back-propagation through time, takes the following format: Firstly, a neural network is trained to emulate the non-linear dynamics of the plant and thus the plant derivatives can now be calculated from the neural network. The control process is considered as a sequence of steps. For the above truck example, a step constituted the selection of a steering angle and the reversing of the tractor unit a fixed distance equivalent to 1m, to a new state position. The steps are continued until either the goal is reached, or the plant exceeds some restriction (e.g. the truck jack-knifes), or a predetermined maximum number of steps is exceeded. Each step is considered as a separate network consisting of the controller and the emulator. Each network forms part of a much larger network that provides control for the whole sequence. Signals are passed from layer set to layer set. By back-propagating the error through the composite network (back-propagation through time), changes to the controller weights at each step can be calculated. These are averaged and the net change applied to the controller weights.

Over a number of initial condition runs, the controller will become sufficiently generalised as to provide satisfactory control even for initial conditions that have not yet been experienced. Many examples of the truck backer-up can be found in simulation on the web, for example;- http://www.handshake.de/user/blickle/Truck

Although very successful in many applications, particularly theoretical, back-propagation of utility cannot efficiently account for noise, as it cannot be included in the system emulator. With larger problems, it also becomes difficult to implement in real time. However, real time learning is not always appropriate or worthwhile. Hampo and Marko [37] also identify further problems in relation to active suspension: the method relies on un-measurable states in the development of a realistic system emulator. They also found the development and validation of the system emulator time consuming.

3.3.4 Adaptive Critic

Adaptive Critics combine the theories of dynamic programming, reinforcement learning, and back-propagation (section 3.2, not 3.3.3) to create action networks that control the plant. Dynamic Programming is based on the statement that for any given optimal trajectory passing through an intermediate point, the optimal trajectory from that point onwards will be the same should the system be restarted at the intermediate point. This is combined with the theories behind reinforcement learning, where the probability of repetition of an action is governed by a successful result having occurred previously when that action was applied. The critic function adapts the action network weights, using back-propagation, to minimise the system performance cost. Therefore, over a number of successive simulations it improves the controls applied by the action network. Although adaptive critics have proven successful in a number of fields [60-62], they are still in their infancy and little work has been done in relation to the field of this PhD. It has also proven difficult to establish a stable controller from which to evolve the optimal control strategy when working on plant dynamics similar to those of the quarter vehicle.

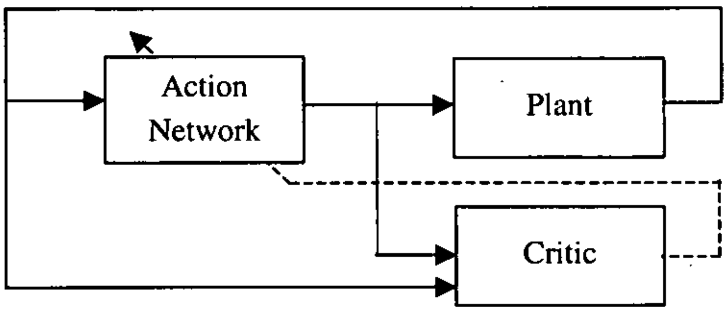


Figure 3.6 - Adaptive Critic Implementation

3.3.5 Supervised Control

Here the neural network is used to map the systems' sensor output to the control actions of a more complex controller [63, 64], or those of a human operator [65, 66]. This process can be used to apply control techniques that are too complex to be implemented in real time but where the resulting control is on a one to one basis, and therefore easily learnt by the neural network. Equally, this technique can be used to reproduce the actions of a human operator, where the network is used to find a relationship between the sensor measurements and the operators' actions. Therefore, the operator is taken out of the control process.

This is one of the simplest routes to using neural networks in control. For this reason it will be used to validate their use with active suspension. In Chapter 4, supervised learning is used to replicate the actions of the control strategies developed by Marsh.

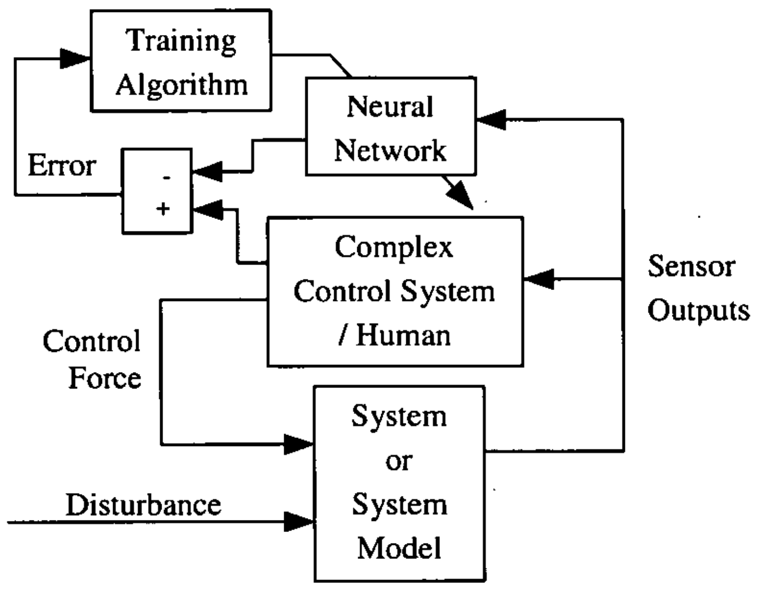


Figure 3.7 - Supervised Learning Process

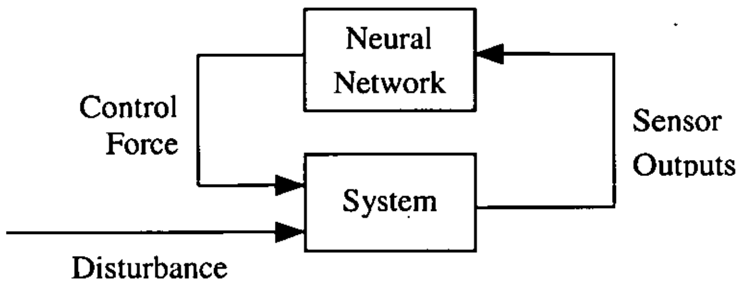


Figure 3.8 - The Trained Network in Operation

Chapter 4 Neural Network Training

Chapter 4 Neural Network Training

In Chapter 2 a method for the calculation of the optimal non-linear control force was introduced. However, the complexity of the method means it cannot feasibly be used on-line. In this chapter, the golden line search and Levenberg Marquardt optimisation methodologies will be assessed for the training of neural networks through supervised learning to learn non-linear input - output relationships, including the state - force relationship generated using the optimal non-linear control method introduced in Chapter 2. In Chapter 5 a number of enhancements to the training process will be made and the performance of neural networks in the control of a quarter vehicle model will be compared with the non-linear optimal open loop control and LQR control introduced in Chapter 2.

4.1 Supervised Learning Process

In order to train a neural network using a combination of supervised learning and least squared error criterion, a training data set must first be created. This consists of a set of inputs (the system states for the non-linear control problem) and the corresponding outputs (the control forces) to cover the complete input range under which the neural network will be expected to operate. To create the control force data set the Fortran code developed by Marsh, is used to optimise the control force for the quarter vehicle, applying the following non-quadratic dynamic cost function.

$$L = 0.5\left(\alpha_1 x_1^2 + \beta_1 x_2^2 + \alpha_2 x_1^{n_1} + \beta_2 x_2^{n_2} + \left(\frac{u}{m_b}\right)^2\right) \tag{4.1}$$

$$\alpha_1 = 16000 \quad \beta_1 = 500 \quad \alpha_2 = 2\times10^{12} \quad \beta_2 = 5\times10^{11} \quad n_1 = 6 \quad n_2 = 10$$

The training data set was created by finding the optimal control force for 11^4 = 14641 initial condition points evenly distributed within a rectangular region of the four dimensioned state space. This equates to 11 points on each state axis.

Having created the training data set, the neural network is optimised using least squared errors to minimise the error between the neural network output and the optimal force, calculated using the discrete control method, Figure 4.1. The neural network can then be implemented in place of the LQR gain matrix in the linearly controlled quarter vehicle model, Figure 4.2.

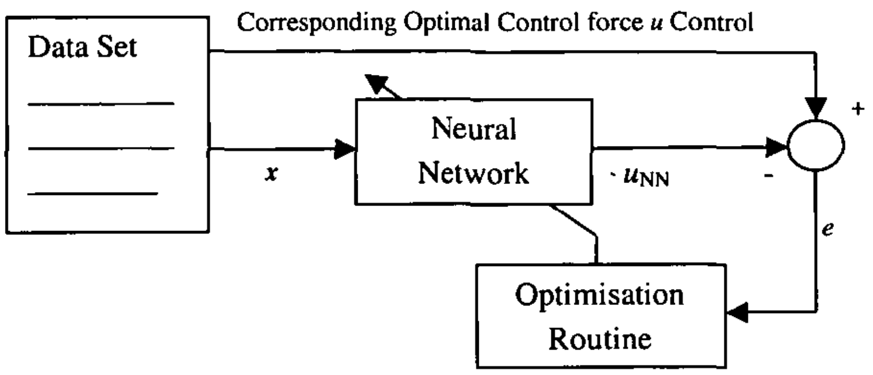


Figure 4.1 - Training the Neural Network using Least Squared Error Methods.

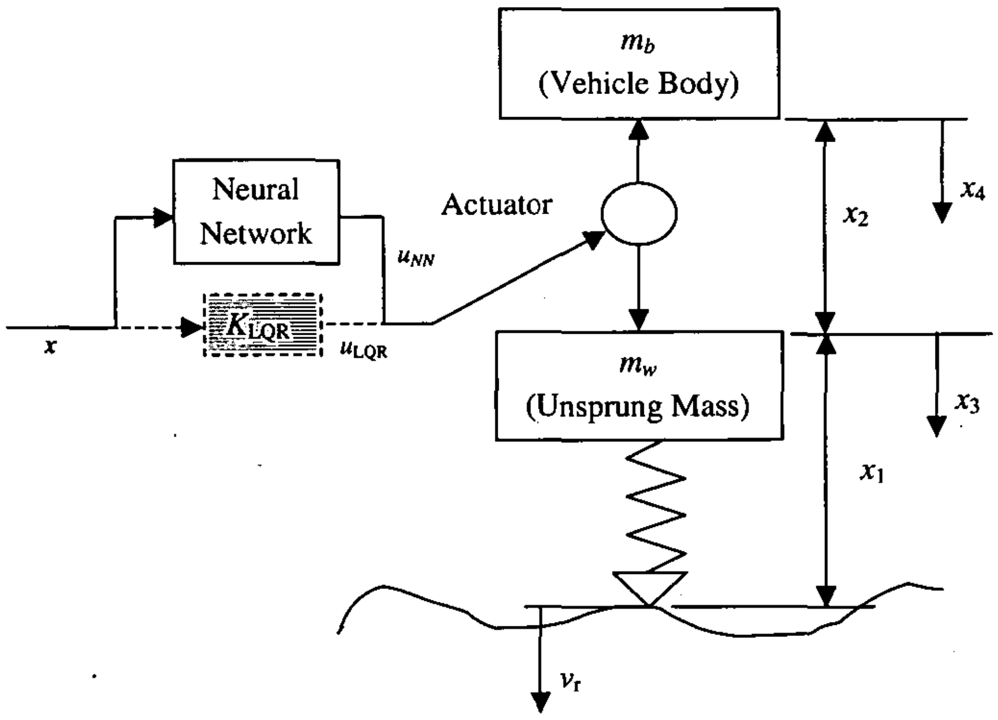


Figure 4.2 - Implementation of the Neural Network Controller in the Control of the Quarter Vehicle Model.

4.2 Training Routines

The terms "training" or "learning", when used in relation to neural networks, refer to the optimisation of the network weights to minimise the error criterion. In the case of feed forward MLPs, this can be carried out using gradient descent through the employment of the back-propagation algorithm. However, pure gradient descent is known to be poor [67], being both slow and potentially inaccurate depending on the step size chosen. Because of this, a number of enhancements to the basic gradient descent algorithm have evolved. The Golden Section line search method is a simple evolution of the gradient descent algorithm, ensuring that an optimal step size is taken within the line search. The Levenberg Marquardt method is more advanced employing both first and second derivative cost surface information to improve the optimisation process. The Golden Section line search method is described in detail in Numerical Recipes with example coding for both Fortran© and C© [67]. The Levenberg Marquardt method is described below and a comparison of their performance for the optimisation of the neural network weights is made.

4.3 Levenberg Marquardt

The Levenberg Marquardt optimisation method is designed for the optimisation of non-linear functions and based around the principles of steepest descent and the Inverse Hessian method, varying smoothly between the two.

4.3.1 Inverse Hessian

The Inverse Hessian method forms a quadratic approximation to the error surface around the current function parameters, based on the current cost value and the first and second derivatives of the error surface. The surface of a smooth continuous function *f*(*x*) can be expressed around the point *x* using the standard Taylor series approximation:-

$$f(x+a) \approx f(x) + f'(x).a + \frac{1}{2!}f''(x).a^2 + \ldots + \frac{1}{n!}f^{(n)}(x).a^n \qquad (4.2)$$

where *f'(x)* is the gradient vector at *x* and *f''(x)* the matrix of second derivatives (Hessian matrix)

$$f''(x) = H_{LM} = \frac{\partial^2 f}{\partial x_i \partial x_j} \tag{4.3}$$

Although the Taylor series can be calculated to the nth derivative, it is unusual for the purposes of optimisation to go beyond the second derivative due to the computational expense involved. With the Taylor series calculated to the second derivative, the derivative of the series can easily be solved to locate the approximate minimum.

Assuming the function to be minimised is of a least squares form, the square of a single function *y*, the aim being to minimise y^2.

$$f(x) = \frac{1}{2}\sum_{i=1}^{m} y_i(x)^T y_i(x) \tag{4.4}$$

for which the Hessian matrix is composed of two parts, the square of the Jacobian matrices

$$\frac{\partial y}{\partial x}^T \frac{\partial y}{\partial x} \tag{4.5}$$

and

$$S(x) = y(x).y''(x) \tag{4.6}$$

the Taylor Series becomes

$$\begin{aligned} f(x+a) &\approx f(x) + f'(x).a + \frac{1}{2}a^T f''(x).a \\ &\approx \frac{1}{2}\sum_{ij} y_j(x)y_i(x) + y_j(x)y_i'(x)a_i + \frac{1}{2}a_j\left(y_j'(x)y_i'(x) + S_{ij}(x)\right)a_i \end{aligned} \tag{4.7}$$

The second order form of the Taylor series can be differentiated in terms of *a* to find the local minimum.

$$\begin{aligned} 0 &= \frac{\partial f(x+a)}{\partial a} = \sum_{ij} y_j(x)y_i'(x) + \left(y_j'(x)y_i'(x) + S_{ij}(x)\right)a_i \\ a &= \sum_{ij} -\left(y_j'(x)y_i'(x) + S_{ij}(x)\right)^{-1} y_j(x)y_i'(x) \end{aligned} \tag{4.8}$$

Certain assumptions about the form of the Hessian matrix are commonly made in order to make its calculation less computationally challenging when applied to a least squares problem. The use of both parts of the Hessian matrix is known as the Full Newton Approach. However, *y″(x)* is usually either incalculable or inconvenient to obtain, and can be expensive to obtain using finite difference methods.

It can be shown that when the optimal solution has zero or minimal residual error, $S(x)$ will be small and can therefore be ignored, where as optimal solutions with large residual errors will result in large values of $S(x)$, and therefore the calculation of $S(x)$ cannot be ignored. This is best-done using the secant method [68].

4.3.2 Steepest Descent

The Levenberg Marquardt method avoids errors in the second order approximation by varying between the Inverse Hessian method and gradient descent depending on the accuracy of the Hessian matrix to the true surface. The selection between the two is made by varying a scalar parameter λ, depending on whether a cost reduction has occurred or not, due to the preceding step calculation. When λ is large, the Levenberg Marquardt search direction tends to that of steepest descent, whilst for small values tends to that of the Inverse Hessian matrix method / Gauss-Newton [69].

The Inverse Hessian Step takes the form

$$\lambda 0.5 H_{LM}.a = y\frac{\partial y}{\partial x} \tag{4.9}$$

While the Steepest Descent step is

$$a = \lambda y \frac{\partial y}{\partial x} \tag{4.10}$$

and therefore the Marquardt formula for varying between the two is,

$$\begin{aligned} H'_{LM\,jj} &\equiv H_{LM\,jj}(1+\lambda) \\ H'_{LM\,jk} &\equiv H_{LM\,jk} \quad (j \neq k) \end{aligned} \tag{4.11}$$

$$H'_{LM}a = y\frac{\partial y}{\partial x} \tag{4.12}$$

There have been a number of methods put forwards for adaptation of λ, the simplest of which takes the following format:

- Select an initial set of parameters x for your function f.
- Compute the current cost at x
- Pick a modest value for λ, say $\lambda = 0.001$
- Solve the Marquardt step equation for a and calculate the cost of the new parameter values $x + a$
- If the cost at $x + a$ is greater than, or equal to, the cost at x, then increase λ by a factor of 10 and repeat the above step again.
- If the cost at $x + a$ is less than at x, then λ can be reduced by a factor of 10, and the parameter values updated, $x = x + a$, and a new Marquardt step calculated.

4.4 Training Results

Four supervised learning cases, of varying complexity, were used to test the performance of the optimisation routines.

1. $y = \cos\theta$, $-2\pi < \theta < 2\pi$. Trained for 15 minutes on 1000 data points, Neural network structure: input layer - 10 tansig functions, output layer - 1 purelin.

2. $y = \cos\theta \cos\phi$, $-\pi < \theta < \pi$, $-\pi < \phi < \pi$. Trained for 15 minutes on 1000 data points, Neural network structure: input layer - 10 tansig functions, output layer - 1 purelin.

3. $y = 1000.\cos\theta \cos\phi$ $-\pi < \theta < \pi$, $-\pi < \phi < \pi$. Trained for 15 minutes on 1000 data points, Neural network structure: input layer - 10 tansig functions, output layer - 1 purelin.

4. The non-linear optimal state - force relationship for a quarter vehicle model (Section 4.1) for 1 hour. , Neural network structure: input layer - 20 tansig functions, output layer - 1 purelin.

The results in Table 4.1 and Figure 4.3 are not definitive, but can be considered typical of the optimisation performance for each case considered. The success of the optimisation process is to a degree dependant on the initial network weights, (see Figure 4.4), even when chosen at random. Some optimisation runs can stop prematurely in comparison to the results shown in Table 4.1. The Levenberg Marquardt technique performs significantly better than the golden section line search routine in all four cases; the utilisation of second derivative cost surface information gives it a distinct advantage. The Levenberg Marquardt technique is also more tolerant of initial conditions as it is able to achieve reasonable results from almost purely random initial weighting values. The line search technique needs more careful selection of the weights, particularly in the input and output layer when the training data has not been normalised to fall in the range -1 to $+1$, such as cases 3 and 4. The initial weights need to be selected carefully to ensure that the Tansig functions will operate in their active range. Saturation limits the opportunity for cost reduction; once saturated, the gradient values become small.

With any form of line search, this results in the change in the related weights being very limited as the step size is dependant on more significant directions, for which if a large step were taken, the cost would almost certainly be negatively affected.

Now that current computing technology is able to accommodate the memory and performance requirements needed to use the Levenberg Marquardt technique with neural networks, there is clear evidence that it is appropriate to do so. Line search techniques have only limited value when dealing with off-line learning problems, however their use is still advocated for on-line learning [70]. The benefits in speed and flexibility far outweigh any limitations the Levenberg Marquardt technique may have for the quarter vehicle problem. From this point on, the Levenberg Marquardt technique will be the predominant optimisation technique used.

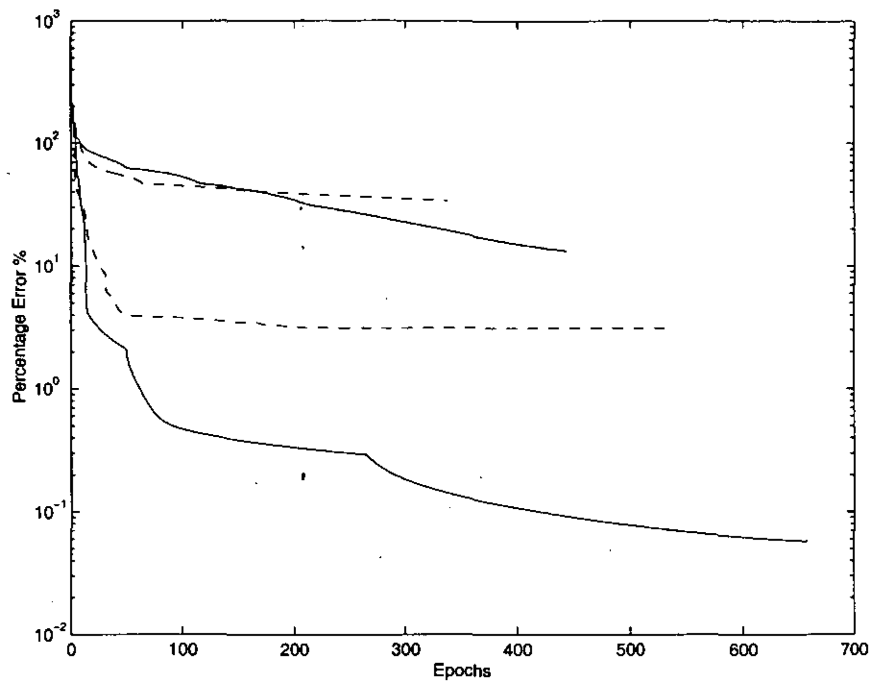
Table 4.1 - Golden Section versus Levenberg Marquardt Optimisation

CASE	Optimisation Routine	Weights Initial Condition	Optimisation Cost E_s	Percentage Error	# of Saturated Neurons / # of Tansig Neurons
1	Golden Section	Uniform, Random	8.60×10^{-3}	13.19 %	0
	Levenberg Marquardt	Uniform, Random	1.67×10^{-7}	0.057 %	0
2	Golden Section	Uniform, Random	8.30×10^{-3}	26.14 %	0
	Levenberg Marquardt	Uniform, Random	1.29×10^{-5}	1.03 %	0
3	Golden Section	Uniform, Random	86985	86.52 %	0
	Golden Section	Weighted, Random	3724	16.91%	0
	Levenberg Marquardt	Uniform, Random	8194	26.05 %	4 / 10
	Levenberg Marquardt	Weighted, Random	1.383	0.34 %	0
4	Golden Section	Uniform, Random	0.6905	73.50 %	15 / 20
	Golden Section	Weighted, Random	0.4798	61.27 %	0
	Levenberg Marquardt	Uniform, Random	0.1043	28.56 %	19 / 20
	Levenberg Marquardt	Weighted, Random	4.73×10^{-3}	6.09 %	0

Uniform, Random: The initial network weights were uniformly distributed in the range –1 to +1.

Weighted, Random: The initial input layer weights mapped the maximum inputs to the active range of the input layer transfer function, the output layer weights reflected the output range

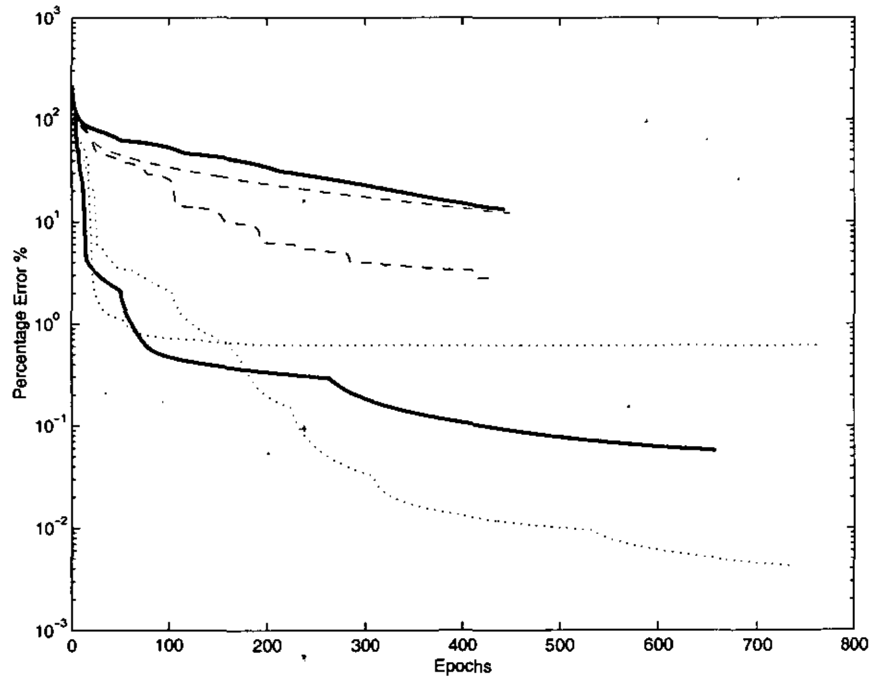
Saturated: When the typical input range does not result in the majority of neuron outputs falling in the range -0.999 to 0.999.



—Case 1 Golden Section, —Case1 Levenberg Marquardt,

- - Case 2 Golden Section, - - Case 2 Levenberg Marquardt

Figure 4.3 - Optimisation of Cases 1 & 2



— Golden Section (table 4.1), — Levenberg Marquardt (table 4.1),

- - - Other Golden Section optimisations,··· Other Levenberg Marquardt optimisations

Figure 4.4 - Comparison of Optimisations Performed for Case 3

CHAPTER 5 ANALYSIS OF NEURAL NETWORK CONTROLLED ACTIVE SUSPENSION

Chapter 5 Analysis of Neural Network Controlled Active Suspension

Following the results of Chapter 4, two neural networks were trained successfully with respect to the cost function, using the Levenberg Marquardt optimisation technique and supervised learning. The first network, NNF (Neural Network Force), learned the relationship between the quarter vehicle states and the control force. The second network, NNC (Neural Network Costate), learnt the relationship between the quarter vehicle states and the costate values, from which the control force can be calculated, Equation 2.24.

In Chapter 5, the performance of these networks will be analysed in the dynamic control environment of a quarter vehicle model. Their performance will be analysed for a number of scenarios, including initial condition settling problems and dynamic simulation over a selection of road models. The initial condition evaluations highlight three problems: The influence of network bias values on the network output when the system states are zero, over-fitting causing poor generalisation between training data points, and large amplitude values dominating the optimisation and preventing small value accuracy. The bias effect can be resolved using bias cancelling, a technique successfully implemented by Hiroki Nakanishi. [71] The corrected networks give zero output for zero states, without degrading the overall performance. Weight decay is used to prevent neuron transfer function saturation and over-fitting of the training data. Cost weighting on the basis of state magnitude is used to reduce the influence of large values, and minimise small value error. The neural network controllers' performance is compared with the Linear Quadratic Regulator introduced in Chapter 2. The state cost matrix is redefined so as to make the performance of the Linear Controller more comparable.

5.1 Equivalent LQR Controller

The non-quadratic cost function has a quadratic component, Equation 2.22, this could be used as the cost function to develop the linear controller for comparison. However, this controller would be inherently less constrained in suspension and tyre deflection control, in order to make it a suitable basis for the addition of non-linear control and the non-quadratic cost elements involved. Having established the non-linear control process, and in an awareness of its likely operating range, it is possible to define an equivalent LQR control. The non-linear control approach, if correct, should still be superior but the LQR control has been redefined to give the closest possible equivalent linear system. The Q matrix tuned in LQR design is now redefined using the following generalised process.

- Using the training range of the neural networks, calculate the non-quadratic cost function values along the state axis for tyre and suspension deflection. Figure 5.1.
- Using linear regression, establish the LQR Q matrix terms that give the best approximation to the non-quadratic cost for each axis. Table 5.1.

Applying the above process the following values were established for the redefinition of the Q matrix, Q_e resulting in the quadratic state, cost relationship, Figure 5.1.

Table 5.1 - Equivalent Q matrix

State	Assumed Operating Range	The Quadratic values used in Equation 2.22	Equivalent Q Matrix
Tyre Deflection (x_1)	+/- 0.0282m	$Q_{(1,1)} = 16000$	$Q_{e(1,1)} = 7.09\times10^5$
Suspension Deflection (x_2)	+/- 0.1095m	$Q_{(2,2)} = 500$	$Q_{e(2,2)} = 4472$

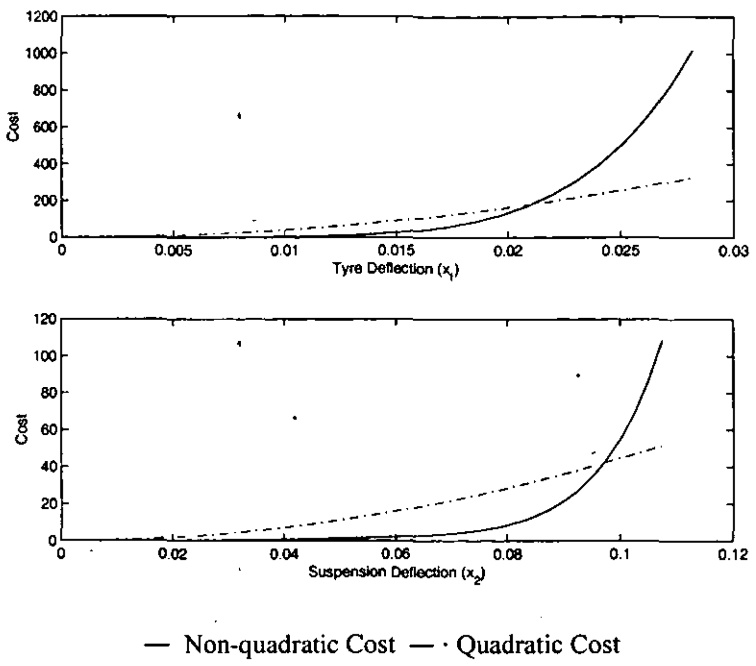


Figure 5.1 - Equivalent Linear Cost Function

The performance of the neural network non-linear controllers will be analysed on two initial condition problems, the first designed to promote the wheel hop mode, x = [0 0 2.5 0]^T, (Figure 5.2), and the second, the body bounce mode, x = [0 0 0 1.25]^T, (Figure 5.3). Firstly, a linear controller based on the quadratic components of Equation 2.22 which will be refer to as the original controller, the new equivalent linear controller and the open loop non-linear control trajectories are compared to assess the changes made by redefining the Q matrix as shown in Figures 5.2 & 5.3

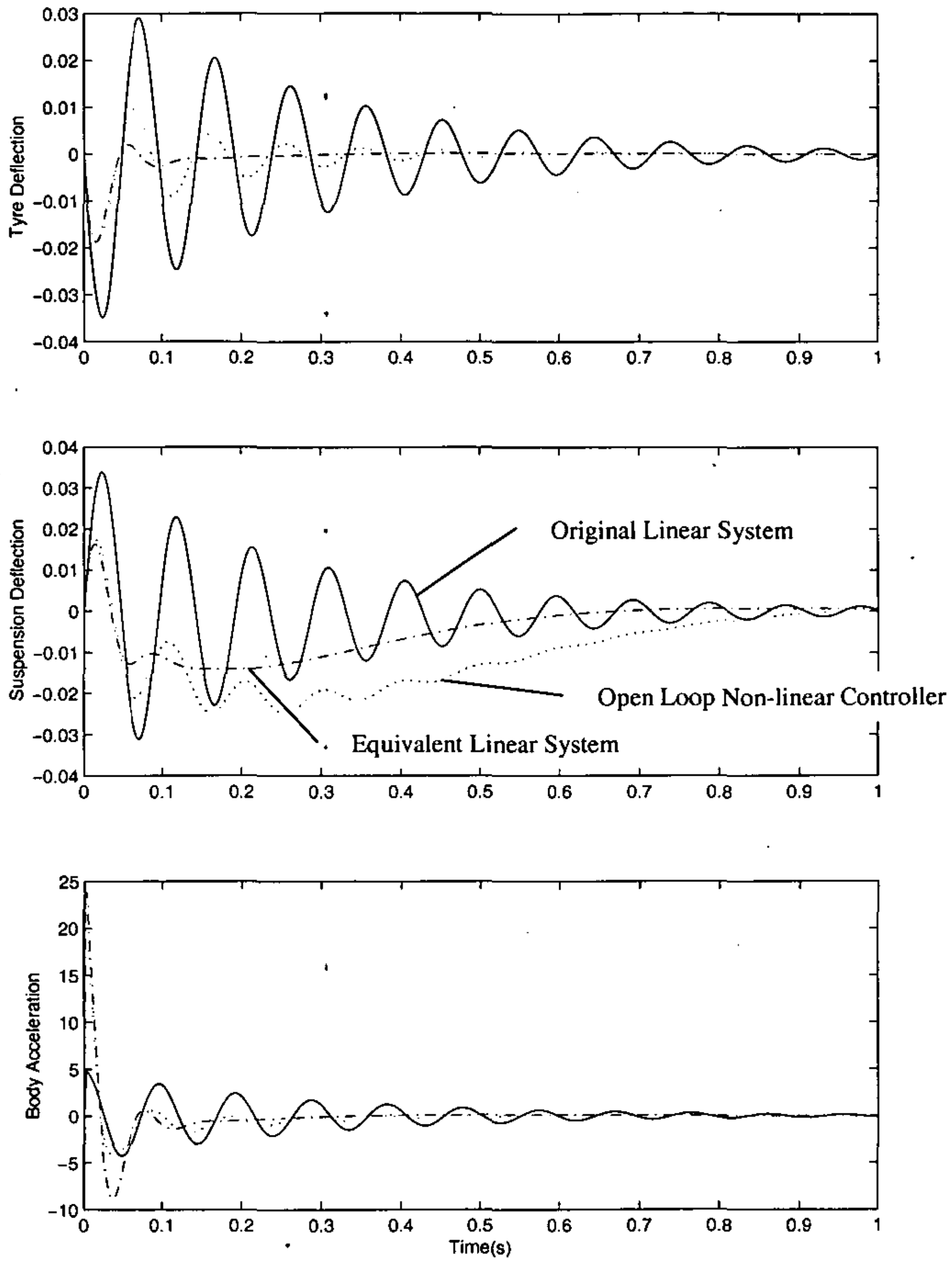
The equivalent linear system results are, by comparison, much closer to those of the non-linear open loop controller used for training and validation than the original linear system, (Tables 5.2 & 5.3). Body acceleration has been sacrificed to achieve a significant improvement in the control of tyre and suspension deflection. The resulting initial condition state trajectories fall in a crucial region for the non-linear controller. Therefore the benefits of non-linear control are less clearly defined by Table 5.2. alone. Careful examination of Figures 5.2 & 5.3 reveal better utilisation of the suspension workspace in order to better control body acceleration. For example the second peak in body acceleration following the initial hub velocity is significantly lower than that of the equivalent linear controller. Following the initial body velocity case, tyre and suspension deflections are better controlled without any increase in the peak levels of body acceleration experienced.

Table 5.2 - Equivalent Linear System, Initial Hub Velocity Response

Hub Velocity Response	R.M.S Values for First Second			Peak Tyre Deflection (mm)
	Tyre Deformation (mm)	Suspension Deflection (mm)	Body Acceleration (m/s^2)	
Original Linear Controller	**9.92**	10.02	1.30	**34.92**
Equivalent Linear Controller	2.72	7.63	**2.46**	18.70
Non-linear Open Loop Control	3.75	**13.65**	2.45	19.58

Table 5.3 - Equivalent Linear System, Initial Body Velocity Response

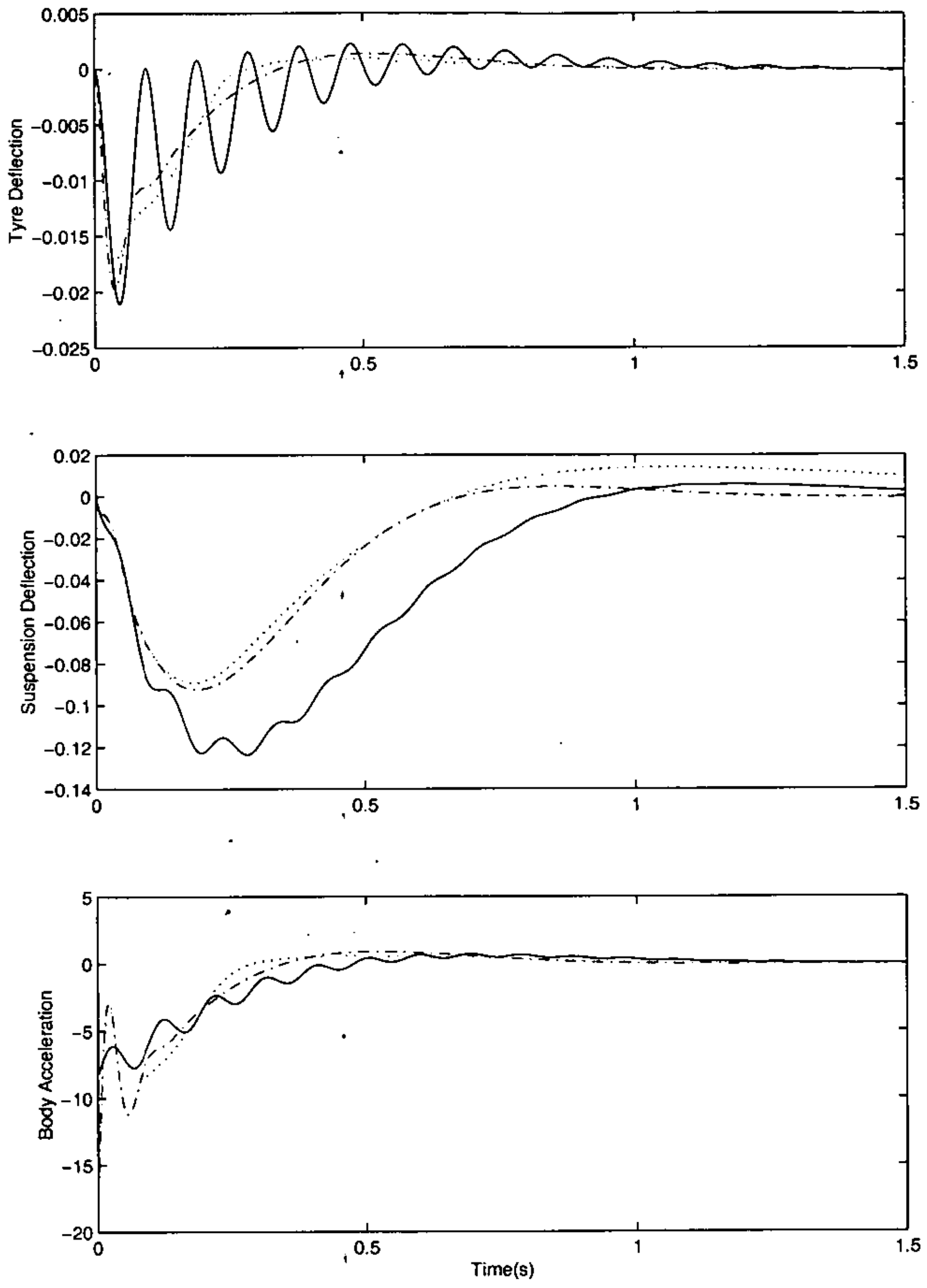
Body Velocity Response	R.M.S Values for First 1.5 Seconds			Peak Suspension Deflection (mm)
	Tyre Deformation (mm)	Suspension Deflection (mm)	Body Acceleration (m/s^2)	
Original Linear Controller	**4.33**	**59.20**	2.27	**123.61**
Equivalent Linear Controller	4.18	38.52	2.67	92.40
Non-linear Open Loop Control	4.24	37.5	**2.95**	89.24



— Original Linear System, — · — Equivalent Linear System,

· · Open Loop Non-linear Control.

Figure 5.2 - Equivalent Linear System, Hub Velocity Response



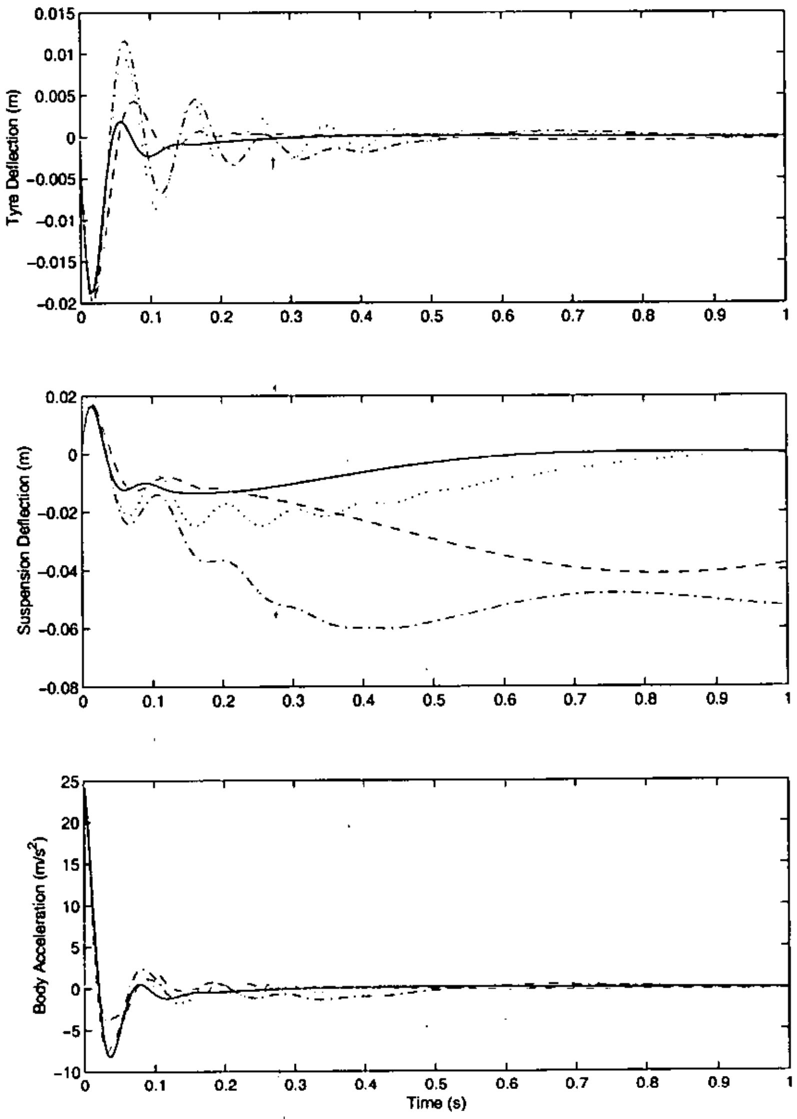
— Original Linear System, — · Equivalent Linear System,

· · Open Loop Non-linear Control.

Figure 5.3 - Equivalent Linear System, Body Velocity Response

5.2 Initial Condition Performance

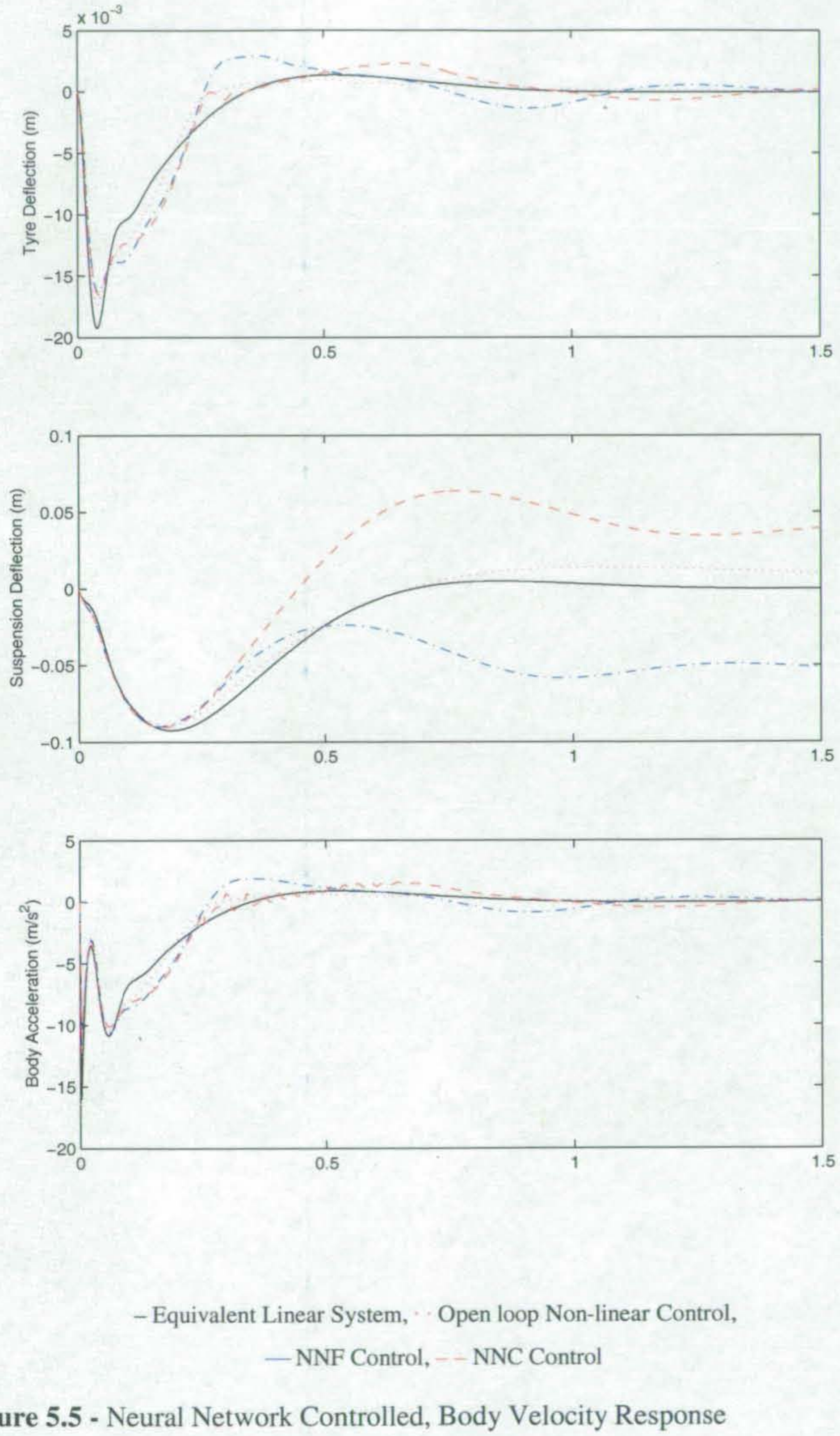
The performance of the neural network non-linear controllers will be analysed on two initial condition problems, the first designed to promote wheel hop, Figure 5.4, and the second, body bounce, Figure 5.5.



- Equivalent Linear System, · · Open loop Non-linear Control,

— NNF Control, – – NNC Control

Figure 5.4 - Neural Network Controlled, Hub Velocity Response



– Equivalent Linear System, ··· Open loop Non-linear Control,

— NNF Control, – – NNC Control

Figure 5.5 - Neural Network Controlled, Body Velocity Response

Initial condition runs, performed with the original networks trained following Chapter 4 highlight a number of problems with both the network architecture, and the optimisation process. The major failing of the original neural network controllers is very apparent for both networks; neither network stabilises the suspension to the intended equilibrium position. This is caused by two main factors. Firstly, both networks generate a non-zero output when at the equilibrium position. Secondly, the training data is evenly distributed across the operational range, which results in the high value points dominating the optimisation so that relatively large errors occur around the origin, where the training values are small.

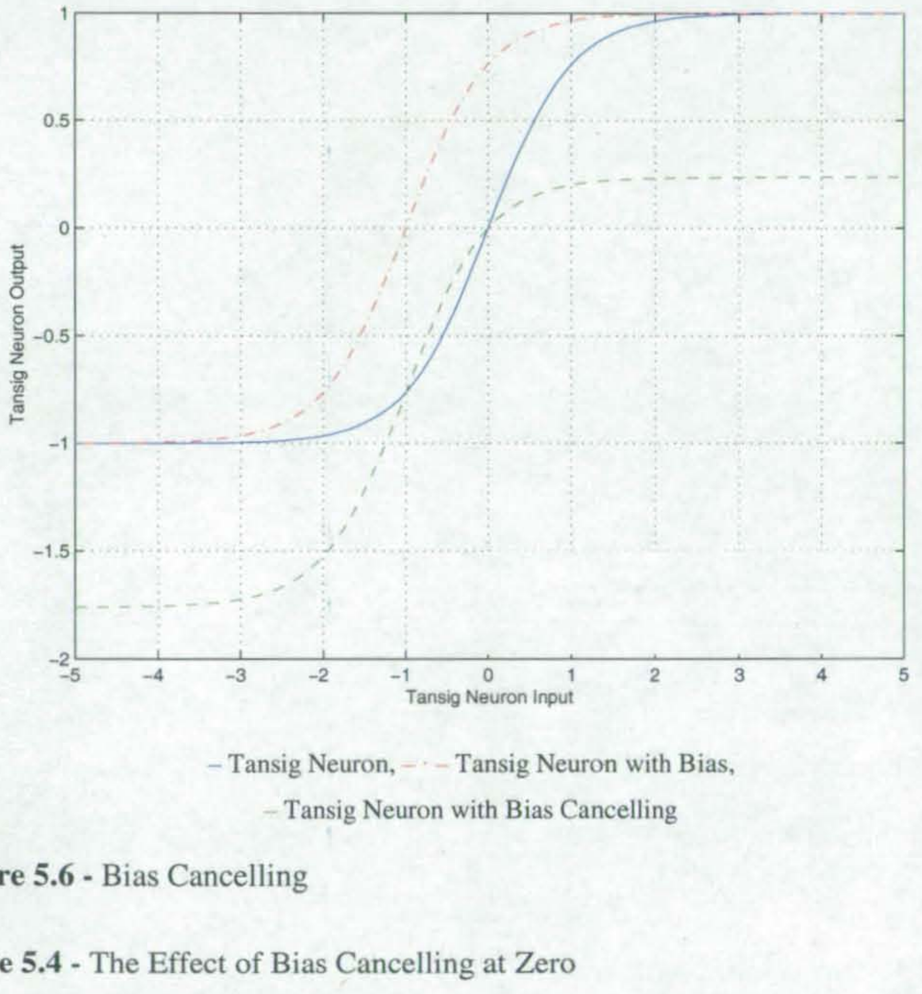
A second failing, which is less noticeable, is that the force surfaces are not as smooth as they should be. This is indicated by the body acceleration response shown in Figure 5.5, in particular the response produced by the NNC controller. This disrupts the settling process and causes unexpected body accelerations. This can be observed for the NNC controller, 0.25seconds after the initial condition settling process starts.

5.3 Bias Cancelling

In many control problems, the force feedback or costate values for zero states are also zero. Because of the bias terms, zero out for zero in, cannot be guaranteed for feed forward neural networks of the standard form. Nakanishi *et al* [71] propose the following solution, which ensures the network output is zero for a zero input, whilst maintaining the functionality of the bias values to add additional non-linear flexibility to the network. The layer calculation equation 3.1, is redefined as follows

$$y_j^{(k)} = f_j^{(k)}\left(\sum_i w_{j,i}^{(k)} u_i^{(k)} + \mathscr{B}_j^{(k)}\right) \xrightarrow{redefine} \begin{array}{c} f_j^{(k)}\left(N_j^{(k)}\right) - f_j^{(k)}\left(\mathscr{B}_j^{(k)}\right) \\ N_j^{(k)} = \sum_i w_{j,i}^{(k)} u_i^{(k)} + \mathscr{B}_j^{(k)} \end{array} \quad (5.1)$$

This does not affect the ability to differentiate the layer with respect to either the weights, or the bias terms and therefore allows the continued use of back propagation based optimisation techniques. Figure 5.6 shows the effect of bias cancelling on a Tansig neuron with original weighting and bias values of one. The influence of the bias term is different. Originally the bias term applied a lateral shift to the output, but now the shift is two-dimensional in order to maintain the "0 ⇒ 0" condition. While the technique for some problems may affect the number of neurons required to accurately learn a relationship, the premise that with sufficient neurons the behaviour of any non-linear system having a zero input, zero output relationship can be replicated remains true. With bias cancelling now applied, Table 5.4 shows the network output is now correctly zero for both the NNF and the NNC.



– Tansig Neuron, – · – Tansig Neuron with Bias,
– Tansig Neuron with Bias Cancelling

Figure 5.6 - Bias Cancelling

Table 5.4 - The Effect of Bias Cancelling at Zero

Network Input	Network Output			
	NNF	NNC	NNF with Bias Cancelling	NNC with Bias Cancelling
[0 0 0 0]	-37.591	[-14.67 –0.31 –0.07 –0.70]	[0]	[0 0 0 0]

5.4 Generalisation / Regularisation

One of the most common problems with neural networks is that of over-fitting, or generalisation. Because neural networks are normally trained on a fixed data set, the ideal result is for the network to interpolate smoothly between the data points provided for training. However, without careful control, over-fitting will occur and the network will give inaccurate results for points it hasn't been trained for. This is illustrated in Figure 5.7 below.

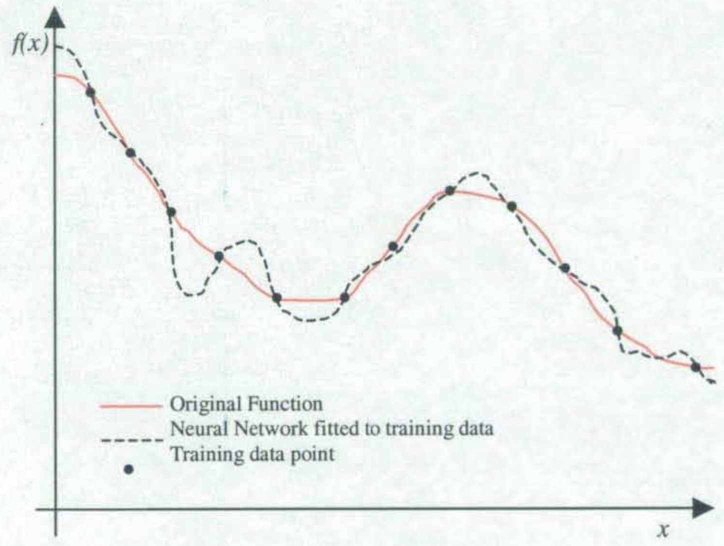


Figure 5.7 - Neural Network Over-Fitting

There are a number of methods for the prevention or control of this. The first is to use a validation data set against which the neural network output is checked. Initially both the training data error and the validation data set error should reduce simultaneously. However, as the network begins to over-fit the training data, the validation error will increase. This can work well but does nothing to actively encourage the smoothing of the network during training. It can result in the network training stopping prematurely unless sufficient leeway is given to allow variations in the validation training set error as the optimisation proceeds.

5.4.1 Regularisation

Regularisation is an alternative, or additional, method to encourage smoothing, which incorporates in the cost function a term that penalises weight complexity. It is commonly used in multi-layer perceptron networks to reduce the possibility of over-fitting [72]. Typically, it is used to make an appropriate trade off between the reliability of the training data and that of the model created. This can be done by creating a trade-off function between data fit and network output smoothness.

$$E(x,w) = E_s(x,w) + \Psi . E_c(w) \tag{5.2}$$

The first term, E_s, is the standard performance function (e.g Equ.3.4), typically in the form of a least squared error function. The second term is a measure of the complexity of the network, which is dependent on the network model only. Its inclusion will attempt to impose prior knowledge of the problem and its solution, whether that is about the symmetry of the solution, or simply some measure of smoothness. The form of the above equation follows that of Tikhonov's regularisation theory [72, 73]; where the regularisation parameter Ψ controls the relative importance of the complexity function with respect to the standard performance function. When Ψ is small the training is almost entirely limited by the standard performance function and the choice of weights is unconstrained. When Ψ is large, the form of the network will be almost totally defined by the complexity function. In effect, little value is put on the standard training function. In the case of a data-fitting problem, this would imply that the original data was considered unreliable. In practice, a value that is the region of near equality for the two cost measures is more typical.

5.4.2 Example Complexity Functions

(a) *k*th Order smoothing Integral

$$E_c(w,k) = 0.5\int \left\| \frac{\partial^k}{\partial x^k} F(x,w) \right\|^2 \mu(x)dx \tag{5.3}$$

$\mu(x)$ is a weighting function that applies particular importance to certain regions of the input space, for which the smoothness of the data is more important. The result of the application of this function is to make the *k*th derivative of $F(x,w)$ with respect to the input x small, thus smoothing the output of the network [73]. However, it can be difficult to apply this technique when using gradient based optimisation, as the evaluation of the derivative of the complexity function with respect to the network weights is often hard to obtain.

Weight Elimination and Weight Decay

While Weight Elimination (b) and Weight Decay (c) both cost the weights directly, their influence is restricted because of their relative importance to the standard performance function, E_s, defined by the regularisation parameter, Ψ.

(b) Weight Elimination.

$$E_c(w) = \sum \frac{(w_i/w_o)^2}{1+(w_i/w_o)^2} \tag{5.4}$$

Here w_o is a pre-assigned parameter defining the bounds of contributing network weights. Weights that fall below w_o result in a low complexity term thus resulting in a reduction in their contribution to the network output. Weights that are greater than w_o will result in a complexity term of one. The selection of w_o is important to the form of the network output. The benefit of this procedure is that it allows larger weight values than the weight decay method [74], while still pruning from the network weights that do not make a useful contribution. This in turn reduces the likelihood of over fitting through excess weights.

(c) Weight Decay

Weight decay makes use of the sum of squares of the free network parameters.

$$E_c(w) = \sum w_i^2 \tag{5.5}$$

This procedure is effective because it forces the weights of non-active neurons to zero, while the important weights retain their relatively large values. Weights that would normally be defined as excess, and result in poor generalisation by virtue of their likelihood to take on arbitrary values or be used to obtain a slight reduction in the cost function, will now tend to zero, which in turn leads to improved generalisation.

Weight decay does not work in all cases [75] and is not considered a strictly suitable form of regularisation for multi-layer perceptron networks as it tends to promote networks with many smaller weights rather than an equivalent larger single weight. However, it has proved successful in many cases [76] and is simple to implement.

5.4.3 Implementation of Weight Decay Control for Levenberg Marquardt Optimisation

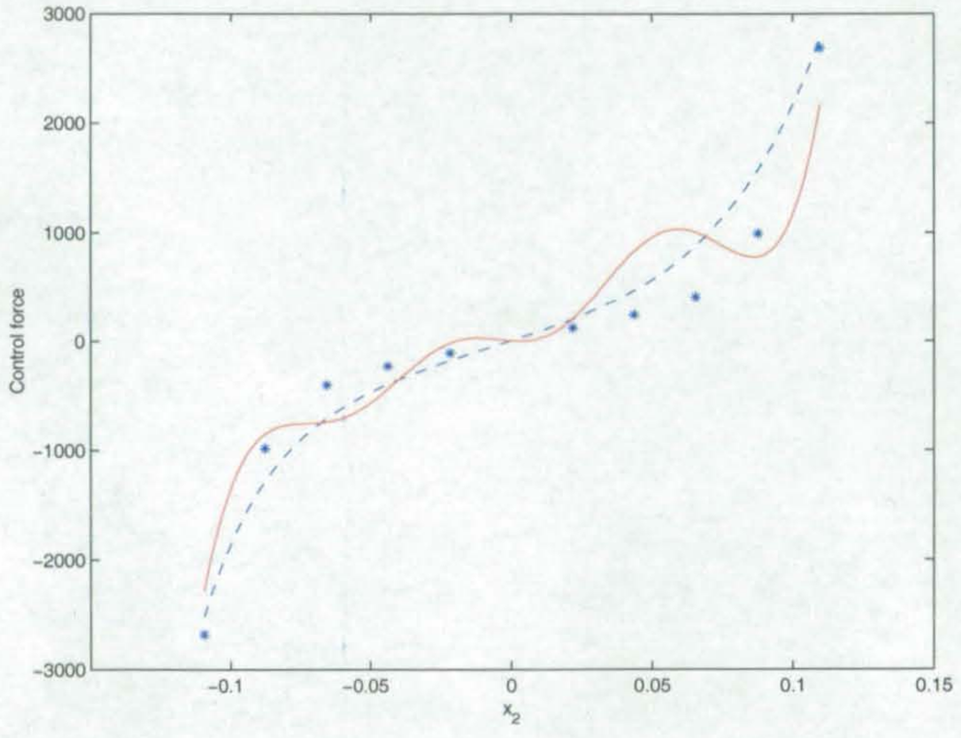
Possible problems resulting from Tansig functions becoming saturated can be much reduced through the use of weight decay. Furthermore, weight decay is easily implemented within the Levenberg Marquardt algorithm and because of its simple squared format, it works well as an addition to least square based optimisation problems and easily fits the format of Equation 4.4.

$$\begin{aligned}
\chi^2 &= \Psi^2 E_c = \Psi^2 w^2 \\
\frac{\partial \chi^2}{\partial w} &= 2\Psi^2 w \\
E_c \frac{\partial E_c}{\partial w} &= \Psi^2 w \\
H_{LM} &= \left(\Psi \frac{\partial E_c}{\partial w}\right)^2 = \Psi \frac{\partial w}{\partial w} \Psi \frac{\partial w}{\partial w} = \Psi^2
\end{aligned} \tag{5.6}$$

Its simplicity also means it has little effect on the computational expense for a training epoch. Although it can affect the cost reduction during a fixed period of time, (because it helps ensure none of the Tansig functions become redundant), with careful management of its contribution to the overall cost in the long term it is likely to give a much improved level of fit.

Although Weight Elimination is more advanced [75] in light of the fact that the main problems stem from excessively large Tansig neuron weights, resulting in continuous saturation of the neuron and switching effects, the allowance of larger weights for these neurons does not seem appropriate. The weights of the final layer are limited by the desired output, therefore it was felt intuitive to apply weight decay only to neurons having either a Tansig or Logsig transfer function, for which saturation is likely.

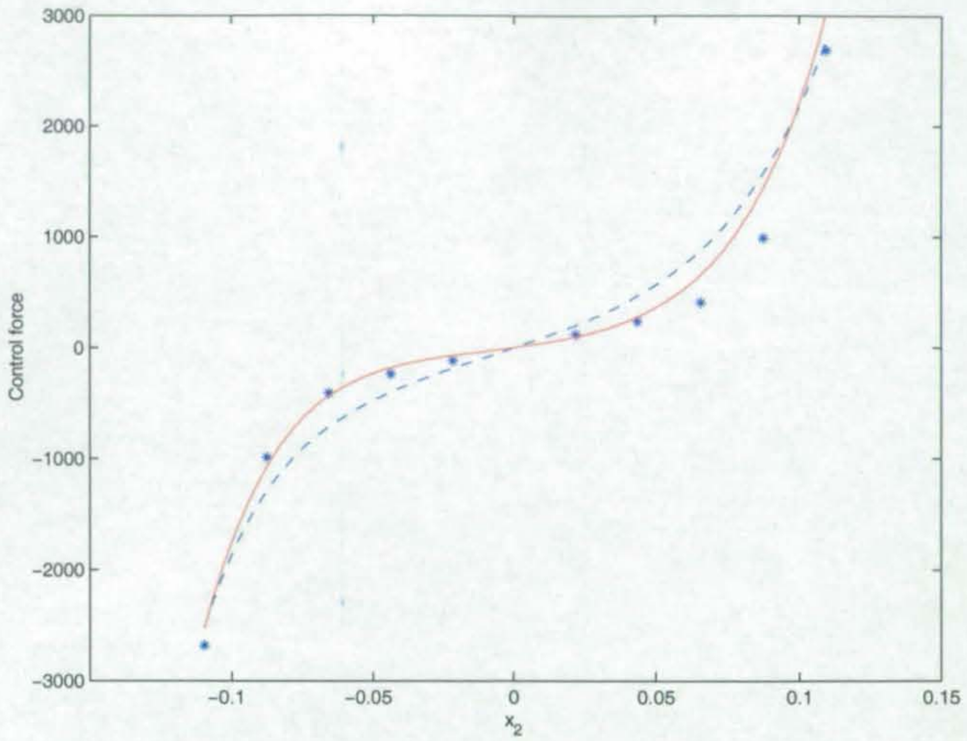
Figures 5.8 and 5.9 show the effect weight decay can have on the trained network. These figures were created by fixing all other states as zero and calculating the force for x_2, (suspension deflection) values at intervals of 0.001. It is worth noting at this point that the forces calculated from the costate training data set do not correspond exactly with the force values calculated by the parameter optimisation technique. Therefore, forces calculated by the two networks do not correspond directly. (The error between them is a function of the calculation tolerances and slight differences between the discrete and continuous time processes used. The error can be reduced, but this will be at the expense of calculation time. Given the number of data points needed to train the network, this must be kept low.)



* Force Calculated from Costate Training Data, -- NNF Calculated Control force,

- NNC Calculated Control Force.

Figure 5.8 - Training Without Weight Decay



* Force Calculated from Costate Training Data, -- NNF Calculated Control force,
- NNC Calculated Control Force.

Figure 5.9 - Training With Weight Decay

Figure 5.8 shows that weight decay is not always essential to obtain a smooth output surface; the NNF network output is naturally smooth. However the NNC output is clearly not satisfactory. Its output cannot be strictly defined as having over-fitted the training data, as clearly the network does not pick up every point as in Figure 5.7. This example of the NNC network output remains smooth along other state axes, but the x_2 axis is of particular importance. Also, in simulation, the NNC trained without weight decay does not settle to zero from initial conditions, but settles instead to approximately $\pm$ 0.02.

The addition of weight decay with a well-chosen regularisation parameter Ψ, has no significant effect on the degree of accuracy that can be achieved. The force error is in fact reduced though the application of weight decay in the example shown. The likelihood of a successful optimisation is also improved, as the Tansig neurons do not remain saturated. Thus switching is less likely and gradients remain significant. With weight decay, Tansig functions are active across a much larger range of the input space.

5.4.4 Point Weighting

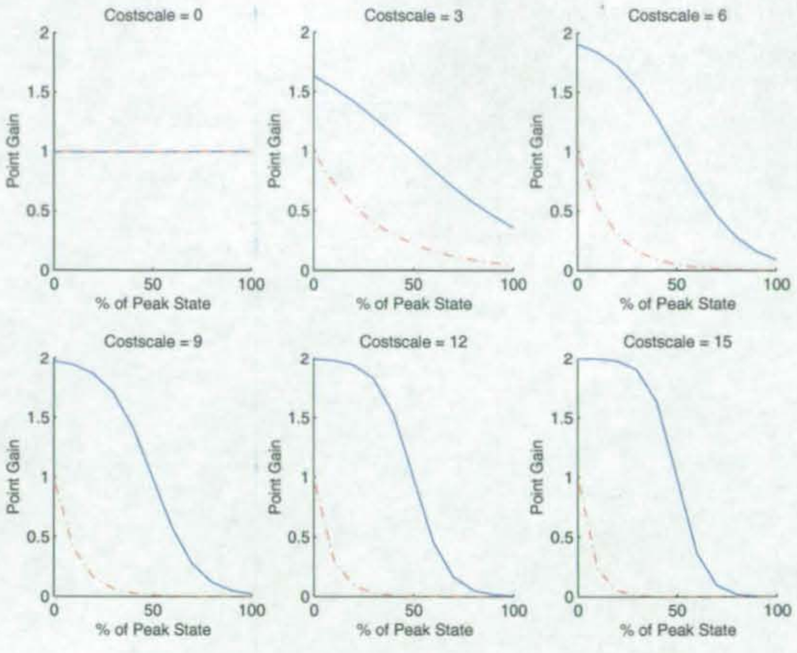
Because some extreme points in the force / costate map have very high values, they can dominate the optimisation process. As the error function is not normalised with respect to the absolute value of the fitted point, large percentage errors can occur for points of low relative amplitude. This may result in poor control around the origin, which can in turn result in poor final settling for initial condition problems. This can be expressed in a number of forms; non-zero settling, even when bias cancelling is used, due to an inflection in the control surface, slow settling or an oscillatory final settling period. To decrease the influence of these points, the error is weighted using one of the following cost functions.

$$\hat{x} = \frac{x_i}{|x_i|_{\text{MAX}}}$$

$$g_1 = \frac{2}{\left(1 + e^{\gamma(\hat{x}-0.5)}\right)} \tag{5.7}$$

$$g_2 = e^{-\gamma(\hat{x})} \tag{5.8}$$

Equation 5.7 based on the Logsig function (Table 3.1) and Equation 5.8 are both adapted by one variable, λ (the costscale), to vary the importance of initial conditions x_i to the optimisation. The two functions allow different shaped bands of importance to be defined and compared.



– Gain Equation (5.7), – · – Gain Equation (5.8)

Figure 5.10 - Point Gain Vs Magnitude

By changing the cost scaling value, γ, the importance of peripheral values in the state space can be adjusted. This helps to enforce the importance of the low value points around the origin.

5.5 Network Training Results

Fifteen neural networks were trained to determine the state - force relationship using various combinations of weight decay, point weighting and bias cancelling. During the 5 hour training period, approximately 310 epochs were completed per optimisation using an 866MHz Athlon processor with 256mb of 100MHz RAM. Work previous to this had shown that a fixed level of weight decay throughout the optimisation was not viable. If the weight decay variable Ψ (Equation 5.5) is small enough to allow complete optimisation, its significance is reduced early in the optimisation and the Tansig functions become saturated. This restricts the ability of the network to optimise further. If it is too large, as the relative value of the least squares cost becomes smaller, it then dominates the optimisation, preventing further reductions in the least squares cost.

To counter this effect, the weight decay variable, Ψ, was adapted at the beginning of each epoch to maintain a constant ratio between the least squares cost and the weight decay cost of 1, 2 or 5 respectively. The maximum value of Ψ was capped allowing greater freedom to the minimisation of the least squares cost in the early stages of the optimisation. A capping value of 50 was chosen on the basis of earlier experimentation with fixed values of Ψ.

Table 5.5 - Least Squares Optimisations for the State - costate Relationship

	Bias Cancelling	E_s/E_c	Point Weight Cost scale, γ	Final Cost E	% Error on Force values		Number of Saturated Tansig functions	
					Original data	Valid-ation data	Dual Sided	Single Sided
1	no	No WD	0	3.8×10^{-2}	17.78	17.61	17	0
2	no	5	0	4.84×10^{-4}	1.99	2.19	0	0
3	yes	No WD	0	7.01×10^{-4}	2.43	2.56	4	0
4	yes	5	0	4.56×10^{-4}	2.02	5.23	0	0
5	yes	No WD	9	1.80×10^{-2}	14.01	10.81	13	0
6	yes	5	9	2.48×10^{-3}	6.581	6.01	0	0
7	yes	2	9	5.15×10^{-3}	8.52	8.31	0	0
8	yes	No WD	12	8.81×10^{-2}	35.58	29.49	14	0
9	yes	5	12	1.66×10^{-3}	7.59	10.04	0	0
10	yes	2	12	1.08×10^{-3}	4.90	4.96	0	0
11	yes	No WD	15	1.59×10^{-3}	8.28	6.54	1	2
12	**yes**	**5**	**15**	$\mathbf{5.94\times10^{-4}}$	**6.27**	**4.43**	**0**	**0**
13	yes	2	15	3.03×10^{-3}	11.63	16.45	0	0
14	yes	1	15	3.32×10^{-3}	10.01	8.07	0	0
15	**yes**	**5**	**7 †**	$\mathbf{1.38\times10^{-3}}$	**4.116**	**3.09**	**0**	**0**

† Gain Equation (5.7)

The importance of weight decay is indicated by the results given in Table 5.5. Saturating tansig functions, caused a significant problem for the networks trained without weight decay. The relationship between poor performance and saturated transfer functions is indicated by networks 1, 5, 8, 11 in Table 5.5. Weight decay has a clear part to play in the prevention of neuron saturation and, despite being in conflict with the main least squares cost function, often results in improved fitting performance.

The correct degree of weight decay to use is less clear from the results in Table 5.5, without further information about the level of fit achieved. As well as the original training data set, a second validation data set, created using a uniform random distribution, is used to test the degree of generalisation achieved by the network. In addition, the force output was plotted along the state axes and against state pairs to give a visual indication of the degree of generalisation. e.g. Figure

5.11. From the visual analysis, the use of an evenly distributed training data set appears to increase the likelihood of over-fitting. The error ridges run in strips across the force surface between the data points. This can be seen in the surface plots of force against x_3 in Figure 5.11.

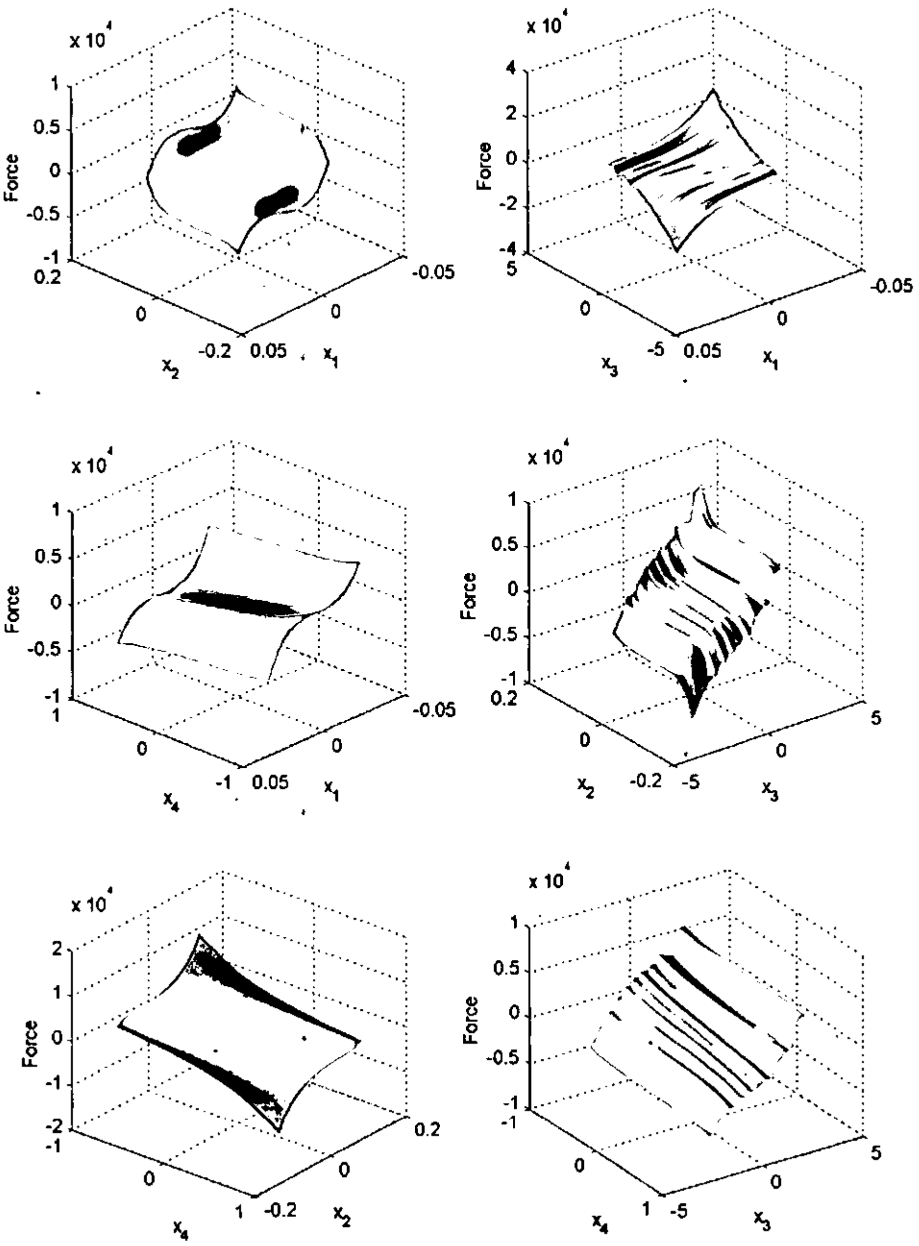


Figure 5.11 - Error Ridges in the Force Surfaces

5.11. From the visual analysis, the use of an evenly distributed training data set appears to increase the likelihood of over-fitting. The error ridges run in strips across the force surface between the data points. This can be seen in the surface plots of force against x_3 in Figure 5.11.

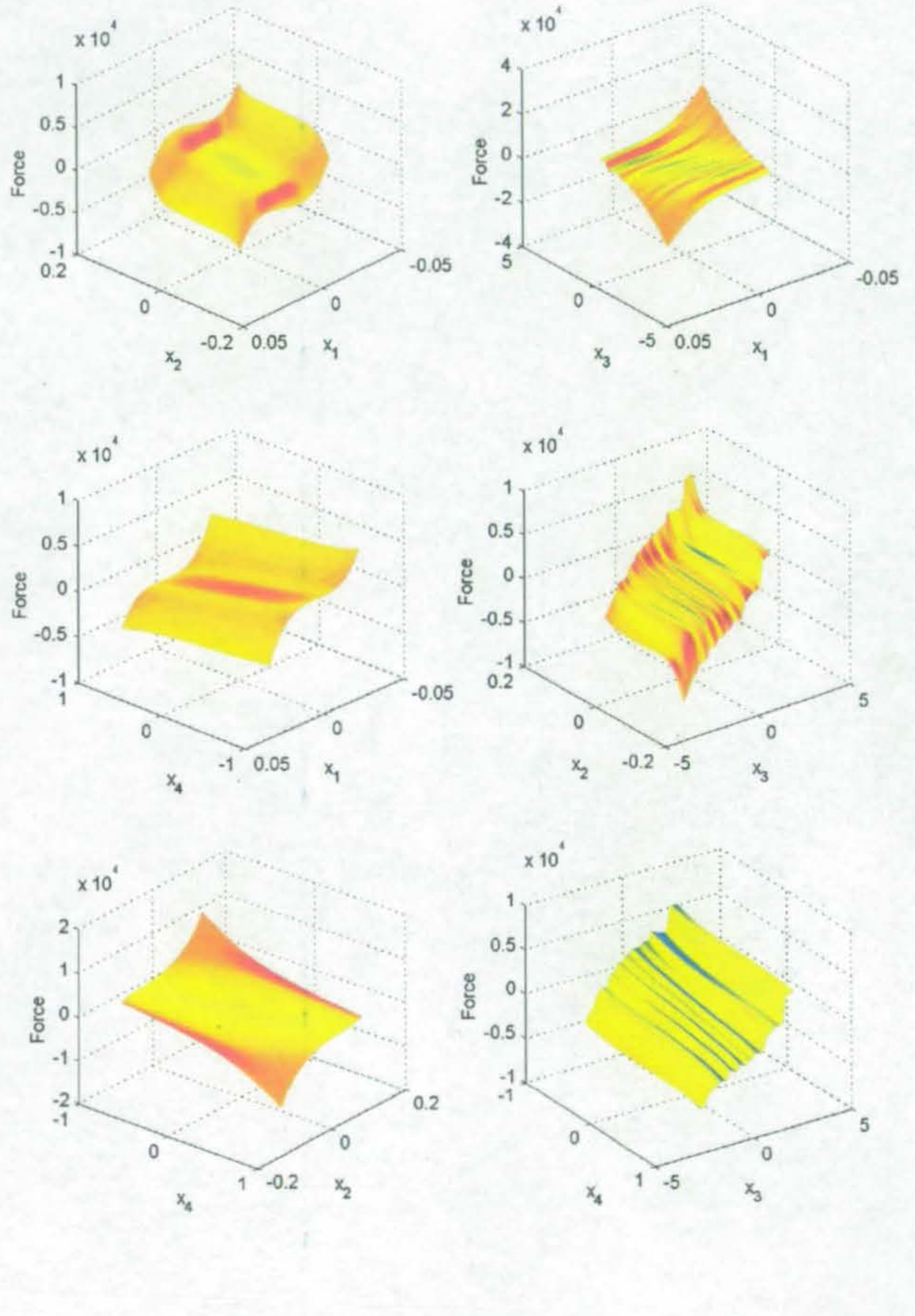


Figure 5.11 - Error Ridges in the Force Surfaces

From the percentage error results in Table 5.2 it is possible to conclude that point weighting is beneficial. The networks without point weighting have achieved higher levels of overall fit. However, the degree of fit at low state values is more important than at the extremes of the state range. Low value errors can prevent the system settling. Figure 5.12 shows that the level of fit achieved at low magnitudes is improved though the use of point weighting and this will ensure better dynamic performance of the neural network.

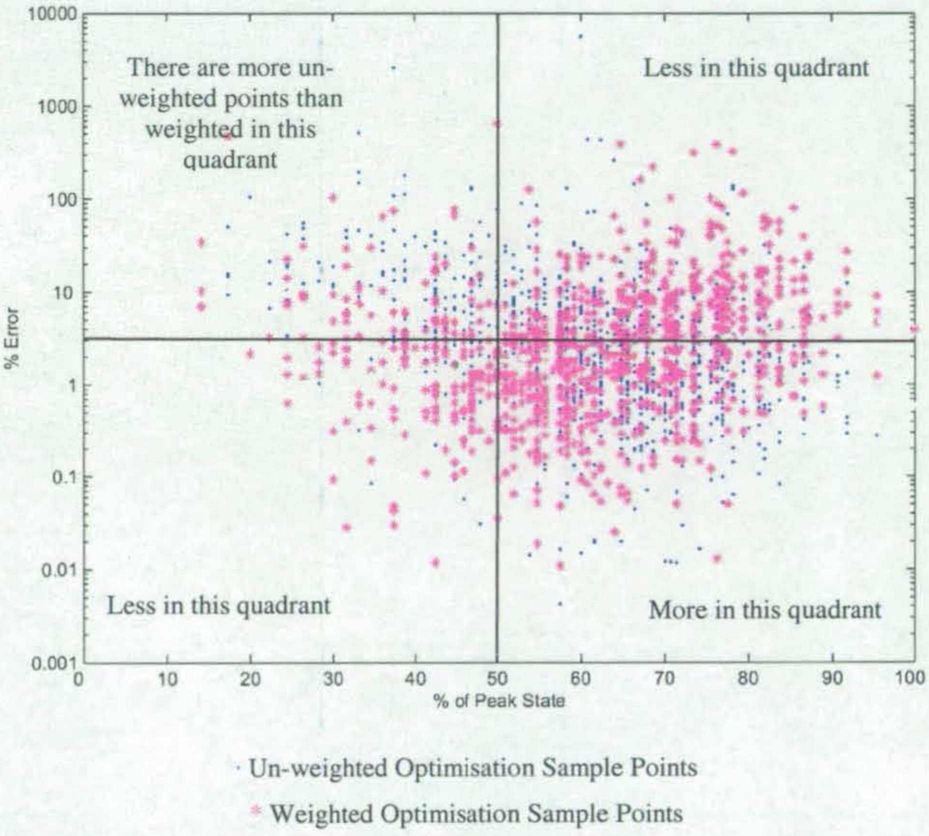


Figure 5.12 - The Effect of Point Weighting on Network Error.

Because there are a number of sources of randomness in the training of neural networks - initial conditions, training data selection and small computational differences - no given optimisation is fully repeatable. In particular, initial conditions contribute significantly to the end result. Although the result may be very similar, no network is going to be exactly the same without restricting the optimisation to repeat the same changes from the same initial conditions. The time constraints involved prohibit training large numbers of networks to achieve a truly generalised result for each combination of weight decay, bias cancelling and point weighting. Therefore, the best result from the 15 networks was chosen to analyse the performance of the network as a feedback controller. The network was chosen on the basis of percentage error, final cost *E*, a visual inspection of the force surfaces as plotted in Figure 5.11, and simulation runs of both initial condition cases. The dynamic performance of this network and the non-linear control strategy employed is now further analysed in Chapter 6.

CHAPTER 6 PERFORMANCE COMPARISON AND ANALYSIS

Chapter 6 Performance Comparison and Analysis

In Chapter 5, a training strategy for the supervised learning of the state - force relationship was developed and a number of neural networks trained. In this chapter, the performance of one of those networks will be assessed as a controller in dynamic simulation. A second network trained using supervised learning to learn the state - costate relationship was also established and its performance will be assessed in parallel. The dynamic correlation between the open-loop discrete control technique that generated the training data set and the neural networks in control will first be assessed using two initial condition examples.

Having established the validity of using the neural networks in dynamic simulation, they will then be used to assess the performance of the non-linear control strategy for a number of test cases; sinusoidal responses, simulated and real road responses.

6.1 Initial condition Performance of Retrained Networks

Network 12 was chosen from Table 5.2 to investigate the use of a neural network feedback controller. Although other networks achieved lower LSQ cost and better overall levels of fit, in simulation low magnitude inaccuracies prevent them settling correctly. Networks 13 and 14 were restricted in training by the higher level of weight decay used and as such do not achieve comparable levels of control to network 12.

A second set of neural networks were trained using weight decay, bias cancelling, and point weighting to learn the state - costate relationship. The best of these was selected using the same methods. The chosen network had the following format, Table 6.1.

Table 6.1 - State - Costate Neural Network Details.

Network Architecture	Details
Input Layer	30 Tansig Function
Output Layer	4 Purelin Functions
Bias Cancelling	Yes
Training Process	
Duration	19 Hours (The significant increase in optimisation time is due to the limited memory available.)
Number of Epochs	253
Weight Decay Ratio	5
Point Weighting, γ	15
Performance	
Least Squares cost E	2.36×10^{-4}
Percentage Error on Force values	
Original data	9.27%
Validation data	6.45%

The performance of the neural networks as controllers in their own right is more than satisfactory (Figures 6.1 & 6.2). However, they do not replicate the exact state trajectories of the open loop controller for either initial condition (Table 6.2 & 6.3). This can be explained by a number of factors; The original open loop control uses zero order held control inputs. The neural network control is continuous, while the hold period is small and the error caused not significant, the accumulative error is of a measurable size. Secondly because the network training data is evenly distributed across the working range, the actual range of training points covering the final settling period of the simulation is very limited. It is at this point that most of the variation between the original open loop control and that of the neural networks occurs. If the suspension deflection axis is considered during the initial hub velocity response (Figure 6.1), only one non-zero x_2 training point is passed through during the simulation. The degree of accuracy is a measure of the generalisation achieved by the network through the use of weight decay. Despite the additional weighting applied to the limited points available, their significance is still not enough to ensure the exact settling trajectory is followed. This could be further improved by an increased point's density in critical regions. The resulting performance is significant enough to clarify that neural networks are a viable means of implementing and verifying the performance of the non-linear control strategy.

The variations between the NNF and the NNC are caused by a number of factors: The level of accuracy achieved by the NNC is lower due to the limitations on the network size. No neural network would have given exactly the same responses, as each will generate a different error distribution across the cost surface depending on the initial conditions used. Marsh's approach for the non-linear force calculation does not employ any means of ensuring the direct relationship, between the costates and the control force, Equation (2.27), is enforced. They are, in effect, optimised independently of each other.

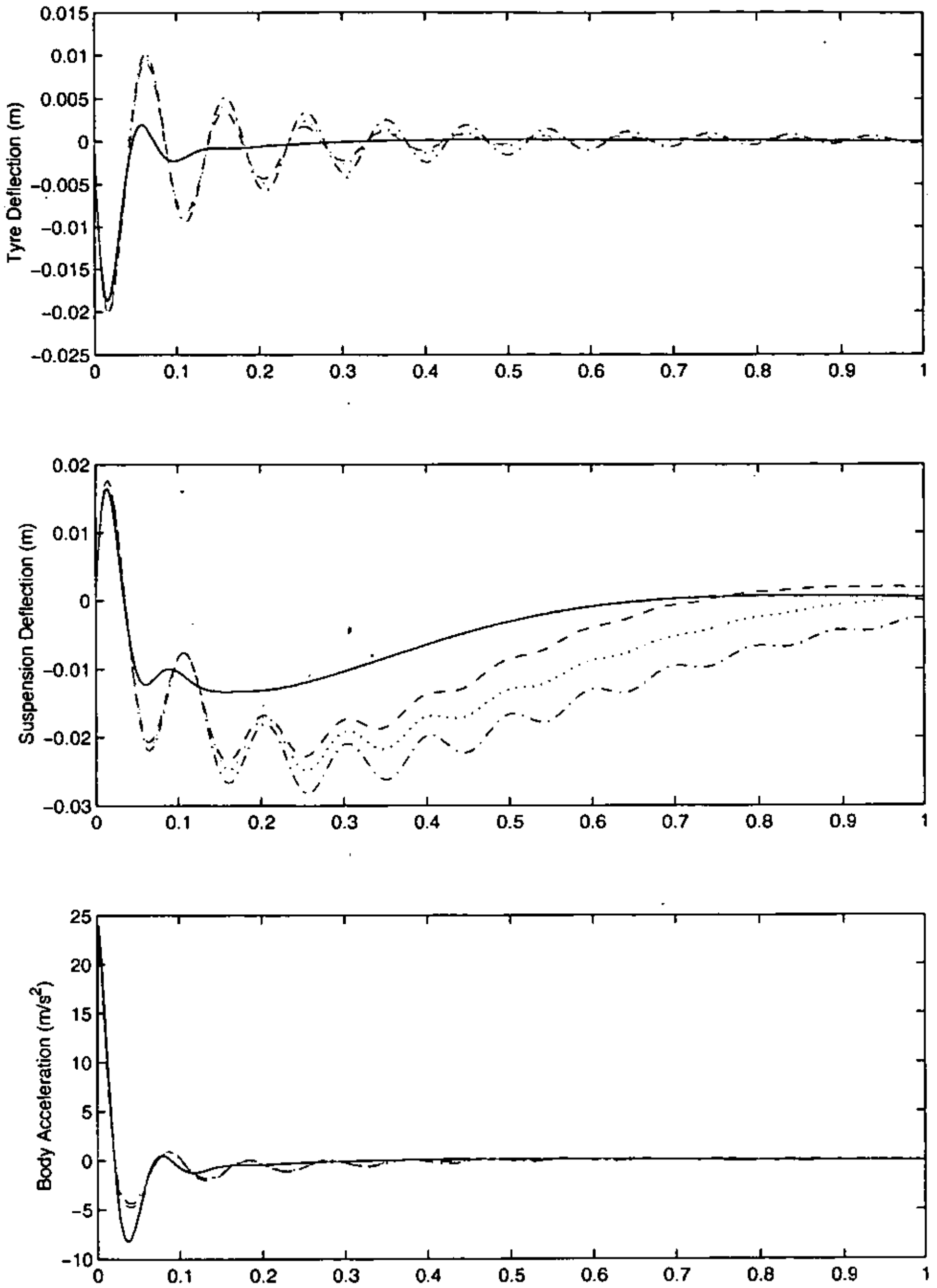
Table 6.2 - Initial Hub Velocity Responses of Retrained Networks

	R.M.S Values for First Second			
Hub Velocity Response	Tyre Deformation (mm)	Suspension Deflection (mm)	Body Acceleration (m/s^2)	Peak Tyre Deflection (mm)
Open Loop Non-linear Control	3.75	13.65	2.45	19.58
NNF Control	3.98 [6%]	16.35 [20%]	2.06 [16%]	20.02 [2%]
NNC Control	3.66 [3.4%]	11.92 [13%]	2.07 [15.5%]	20.04 [2%]

Table 6.3 - Initial Body Velocity Response of Retrained Networks

	R.M.S Values for First 1.5 Seconds			
Body Velocity Response	Tyre Deformation (mm)	Suspension Deflection (mm)	Body Acceleration (m/s^2)	Peak Suspension Deflection (mm)
Open Loop Non-linear Control	4.15	37.52	2.95	89.24
NNF Control	4.28 [3%]	36.50 [0.05%]	2.77 [6%]	89.21 [0.04%]
NNC Control	4.24 [2%]	36.65 [0.35%]	2.75 [7%]	89.79 [0.6%]

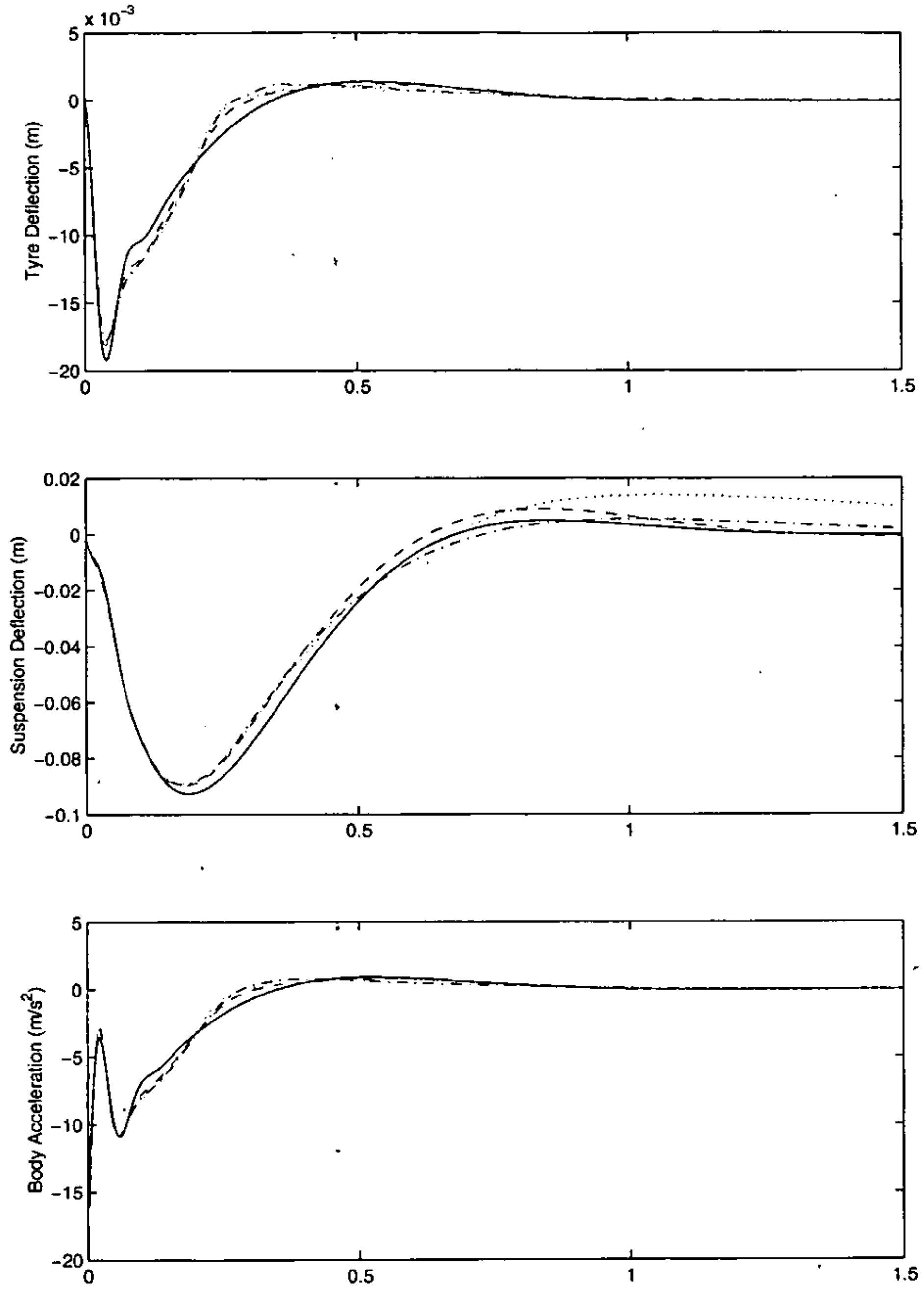
[percentage deviation from open loop non-linear control case is shown in square brackets]



– Equivalent Linear System, · · Open loop Non-linear Control,

– · NNF Control, – – NNC Control

Figure 6.1 - Neural Network Controlled, Hub Velocity Response



– Equivalent Linear System, · · Open loop Non-linear Control,

– · NNF Control, – – NNC

Figure 6.2 - Neural Network Controlled, Body Velocity Response

6.2 Sinusoidal Responses

As discussed in Chapter 2, it is inappropriate to use standard frequency analysis techniques for non-linear systems. It is however still possible to assess the frequency response of a non-linear system through analysis of the system for fixed frequency sine wave disturbances. The suspension control strategies were simulated for a range of fixed frequency disturbances (0.25Hz – 16Hz), amplitude 0.5m/s, the peak and RMS values calculated to create Figure 6.3 and Table 6.3.

The neural networks give improved control at the body bounce frequency, reducing the level of body acceleration through increased utilisation of the suspension workspace. They significantly improve the levels of body acceleration at frequencies between body bounce and wheel hop. The high levels of damping used in the equivalent linear control cause a broadening of the two resonances, which allows a significant improvement using non-linear control. At the wheel hop frequency the disturbance amplitude is relatively large and this causes the tyre deflections to become very large in respect to the non-linear cost-function. This results in slightly worse control of body acceleration at the wheel hop frequency. However beyond the wheel hop frequency, body acceleration is again significantly reduced. The tyre deflection always remains within the allowable range, with respect to the non-linear cost function, but is significantly worse than the Equivalent Linear Controlled system (ELC) above 5Hz. Below 5Hz tyre deflection is lower. This is significant as load transfer due to handling related disturbance inputs are normally below this frequency. This should mean that during a dynamic manoeuvre, when control of the tyre contact patch vertical load is most important, the non-linear controller is also beneficial to handling performance.

The frequency response of the original linear system is similar to that of the non-linear system, however high levels of suspension and tyre deflection are exhibited at the body bounce and wheel hop frequencies respectively. These would seriously compromise both ride and handling at their respective frequencies. The non-linear system combines the best of both the original and the equivalent linear control

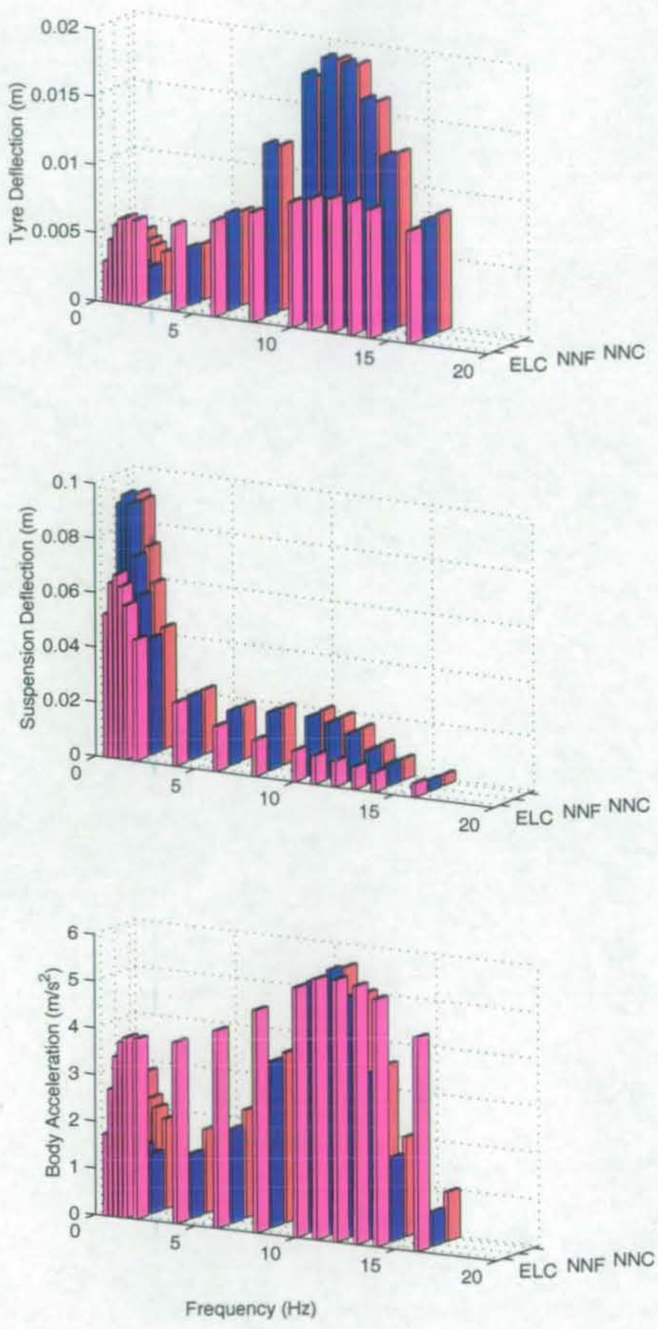


Figure 6.3 - Peak Values for a Range of Disturbance Frequencies

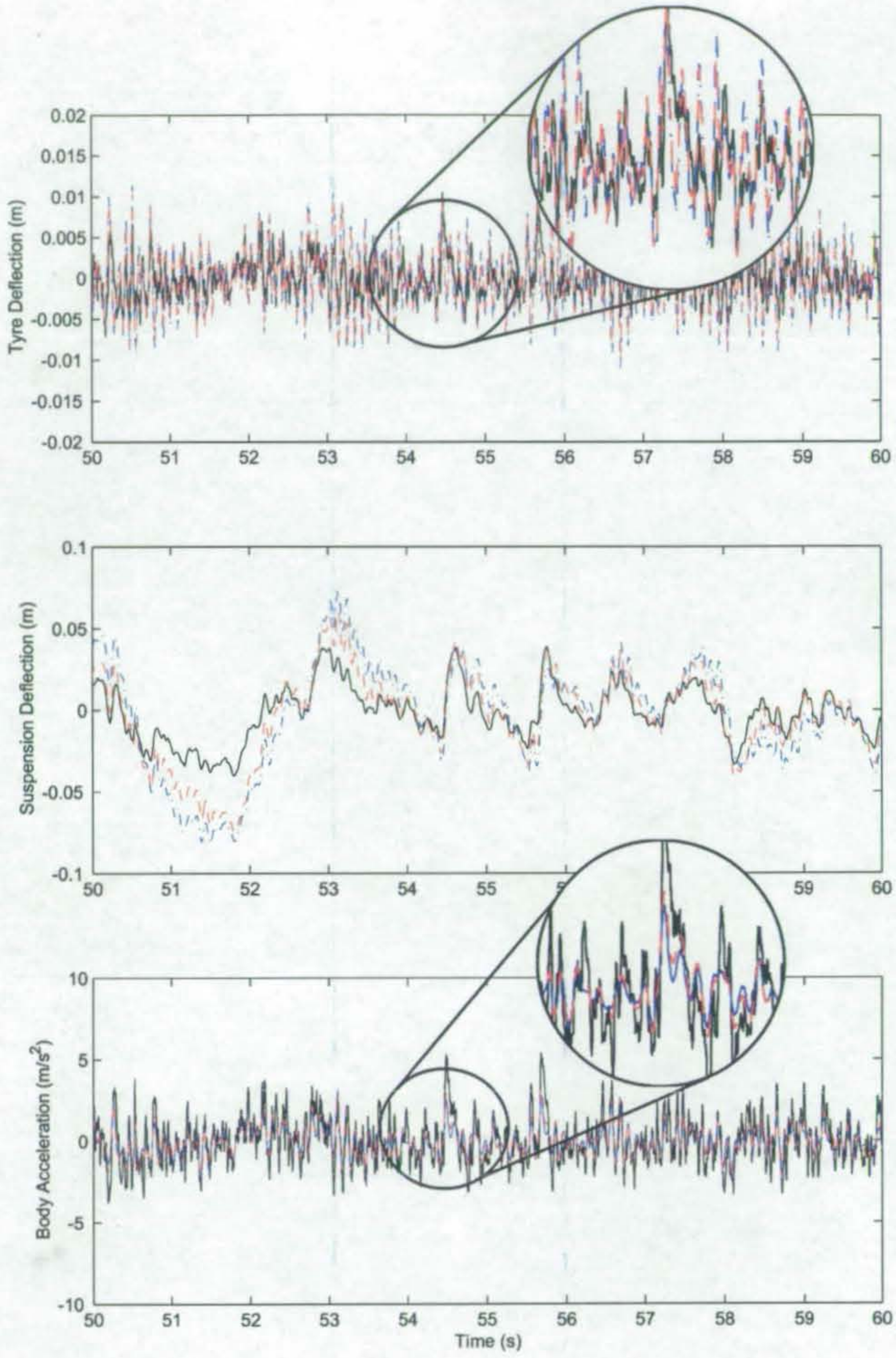
Table 6.4 - Selected Peak and RMS Values for a Range of Frequencies.

Frequ-ency	Equivalent Linear System		NNF control		NNC control	
	Peak	RMS	Peak	RMS	Peak	RMS
Tyre Deflection (mm)						
1	5.70	4.04	4.34	3.29	4.44	3.48
4	6.32	4.47	4.23	2.83	4.07	2.83
11	9.64	6.81	19.56	13.81	19.03	13.44
Suspension Deflection (mm)						
1	67.29	47.56	90.7	63.79	90.37	63.49
4	23.26	16.46	23.61	16.57	23.58	16.54
11	10.48	7.39	20.38	13.88	20.38	13.53
Body Acceleration (m/s^2)						
1	3.43	2.42	3.00	2.05	2.87	2.16
4	3.88	2.74	1.38	0.94	1.78	1.25
11	5.57	3.93	5.66	4.24	5.60	4.23

6.3 Road Responses

6.3.1 Real Road Response

The Equivalent Linear Controller, NNF and NNC were simulated using real road data: Breakback Road and the A6 at 20m/s and 30m/s respectively (see Section 2.2.3). The real roads cover a wide range of frequencies and highlight the true benefits of the non-linear controller. Although not adaptive, the control force is in many respects dependent on the magnitude of the disturbance. The non-linear control allows greater utilisation of the suspension workspace for small amplitude disturbances, which reduces their influence on body acceleration relative to the equivalent linear control. For larger amplitude disturbances, the non-linear control prevents excursions outside the suspension workspace that would result in large body accelerations in the original linear system. This results in a significant reduction in RMS body acceleration for both roads over the equivalent linear controller, though better utilisation of the suspension workspace.



– Equivalent Linear System, – · NNF Control, – – NNC Control

Figure 6.4 - BreakBack Road Response

Table 6.5 - RMS and Peak values for Breakback Road Response

	Tyre Deformation (mm)	Suspension Deflection (mm)	Body Acceleration (m/s^2)
RMS Values			
Equivalent Linear Controller	2.42	15.32	1.40
NNF Control	4.17	31.20	0.85
NNC Control	3.76	24.17	0.91
Peak Values			
Equivalent Linear Controller	11.41	65.47	7.14
NNF Control	19.53	98.00	7.71
NNC Control	18.41	88.79	6.03

Table 6.6 - RMS and Peak values for A6 Road Response

	Tyre Deformation (mm)	Suspension Deflection (mm)	Body Acceleration (m/s^2)
RMS Values			
Equivalent Linear Controller	1.16	3.93	0.65
NNF Control	2.23	8.2	0.27
NNC Control	1.81	5.69	0.32
Peak Values			
Equivalent Linear Controller	7.19	18.46	4.98
NNF Control	9.70	39.96	2.14
NNC Control	9.28	29.11	2.34

One feature that is observable in both the linear and non-linear system is the influence of low frequency disturbances on suspension deflection. This is accountable for the higher peak acceleration of the NNF on the Breakback Road. Low frequency disturbances appear to cause a suspension drift that reduces the available suspension workspace to absorb additional disturbances. To investigate suspension drift further, a number of simulated roads were established that seek to highlight the effect.

6.3.2 Simulated Roads

It is important here to define road inputs with sufficient amplitude at long wavelengths to influence the suspension drift response. The simplest case is a step in the vertical velocity

$$\dot{Z}(t) = V(t) = \begin{cases} 0 & (t \leq 0) \\ 0.5 & (t > 0) \end{cases} \tag{6.1}$$

At a vehicle horizontal speed U = 20 m/s, $\dot{Z}(t)$ corresponds to a sudden change from flat road to a uniform 2.5% slope downhill. This input will be referred to as *Road A*. In Chapter 7, for simplicity, a unit step will be used. This would cause the neural networks to exceed their original training range so cannot be applied here.

Road B, Figure 6.6, provides a very low frequency excitation, but without the initial transient disturbance. It takes the form of a very long wavelength downhill slope, dropping 30 metres in height over a distance of 1.2 km. Again assuming a vehicle horizontal speed of 20 m/s, we have

$$V(t) = \begin{cases} \frac{\pi}{4}\sin\left(\frac{\pi t}{60}\right) & (0 \leq t \leq 60) \\ 0 & \textit{otherwise} \end{cases} \tag{6.2}$$

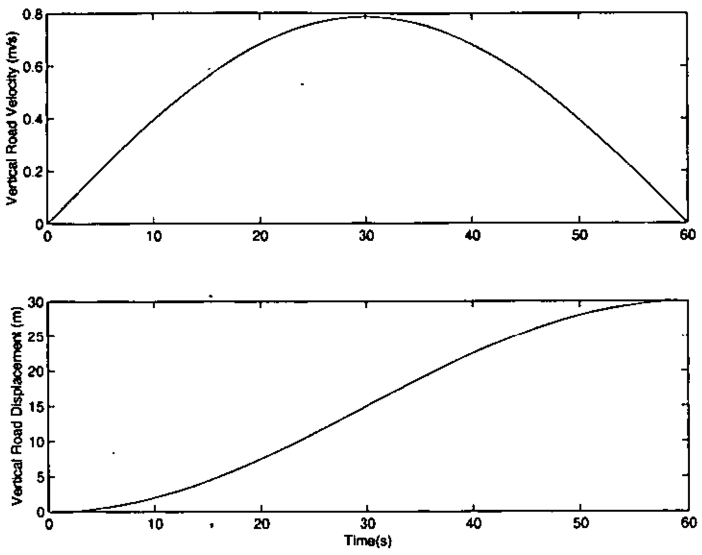


Figure 6.5 - Road B, Displacement, Velocity Profile

Road C combines both short and long wavelengths, in the form of a stochastic road model. Consider first a reference stochastic model ("Reference Road") [77], based on the differential equation

$$\dot{Z} + 2\pi f_0 Z = \sqrt{GU}\,w \tag{6.3}$$

where w is a unit amplitude white noise process. This results in the following PSD for Z

$$S_z(f) = \frac{GU}{4\pi^2} \cdot \frac{1}{f^2 + f_0^2} \tag{6.4}$$

Comparing this PSD of $Z(t)$ with that of a real road (the Copt Oak Road, a minor road South of Loughborough, England) which was sampled at 0.1m intervals and traversed in simulation with $U = 20$ m/s, the theoretical PSD is very much reduced at low frequencies - Fig. 6.7. To overcome this limitation, *Road* C is a modified form of stochastic road, based on a combination of two independent unit amplitude white noise processes w_1 and w_2:-

$$\begin{aligned} \dot{Z} &= V_l + A_1 w_1 \\ \dot{V}_l + \lambda V_l &= A_2 w_2 \end{aligned} \tag{6.5}$$

where λ is a very small low-frequency cut-off. The interpretation is that w_1 is integrated once to provide a road roughness component for $Z(t)$, while w_2 is integrated twice to provide a low frequency undulation, for which V_l is the associated vertical velocity.

Here A_1 and A_2 were adjusted to provide a best-fit to the measured PSD. The corresponding theoretical PSD, is (for $\lambda \approx 0$)

$$S_z(f) = \frac{A_1^2}{4\pi^2} f^{-2} + \frac{A_2^2}{16\pi^4} f^{-4} \tag{6.6}$$

A_1 and A_2 were chosen to minimise the squared logarithmic deviations from the sampled PSD $\overline{S}_z(f)$ obtained from the road data:

$$E(A_1, A_2) = \sum_f \left(\log_{10}(\overline{S}_z(f)) - \log_{10}(S_z(f))\right)^2 \tag{6.7}$$

This results in a PSD that matches well to the measured spectrum Figure 6.7. Although the very low frequencies are slightly over-estimated, the accuracy of the data is not entirely clear at these very long wavelengths ($\lambda \approx$ 2km), and the resulting Road C is ideally suited to the present investigation.

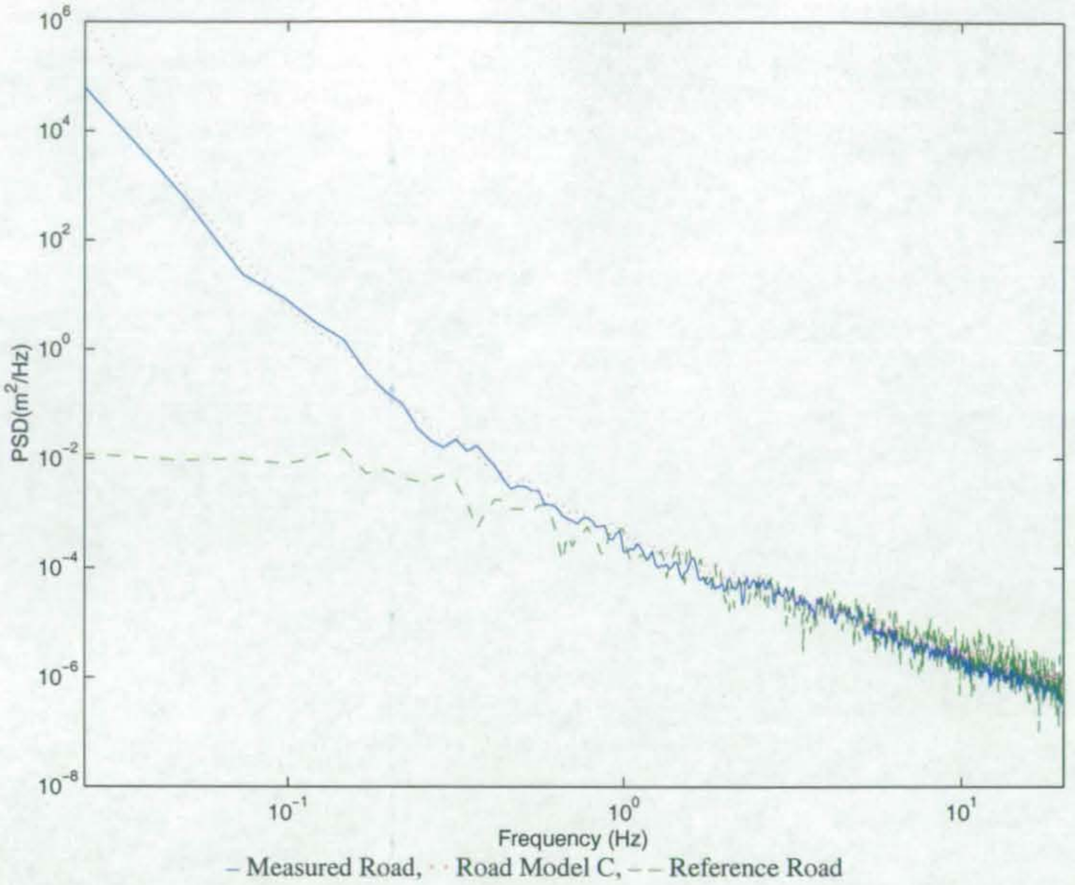


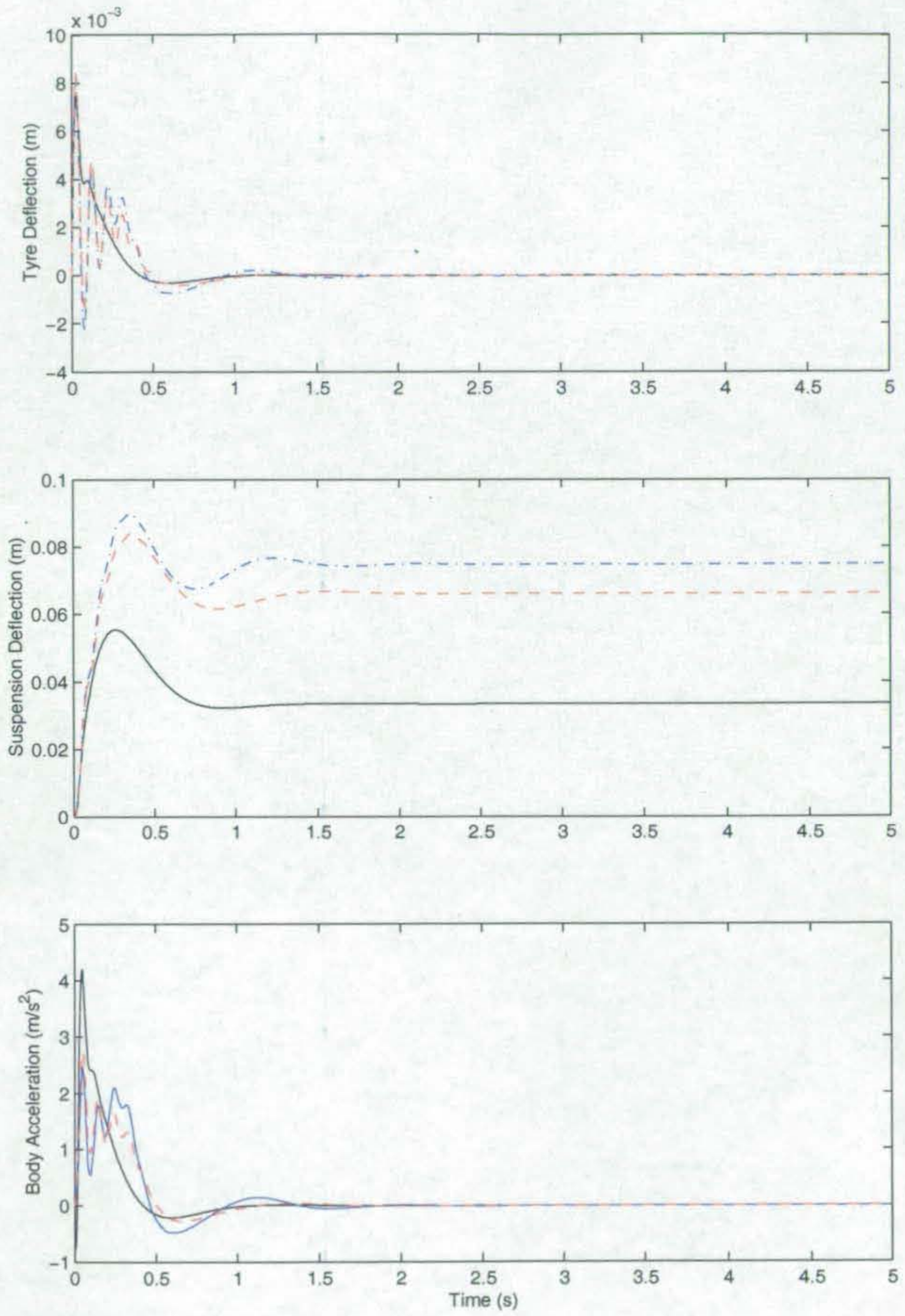
Figure 6.6 - Comparison of Road Model C with the Reference Stochastic Model, Equation (6.5) and a Real Road

6.3.3 Road A Response

The step in road velocity clearly highlights the problem in both the linear and non-linear controller, Figure 6.7. Because the non-linear control is effectively softer than the linear system at the final state condition to which the suspension settles, the suspension drift is larger. This makes it more susceptible to further disturbance as occurred in the Breakback Road simulation, thus resulting in higher body acceleration. Therefore, in all cases it is clearly important to develop a technique that is capable of recovering the lost suspension workspace.

Table 6.7 - Steady State Suspension Deflection Values on Ramp Road

Suspension Deflection	Equivalent Linear System	NNF	NNC
(mm)	33.38	75.00	66.14

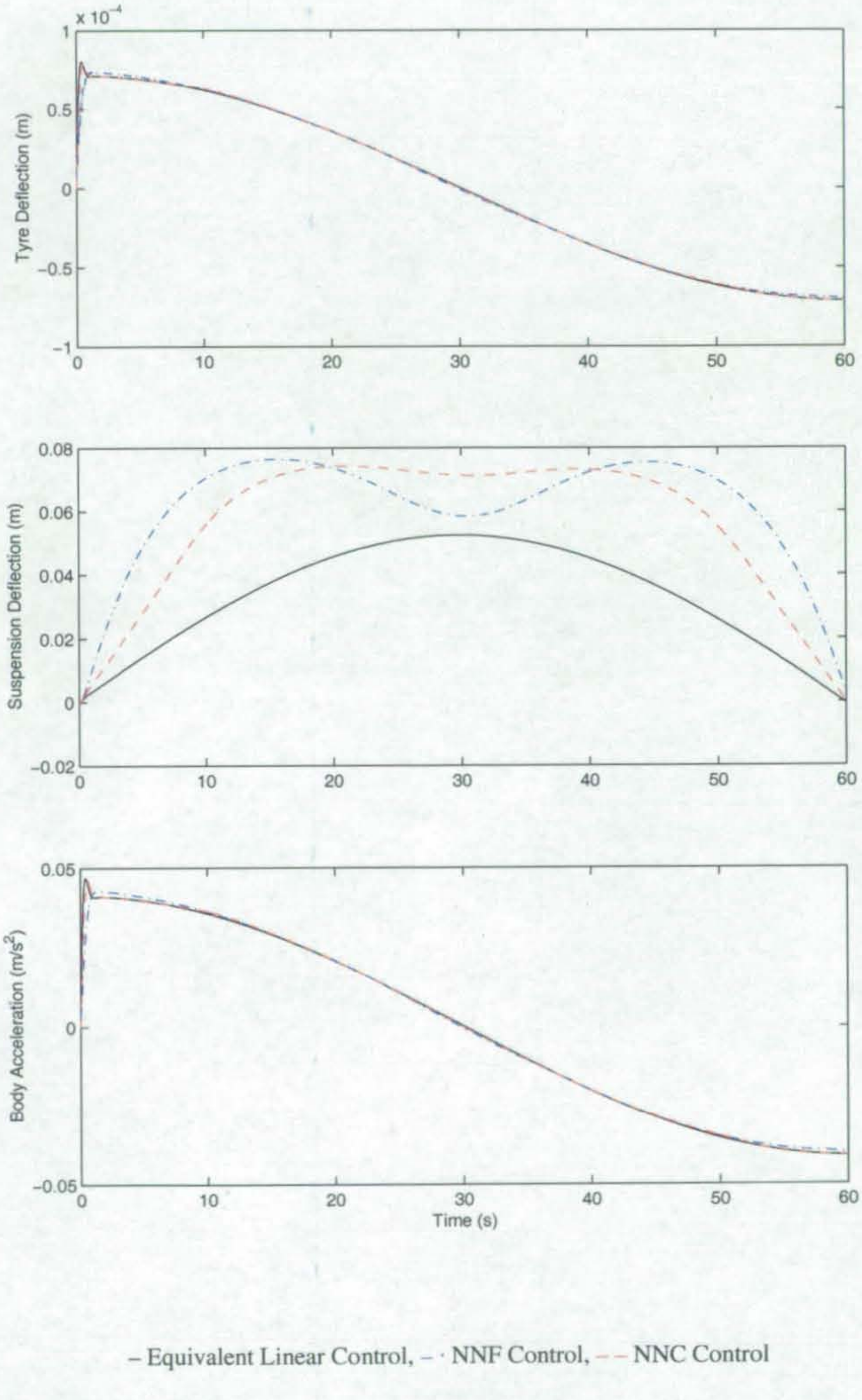


– Equivalent Linear Control, – · NNF Control, – – NNC Control

Figure 6.7 - Road A Response

6.3.4 Road B Response

The Response to a low frequency single event is now considered in Figure 6.8



– Equivalent Linear Control, – · NNF Control, – – NNC Control

Figure 6.8 - Road B Response

The sine wave road response reinforces the results shown in response to the step in road velocity. However, the non-linear control clearly limits the possible suspension drift when x_2 reaches the defined limits of the suspension workspace described by the non-linear cost function. Passive suspension on this road profile exhibits very little disturbance and highlights the magnitude of the problem.

Table 6.8 - Peak Suspension Deflection Values on Sine Wave Road

Peak Suspension Deflection (mm)	Passive	Equivalent Linear System	NNF	NNC
	0.7	52.44	76.53	74.2

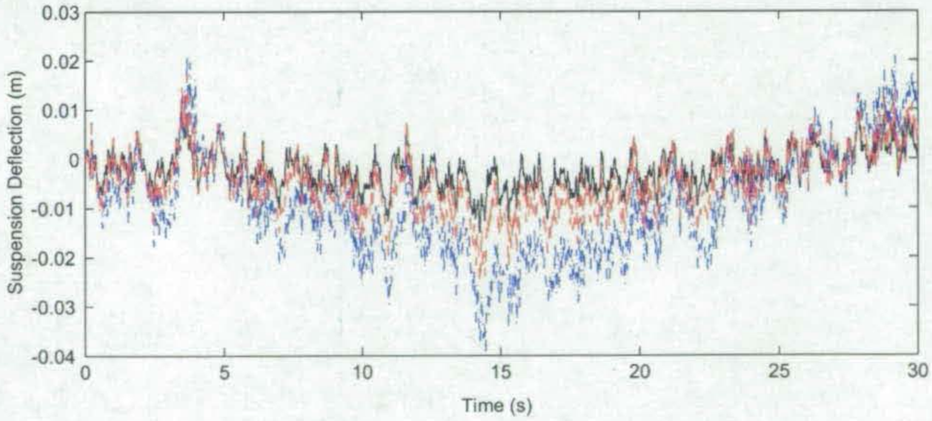
6.3.5 Road C Response

Road C is the most realistic of the three simulated road profiles and highlights clearly the benefits of the non-linear control strategy employed by the neural network controllers. The neural networks indicate a significant improvement can be achieved through the implementation of the non-linear control strategy, both peak and RMS body acceleration values are reduced through greater utilisation of the suspension workspace, Table 6.9. This is key to the success of the non-linear strategy in real road simulation, greater utilisation of the suspension workspace is allowed in order to reduce body acceleration under normal operating condition. However, unlike the linear system on which the strategy is based, careful control of suspension deflection is still maintained in extreme condition in order to prevent bump stop contacts. Better control could still be achieved as the suspension deflection diverges from the equilibrium position for the NNF controller due to a low frequency disturbance of around 0.05Hz, Figure 6.9. The road disturbances that occur during this period are not significant enough to generate a suspension deflection that results in higher body acceleration, but the likelihood of this is increased by the suspension drifting close to the workspace limits.

Road C also highlights variations between the two neural network controllers, further training with extra data points and possibly larger network structures would be necessary to reduce the difference between them. The NNC network exhibits less drift simply as a function of these learning errors and not as a feature of the method by which control is implemented.

Table 6.9 - RMS and Peak values for Simulated Road Response

	Tyre Deformation (mm)	Suspension Deflection (mm)	Body Acceleration (m/s^2)
RMS Values			
Equivalent Linear Controller	1.26	4.20	0.71
NNF Control	2.6	12.17	0.33
NNC Control	2.11	6.90	0.38
Peak Values			
Equivalent Linear Controller	5.22	15.05	2.83
NNF Control	11.06	39.38	1.53
NNC Control	9.29	25.14	1.48



– Equivalent Linear Control, – · NNF Control, – – NNC Control

Figure 6.9 - Road C, Suspension Deflection Response

6.3.6 Summary and Conclusions

It has now been shown that neural networks can be used to implement complex non-linear control strategies through the application of supervised learning. Neural networks having successfully learned both the non-linear state, force and state, costate relationships. In order to be sure of a successful training run a number of key factors have been highlighted; It is important to ensure good generalisation between training data points, this can be achieved using weight decay. The distribution of the training data points through careful positioning to reflect the most likely operating regions, or weighting of an already defined training data set, will increase the likelihood of a neural network that not only gives good optimisation results but also performs well dynamically. Weighting of the training data is also important where the relative magnitude of certain data points is small. To ensure no errors occur at zero and the controlled system settles fully, bias cancelling can be used to enforce the zero for zero relationship of the non-linear feedback controller.

The successfully trained neural network controllers have allowed further investigation of the benefits of the non-linear control strategy. The sinusoidal and real road simulations both reveal the same benefit. Body acceleration is better controlled through more effective utilisation of the suspension workspace. Tyre deflection is increased, but the non-quadratic form of the cost function ensures it does not become excessive.

The dynamic simulations have however highlighted the problem of suspension drift due to low frequency disturbances. An approach to resolve this is described in Chapter 7.

CHAPTER 7 ON-LINE ESTIMATION OF LOCAL ROAD GRADIENT FOR IMPROVED STEADY STATE DEFLECTION CONTROL

Chapter 7 On-line Estimation of Local Road Gradient for Improved Steady State Deflection Control.

In Chapter 6 the performance of the neural network controllers was compared over three simulated road profiles. A significant factor arising from the analysis of this performance is that, like their linear control based counterparts, the neural network controllers do not resolve the problem of steady state deflection arising from the low frequency component of the road profile. This problem is not a function of the controller's implementation through the use of neural networks, but is inherent in the active control technique from which they were trained.

In this chapter, a new approach will be presented that resolves this problem and can be implemented in parallel with the neural network controllers. This approach was previously presented by Fairgrieve and Gordon [78]. The majority of the work in this chapter is carried out using linear modelling and control so as to benefit from the inherent reduction in development time through doing so. When the development is complete, the resulting method is implemented with the neural network controllers, and is shown to result in a significant improvement in all round suspension performance.

7.1 Suspension Drift.

The problem of steady-state offset and low-frequency drift in vehicle suspension deflections arises whenever an active suspension controller employing feedback of absolute vertical body velocity is used. The most common example of this is the so-called 'sky-hook damping' principle introduced by Karnopp [79]. Sky-hook damping is an idealised concept of connecting a 'damper' between the vehicle body and an inertial reference frame, which in practice may be implemented via inertial sensors and an electronic suspension controller. Therefore, for an ideal vehicle travelling at a fixed speed on a constant road gradient, the sky-hook damper exerts a constant force. In steady-state, this must be offset by a constant spring force – either a physical spring, or a corresponding term in the control law – and this implies an offset in suspension travel from the static equilibrium. For long wavelength undulations, the effect leads to poor control (drift) in the suspension motion, and hence unnecessarily large suspension deflections.

If this offset, or drift, is not compensated for, there are obvious implications for the active suspension performance. In the worst case, the suspension will ride close to its bump or rebound stops, with potential impacts resulting from the input of any additional road roughness. Even where this is not a problem, there is a well-known trade-off between available workspace and achievable ride comfort [42] (at the relevant primary ride frequencies, an additional trade-off with dynamic tyre load variation is less of an issue). Therefore, performance degradation occurs whenever offset and drift reduce the available dynamic workspace.

The significance of these effects is often masked by a number of factors. Most simply, there may be a low-frequency cut-off in the input road excitation – imposed either implicitly or explicitly (see Section 6.3.2, Chapter 6). Thus far, it has been assumed that the vertical velocities of the vehicle can be measured directly. However, current sensor technology does not allow this, and estimating absolute vertical body velocity may, and generally will, involve a low-frequency cut-off. In some cases the suspension actuator is incapable of delivering such offsets – as in semi-active control. In all of these cases the most obvious effects of suspension drift are absent, but it is far from clear that the dynamic suspension performance is not compromised.

7.1.1 Passive Suspension Model

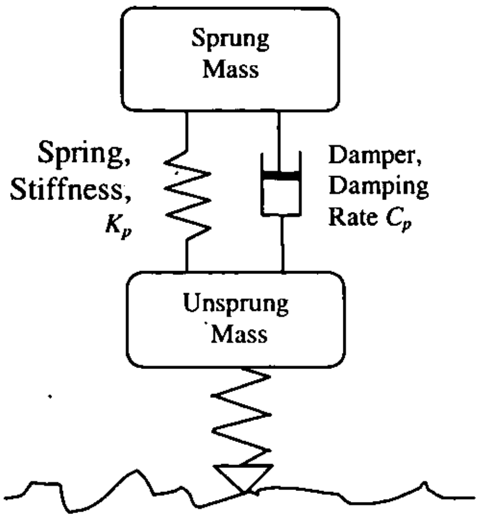


Figure 7.1 - Passive Quarter Vehicle Model

Thus far this thesis has been concerned with purely active suspension. For the purpose of reference, a passive system is now introduced. This has a simple spring and damper instead of the actuator of the quarter vehicle model introduced in Chapter 2. The spring and damper forces are represented as linear functions of the suspension deflection and the rate of suspension deflection respectively.

$$F_s = K_p x_2 + C_p(x_3 - x_4) \tag{7.1}$$

It is this method of suspension deflection control that is used on most current production vehicles. Because the velocity related control element, the damper's force, is a function of the relative difference in the vertical velocity of the sprung and unsprung mass and not the individual velocities, passive suspension does not experience suspension drift. This can be seen by comparing the relative suspension deflection of the passive system with the LQR based controller on Road B, Chapter 6, shown below in Table 7.1

Table 7.1 - Suspension Deflection on Road B

Control Method	Peak Suspension Deflection (mm)
Passive Suspension	0.7
Original Linear Active Suspension	167.14
NNF	76.53

7.1.2 Reference LQG Controller

A reference linear state feedback controller is defined, the gains for which are to be optimised using LQR control, Section 2.1,Chapter 2. Weighting parameters α and β, control the balance between tyre and suspension deflections and body acceleration. α and β are adjusted to meet reference criteria based on the passive system's response to the following initial conditions: (a) unit body velocity (b) unit hub velocity. The passive suspension model following these two initial condition events gives peak deflection values of (a) suspension deflection = 0.0949m, (b) tyre deflection = 0.0103m.

Table 7.2 Shows the resulting gain matrix values *K*, and cost function coefficients required to match these criteria. The resulting Perfect State Feedback (PSF) system forms a simple reference active suspension system.

Table 7.2 - PSF and Passive Model Feedback Gains

	Cost Matrix values		Gain Matrix Values			
Model	α	β	K_1	K_2	K_3	K_4
Passive	-	-	0	18000	1200	-1200
PSF	70000	658	-16956	8208	1149	-2387

7.2 Kalman Filter

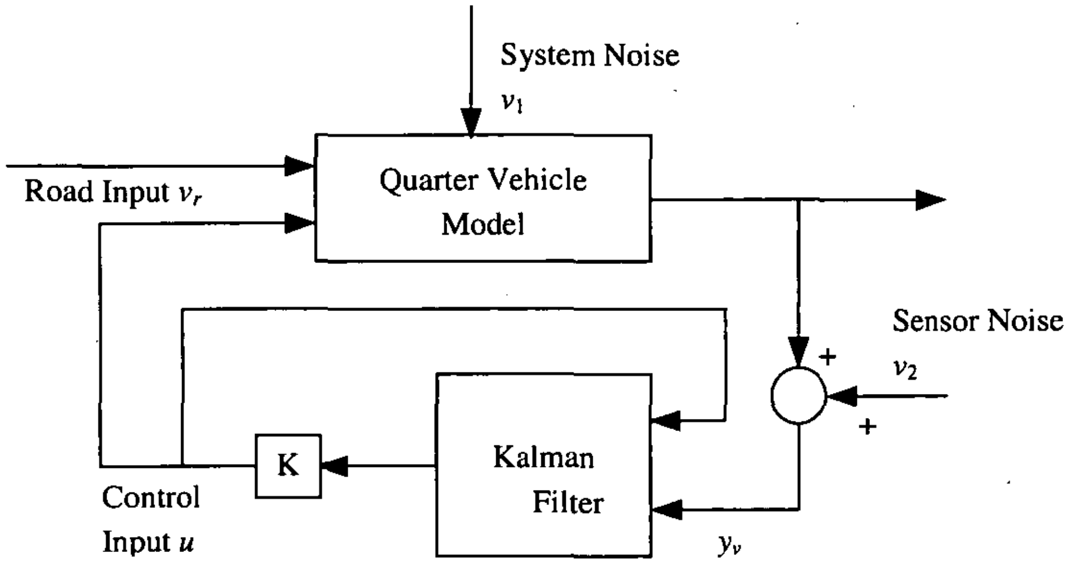


Figure 7.2 - Kalman Filter Implementation

Although assumed in previous chapters, it is unrealistic to assume perfect state feedback, so a Kalman filter [77] is introduced. To construct the filter, white noise, v_1, is included in the system model and sensor noise, v_2, is added to the outputs, y_v

$$\dot{x} = Ax + Bu + v_1 \quad y_v = Cx + Du + v_2 \tag{7.2}$$

In standard fashion the Kalman filter design assumes the following statistics for these stationary random processes:-

$$E(v_1) = E(v_2) = 0 \quad E(v_1 v_1^T) = Q_e \quad E(v_2 v_2^T) = R_e \quad E(v_1 v_2^T) = N_e \tag{7.3}$$

From this the Kalman filter is designed to estimate unmeasured states and minimise the effect of the added noise in an optimal way. The estimated states, $\hat{x}$, are calculated using the following continuous-time equation

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y_v - C\hat{x} - Du) \tag{7.4}$$

where

$$L = R_e^{-1}\left(C^T S_e + N_e^T\right) \tag{7.5}$$

and S_e is the solution to the Riccati equation,

$$A^T S_e + S_e A - (S_e C + N_e) R_e^{-1}\left(C^T S_e + N_e^T\right) + Q_e = 0 \tag{7.6}$$

It is assumed that sensors are fitted to directly measure sprung and unsprung mass accelerations $\dot{x}_4$, $\dot{x}_3$ and suspension deflection, x_2. These measurements are used as the input to the Kalman filter with the sensor errors forming the diagonal elements in the covariance matrix, R_e, and the system disturbances forming those of the Q_e matrix. N_e has been set to zero here because of the simple nature of the system noise, arising purely from the road disturbance. In general, where accelerometers are used, N_e will be non-zero [80]

$$Q_e = \begin{bmatrix} A_1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad R_e = \begin{bmatrix} 10^{-6} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.7}$$

The estimated states are now used in the feedback force, $F = K\hat{x}$, combining the state feedback model with a Kalman filter; the resulting controller provides a second reference active suspension system with Kalman State Feedback (KSF).

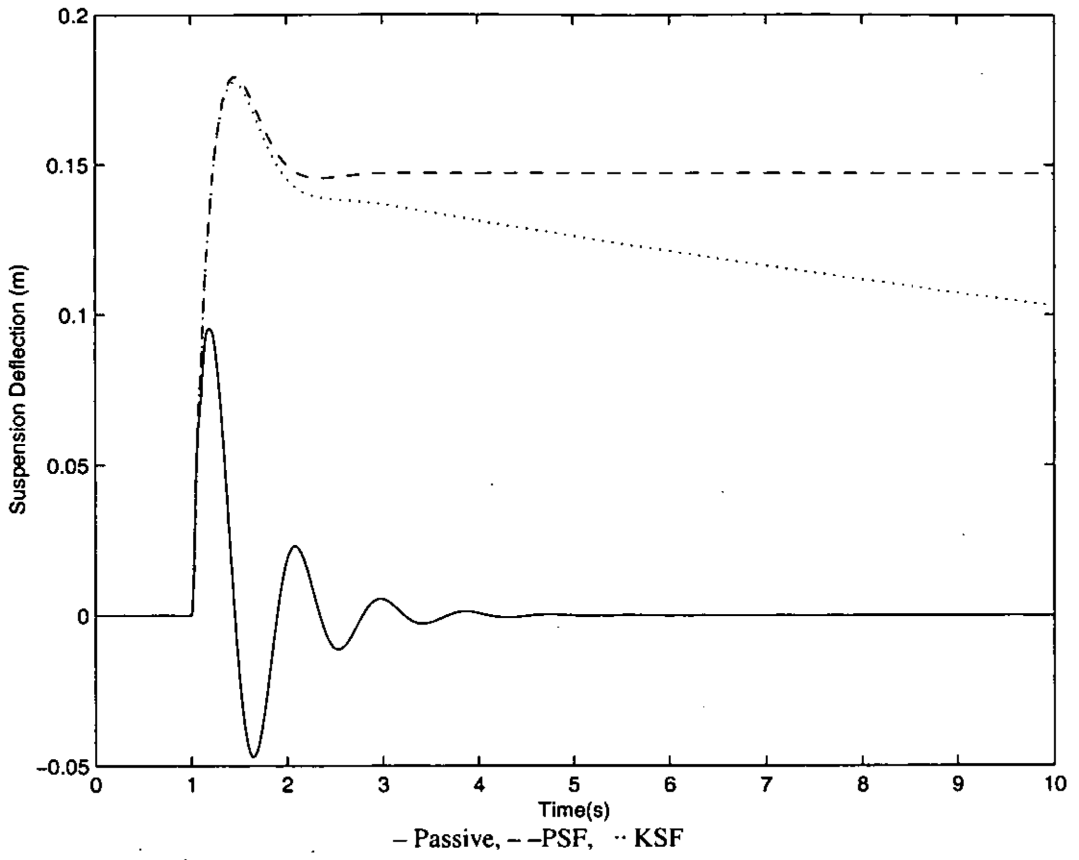


Figure 7.3 - Suspension Deflection of the Reference Models on Road A

Figure 7.3 clearly illustrates the problem of suspension drift on Road A, where the Passive suspension has no problem negotiating the unit incline. However, for the PSF model, there is a steady state suspension deflection offset as expected. The KSF model does not exhibit the same steady state offset, due to the inherent stability of the Kalman filter dynamics. However, the settling time is unacceptably long. A possible solution to this is to increase the assumed noise covariances in Equation (7.6). This will increase the speed of the Kalman Filter dynamics and therefore give a shorter settling time. Unfortunately, this will also result in a degradation of the accuracy of the state estimation. Figure 7.4 was obtained by varying the assumed sensor covariance matrix, to $\overline{R}_e$ where,

$$\overline{R}_e = 10^{A} R_e \tag{7.7}$$

and simulating the modified KSF system on Road C (Note that the actual Sensor noise levels in the model are not varied). For positive values of A, the suspension deflection is reduced, as predicted; however, body acceleration is negatively affected by the resulting poor level of feedback control. If ρ is further increased, then poor state feedback eventually degrades the suspension deflection control, and the trend is reversed.

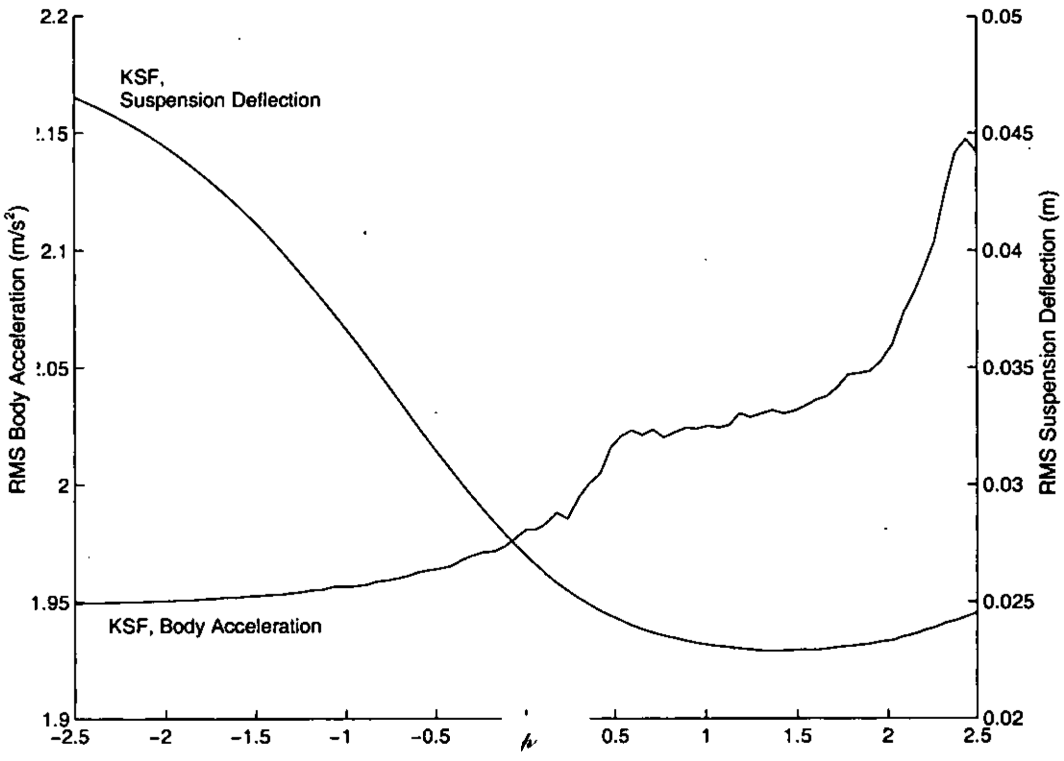


Figure 7.4 - The Effect of Sensor Error on the RMS Body Acceleration and Suspension Deflection on Road C

7.3 Extended Control Scheme

The suspension offset seen in Figure 7.3 for the PSF system is easily understood in terms of the equations of motion (2.2). Using the force feedback, $F_s(t) = Kx$, gives

$$F_s = K_1x_1 + K_2x_2 + K_3x_3 + K_4x_4 \tag{7.8}$$

In the steady state, where the vertical road velocity tends to a constant, v_l, the state suspension deflection x_2 tends to a constant value,

$$x_2 = -(K_3 + K_4)v_l / (K_2) \tag{7.9}$$

as any transients settle and x_3 and x_4 tend to v_l [81]. Once again there is clearly no drift in the passive systems as $K_3 = -K_4$.

The steady state offset can be simply removed by redefining the system velocity states, in Equation (7.10)

$$x_3 \rightarrow x_3 - v_l \quad x_4 \rightarrow x_4 - v_l \tag{7.10}$$

where v_l is the steady-state vertical road velocity. Although this is not feasible in general, it motivates an extended form of the feedback control law:-

$$F_s = \sum_{i=1}^{5} K_i x_i \tag{7.11}$$

where $x_5 = v_l$ is to be an estimate of the "low frequency" or underlying road velocity, and the zero steady state condition becomes

$$K_3 + K_4 + K_5 = 0 \tag{7.12}$$

assuming that x_3 and x_4 tend to v_l in the steady-state. Note that for a fixed forward speed, x_5 is equivalent to estimating the underlying road gradient. The concept is made more precise in the following section, where a Kalman filter estimator for the extended state vector is based on the underlying model of the vehicle driving at a fixed speed on Road C.

The system equations are as follows:-

$$\begin{aligned}
\dot{x}_1 &= -x_3 + x_5 + A_1 w_1 \\
\dot{x}_2 &= x_3 - x_4 \\
\dot{x}_3 &= (F_t - F_s) / m_w \\
\dot{x}_4 &= F_s / m_b \\
\dot{x}_5 &= -\lambda x_5 + A_2 w_2
\end{aligned} \tag{7.13}$$

As is standard, the disturbance inputs w_1 and w_2 are set to zero in the LQR controller synthesis. Using the same cost Function (2.5) as for the PSF system, the following gain matrix values result.

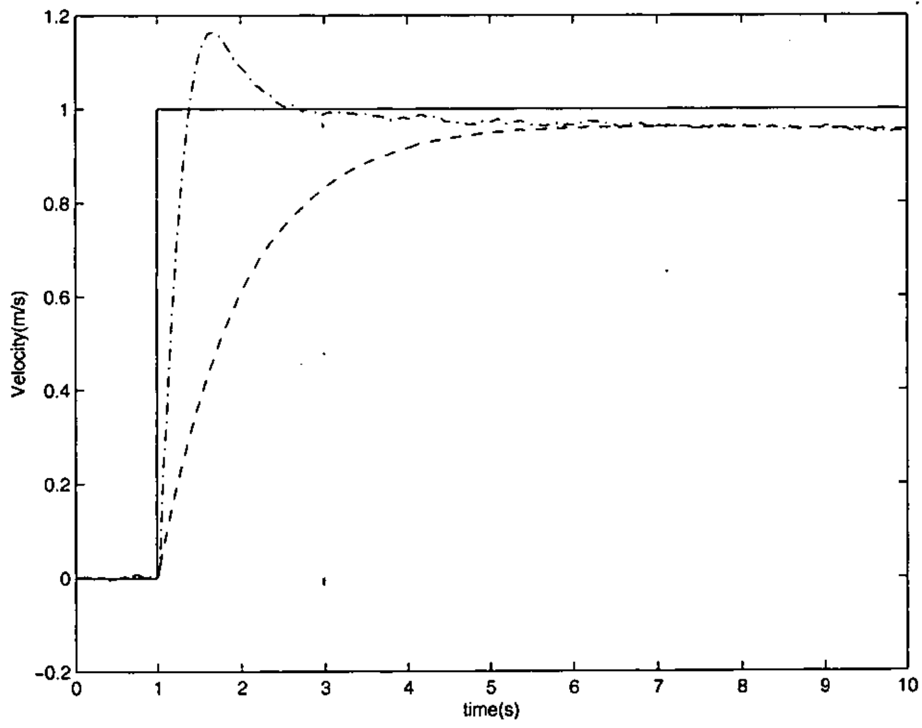
Table 7.3 - Model Feedback Gains

		Cost Matrix values		Gain Matrix Values				
Model	λ	α	β	K_1	K_2	K_3	K_4	K_5
KSF	-	70000	658	-16956	-4792	1049	-2287	-
LGF	-7×10^{-4}	70000	658	-16956	-4792	1049	-2287	1238

Note that requirement (7.12),.though not explicitly imposed, is actually satisfied (to 6 significant figures) via the controller design method, and so we can expect the steady-state offset to be removed automatically. Although $x_5 = v_l$ cannot be directly measured, it can now be estimated by a Kalman Filter. Road C is assumed as in Equations (7.13) and the feedback force becomes

$$F = \sum_{i=1}^{5} K_i \hat{x}_i \tag{7.14}$$

and the resulting "Local Gradient Feedback" (LGF) system provides a third active suspension system.



– Road Velocity, – – Velocity Estimate - Local Gradient,

– · – Velocity Estimate -Vehicle Body

Figure 7.5 - Kalman Filter Estimate of x_5 on Road A (LGF system).

Figure 7.6 shows the LGF estimate of v_l on Road A. The estimated road velocity does not follow the step input precisely, due to the disturbing influence of the initial high frequency event. The important fact is that the estimated velocity $\hat{x}_5$ tends towards that of the estimated body velocity $\hat{x}_4$, immediately after the transient (and a similar result applies to the estimated hub velocity $\hat{x}_3$). As noted above, this results in the velocity components of the suspension force F tending to zero, removing the suspension drift. The peak suspension deflection of the LGF is also 5% less than the KSF, with a negligible change in body acceleration.

7.4 Integral Control Scheme

A second proposed solution to the drift problem is based on a concept from classical control theory: steady-state errors can be removed by the introduction of an additional integrator. Integral control is adopted [82, 83], with the introduction of an additional state, defined as the integral of suspension deflection:

$$\dot{x}_5 = -\lambda x_5 + x_2 \tag{7.15}$$

This again creates a 5th order system and a fifth feedback gain K_5. As with classical PID control, the use of an additional integrator may be expected to result in increased peak suspension deflection, greater oscillation and longer settling times.

The optimisation of the gain matrix K is made more complex for the integral feedback controller (IFC), since an additional cost function parameter, γ, is needed in the cost function:-

$$J = \int_0^\infty \left(\alpha x_1^2 + \beta x_2^2 + \gamma x_5^2 + \dot{x}_4^2\right)dt \tag{7.16}$$

This additional parameter γ is necessary because feedback of x_5 also has a strong influence on the ability of the system to meet the passive system requirements of Section 7.1.1 A third condition is needed to determine the three parameters, and this is based on the low–frequency dynamic response. The maximum suspension deflection on Road B is limited to 19% of the suspension deflection range. This then allows the integral feedback controller (IFC) to match the peak deflection of the LGF system on Road B when both are combined with a Kalman filter. The resulting gain matrix values are shown in Table 7.4, together with values for the other models.

Table 7.4 - Model Feedback Gains

	Cost Matrix values			Gain Matrix Values				
Model	α	β	γ	K_1	K_2	K_3	K_4	K_5
Passive	-	-	-	0	18000	1200	-1200	-
KSF	70000	658	-	-16956	8208	1149	-2387	-
LGF	70000	658	-	-16956	8208	1149	-2387	1238
IFC	70000	484	106	-17013	8074	1149	-2371	3295

A Kalman filter is again used for estimation of the four physical states, and the active control force is now determined by

$$F = K\hat{x} + K_5 \int \hat{x}_2 dt \tag{7.17}$$

7.5 Performance Comparisons

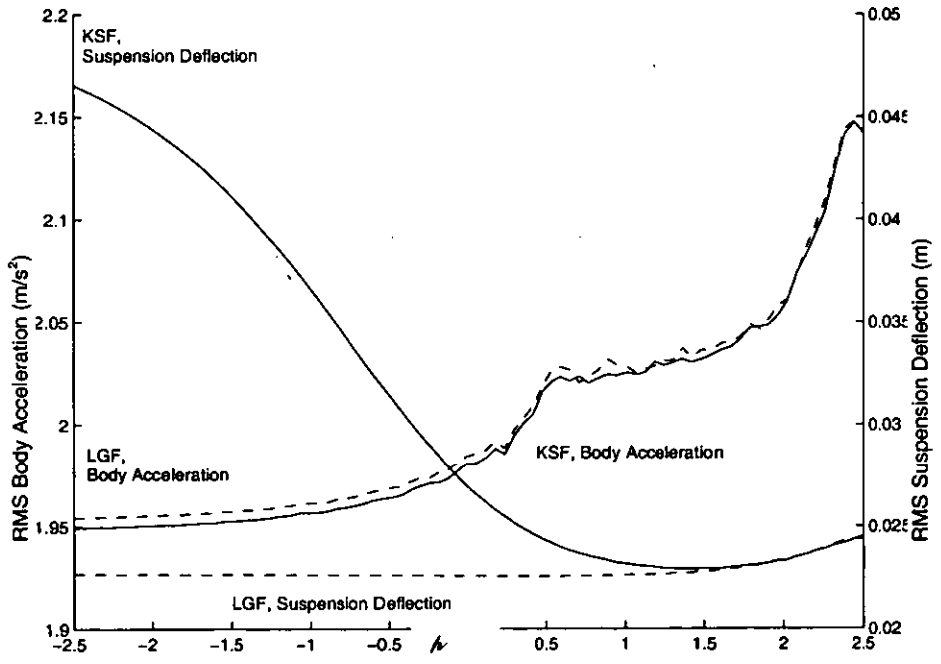


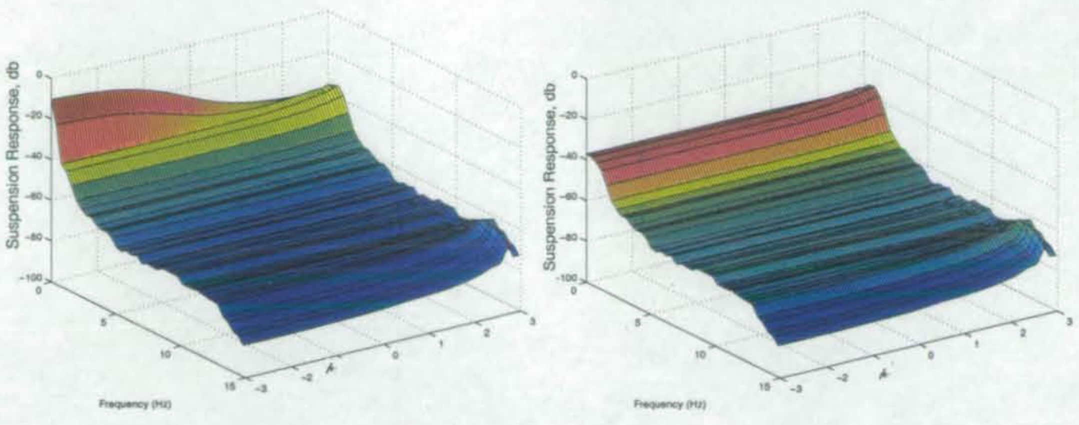
Figure 7.6 - RMS Suspension Deflections and Body Accelerations on Road C – KSF and LGF Systems.

The process of obtaining Figure 7.5 is repeated with the LGF model. The resulting plot reflects the expectations outlined in Section 7.2 – see Figure 7.7. When compared with the KSF model, LGF always benefits from low sensor noise, and accurate estimation of the systems states. The marginally increased body acceleration, compared to KSF, for negative values of p, is neither surprising nor significant.

The influence of sensor noise is further investigated by analysing its effect across the frequency range for both the KSF and LGF models – Fig. 7.8. Here we define a logarithmic suspension response.

$$\sigma(f,p) = 20\log_{10} S_{x_2}(f) \tag{7.18}$$

where $S_{x_2}(f)$ is the PSD of the suspension response on Road C. The plots show just how influential low-frequency suspension deflection is on the RMS values of Figure 7.7, where only frequencies below 5Hz are seriously affected by the changing sensor error. For KSF, suspension deflection power below 1.5Hz rises as assumed sensor error is reduced to a point where the body bounce mode can no longer be identified. The power of the suspension deflection of the LGF model is virtually unaffected by changing assumed sensor error. Only at very high sensor noise levels does the suspension performance deteriorate significantly. The suspension response of the LGF model is more adversely affected by extremely high sensor noise values, and it appears that the KSF system is less dependent on accurate state estimation. However, both systems suffer badly in terms of body acceleration response for high sensor noise, and this is a more significant limiting factor than suspension deflections.



KSF LGF

Figure 7.7 - Frequency Dependence of Suspension Responses on Road C - $\sigma(f,p)$

We now include the IFC system in the performance comparison, and revert to the standard (ρ=0) estimate for sensor noise. The frequency responses of the three system models (KSF, LGF, IFC) are shown in Figure 7.9. The body accelerations are very similar in all three cases. In terms of suspension deflection, KSF gives poor control at low frequencies, as would be expected. Also, around the body bounce frequency the IFC system gives an amplitude gain approximately 10% higher than LGF.

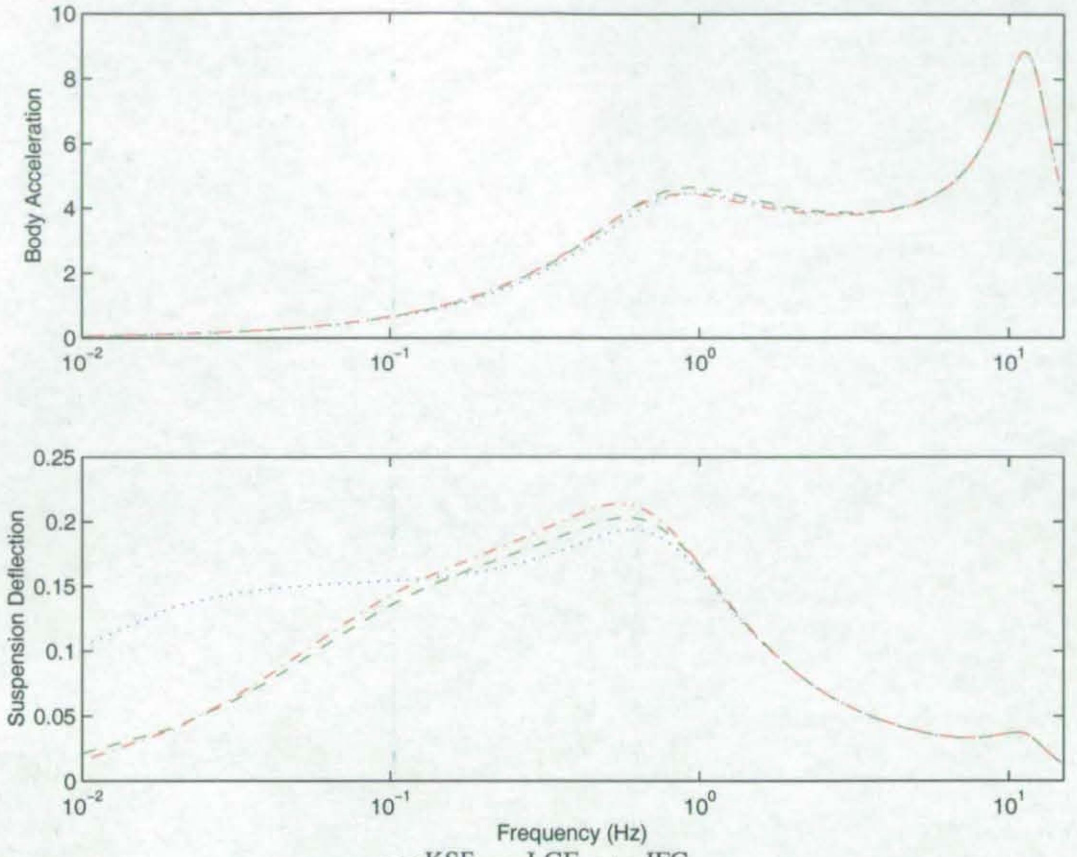


Figure 7.8 - Theoretical System Frequency Responses

A more significant problem with the IFC system can be identified in the results from Road A not shown; because the integration feedback gain is relatively small, the system takes a long time to settle, and although considerably better than the KSF model, the settling is much slower than the LGF Model. The IFC model can be re-tuned to improve this settling time, but it then no longer satisfies the design criteria of Sections 7.2 and 7.5.

The performance of KSF on Road B, Table 7.5, gives a clear indication of the need for suspension drift control; the KSF model uses 75% of the available ±100mm suspension workspace even though the road contains no significant ride disturbance. This reduces the space available to absorb further road disturbances, and increases the probability of bump stop contacts. The performance of the IFC and LGF systems are very similar on Road B, with negligible body acceleration and suspension deflections.

Road C, the most realistic of the three road profiles, ascends 15m during the 60-second simulation sample used. The mean suspension deflection is a simple measure of suspension drift, as the road is effectively a continuous ascent. On average the KSF model operates 14mm away from the equilibrium point while the others operate within 3mm. In fact, the IFC system has a mean closer to 1mm, but this is to be expected, due to the integral feedback, and the reduction is relatively insignificant. A summary of performance on the three roads is given in Table 7.5.

Table 7.5 - A Comparison of System Performance on the Three Road Models

Road Profile	Measurement	Passive	KSF	LGF	IFC
A	Peak x_2 (mm)	-	180.3	171.9	175.8
	Peak $\dot{x}_4$ (m/s^2)	-	5.2023	5.1581	5.1996
	x_2 after 10 seconds (m)	-	77.2	-5.6	-13.8
B	Peak x_2 (mm)	-	75.8	1.94	18.5
	Peak $\dot{x}_4$ (m/s^2)	-	0.2101	0.1930	0.2093
C	RMS x_2 (mm)	22.9	26.8	22.6	22.9
	RMS $\dot{x}_4$ (m/s^2)	2.3698	1.9827	1.9869	1.9806

7.6 The Extended Control Scheme with the NNF Controller

Having established a viable means of reducing the influence of low frequency disturbances on suspension deflection the strategy can be combined with the neural network controllers using one of two methods.

Method 1.

The neural network is retrained using training data, which includes the additional state representing the underlying road velocity. The way the neural network utilises the additional road information is dependent on the non-linear control strategy used to create the training data. The neural network and Kalman filter are then incorporated in the control system in the standard format, Figure 7.9.

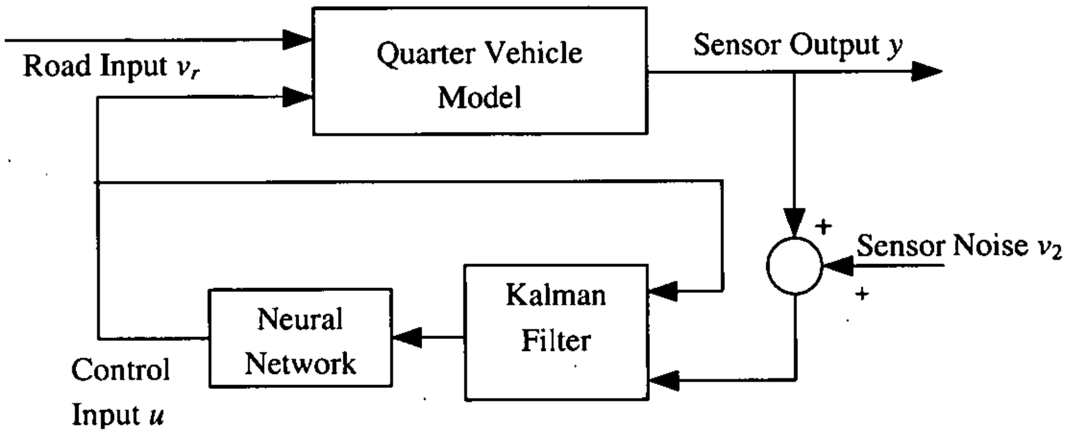


Figure 7.9 - Retrained NNF, Implementation

Method 2.

The inputs to the neural network are redefined

$$\begin{aligned}\bar{x}_1 &= \hat{x}_1 \\ \bar{x}_2 &= \hat{x}_2 \\ \bar{x}_3 &= \hat{x}_3 - \hat{v}_{rl} \\ \bar{x}_4 &= \hat{x}_4 - \hat{v}_{rl}\end{aligned} \tag{7.19}$$

thus Equations 7.14 & 7.16 are applied externally to the controller and LFC is indirectly implemented, Figure 7.10. The network remains standard, the changes applied outside the original controller. This avoids retraining of the neural network and therefore is a much faster approach to the incorporation of the underlying road gradient information.

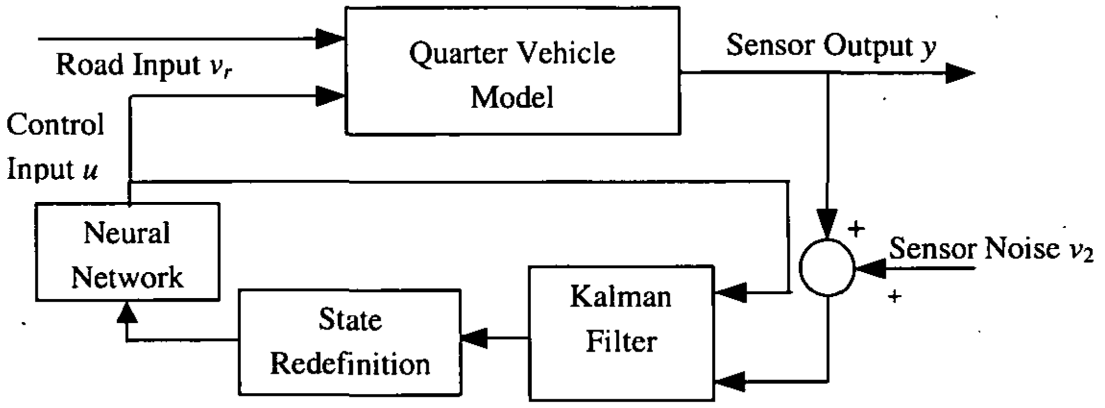
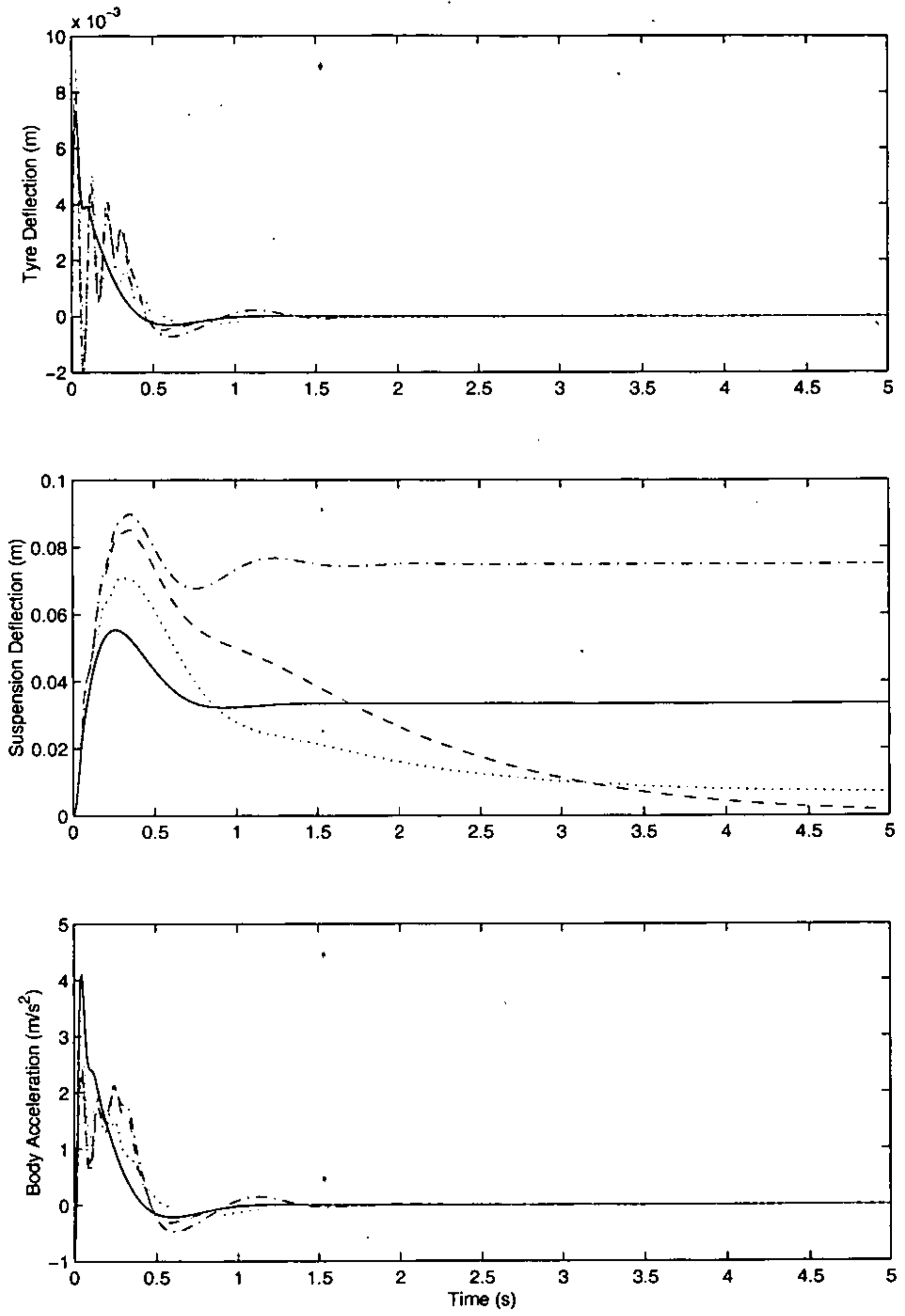


Figure 7.10 - LFC Implementation with State Redefinition

Both approaches successfully control suspension deflection drift (Figure 7.11) minimising the error, Method 2 eventually doing so more successfully, due to the direct enforcement of Equation 7.14 during the state redefinition process (Table 7.6). Method 1 exhibits less control as the relationship is not directly enforced, but responds more quickly despite the Kalman filter being the same in both cases. This leads to higher RMS body acceleration in both example cases, but may not always be the case depending on the severity of the road profile considered. Because of the simplicity of the approach and the speed of implementation, Method 2 appears to be the better of the two approaches. The results produced from the simulation of Road C indicate that the effect of the implementation of LFC are less significant using Method 2. Only a minor increase in RMS body acceleration (Table 7.7) occurs while suspension deflection drift is clearly reduced, Figure 7.12.

Table 7.6 - Suspension Deflection after 5 Seconds on Road A

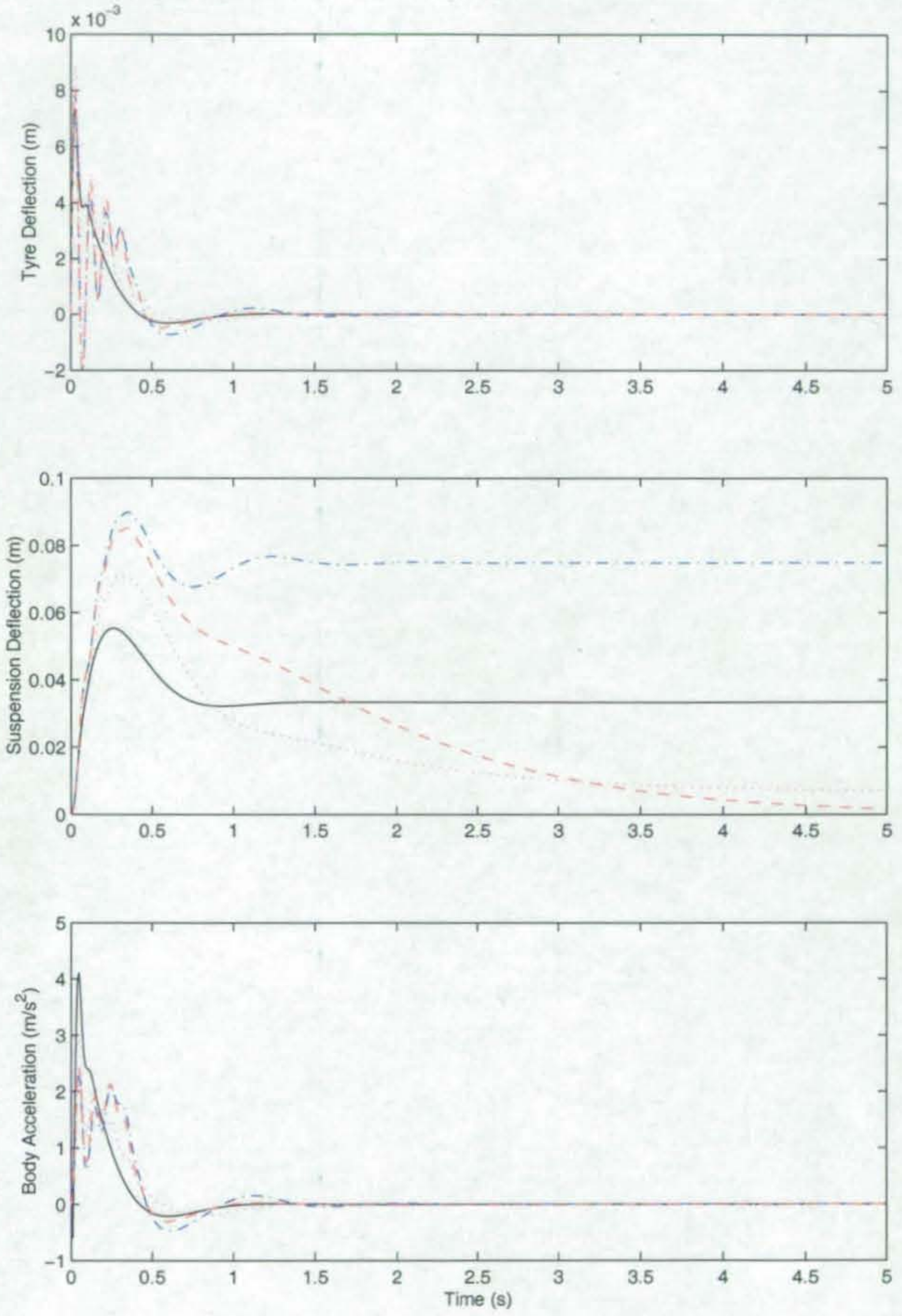
Suspension Deflection (mm)	Equivalent Linear System	NNF	Retrained NNF with LFC (Method 1)	NNF with LFC using Redefined States (Method 2)
	33.38	75.00	6.53	1.60



– Equivalent Linear System, – · – NNF, · · · Retrained NNF with LFC,

– – NNF with LFC using Redefined States

Figure 7.11 - Road A, Neural Network Response



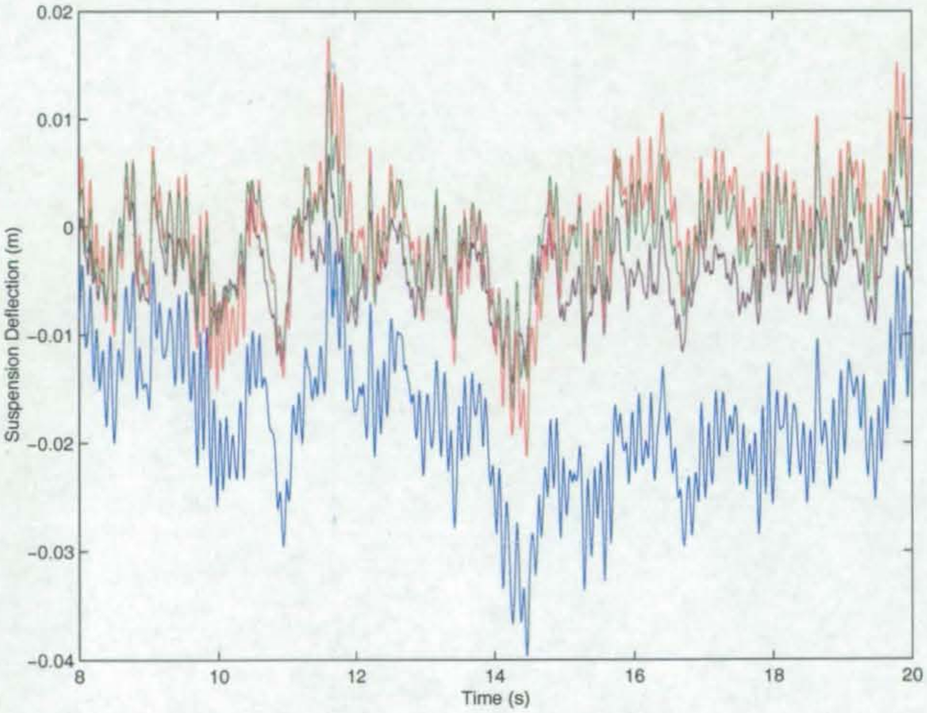
– Equivalent Linear System, – · – NNF, ··· Retrained NNF with LFC,

– – NNF with LFC using Redefined States

Figure 7.11 - Road A, Neural Network Response

Table 7.7 - Road C Performance[1]

	Equivalent Linear System	NNF	Retrained NNF with LFC (Method 1)	NNF with LFC using Redefined States (Method 2)
RMS Suspension Deflection (mm)	4.17	12.16	4.80	6.75
RMS Body Acceleration (m/s^2)	0.679	0.306	0.412	0.307



– Equivalent Linear System, – NNF, – Retrained NNF with LFC,

– NNF with LFC using Redefined States

Figure 7.12 - Simulated Road C Response

[1] Not the same section of Road C as used to create the data in Table 7.5

CHAPTER 8 DIRECT LEARNING

Chapter 8 Direct Learning

In Chapters 4 & 5, the principles of training through the use of supervised learning techniques were investigated and proven to be successful for the quarter vehicle control problem. In Chapter 8, a new approach termed H_{norm}, will be proposed for a neural network to learn the state - costate relationship directly, using learning techniques based around what has been termed "grad" based learning [84]. In Section 8.1 the principles of the algorithm for the optimisation of the network weights will be proposed. Section 8.3 will present an initial application of the approach to solve for linear quadratic regulation, for which there is a known solution given by the Riccati technique.

8.1 H_{norm} Based Optimisation of Neural Networks for Costate Estimation.

In order to develop the H_{norm} based optimisation, the following assumptions are made about the system to be controlled. The system is considered as a regulator, settling from an initial condition with no disturbances, described in the following format as a function of its current states and the current control input.

$$\dot{x} = f(x,u) \tag{8.1}$$

For the remainder of this thesis, the system is considered to be linear in form though the cost function is not necessarily quadratic. However, it is considered by the author to be feasible, with more time, to develop the technique for non-linear system models. The continuous time model incurs cost via the following cost function.

$$J = \int_0^{t_f} L dt \tag{8.2}$$

However, no neural network technique exists for the optimisation of the network weights through continuous time simulation for this type of problem. Therefore, the following technique put forward, involves no simulation, but assumes both a continuously varying system and controls.

This is advantageous in a number of factors, as it

- Allows for optimisation of continuous control problems.
- Reduces the optimisation time as simulation is often computationally expensive.
- Allows for the theoretical complete settling of the model as time tends to infinity.

In the standard form, if Lagrange multipliers are applied [85], a cost function can be developed for the optimisation of costate estimators through the minimisation of the cost given below.

$$J_p = \int_0^{t_f} \left[L + p \cdot \left(-\dot{x} + f \right) \right] dt \tag{8.3}$$

It will be assumed that $p = P(x, w)$ is the output of a neural network costate model.

Using integration by parts[2], part of the cost function (8.3) can be redefined as the Hamiltonian equation, H.

$$\begin{aligned} J_p &= \int_0^{t_f} [H + \dot{p}x] dt - [px]_0^{t_f} \\ H &= L + p.f \end{aligned} \tag{8.4}$$

Assuming that the model does not change with time, it is now possible to minimise J_p using the following gradient calculation.

$$\delta J_p = p_0 \delta x_0 - p_f \delta x_f + \int_0^{t_f} \left(\frac{\partial H}{\partial x_i} + \dot{p}_i \right) \delta x_i + \int_0^{t_f} \frac{\partial H}{\partial u_i} \delta u_i \tag{8.5}$$

[2] $\int u \frac{dv}{dx} dx = uv - \int \frac{du}{dx} v dx$

Step 1 Minimisation of J_p

The following assumptions are made

1. x_0 the initial condition, is held fixed.
2. $x_f = 0$, the final state will, if the control strategy is stable after sufficient time, tend to zero.
3. t_f is sufficiently large, as to allow x_f to tend to zero and is of a fixed length.
4. x(t) and u(t) are, at this stage, subject to free choice.

8.1.1 Hamiltonian Derivatives and their Optimal Conditions

Assuming the minimisation problem to be unique in its solution, and the cost function to be convex in form then the minimum is defined by the zero gradient condition $\delta J_p = 0$. Therefore the derivatives of the Hamiltonian give the following two conditions for optimality.

$$\frac{\partial H}{\partial x} + \dot{p} = 0 \qquad (8.6)$$

and

$$\frac{\partial H}{\partial u} = 0 \qquad (8.7)$$

The derivative of the Hamiltonian equation with respect to the changes in system states is given by; -

$$\frac{\partial H}{\partial x} = \frac{\partial L}{\partial x} + \sum_i p_i \frac{\partial f_i}{\partial x} \qquad (8.8)$$

The costate derivatives can be calculated in the following manner.

$$\dot{p} = \frac{d}{dt}\left(P(x,w)\right) = \frac{\partial P}{\partial x_i}\dot{x}_i \qquad (8.10)$$

$$S_{i,j} = \frac{\partial P_i}{\partial x_j} = S(x,w) \qquad (8.11)$$

The derivative of the Hamiltonian can be solved simply for u, as both the cost function L and the system model f are a function of the control force u.

$$\frac{\partial H}{\partial u} = 0 \Rightarrow \frac{\partial L}{\partial u} + \sum_j p_j \frac{\partial f_j}{\partial u} = 0 \qquad (8.9)$$

Solving this, the control force is assumed to be an explicit function of the current system states and the related costate function output, which is itself a function of the states, and the coefficients of the neural network used to calculate them.

$$u = U(x,p) = U(x, P(x,w)) \quad (8.12)$$

However at this stage it is not possible to apply the implied relationship $\dot{x} = f$ between the change in states and the system model.

Assuming all of the above, then $\frac{\partial H}{\partial x} + \dot{p} = 0$ implies the "Jp-optimal" trajectory and will satisfy the equations

$$\frac{\partial H}{\partial x} + S\dot{x} = 0 \qquad (8.13)$$

Assuming S is invertible

$$\dot{x} = -S^{-1}\frac{\partial H}{\partial x} \qquad (8.14)$$

Step 2: The costate estimator *P*(*x*,*w*) is improved through the adaptation of the network weights.

The Lagrange multiplier approach [85] suggests that Equations 8.12 and 8.14 should be imposed to get a J_p – optimal solution set of "all possible" solutions P(x,w), before finally choosing the one that also coincides with the constraint $\dot{x} = f$. This, in reality is clearly not feasible. A more realistic approach is to choose an initial model *P*(*x*,*w*), and then attempt to improve the fit of Equation 8.13 to the desired state equations $\dot{x} = f$. This motivates a subsidiary optimisation of the form.

$$F = \int_0^{t_f} \left|-\dot{x} + f\right|^2 dt \quad \text{or more generally} \quad F = \int_0^{t_f} \left(-\dot{x} + f\right)^T Q\left(-\dot{x} + f\right) dt \quad (8.15)$$

where the derivatives of x are given by (8.1), and Q is some positive definite or positive semi-definite matrix.

The fitting criterion can be summed over a large number of trajectories, or more simply over a large number of points in the state space, in order to ensure a generalised solution. Therefore, the cost function for the optimisation of the costate estimator takes the following form

$$\begin{aligned} F &= \sum_x (-\dot{x}+f)S^T S(-\dot{x}+f) \\ &= \sum_x \left|S(-\dot{x}+f)\right|^2 \\ &= \sum_x \left|\frac{\partial H}{\partial x}+S.f\right|^2 \end{aligned} \tag{8.16}$$

Making the following definition

$$h_i(x,w)=\left(\frac{\partial H}{\partial x_i}+S_{ij}f_j\right) \tag{8.17}$$

Equation 8.16 can be simplified

$$F=\sum_x h^2 \tag{8.18}$$

Whilst other approximation techniques exist, this thesis deals solely with the use of neural networks for identification of the relationship between system state and costate. Although techniques do exist for the optimisation of neural networks without knowledge of the cost gradient, whenever it is feasible to calculate the cost function gradient with respect to the network weights, it has proven prudent to do so, for example using gradient descent methods for the optimisation. For convenience let δ_w represent a small change in any quantity induced by a variation in the network weights e.g. $\delta_w P_m = \frac{\partial P_m}{\partial w_n}\delta w_n$. Then

$$\delta_w F = 2\sum_x h_i \delta_w h_i \tag{8.19}$$

Therefore, we must calculate the derivative Equation 8.17 with respect to the neural network weights, the first part of which is given as follows.

$$\delta_w \frac{\partial H}{\partial x} = \frac{\partial}{\partial p_i}\left(\frac{\partial H}{\partial x}\right)\delta_w p_i + \frac{\partial}{\partial u_j}\left(\frac{\partial H}{\partial x}\right)\frac{\partial U_j}{\partial p_i}\delta_w p_i \qquad (8.20)$$

The second element of Equation 8.20 is identically zero as

$$\frac{\partial}{\partial u_j}\left(\frac{\partial H}{\partial x}\right) = \frac{\partial^2 H}{\partial u_j \partial x} = \frac{\partial}{\partial x}\left(\frac{\partial H}{\partial u_j}\right) = 0 \qquad (8.21)$$

and therefore, given the derivative of the Hamiltonian equation with respect to the costate values, we obtain

$$\delta_w \frac{\partial H}{\partial x_i} = \frac{\partial}{\partial x_i}\frac{\partial H}{\partial p_j}\delta_w P_j = \left(\frac{\partial f_j}{\partial x_i}\right)\delta_w P_j \qquad (8.22)$$

The derivative of the second component of Equation 8.17 is

$$\delta_w(S_{ik} f_k) = \delta_w\left(\frac{\partial P_i}{\partial x_k} f_k\right) = \left(\frac{\partial}{\partial x_k}\delta_w P_i\right) f_k + \frac{\partial P_i}{\partial x_k}\frac{\partial f_k}{\partial u_l}\frac{\partial U_l}{\partial p_m}\delta_w P_m \qquad (8.23)$$

Therefore $\delta_w\, h_i$ is

$$\delta_w h_i = \frac{\partial f_j}{\partial x_i}\delta_w P_j + \frac{\partial P_i}{\partial x_k}\frac{\partial f_k}{\partial u_l}\frac{\partial U_l}{\partial p_m}\delta_w P_m + \left(\frac{\partial}{\partial x_k}\delta_w P_i\right) f_k \qquad (8.24)$$

If the dummy variable m is substituted for j in the first term of Equation 8.24 the following rearrangement can be made

$$\delta_w h_i = \left(\frac{\partial f_m}{\partial x_i} + \frac{\partial P_i}{\partial x_k}\frac{\partial f_k}{\partial u_l}\frac{\partial U_l}{\partial p_m}\right)\delta_w P_m + f_k\left(\frac{\partial}{\partial x_k}\delta_w P_i\right) \qquad (8.25)$$

To simplify, we make the following definition from first part of Equation 8.25

$$\mathscr{G}_{im}(x,w) = \frac{\partial f_m}{\partial x_i} + \frac{\partial P_i}{\partial x_k}\frac{\partial f_k}{\partial u_l}\frac{\partial U_l}{\partial p_m} \qquad (8.26)$$

This allows the derivative of the cost Function 8.16 with respect the network weights to be written as follows

$$\begin{aligned}
\delta_w F &= 2\sum_x h_i\left[\mathscr{g}_{im}(\delta_w P_m) + f_k \delta_{im}\frac{\partial}{\partial x_k}(\delta_w P_i)\right] \\
&= 2\sum_x \left[h_i \mathscr{g}_{im}(\delta_w P_m) + h_i f_k \delta_{im}\frac{\partial}{\partial x_k}(\delta_w P_i)\right] \\
&= 2\sum_x \left[h_i \mathscr{g}_{im} + h_m f_k \frac{\partial}{\partial x_k}\right]\delta_w P_m
\end{aligned} \tag{8.27}$$

Having made the required substitutions of the dummy variables, the derivative of the cost function with respect to the networks weights can now be defined as follows.

$$\begin{aligned}
\delta_w F &= 2\sum_x \left\{h_i \mathscr{g}_{im}\frac{\partial P_m}{\partial w_n} + h_m f_k \frac{\partial^2 P_m}{\partial x_k \partial w_n}\right\}\delta w_n \\
G_n &= 2\sum_x \left\{h_i \mathscr{g}_{im}\frac{\partial P_m}{\partial w_n} + h_m f_k \frac{\partial^2 P_m}{\partial x_k \partial w_n}\right\}
\end{aligned} \tag{8.28}$$

G_n could now be used in a simple gradient descent algorithm such as back-propagation or the direct delta method.

8.2 Implementation of H_{norm} Optimisation using the Levenberg Marquardt Method

The Levenberg Marquardt method is based around the least squares cost function 8.18 but does not require the derivative of the cost function as a whole with respect to the network weights, Equation 8.28. In order to calculate the approximate Hessian term, Equation 4.10, only the derivative of the squared element, *h* is required. This is defined in Equation 8.25 and having applied the simplification, Equation 8.26, is resolved through the following steps to the simplified term 8.31

$$\begin{aligned}
\delta_w h &= \sum_x \left[\mathscr{g}_{im}(\delta_w P_m) + f_k \delta_{im}\frac{\partial}{\partial x_k}(\delta_w P_i)\right] \\
&= \sum_x \left[\mathscr{g}_{im} + f_k \frac{\partial}{\partial x_k}\right]\delta_w P_m
\end{aligned} \tag{8.29}$$

Having made the required substitutions of the dummy variables, the derivative of h with respect to the networks weights can now be defined as follows with the assumption $\delta_w P_m = \frac{\partial P_m}{\partial w_n} \delta w_n$.

$$\delta_w h = \sum_x \left\{ \mathscr{g}_{im} \frac{\partial P_m}{\partial w_n} + f_k \frac{\partial^2 P_m}{\partial x_k \partial w_n} \right\} \delta w_n \tag{8.30}$$

$$\frac{\partial h_i}{\partial w_n} = \mathscr{g}_{im} \frac{\partial P_m}{\partial w_n} + f_k \frac{\partial^2 P_m}{\partial x_k \partial w_n} \tag{8.31}$$

8.3 LQR Optimisation and Confirmation

During the remainder of this chapter and in the subsequent chapter, the technique will be evolved for application to the quarter vehicle model with non-quadratic cost function. The technique is applied initially to the problem of linear quadratic regulation for which a known solution can be obtained using the Riccati equation (a one step linear optimisation technique). This will serve to verify the approach and highlight a number of key factors to be resolved in the final implementation of the technique. In Chapter 9 the technique will be further developed, before being applied to the final problem of the non-quadratic cost function, quarter vehicle problem.

Starting then with a quadratic cost function, Equation 2.4, applied to a quarter vehicle model in Chapter 2, the optimisation is performed on the most basic of networks - a single layer network of four purelin neurons having 16 weights. The 16 weights form a 4 by 4 matrix that if correct will map directly to the matrix solution given by the Riccati Equation 2.6.

The system is modelled using standard linear state space modelling techniques, with

$$f = Ax + Bu \tag{8.32}$$

The quadratic cost function is defined via matrices Q and R

$$J = \frac{1}{2}\int_0^{t_f}\left(x^T Qx + u^T Ru\right)dt \tag{8.33}$$

This gives the following Hamiltonian function.

$$H = p\left(A.x + B.u\right) + \frac{1}{2}\left[x^T Qx + u^T Ru\right] \tag{8.34}$$

The derivative of the Hamiltonian with respect to the control force *u*, Equation 2.26, can be rearranged to give *u* the actuator force as a function of *p*, the neural network output.

$$\begin{aligned}
&\frac{\partial H}{\partial u_k} = p_i B_{ik} + R_{ik} u_i = 0 \\
&B_{ki}^T p_i + R_{ki}^T u_i = 0 \quad R^T = R \\
&B^T p + Ru = 0 \\
&\therefore u = -R^{-1} B^T p
\end{aligned} \tag{8.35}$$

The costate model is a linear function of the system states for the quadratic cost function and is therefore of the following form:

$$p = Sx = Wx \tag{8.36}$$

This can be represented as a single layer network as described above, with weights *W*. Given the linear form of the state model and cost function, the derivative of the Hamiltonian equation with respect to the system states is derived as follows.

$$\frac{\partial H}{\partial x} = A^T p + Qx \tag{8.37}$$

8.3.1 Linear Least Squares Cost Function

Equation 8.30 gives the following form for h in the least squares cost function.

$$\begin{aligned} h_i &= \frac{\partial H}{\partial x_i} + S_{ij} f_j \\ &= \left(A^T S + Q\right)_{ij} x_j + S_{ij}\left(Ax + Bu\right)_j \\ &= \left(A^T S + Q\right)_{ij} x_j + S_{ij}\left(Ax - BR^{-1}B^T Sx\right)_j \\ &= \left[A^T S + SA - SBR^{-1}B^T S + Q\right]x \end{aligned} \tag{8.38}$$

This closely resembles the algebraic Riccati equation. In order to perform Levenberg Marquardt least squares optimisation the derivative of h with respect to the network weights, w must be calculated.

$$\begin{gathered} \frac{\partial h_i}{\partial w_n} = \mathscr{g}_{im} \frac{\partial P_m}{\partial w_n} + f_k \frac{\partial^2 P_m}{\partial x_k \partial w_n} \\ \mathscr{g}_{im}(x) = \frac{\partial f_m}{\partial x_i} + \frac{\partial p_i}{\partial x_k} \frac{\partial f_k}{\partial u_l} \frac{\partial U_l}{\partial p_m} \qquad \mathscr{g}_{im} = A_{mi} - S_{ik} B_{kl}\left(R^{-1}B^T\right)_{lm} \\ \frac{\partial P_m}{\partial w_n} = x \\ \frac{\partial^2 P_m}{\partial x_k \partial w_n} = 1 \\ \frac{\partial h_i}{\partial w_n} = A_{mi} - S_{ik} B_{kl}\left(R^{-1}B^T\right)_{lm} x_n + \left(Ax + Bu\right)_k \\ S = w \end{gathered} \tag{8.39}$$

Figure 8.1 shows the results of two optimisation runs. Both were started using random initial network weights, and trained for 3 minutes using the Levenberg Marquardt optimisation for 1000 initial condition points, using batch optimisation. The computer used for the training was an 800Mhz AMD Athlon®, with 256Mb of RAM, and the optimisations were run using Matlab® 6.

Only one run found the correct solution, the other an incorrect one. There are many possible solutions to the problem as posed by the cost function h, only one is correct. The probability of any of these solutions occurring would appear to be approximately equal and therefore finding an incorrect solution is more likely.

However the correct solution to the LQR problem can easily be identified as the derivative of the trained neural network, S is positive definite [86][3].

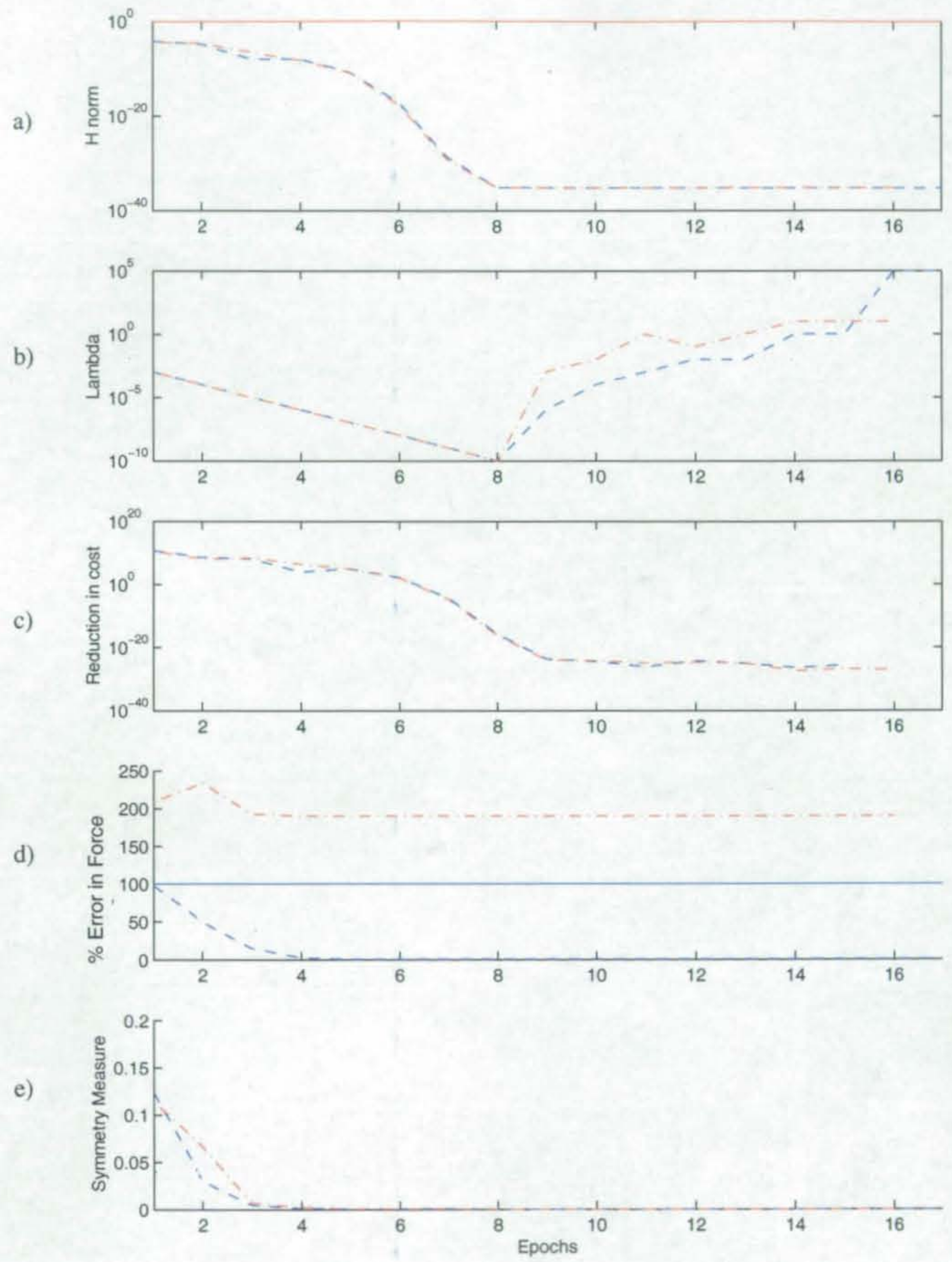
Given that each optimisation run can be completed in 3 minutes or less for the LQR problem, it is possible to complete sufficient runs to identify the correct solution. During the experimental period 10 runs was always sufficient to find at least one correct solution.

The Riccati approach to this problem always finds the correct solution and is a one step process. The approach posed here is not put forward as an alternative, but is used to verify the principles of the optimisation process before progressing to the more complex non-linear problem.

While the results of the LQR optimisation runs indicate that the approach can be successful, when the problem progresses to the non-quadratic cost function the probability of identifying an incorrect solution is likely to be much harder, and the chance of finding the correct solution much reduced.

It should also be noted that aside from the network derivative being positive definite, there is nothing to differentiate between the two runs, successful or unsuccessful. The H_{norm} values for both optimisations being very similar, Figure 8.1.a), there is also no significant difference between the rate of optimisation, Fig. 8.1.c) or the selection of λ during the optimisation process, Fig. 8.2.b). The Symmetry [87] of the neural network derivative, costate calculation $\frac{\partial p_i}{\partial x_j} = \frac{\partial p_j}{\partial x_i}$, is an indication that the optimisation process has completed correctly, and is also a secondary measure as to how the optimisation process is progressing. However, there is nothing to differentiate between the symmetry of the two solutions highlighted in Fig. 8.1.e) when optimisation is complete. The only true indication of how well the network is doing is given by the percentage error between the force calculated by the neural network and that calculated via the LQR method, Fig. 8.1.d). However, correct force information will not be available for comparison when the approach is applied to a non-quadratic cost functions.

[3] For a symmetric, positive definite matrix A, $A=A^T$ and $x^TAx > 0$ for all $x \neq 0$



-·- Unsuccessful Optimisation, - - Successful Optimisation

Figure 8.1 - LQR Costate Estimation Network Training

Chapter 9 Development of the Hnorm Optimisation Routine for the Control of the Quarter Vehicle Model

Chapter 9 Development of the H_{norm} Optimisation Routine for the Control of the Quarter Vehicle Model

In Chapter 8 a "grad" based training technique for neural network costate estimation was proposed and applied to the Linear Quadratic Regulation problem. This highlighted a number of difficulties and a need for further development before the technique could be applied to the more complex task of costate estimation for a quarter vehicle model with a non-quadratic cost function leading to a non-linear feedback controller. In Chapter 9, the technique is developed further using a single degree of freedom mass and actuator model. The principles of neural network shaping, LQR adaptation, point weighting, and weight decay will be applied to increase the probability of successful training of the network as a costate estimator. Finally, the technique is applied to the quarter vehicle problem.

9.1 The Single Degree of Freedom, Mass Actuator Model.

In order to further develop the H_{norm} technique without the relative complexity of the 2-degree of freedom, 4 state, quarter vehicle model, the introduction of a single degree of freedom (SDOF), Figure 9.1, 2 state model is made for this chapter. The SDOF model has the added advantage that data can be plotted as a surface against both the states simultaneously, which allows for greater visualisation of the problem. The model represents a mass supported by a continuously variable force actuator, which may be considered to be either traversing an irregular surface or for example supported on a laboratory shaker.

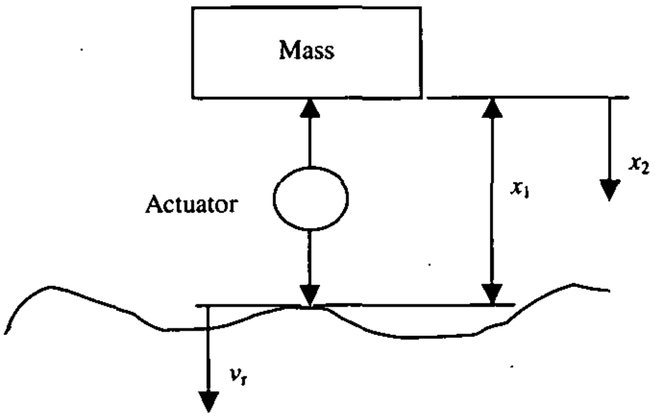


Figure 9.1 - Single Degree of Freedom Model

The model can be represented mathematically using state variables, for which the matrices are shown below including a system disturbance v_r. However as in Chapter 8, for the purposes of the optimisation process the model will be considered as a settling problem from initial conditions over time, involving only matrices A and B.

$$\dot{x} = Ax + Bu + Gv_r$$
$$A = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1/m \end{bmatrix} \quad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{9.1}$$

The cost function for the SDOF model is formulated in a similar way to the cost Function (4.1) in Chapter 4: A quadratic cost is applied to the actuator displacement and supplemented by a higher order cost with the aim of limiting the range of operation for the actuator. This may be considered in a similar context to the bumpstops of the quarter vehicle model introduced in Chapter 2, or simply representative of the limited range of the actuator. These displacement restrictions must then be offset against a cost on the acceleration of the mass and hence the degree of force, u applied by the actuator to form the following cost function.

$$L(x,u) = 0.5\left(\alpha x_1^2 + \beta x_1^n + \left(u/m\right)^2\right)$$
$$\alpha = 500 \quad \beta = 5\times 10^{11} \quad n = 10 \tag{9.2}$$

9.2 SDOF Reference data.

In order to verify the success of the neural network in the learning of the state - costate relationship for the SDOF model, a set of reference data was created using Marsh's method outlined in Chapter 2. The details for the derivation of the appropriate equations are presented below.

The Hamiltonian equation is derived from the cost function (9.2) and state Equations (9.1).

$$H(t) = 0.5\left(\alpha x_1^2 + \beta x_1^{10} + \left(u/m\right)^2\right) + p_1\dot{x}_1 + p_2\dot{x}_2 \tag{9.3}$$

The derivative of the Hamiltonian equation is taken with respect to the states inorder to obtain the derivative equations for the costates.

$$\begin{aligned} p_1\dot{x}_1 &= -p_1x_2 \\ p_2\dot{x}_2 &= \frac{p_2u}{m} \\ \dot{p}_1 &= -\frac{\partial H}{\partial x_1} = -\alpha x_1 - 5\beta x_1^9 \\ \dot{p}_2 &= -\frac{\partial H}{\partial x_2} = p_1 \end{aligned} \tag{9.4}$$

The derivative of the Hamiltonian with respect to the control force u is taken in order to establish the force as a function of the costates.

$$\begin{aligned} \frac{\partial H}{\partial u} &= \frac{u}{m^2} + \frac{p_2}{m} = 0 \\ u &= -mp_2 \end{aligned} \tag{9.5}$$

All other elements of the calculation process are as defined in Chapter 2.

9.3 H$_{\text{norm}}$ Formulation for a Non-Quadratic Cost Function Problem

The non-quadratic cost function is assumed for the SDOF model and leads to the following Hamiltonian equation.

$$H = p_i\left(A_{ij}x_j + B_{ij}u_j\right) + \tfrac{1}{2}\left(Q_{ij}x_j^2 + Q_{10_{ij}}x_j^{10} + R_{ij}u_j^2\right) \tag{9.6}$$

The derivative $\partial H/\partial u$ is rearranged to give u the actuator force as a function of p, the output of the neural network costate estimator, $p = P(W,x)$.

$$\begin{aligned}
&\frac{\partial H}{\partial u_k} = P_i B_{ik} + R_{ik}u_i = 0 \\
&B_{ki}^T p_i + R_{ki}^T u_i = 0 \quad R^T = R \\
&B^T p + Ru = 0 \\
&\therefore u = -R^{-1}B^T p
\end{aligned} \tag{9.7}$$

The derivative of the Hamiltonian equation with respect to the system states is

$$\frac{\partial H}{\partial x} = p_j A_{ji} + Q_{ij}x_j + 5Q_{10_{ij}}x_j^9 = A^T p + Qx + 5Q_{10}x^9 \tag{9.8}$$

from which can be derived the following equation for the calculation of the network performance cost.

$$\begin{aligned}
h_i &= \frac{\partial H}{\partial x_i} + S_{ij}f_j \\
&= A^T p + Qx + 5Q_{10}x^9 + \frac{\partial P}{\partial x}_{ij}\left(Ax + Bu\right)_j \\
&= A^T p + Qx + 5Q_{10}x^9 + \frac{\partial P}{\partial x}_{ij}\left(Ax - BR^{-1}B^T p\right)_j
\end{aligned} \tag{9.9}$$

In order to perform Levenberg Marquardt least squares optimisation of the network performance cost, the derivative of h with respect to the network weights, w, must be defined. From Equation 8.31 the following equations can be established for the SDOF model.

$$\begin{aligned}
&\frac{\partial h_i}{\partial w_n} = g_{im}\frac{\partial P_m}{\partial w_n} + f_k\frac{\partial^2 P_m}{\partial x_k \partial w_n} \\
&g_{im}(x) = \frac{\partial f_m}{\partial x_i} + \frac{\partial p_i}{\partial x_k}\frac{\partial f_k}{\partial u_l}\frac{\partial U_l}{\partial p_m} \\
&g = A - \frac{\partial P}{\partial x}BR^{-1}B^T \\
&\frac{\partial h_i}{\partial w_n} = A_{mi} - \frac{\partial P}{\partial x}B_{kl}\left(R^{-1}B^T\right)_{lm}x_n + \left(Ax+Bu\right)_k\frac{\partial^2 P_m}{\partial x_k \partial w_n}
\end{aligned} \tag{9.10}$$

Because $\partial^2 P_m / \partial x_k \partial w_n = 1$ the calculation of the derivatives of h with respect to the network weights is simplified for the LQR problem in Chapter 8. However, for the non-quadratic problem this is not the case: Its analytical derivation and computation are complex. This is avoided using the following numerical approximation.

Because the overall change in x can be considered proportional to the direction of change in the system states for the current control action u calculated by the costate estimator, the following numerical approximation can be applied: If a small change is made to the current states, which is proportional to the system derivatives at that state.

$$\begin{aligned}
&f = Ax - BR^{-1}B^T P \\
&x_2 = x_1 + \varepsilon f
\end{aligned} \tag{9.11}$$

and the derivative of the costate estimator is again calculated with respect to the network weights, then $f_k \partial^2 P_m / \partial x_k \partial w_n$ can be approximated using the following numerical estimation.

$$f_k\frac{\partial^2 P_m}{\partial x_k \partial w_n} = \frac{\dfrac{\partial P_m(x_2)}{\partial w_n} - \dfrac{\partial P_m(x_1)}{\partial w_n}}{\varepsilon} \tag{9.12}$$

9.4 Training of a standard form Neural Network for the Non-linear SDOF Problem.

Figure 9.2 shows the results of ten training runs, each performed using the standard form of neural network; 20 Tansig neurons in the hidden layer and 2 purelin functions in the output layer, on a fixed data set of 1000 points. Each training run lasted 1 hour using an AMD Athlon® 800Mhz computer with 256 MB of RAM. None of the ten optimisations successfully learned the state - costate relationship for the SDOF model even though the network size had proved sufficient during the supervised learning performed in Chapter 4. As predicted, the likelihood of identifying the incorrect solution for at least part of the operating range of the network is too high. The inability to use symmetry as an indicator of success is also highlighted; while some of the networks with higher levels of fit exhibited low symmetry values, the same is true of less successful training runs. Although the clarity of Figure 9.2 is not good, it is clear that the likelihood of a successful training run is very low indeed.

Figure 9.3, shows one of the more successful training runs in more detail. The error between the force calculated using Marsh's method and that calculated by the neural network costate estimator is about 70%. However, the resulting control actions do not stabilise the mass actuator system.

The upper plots in Figure 9.3 show the costate values; the solid surface, the output of the neural network estimator; the lines, the calculated values using Marsh's method. The centre pair of plots display the error between the two. The lower pair show the distribution of the optimisation cost, h_i, within the operational range of the network. The fact that there is no direct relationship between the costates and the H_{norm} cost is reflected in the lower four plots.

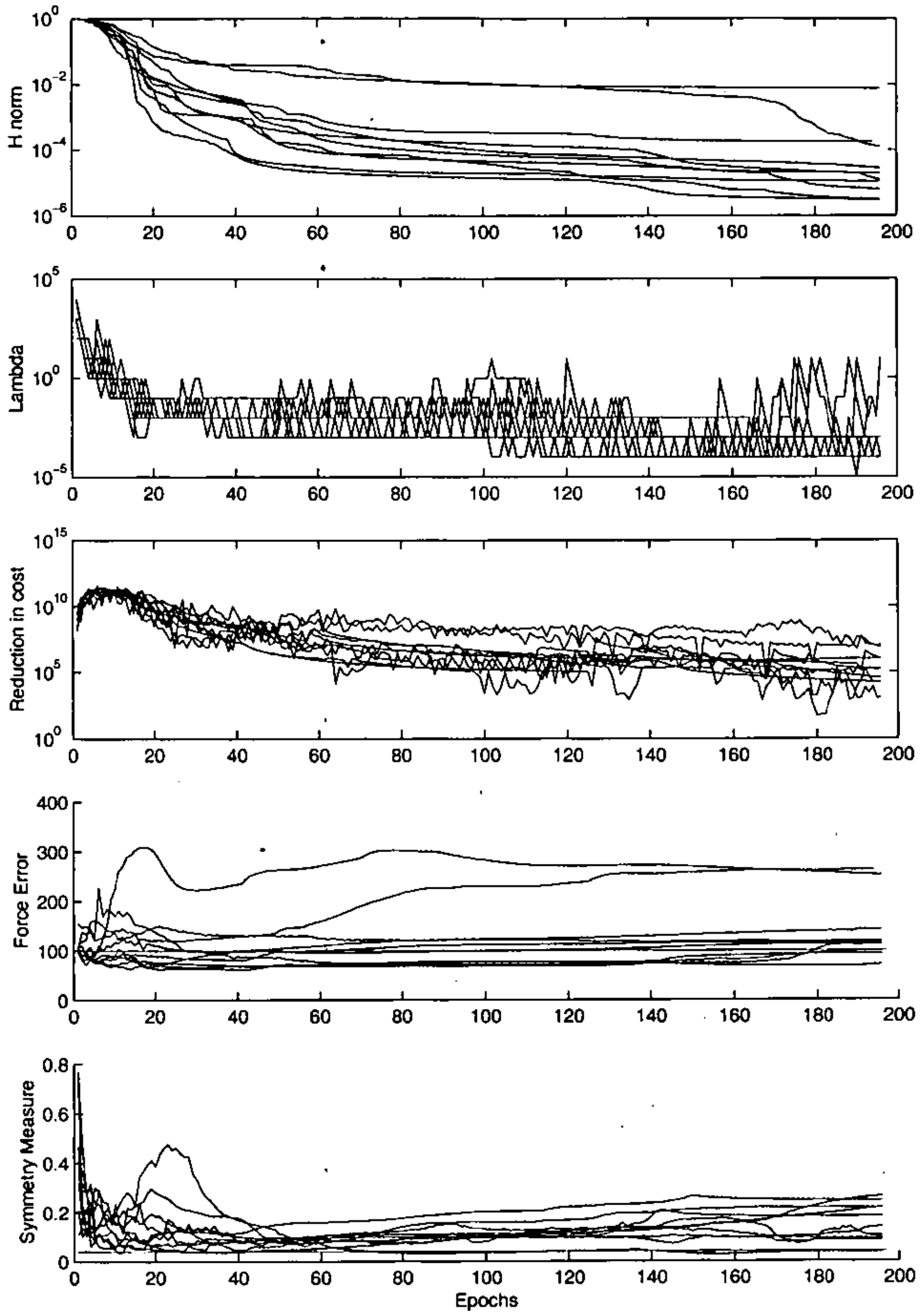


Figure 9.2 - Initial SDOF H_{norm} Optimisations

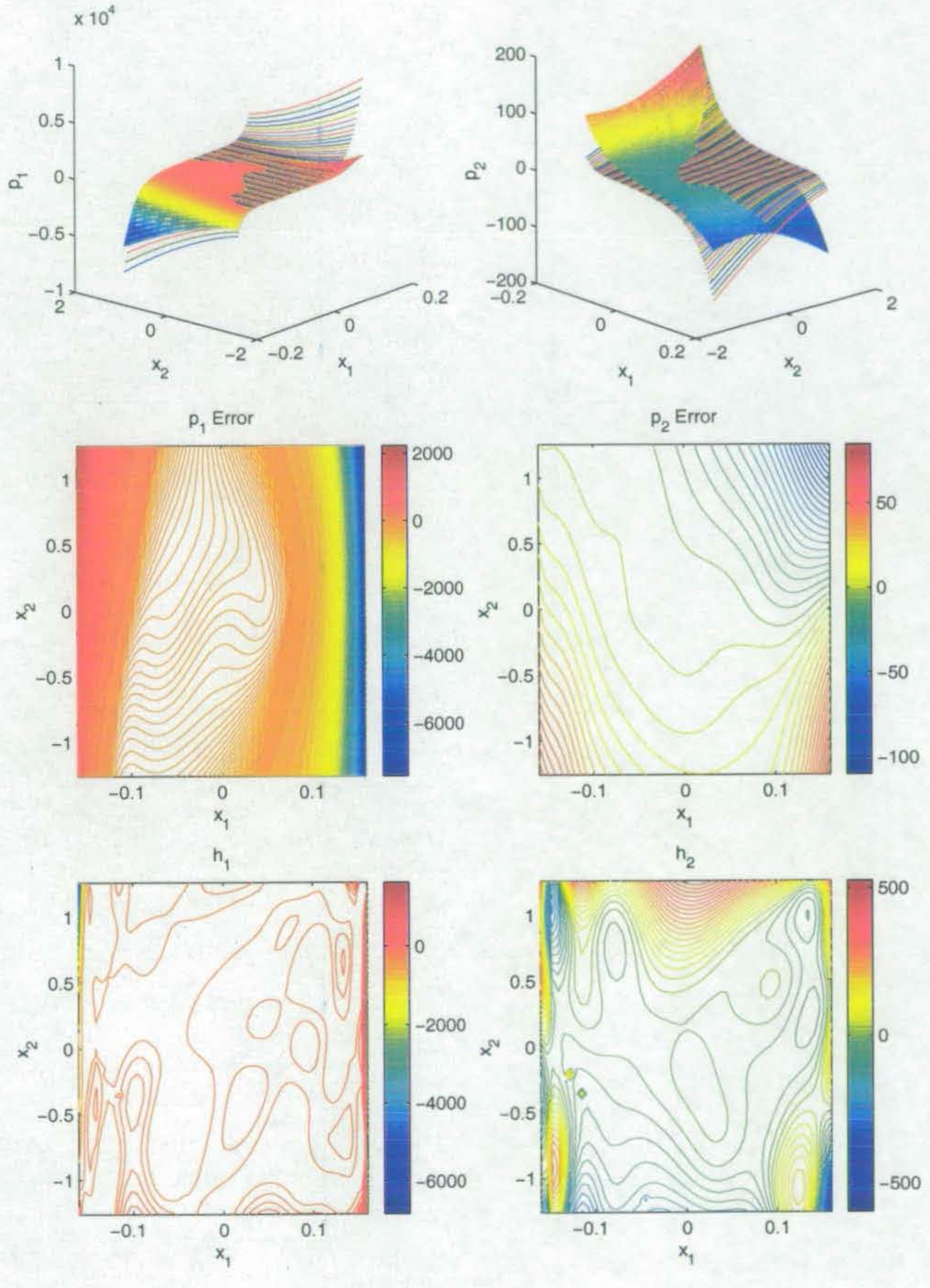


Figure 9.3 - SDOF Costate Surface & Error Distribution

9.5 Further Generalisation

In Chapter 5 the principles of validation were discussed. Validation is a stopping criterion, designed to ensure good network generalisation. In Chapter 5 it was ignored in favour of weight decay. However, in the context of the H_{norm} optimisation, validation can be implemented in a more flexible form.

Validation assumes only a limited data set is available. However, an infinite number of points are available for training or validation during the H_{norm} optimisation process. Because the H_{norm} optimisation process does not require a pre-calculated training data set, in theory an infinite training set can be used. Available time and memory obviously preclude this for every epoch. However from this stems the idea of point replacement: At the end of every epoch a selection of the initial condition points are replaced with new points; so the training data set is continuously changing. In effect, the training data set tends to infinity over sufficient time.

9.5.1 Point Replacement

The procedure for point replacement is outlined below.

1. For the first epoch, a data set is chosen and the change in weights calculated and applied.

2. At the end of the epoch a selection of data points are replaced. For simplicity these were chosen to be the first n points, and 10% of the total number of points was deemed to be sufficient.

3. The new points are added to the end of the data set, the old removed from the beginning. Therefore, if 10% of the points are replaced every epoch, the points will spend 10 epochs in the data set before being replaced.

4. The h values for the new points, are calculated for the current set of network weights, the h values for the other points can be carried over from the previous epoch.

5. The H_{norm} cost must then be calculated before beginning the next epoch to account for the change in cost resulting from the changing data set.

Point replacement allows a reduction in the number of initial condition points processed in a given epoch without compromising generalisation in the long term. This will benefit the speed of the optimisation process particularly in the early stages.

9.6 Network Shaping

In [88] Werbos makes the following comment: "There are many complex problems where it is difficult to find a good controller by adaptation alone, starting from random weights. In such problems, it is crucial to use a strategy called "shaping." In shaping, one first adapts a simpler neuro-control approach or even by talking to an expert; then one uses the weights of the resulting controller as the initial values of the weights of the controller to solve the more complex problem. This approach can, of course, be repeated many times if necessary. One can also build systems that phase in gradually from simpler approach to a more complex approach."

The results established so far suggest this is the case for the H_{norm} problem. Whilst it is very difficult to obtain the correct result using only basic neural network techniques, the comments of Werbos indicate an area for exploration.

Although additional knowledge of the mapping surface is provided by Marsh's work, assumptions made in the remainder of this section could still have been made based on intuition. The shaping process for the costate estimator problem takes three forms: Firstly, the provision of neurons that reflect the basic profile of the mapping surface. Secondly, the adaptation of a network or system of a more basic form. Finally, pre-training of the network to match a simpler system.

9.6.1 Odd Power Transfer Functions

Examination of the state - costate relationship for linear state equation based control problems with cost functions of even powers reveals that the relationship is odd. This can be observed not only in the LQR solution, where the function involved is linear, but also with the benefit of the optimal regulator solutions calculated from Marsh's work in the non-linear solutions. Based on this, an additional neuron has been created in the form of a cubic function. The cubic neurons implemented in parallel or series with the Tansig layer should allow the network to capture more easily the basic form of the costate estimator surface. The Tansig neurons can then be used to adapt this simplistic cubic-based approximation to give greater accuracy. The new cubic function takes the following form to ensure *dy/dN* is one and not zero, where N is the input to the transfer function.

$$\begin{aligned} y &= N + N^3 \\ \frac{\partial y}{\partial N} &= 1 + 3N^2 \end{aligned} \qquad (9.13)$$

This ensures that errors passed through the cubic neurons carry the same importance as those of the Purelin and Tansig neurons when operating near zero. For some problems it maybe worth while experimenting with higher order odd powers of *N*, depending on the form of the original cost functions.

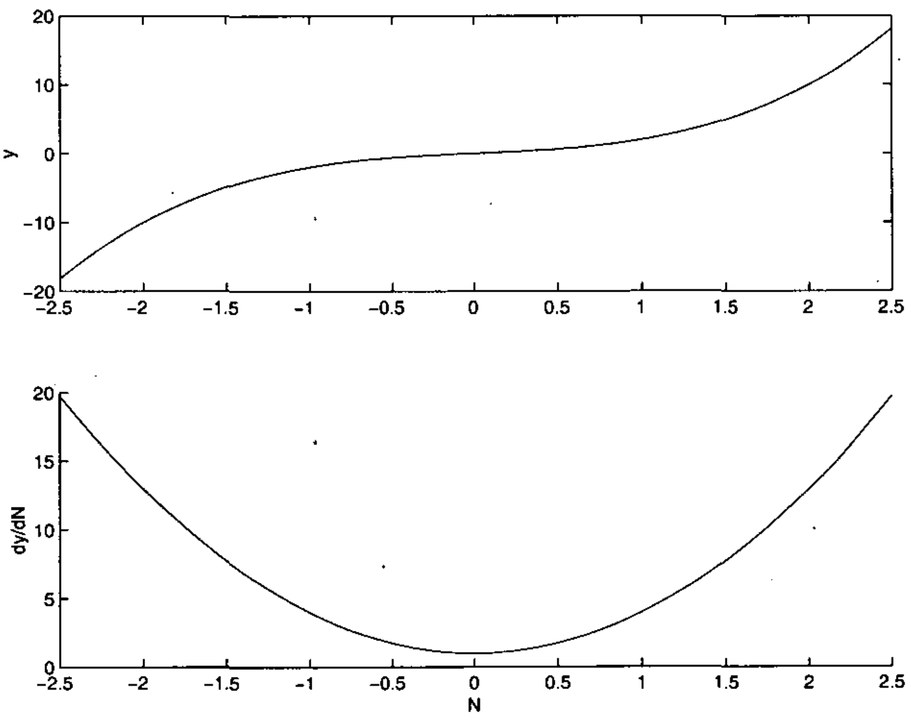


Figure 9.4 - Cubic Transfer Function

9.6.2 LQR Adaptation

The second approach used to shape the network is the adaptation of the Riccati matrix solution. As the cost function 9.2 is itself an adaptation of the LQR cost function, the concept of making the costate estimator a combination of the Riccati matrix and a neural network seemed an appropriate step to take. The non-quadratic cost function tends to the LQR cost function in the region of the origin. In order to incorporate the Riccati matrix within the network, a number of alterations need to be made to the basic feed forward network architecture.

These additions outlined below include the radial propagation neuron (presented for the first time to best of the author's knowledge), direct feed through weighting, and layer bypassing.

9.6.3 Radial Propagation

The Radial Propagation (RP) neuron defines a region where the neural network (Sub-Neural Network, Figure 9.5) that lies beyond it in the network architecture is deactivated. If biases are used, the deactivated region can be positioned anywhere in the networks operational space, x, but typically the RP neuron will operate around the origin.

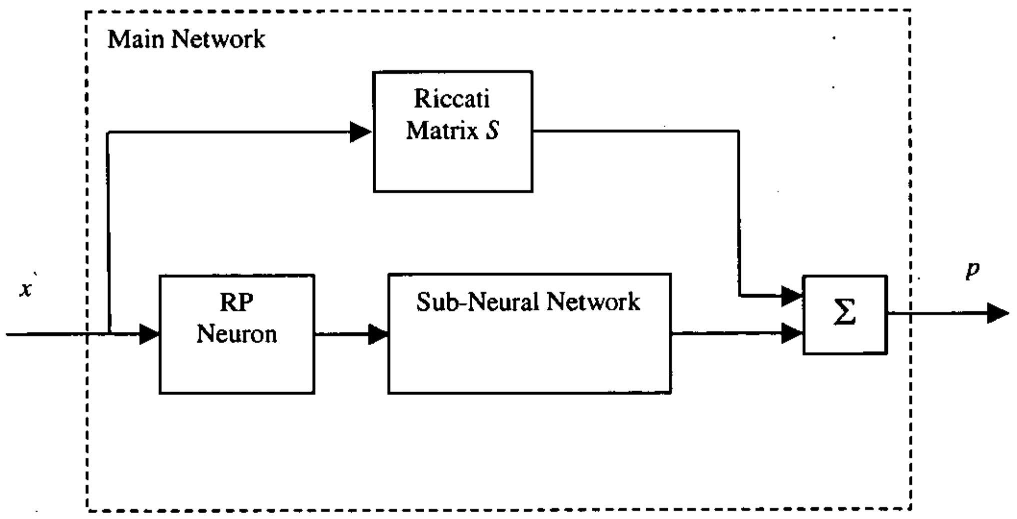


Figure 9.5 - Radial Propagation Transfer Function Utilisation

For LQR adaptation, the non-quadratic cost function tends to the LQR cost function as the states tend to zero. Therefore, the deactivating region of the RP neuron is positioned at the origin. As the states increase, the RP neuron activates the sub-neural network, which adapts the Riccati matrix calculation outputs for the non-linear costate problem.

The RP neuron output is linearly proportional to the input in the majority of its operating range. However, in a definable region around its origin, the output tends to zero. The following equation defines the RP neuron and is continuously differentiable.

$$y_i = N_i \cdot \left(1 - e^{-\sum_i N_i^2}\right) \tag{9.14}$$

Where N for the RP neuron only is a vector of the scaled input $\boldsymbol{x}$

Figure 9.6 shows the input / output relationship of the radial propagation function for a two state case and the related derivatives.

9.6.4 Direct Feed through Weighting

The inputs are scaled to define the operational range of the deactivated zone of the RP neuron output. Acting directly on the input, the RP network will pass approximately 50% of the operational range [±1]. The input range is therefore scaled by the direct feed through layer to account for differences in the operational range of the states. The input to the RP neuron N_i, Equation 9.14 is then a scaled version of the input x_i. The selection of the scaling weights was carried out manually with the direct feed through layer weights held fixed during the optimisation. Potentially in the future, with certain constraints, the scaling weights could be optimised using an appropriate routine.

9.6.5 Layer Bypassing

In order to perform the Riccati matrix calculation requires only one network layer, whilst the sub-net and RP neuron will require a minimum of 3 layers in which to perform the relevant calculations. For this reason some modifications are required to the basic network structure, allowing some data to bypass certain layers. Although the exact structure of this will depend on the form of the original network calculation, consideration should be taken to ensure its implementation remains computationally efficient.

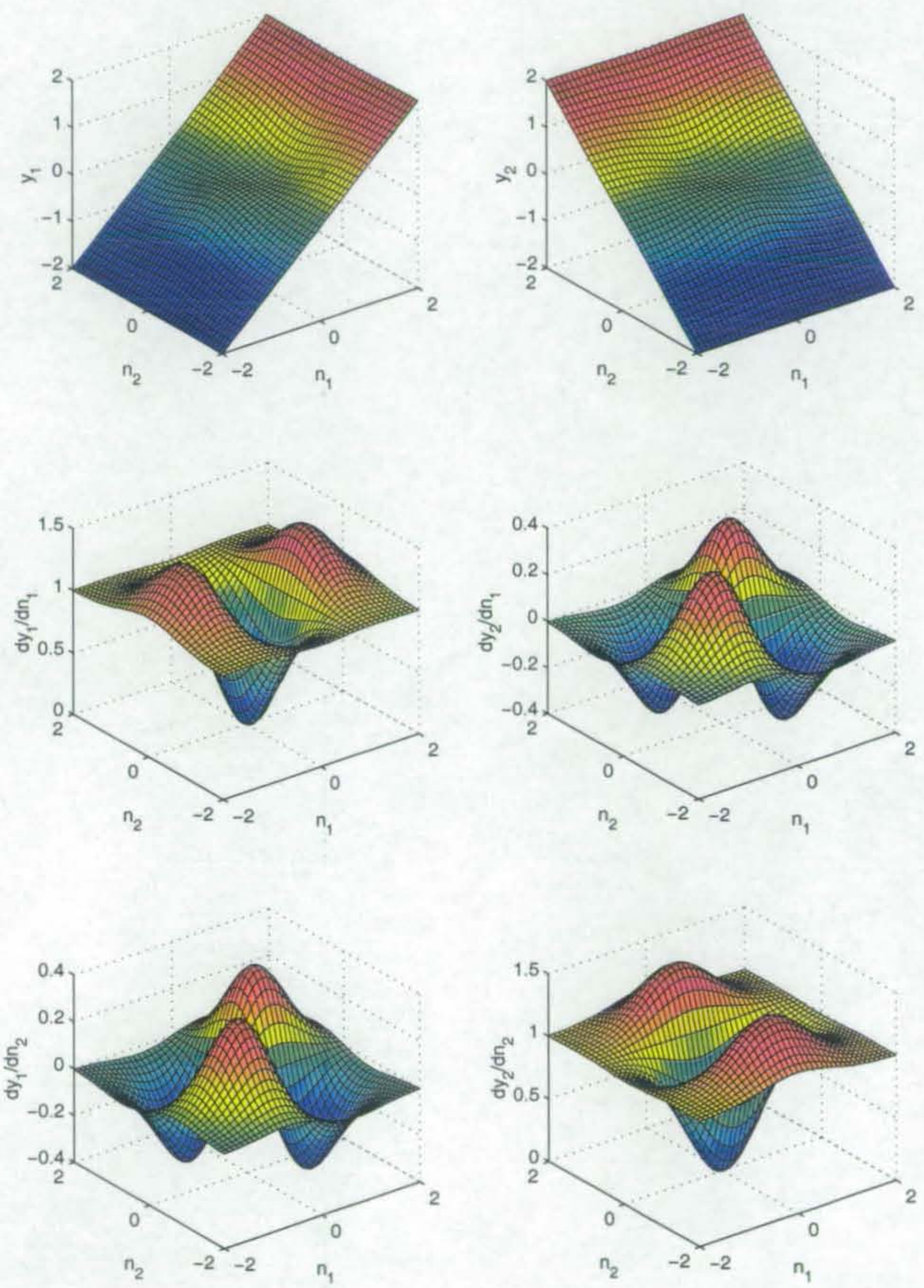


Figure 9.6 - Radial Propagation Transfer Function

9.6.6 Theorem of Extended Symmetry.

By enforcing Linear Quadratic Regulation in the region of the origin R_1, symmetry and "positive definite-ness" are enforced by default in that region. In Figure 9.7,assume $h_i \rightarrow 0$ in the region R_2, and that in the region R_1 (containing the origin) the symmetry condition applies:

$$Z_{ij} \xrightarrow{def} \frac{\partial p_i}{\partial x_j} - \frac{\partial p_j}{\partial x_i} = 0 \tag{9.15}$$

It can be shown that the symmetry condition automatically extends to those points in R_2 from which the state trajectory (– · –) remains within R_2 and eventually enters R_1. The proof is shown below.

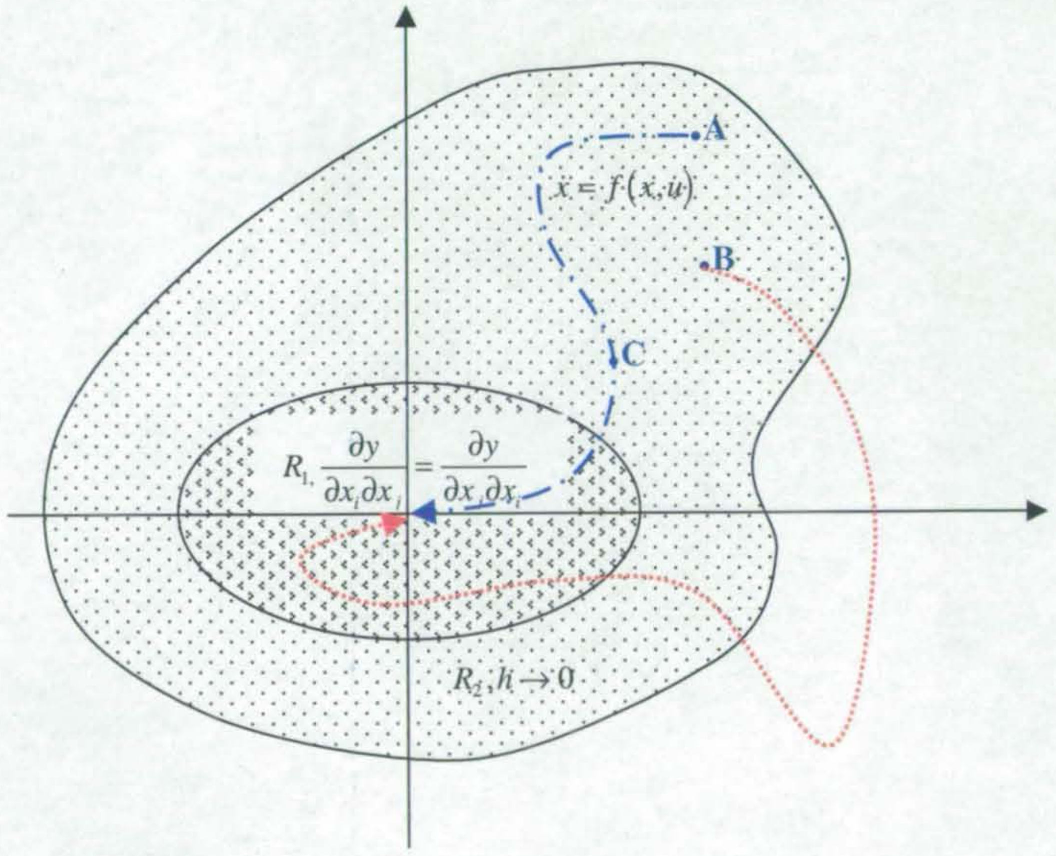

A
$\dot{x} = f(x, u)$
B
C
R_1 $\frac{\partial y}{\partial x_i \partial x_j} = \frac{\partial y}{\partial x_j \partial x_i}$
$R_2, h \rightarrow 0$

Figure 9.7 - Regions of Symmetry

Proof

Assuming optimal conditions in the region R_2, h_i is zero.

$$h_i = \left.\frac{\partial H}{\partial x_i}\right|_{\text{p constant}} + S_{ij} f_j \to 0 \tag{9.16}$$

and therefore

$$\frac{\partial L}{\partial x_i} + p_k \frac{\partial f_k}{\partial x_i} + \frac{\partial p_i}{\partial x_k} f_k = 0 \tag{9.17}$$

where $p_i = P_i(x)$ is the feedback form of the costate functions.

Since h_i is zero in R_2, the following derivative equation holds.

$$\frac{\partial h_i}{\partial x_j} - \frac{\partial h_j}{\partial x_i} = 0 \tag{9.18}$$

This is expanded to form the following equation

$$\begin{gathered}\left(\frac{\partial^2 L}{\partial x_i \partial x_j} + \frac{\partial p_k}{\partial x_j}\frac{\partial f_k}{\partial x_i} + p_k \frac{\partial^2 f_k}{\partial x_i \partial x_j} + \frac{\partial^2 p_i}{\partial x_j \partial x_k} f_k + \frac{\partial p_i}{\partial x_k}\frac{\partial f_k}{\partial x_j}\right) \\ -\left(\frac{\partial^2 L}{\partial x_j \partial x_i} + \frac{\partial p_j}{\partial x_k}\frac{\partial f_k}{\partial x_i} + p_k \frac{\partial^2 f_k}{\partial x_j \partial x_i} + \frac{\partial^2 p_i}{\partial x_i \partial x_k} f_k + \frac{\partial p_j}{\partial x_k}\frac{\partial f_k}{\partial x_i}\right) = 0\end{gathered} \tag{9.19}$$

Rearranging and performing the obvious cancellations then gives the following result

$$\frac{\partial f_k}{\partial x_i}\left(\frac{\partial p_k}{\partial x_j} - \frac{\partial p_j}{\partial x_k}\right) + \frac{\partial f_k}{\partial x_j}\left(\frac{\partial p_i}{\partial x_k} - \frac{\partial p_k}{\partial x_i}\right) + f_k \frac{\partial^2 p_i}{\partial x_k \partial x_j} - f_k \frac{\partial^2 p_j}{\partial x_k \partial x_i} = 0 \tag{9.20}$$

With Z_{ij} as defined above, Equation 9.20 can be rewritten as follows

$$\frac{\partial f_k}{\partial x_i} Z_{kj} + f_k \frac{\partial}{\partial x_k} Z_{ij} - \frac{\partial f_k}{\partial x_j} Z_{ki} = 0 \tag{9.21}$$

Taken along any state trajectory $\dot{x} = f(x,u)$ for any function $q(x(t))$ the derivative of q with respect to time during the trajectory can be formed as follows.

$$\begin{aligned}\frac{dq}{dt} &= \dot{x}_k \frac{\partial q}{\partial x_k} \\ &= f_k \frac{\partial q}{\partial x_k}\end{aligned} \tag{9.22}$$

hence

$$f_k \frac{\partial (Z_{ij})}{\partial x_k} = \dot{Z}_{ij} \tag{9.23}$$

is the time derivative along the state trajectory.

Therefore, Equation 9.19 forms a set of ODEs for the function Z_{ij}

$$\dot{Z}_{ij} + \frac{\partial f_k}{\partial x_i} Z_{kj} - \frac{\partial f_k}{\partial x_j} Z_{ki} = 0 \tag{9.24}$$

Then given that $Z_{ij} \rightarrow 0$ in R_1, and Equation 9.14 holds throughout R_2, the uniqueness theorem for ODE solutions [89] ensures

$$Z_{ij}(x) = 0 \tag{9.25}$$

for all points such as **A** connected to R_1 by a state trajectory , though not necessarily by points such as **B.**

If the principles of uniqueness are applied, given that any stable state trajectory ends in the region R_1 where Z_{ij} is known to be zero when LQR is applied and stability is assured, all points along the trajectory are also symmetrical because of the following principles.

- They satisfy the ODE uniquely and are therefore the only solution at that point.
- Their connection to the region R_1 ensures that they have the same symmetry qualities as points within the region R_1.

- The principles of dynamic programming state that: Within an optimal regulatory system, the path taken from an initial condition point, A, passing through a point, C to the origin will be the same from point C as the path taken to the origin if C was the initial condition. Therefore, all points along the trajectory are also optimal, and in this case satisfy the ODE uniquely and are therefore symmetrical. Points within R1 will converge to zero as it is know to be stable if $h_i \rightarrow 0$.

Hence, the symmetry condition maybe expected to arise naturally out of a satisfactory H_{norm} optimisation, providing a convenient diagnostic test.

9.7 Pre-Training

By pre-training the network to the equivalent linear quadratic system introduced in section 5.1, a set of initial weights are established that already capture the basic form of the state - costate relationship. This is done using supervised learning. The reference points are established using the Riccati matrix solution, Equation 1.2, and therefore need not be fixed. There is no requirement for a high level of fit so the pre-training process does not significantly increase the overall training time.

While pre-training definitely has a beneficial influence on the optimisation process, observation indicates that it is important to be aware of the possibility of linearization of the solution. If the network is pre-trained too harshly, it is possible that the end solution found by the neural network will be either partially, or wholly, a linear approximation to the non-linear problem. Typically, the neural network solution will capture the non-linear form of the solution in some dimensions but not in all. Careful manipulation of the weight decay coefficients and the weights to which they are applied can help to increase the number of dimensions in which a non-linear approximation is achieved.

9.8 Point weighting

As in Chapter 5, point weighting remains an important factor in the optimisation process: Points in the operational surface away from the origin have very high H_{norm} values associated with them. These can dominate the optimisation process, particularly in the early stages when the errors relating to these points are large. Combined with the odd power neurons this can lead to the network taking on an unresolvable set towards one of the predominantly non-positive definite solutions. To decrease the influence of these points, the H_{norm} values are weighted based on relative state position as in Chapter 5.

9.8.1 Uniform Distribution

The original data used in the supervised training was evenly distributed across the predicted operational space of the neural network. With the H_{norm} process, the data points are not restricted to fixed points for the whole of the optimisation process and change continuously as outlined above. There are two methods for the generation of the training points both based on a uniform random number generator. The first forms a distribution of points that is rectangular in the dimensional pairs, for example in a 3 state system this would form a rectoid.

$$x_i = x_{i_{\max}}\left((2u)-1\right) \tag{9.26}$$

where u is a uniformly distributed random number $0 \le u \le 1$.

However, the magnitudes of the state points do not form a uniform distribution using this method. This can result in poor performance around the origin, which is under populated.

9.8.2 Uniform Radial Distribution

An alternative method, which ensures that the points have uniformly distributed magnitudes, is to calculate the points to form a uniform radial distribution in the dimensional pairs. This results in a spherical distribution when points are generated for a three dimensional system. Equations 9.27 & 9.28 define the generation of points for the SDOF and quarter vehicle problem respectively.

SDOF model

$$\begin{aligned} x_1 &= x_{1_{\max}} r\cos\theta \\ x_2 &= x_{2_{\max}} r\sin\theta \end{aligned} \tag{9.27}$$

Quarter Vehicle Model

$$\begin{aligned} x_1 &= x_{1_{max}} r\cos\theta \\ x_2 &= x_{2_{max}} r\sin\theta\cos\phi \\ x_3 &= x_{3_{max}} r\sin\theta\sin\phi\cos\lambda \\ x_4 &= x_{4_{max}} r\sin\theta\sin\phi\sin\lambda \end{aligned} \tag{9.28}$$

where r, θ, ϕ & λ are uniformly distributed random numbers,

$0 \le r \le 1$, $0 \le \theta \le \pi$, $0 \le \phi \le \pi$, $0 \le \lambda \le 2\pi$.

9.9 SDOF H_{norm} Optimisation

Figure 9.8 shows a successful H_{norm} optimisation run for the SDOF system. The upper pair of plots show the neural network output as a solid surface and the target costate values calculated from Marsh's work, as lines that define a surface. The error between the two is shown in the centre pair of plots, with the h_i values shown on the bottom pair of plots. As can be seen there is no direct correlation between the h_i values and the costate error even when a stable solution has been obtained.

The h_i values around the origin are clearly very small and therefore based on Section 9.8.2, the symmetry error should be reasonably small in the majority of the operational range.

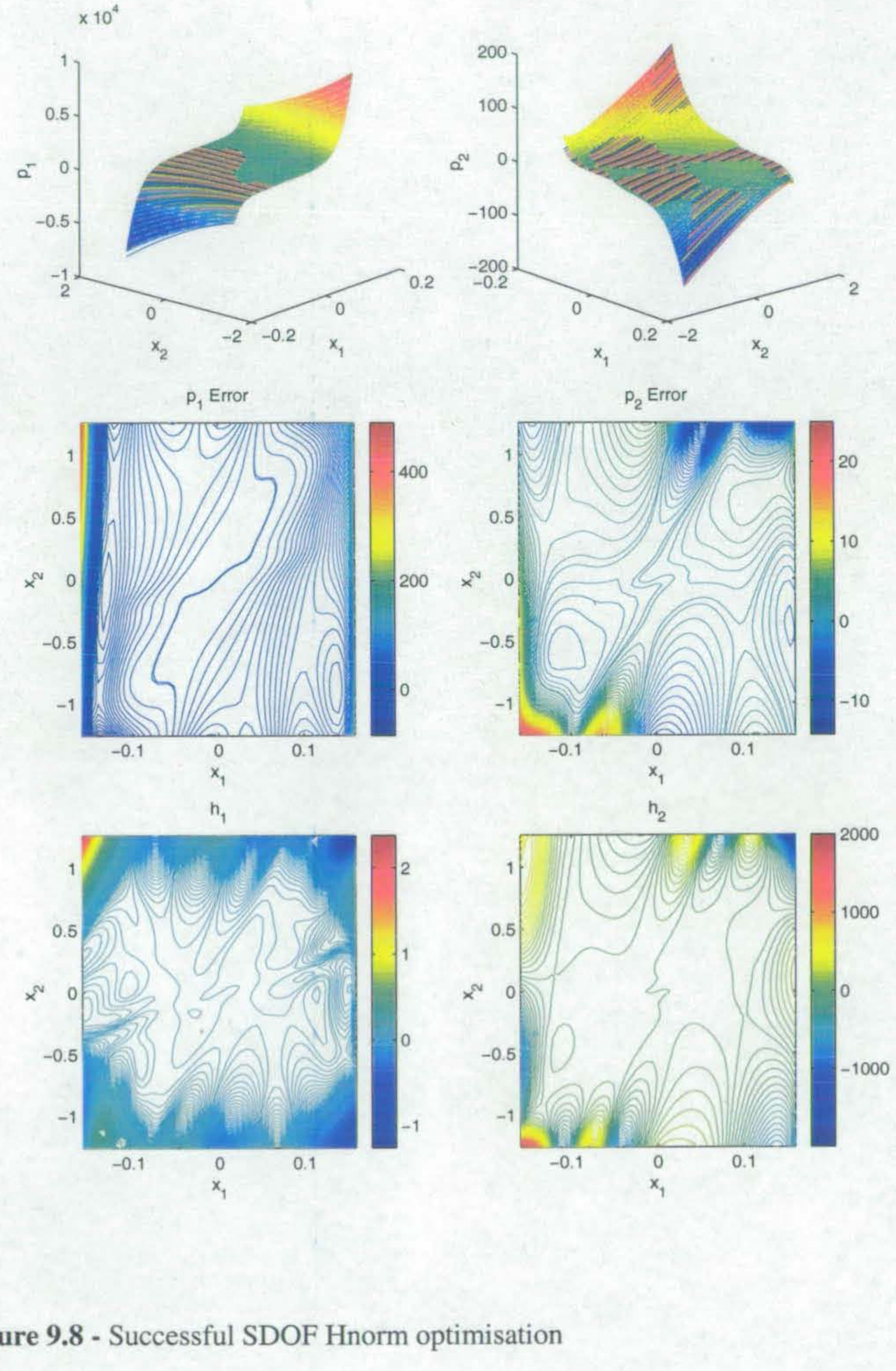
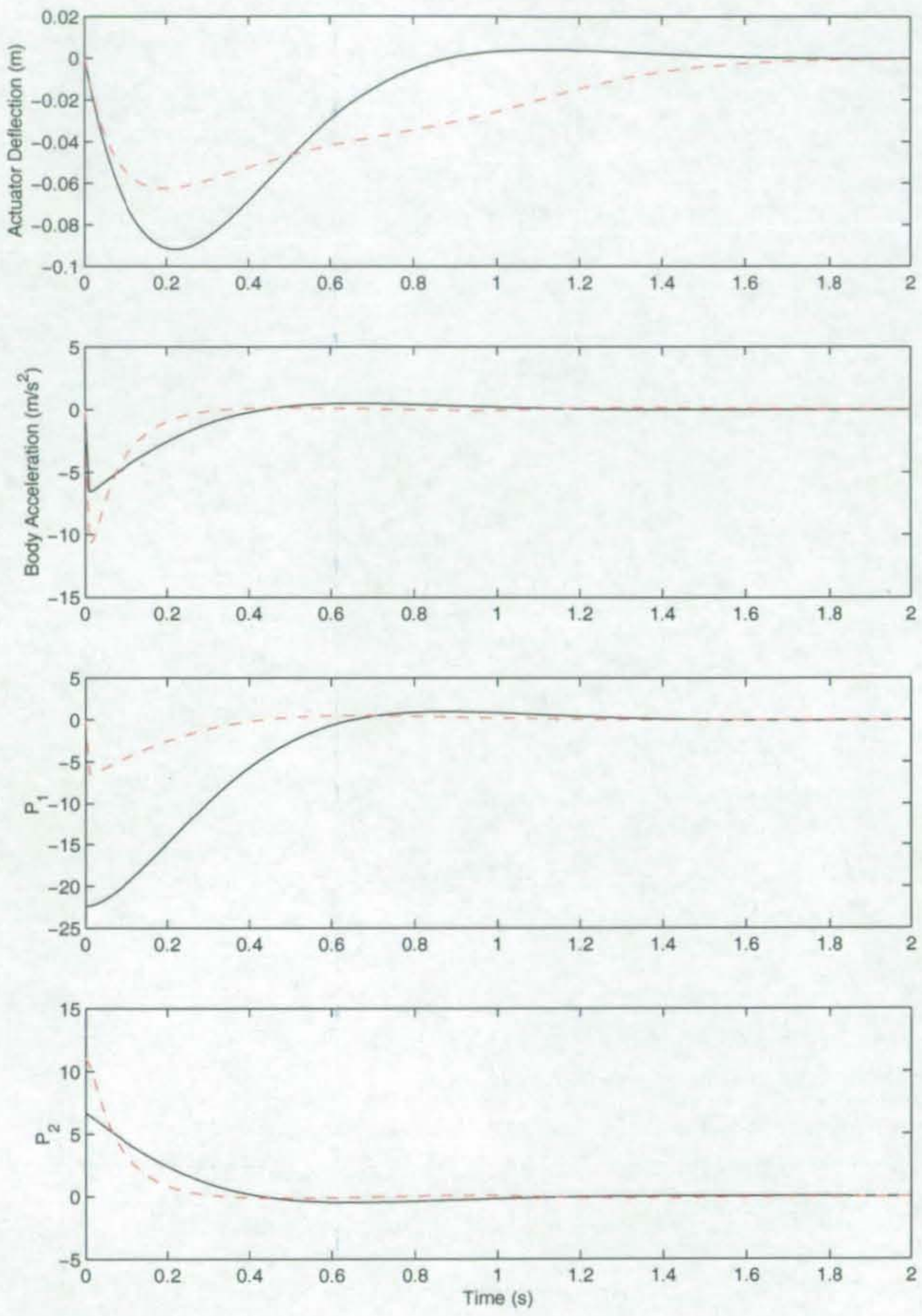


Figure 9.8 - Successful SDOF Hnorm optimisation

9.10 SDOF Simulation

Figure 9.9 shows the unit body velocity response of the single degree of freedom system, controlled by the H_{norm} trained neural network shown in Figure 9.8. The resulting state trajectory is stable and the performance comparable with the linear quadratic regulator control shown for comparison. The credibility of the control strategy is not important, but the result indicates that, under the right circumstances, it is possible to train a neural network to learn the costates for a non-quadratic cost function. Although the costate surfaces are not perfect, the principle of the H_{norm} training process is successfully established. The approach can now be applied to the more complex task of the quarter vehicle model. The additional degrees of freedom introduced by the quarter vehicle model will increase the possibility for failure in the optimisation process.



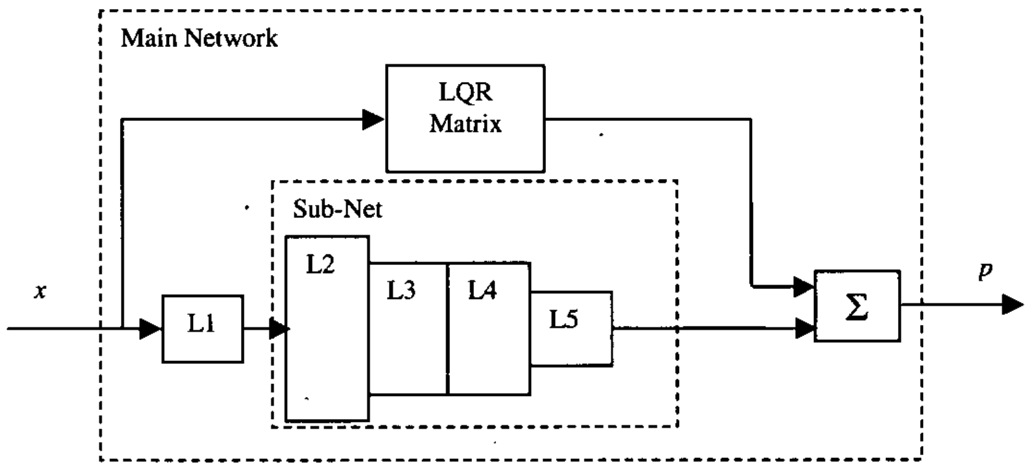
– Linear Control, - - - NNH Control

Figure 9.9 - Unit Body Velocity Response

9.11 The Quarter Vehicle Model.

At this stage a number of limiting factors become apparent. The increased dimensionality of the problem is significant in its effect on calculation time and memory requirements. The computing power of the desktop PC available to the author (an 800Mhz AMD Athlon with a limiting memory capacity of 768Mb) is no longer truly able to cope with processing requirements of the H_{norm} optimisation approach. Increasing the number of degrees of freedom not only increases the size of the network, but also the number of training points that are required to capture the general shape of the costate surfaces.

For the quarter vehicle problem the network structure is increased to give the network additional opportunity to utilise the cubic functions, allowing it to cope with the increased dimensionality of the two-degree of freedom problem.



Layer Type	Number of Neurons
L1 = Radial Propagation Functions + LQR Function	4
L2 = Combination Layer (Purelin and Tansig)	4 purelin, 20 Tansig
L3 = Cubic Layer	8
L4 = Purelin Layer	4
L5 = LQR function	4 Purelin

Figure 9.10 - Quarter Vehicle Network Structure

Most of the terms for the H_{norm} optimisation remain unchanged from those presented for the SDOF problem in section 9.2. All changes stem from the additional power term in the non-quadratic cost function.

$$J = \frac{1}{2}\int_0^{t_f} Qx^2 + Q_6x^6 + Q_{10}x^{10} + u^TRudt \qquad (9.29)$$

which in turn changes the Hamiltonian equation

$$H = p_i\left(A_{ij}x_j + B_{ij}u_j\right) + \tfrac{1}{2}\left(Q_{ij}x_j^2 + Q_{6_{ij}}x_j^6 + Q_{10_{ij}}x_j^{10} + R_{ij}u_j^2\right) \qquad (9.30)$$

affecting the state derivatives of the Hamiltonian equation as follows

$$\frac{\partial H}{\partial x} = p_jA_{ji} + Q_{ij}x_j + 3Q_{6_{ij}}x_j^5 + 5Q_{10_{ij}}x_j^9 = A^Tp + Qx + 3Q_6x^5 + 5Q_{10}x^9 \quad (9.31)$$

The cost element h_i becomes

$$\begin{aligned} h_i &= \frac{\partial H}{\partial x_i} + S_{ij}f_j \\ &= A^Tp + Qx + 3Q_6x^5 + 5Q_{10}x^9 + \frac{\partial P}{\partial x}_{ij}\left(Ax + Bu\right)_j \\ &= A^Tp + Qx + 3Q_6x^5 + 5Q_{10}x^9 + \frac{\partial P}{\partial x}_{ij}\left(Ax - BR^{-1}B^Tp\right)_j \end{aligned} \qquad (9.32)$$

All other elements of the optimisation remain the same as that presented in Section 9.2, the increased number of states aside.

9.12 Quarter Vehicle Results

The supervised learning approach has shown that higher levels of fit can be achieved with smaller networks for the reference data set. The H_{norm} optimisation costs attributes of the network that are not measured during the supervised learning process, in particular, the H_{norm} cost is a function of the network derivatives. This may limit the levels of fit achievable even though the network is larger. The network derivative has an important part to play in the optimisation process; high derivative values can upset the optimisation process even when the reference data error is low. If over-fitting type behaviour were to occur, this would clearly degrade the optimisation process due to the derivative element in the H_{norm} cost function.

The H_{norm} cost is not directly related to either the costates or the force. However, the error between the values calculated using Marsh's method and the neural network suggest there maybe an underlying relationship between the H_{norm} and the force, Table 9.1. The errors in the calculated force are much lower than the errors in the costates from which it is calculated, Equation 8.4.

Table 9.1 - Percentage errors between ELC, Neural Network and FORTRAN® Calculated Values

		Force	Costates			
			P_1	P_2	P_3	P_4
Neural Network	Original training data set	26.86	40.86	75.29	47.87	77.94
	Validation data set	24.15	33.03	70.20	37.58	70.97
ELC approximation		68.10	86.58	92.22	78.33	82.28

This may be attributable in part to the pre-training process as the force error between the ELC approximation and the reference force data set is smaller in comparison to the costate errors. However, this does not account for the breakdown of improvement in the costate estimation: the form of the H_{norm} cost function may be accountable for this phenomenon. The costate elements that contribute to the calculation of the control force form pairs within the H_{norm} cost, Equations 9.31-5.

$$h_i = A^T p + Qx + 3Q_6 x_1^5 + 5Q_{10} x_2^9 + \frac{\partial P_i}{\partial x_j}\left(Ax - BR^{-1}B^T p\right)_j \tag{9.33}$$

$$\begin{aligned} h_1 = {}^{k_t}\!\!/_{m_w}\, p_3 + \alpha_1 x_1 + 3\alpha_2 x_1^5 + \frac{\partial P_1}{\partial x_1}(-x_3) + \frac{\partial P_2}{\partial x_1}(x_3 - x_4)\ldots \\ + \frac{\partial P_3}{\partial x_1}\left(\frac{k_t}{m_w}x_1 - \frac{m_b^2}{m_w^2}p_3 + \frac{m_b}{m_w}p_4\right) + \frac{\partial P_4}{\partial x_1}\left(\frac{m_b}{m_w}p_3 - p_4\right) \end{aligned} \tag{9.34}$$

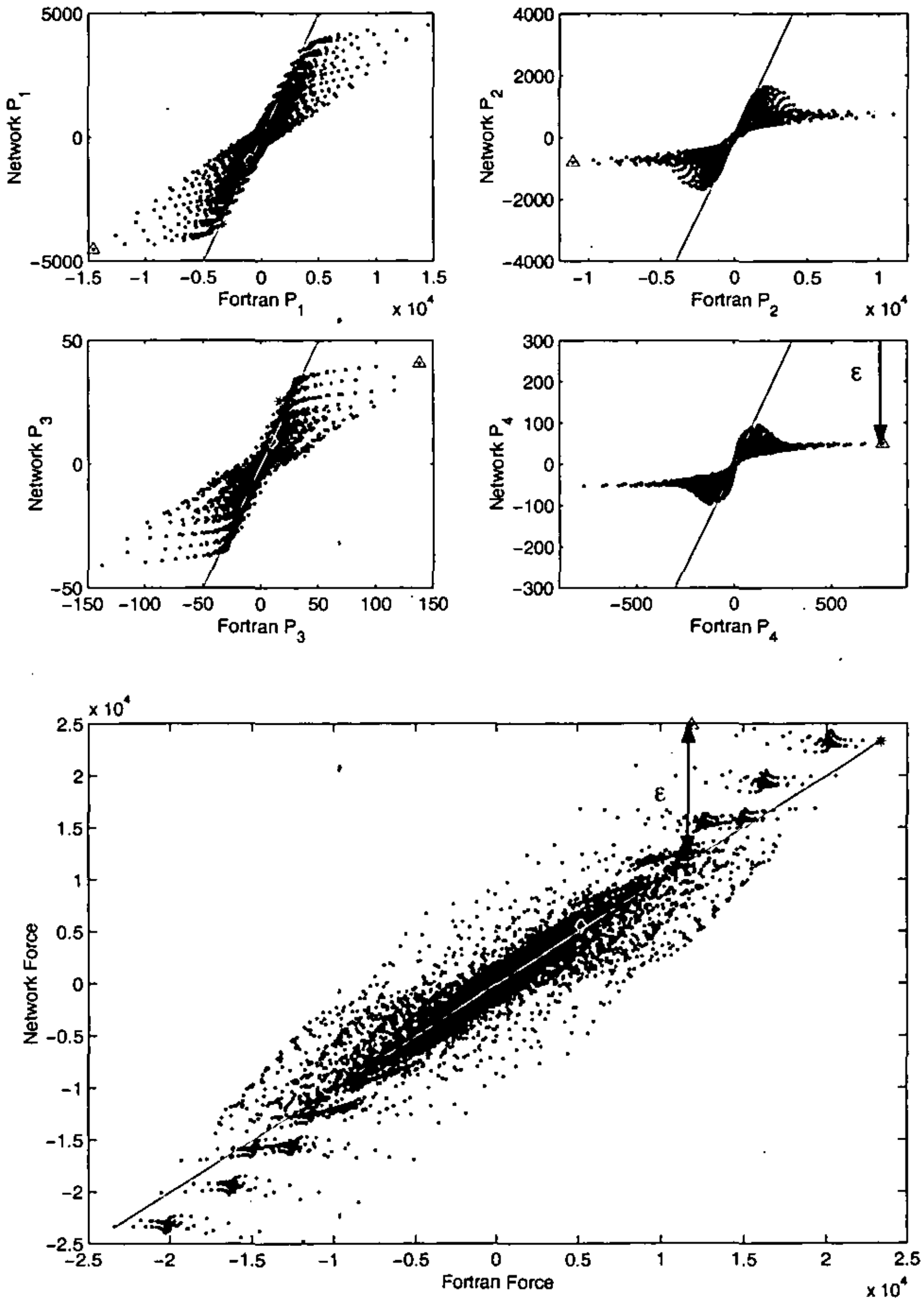
$$h_2 = \beta_1 x_1 + 5\beta_2 x_2^9 + \frac{\partial P_1}{\partial x_2}(-x_3) + \frac{\partial P_2}{\partial x_2}(x_3 - x_4)\ldots$$
$$+\frac{\partial P_3}{\partial x_2}\left(\frac{k_t}{m_w}x_1 - \frac{m_b^2}{m_w^2}p_3 + \frac{m_b}{m_w}p_4\right) + \frac{\partial P_4}{\partial x_2}\left(\frac{m_b}{m_w}p_3 - p_4\right) \tag{9.35}$$

$$h_3 = p_2 - p_1 + \frac{\partial P_1}{\partial x_3}(-x_3) + \frac{\partial P_2}{\partial x_3}(x_3 - x_4)\ldots$$
$$+\frac{\partial P_3}{\partial x_3}\left(\frac{k_t}{m_w}x_1 - \frac{m_b^2}{m_w^2}p_3 + \frac{m_b}{m_w}p_4\right) + \frac{\partial P_4}{\partial x_3}\left(\frac{m_b}{m_w}p_3 - p_4\right) \tag{9.36}$$

$$h_4 = -p_2 + \frac{\partial P_1}{\partial x_4}(-x_3) + \frac{\partial P_2}{\partial x_4}(x_3 - x_4)\ldots$$
$$+\frac{\partial P_3}{\partial x_4}\left(\frac{k_t}{m_w}x_1 - \frac{m_b^2}{m_w^2}p_3 + \frac{m_b}{m_w}p_4\right) + \frac{\partial P_4}{\partial x_4}\left(\frac{m_b}{m_w}p_3 - p_4\right) \tag{9.37}$$

The error in the individual costates can potentially cancel within these costate pairs and thus not contribute to the overall cost. However, an error in the costate pair, which has an indirect association with the force, makes a stronger contribution to the H_{norm} cost function. The costate p_3, is individually referenced within the H_{norm} cost. It is therefore possible to conceive that this might account for the greater accuracy achieved in its calculation. However, p_1 does not occur in the H_{norm} cost in any other respect than its derivative. Therefore, it maybe considered a credit to the H_{norm} process that an improvement occurs in p_1. These features are observable in the sample points shown on Figure 9.11.

Figure 9.11 shows the correlation between the FORTRAN® calculated data on the *x*-axis and the neural network output on the *y*-axis. A perfect correlation between the two would generate lines of points at y = x (-). Several points show the relationship between the costates and the calculated force. The first point, * ,is the maximum correct force. Although the force indirectly has been calculated correctly by the neural network, the costates that contribute to its calculation, Equation (8.28),.are incorrect,·the errors cancelling in the calculation process. The second point, Δ, is the highest neural network calculated force, it exhibits the largest error, ε, across all five plots. The neural network output under estimates all four costates, however the net result is an overestimated force value. The final point selected at random, ◊, shows that when all four states are estimated with minimal error the calculated force will in turn be correct.

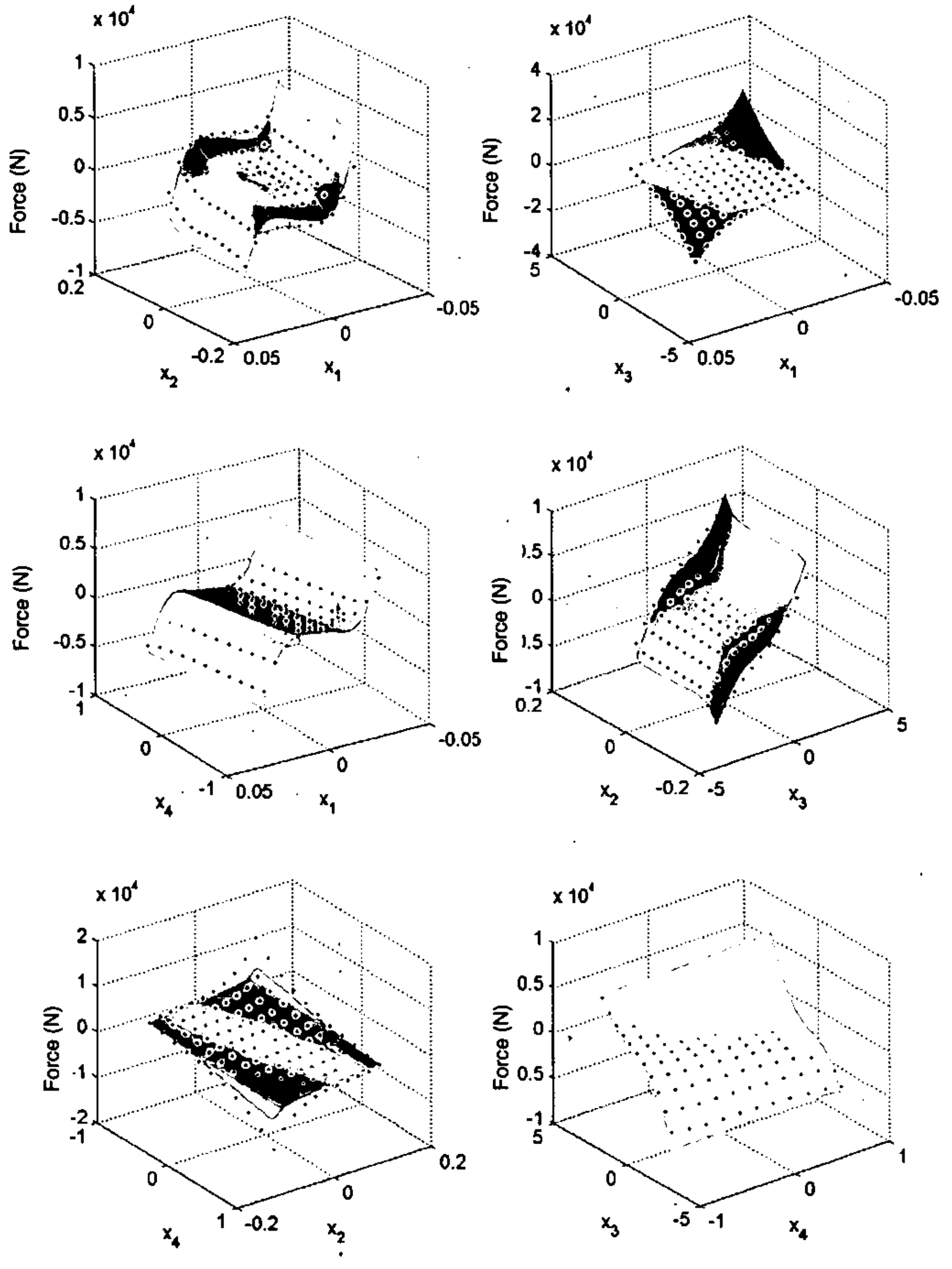


· Points, – Exact Correlation, *,◊,Δ Sample Points

Figure 9.11 - Neural Network Output Versus FORTRAN® Force

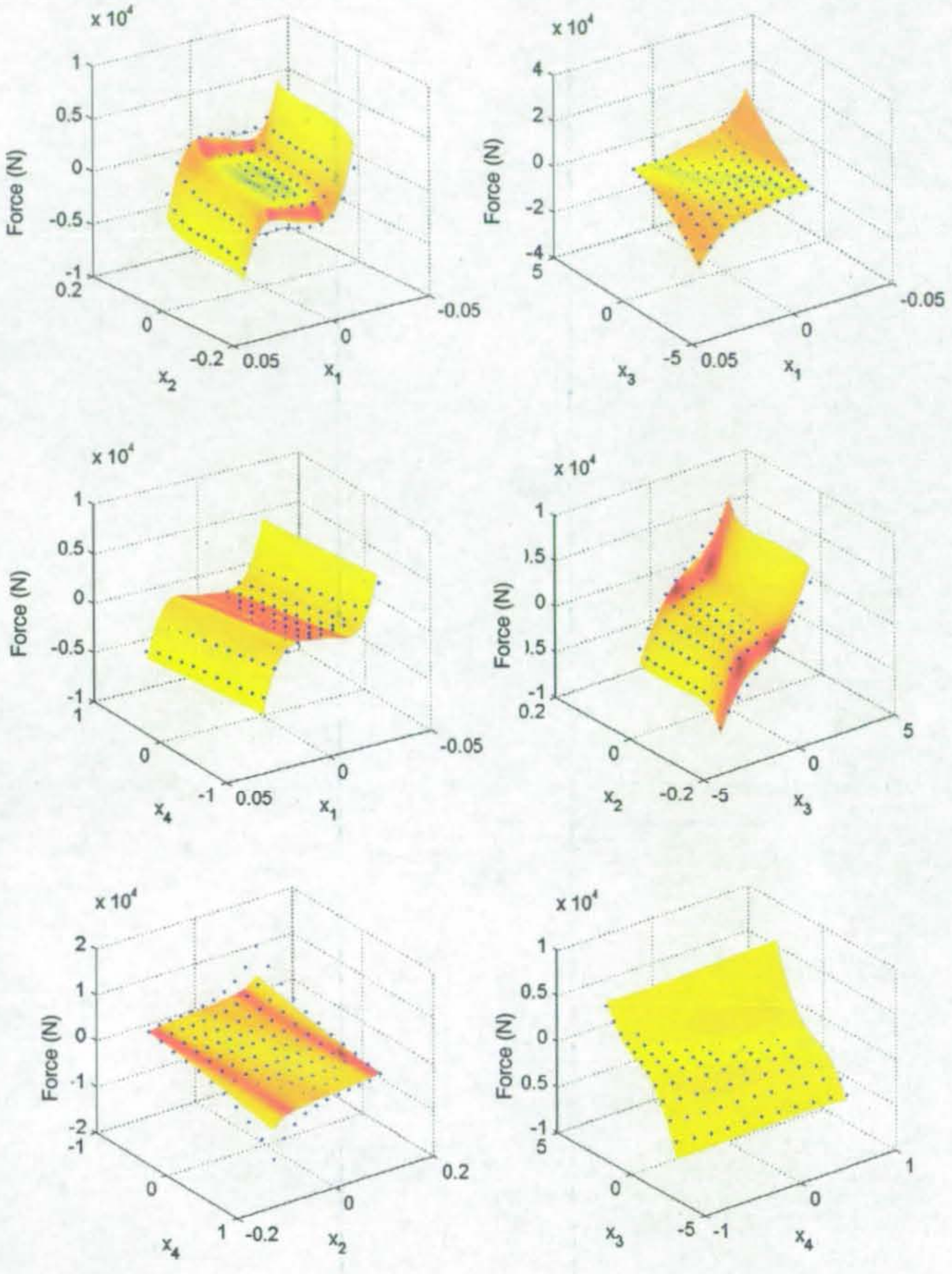
Figures 9.12-16 show that the optimisation process has made significant changes to the pre-trained network. For certain state pairs the network has captured the force or individual costates very well, whilst other state pairs with more complex surfaces are more difficult for the network to learn.

It is possible that a more complex network will be able to learn the surface in more detail and thus reduce the error levels achieved. However, the network used is at the limit of the computer's processing power. The need to calculate the network derivative for every cost calculation made means the H_{norm} process is computationally expensive to perform. This restricts the size of network used in comparison with the supervised learning technique. Advances in computing technology will change this. If a more successful version of the H_{norm} process is ever established it will not be beyond the scope of future computing systems to perform individual optimisation runs; current computing technology makes the development process very tedious.



(Solid Surface) Neural Network Output, · FORTRAN® Calculated Points

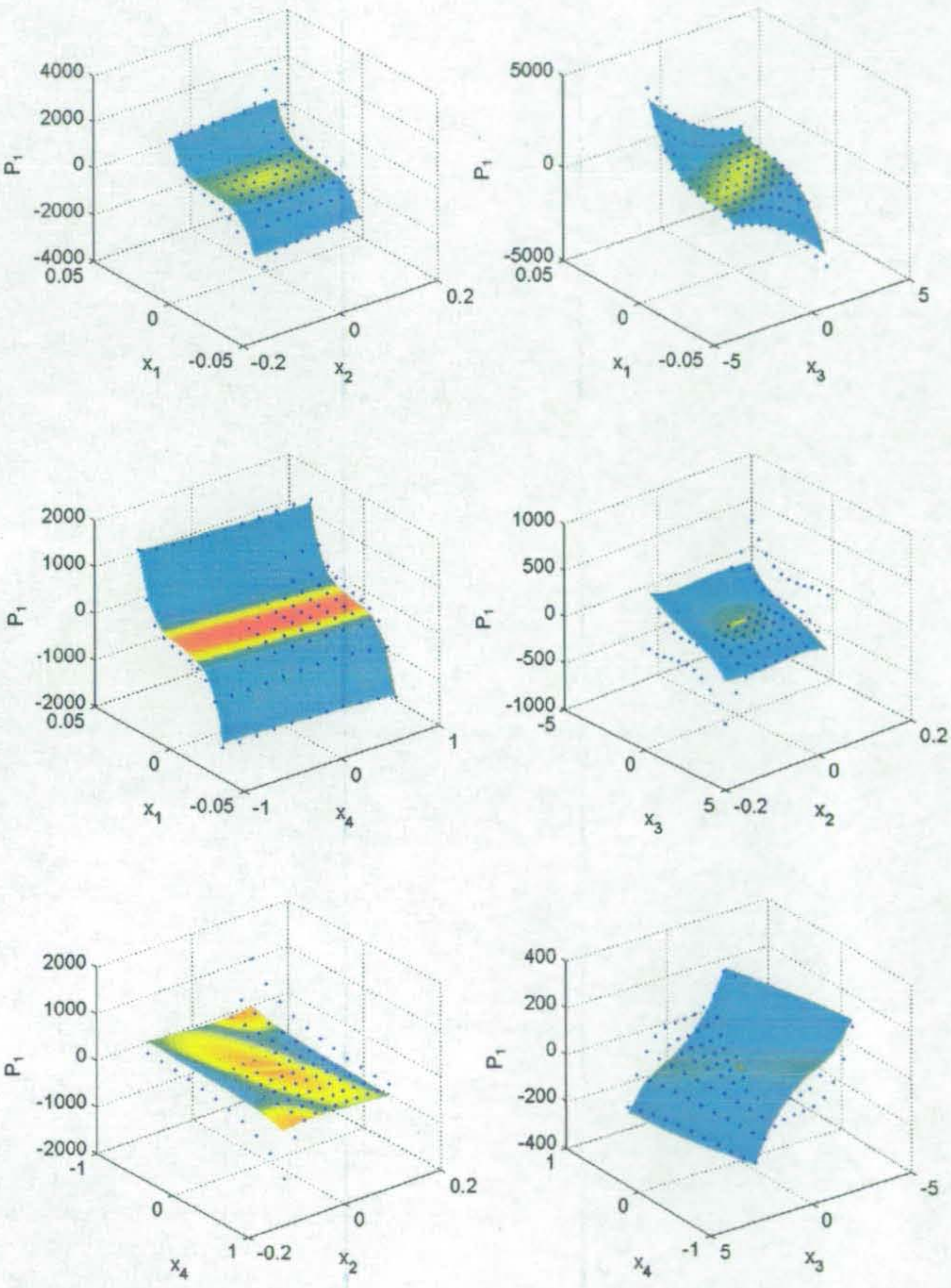
Figure 9.12 - Neural Network Force versus State Pairs



(Solid Surface) Neural Network Output, · FORTRAN® Calculated Points

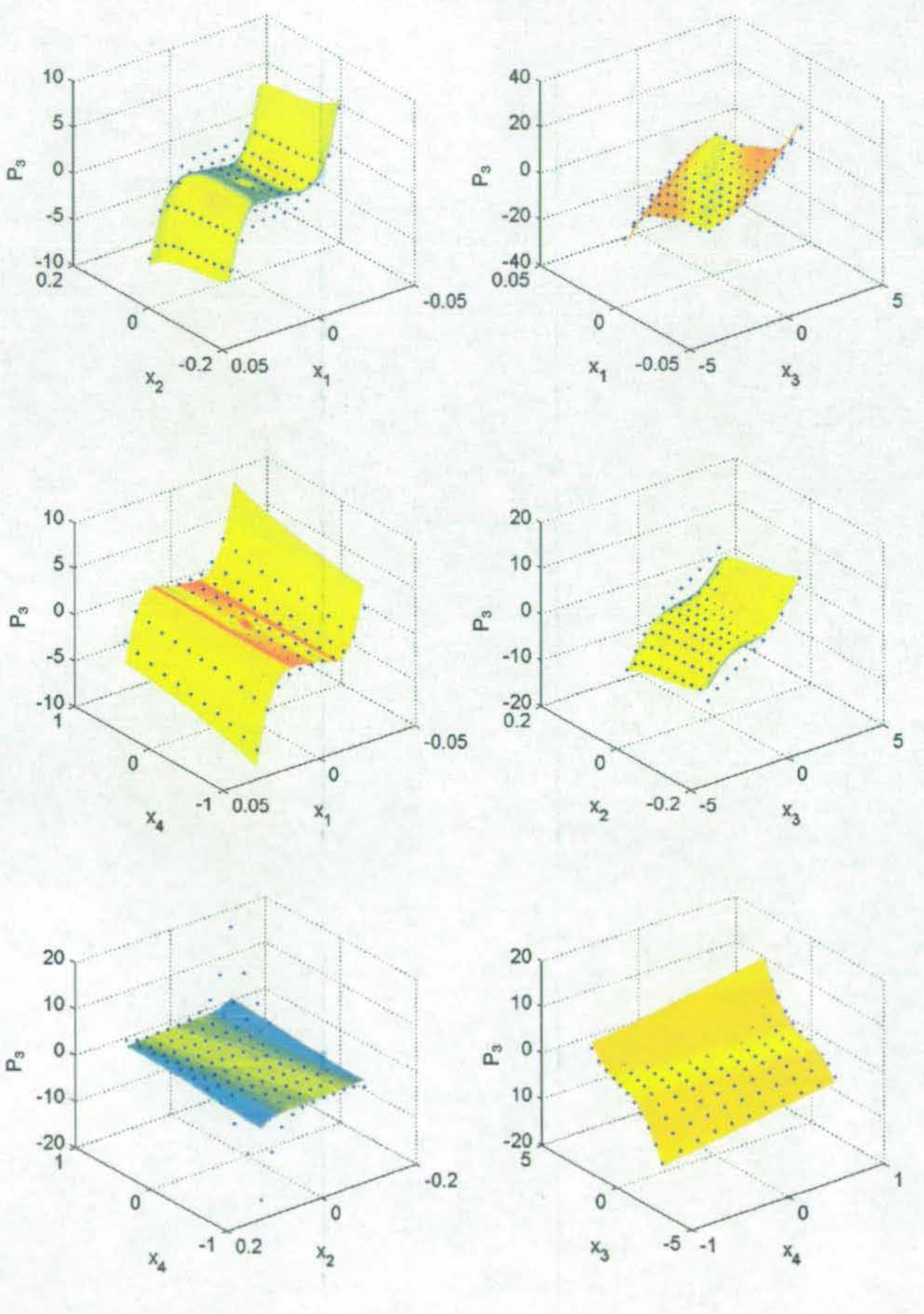
Figure 9.12 - Neural Network Force versus State Pairs

9.12.1 Costates



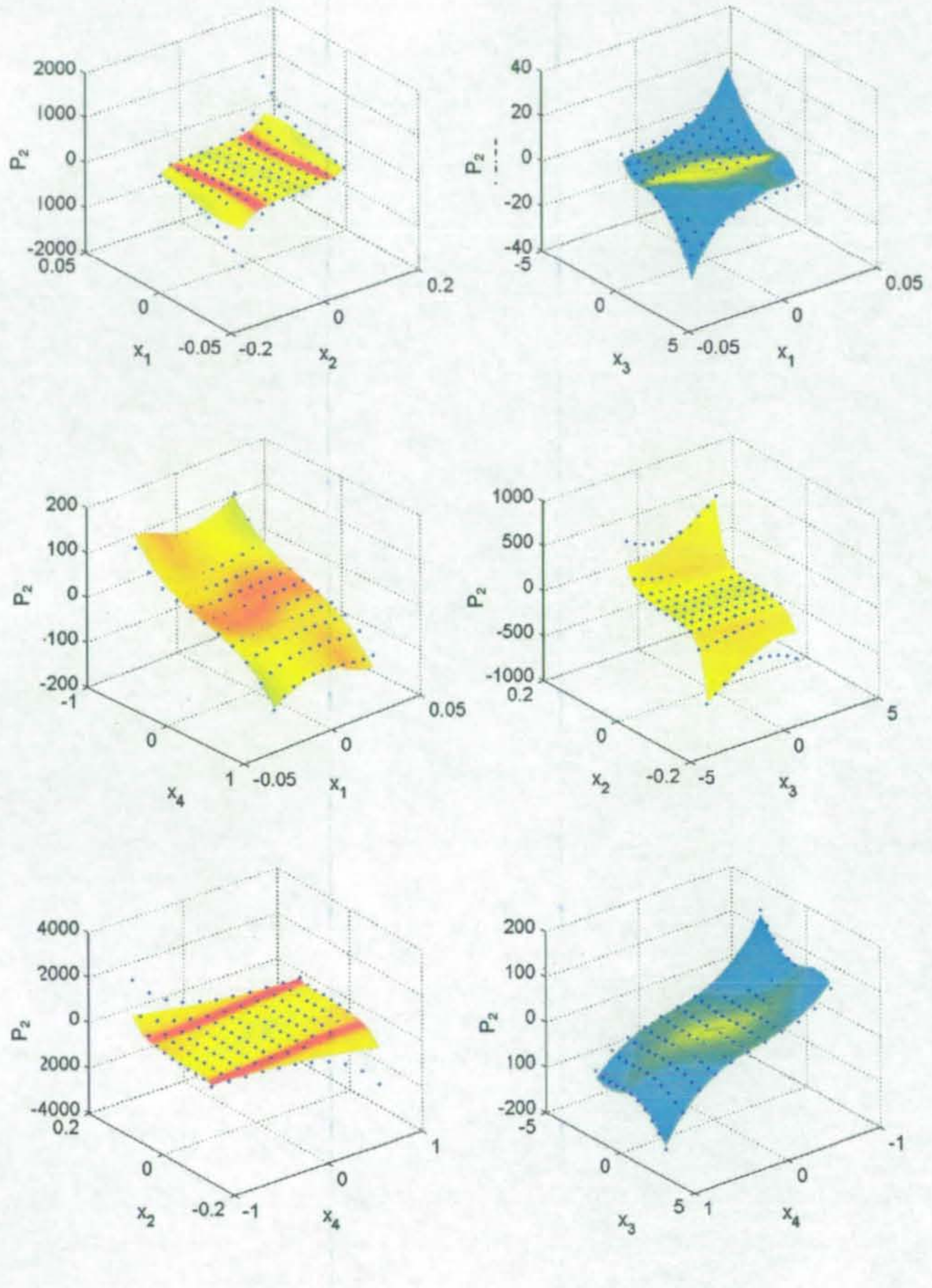
(Solid Surface) Neural Network Calculated Costate, · FORTRAN® Calculated Costate

Figure 9.13 - P_1 versus State Pairs



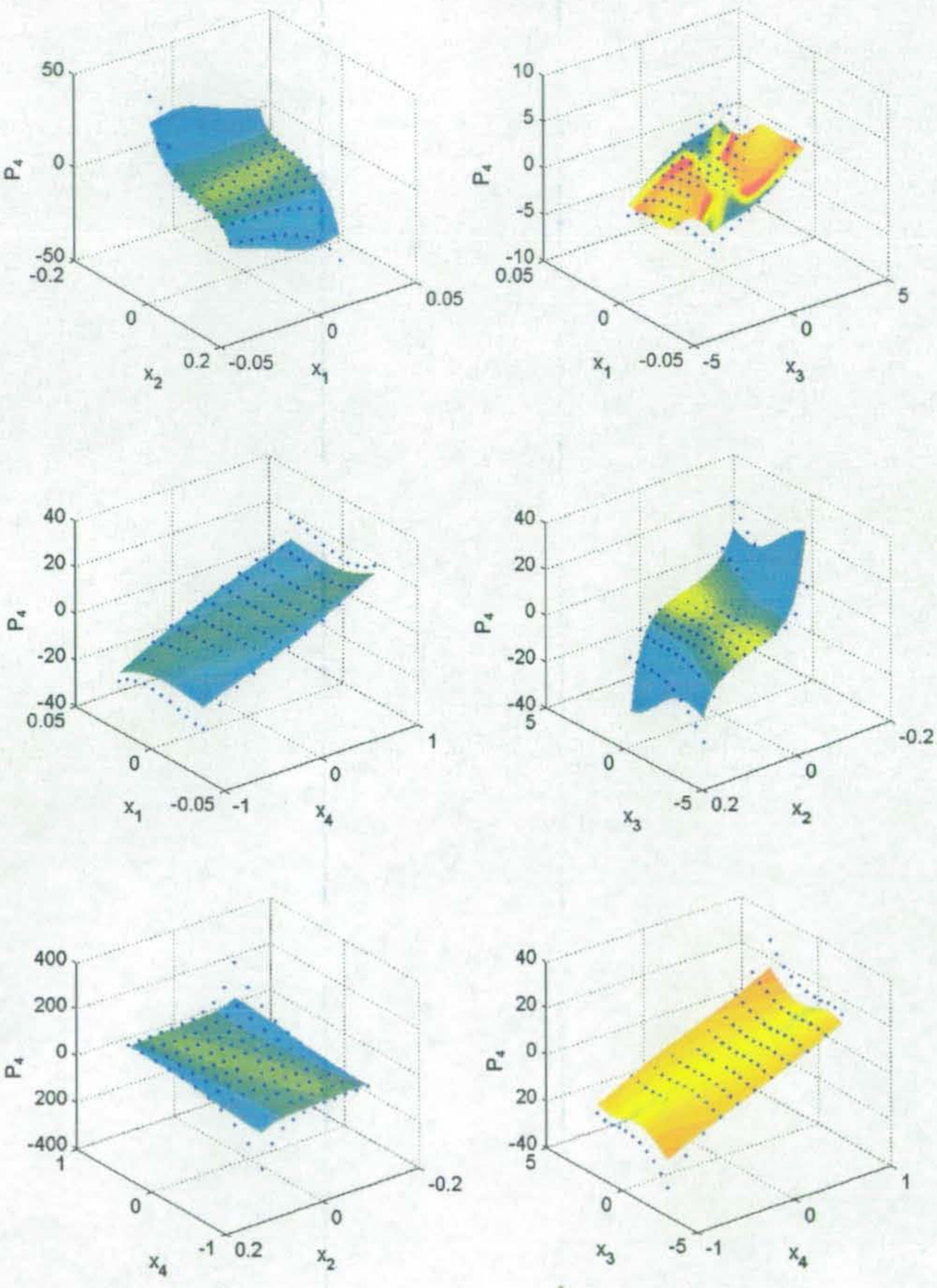
(Solid Surface) Neural Network Calculated Costate, · FORTRAN® Calculated Costate

Figure 9.15 - P_3 versus State Pairs



(Solid Surface) Neural Network Calculated Costate, · FORTRAN® Calculated Costate

Figure 9.14 - P_2 versus State Pairs



(Solid Surface) Neural Network Calculated Costate, · FORTRAN® Calculated Costate

Figure 9.16 - P_4 versus State Pair

Chapter 10 Summary and Conclusions

Chapter 10 Summary and Conclusions

This thesis set out to consider the application of neural networks to the non-linear control of the quarter vehicle model. This has been successfully achieved using supervised learning, with the neural network directly calculating the optimal force and with the neural network estimating the costates from which the control force can be calculated. This has a number of beneficial implications for the study of suspension control algorithms that have thus far only been studied off-line due to the computational effort involved.

The second part of the thesis covered attempts to establish a method that will allow the neural network to learn the costates directly using what has been termed the H_{norm} method. Although successful at a menial level, it has proved difficult to obtain successful results for the quarter car problem. In addition to the work with neural networks, Chapter 7 of the thesis covers methods for the reduction of suspension drift in active suspension vehicles, which are applicable to a wide range of controllers. A discussion of the conclusions drawn from work in relation to the three subject areas is now given, along with suggestions for further research.

10.1 Non-linear Control

The value of non-linear control has been proven many times, but the work presented in Chapter 2 and the investigation that was enabled using neural network controllers in Chapter 6 reiterates this in respect to vehicle suspension control. The constrained environment makes vehicle suspension an ideal system with which to demonstrate the use of non-linear control. Non-linear control results in reduced levels of body acceleration through more effective use of the suspension workspace. A greater awareness in the controller of the presence of the bump-stops can be established through the use of non-linear control and this ensures that the appropriate control force is used depending on the state of the system. Greater utilisation of both the tyre and suspension deflection ranges available to the controller results in reduced body acceleration and therefore, a better ride.

10.2 Supervised Learning and Neural Networks

Chapters 4-7 demonstrate the power of supervised learning and neural networks to learn multivariable non-linear relationships. They also demonstrate some of the fundamental failings of the basic supervised learning, neural network architecture approach for control problems. In order to avoid these a number of additions to the basic approach had to be made to ensure the learning process resulted in a successful working controller.

Bias Cancelling

The standard multi-layer perceptron network architecture does not guarantee a zero output for a zero input. In the majority of control problems, this is expected and is vital if the system is to stabilise to a zero position. The network architecture can be adapted so that a zero output is guaranteed using the principles of bias cancelling. This does not restrict the network's ability to learn suitable non-linear problems given a network of sufficient size, and introduces only a small increase in the calculation time. However, bias cancelling does not guarantee a zero settling point. It is important that around the origin, the network accurately reflects the control strategy that is to be mapped otherwise the use of bias cancelling can introduce multiple non-zero settling points. This can be achieved using point weighting.

Point weighting

It is important that the training points are distributed and / or weighted in such a way as to give appropriate importance to the accuracy of the network in a given area of its operational space, such as the origin. High value points can often incorrectly dominate the optimisation process. This leads to a network that may perform well in terms of the network training cost, but when utilised as a controller does not settle the system correctly. This can also lead to unnecessarily large networks, as the degree of accuracy at low value points is sought but not met because of the higher value points. Point weighting gives the low value points the necessary importance they deserve, ensuring high value points do not dominate the process.

Weight decay

The control of neural network weights is also vital to successful completion of the training process. Neuron saturation can lead to failure of the optimisation or poor network performance. Neurons such as the Tansig and Logsig neurons can saturate in two ways, both of which introduce their own unique problems. If the weights are small and the bias large, the transfer function output becomes constant across the complete network operating range. This can lead to one of two problems; an offset at the output or a reduction in the level of fit achieved, the saturated neuron effectively reducing the size of the network, as the neuron is made redundant. Secondly, the neuron weights can become large and in this case, the neuron becomes a switch. This may be useful in some scenarios, but for costate and force estimation leads to fluctuations in the output surface which negatively affect performance. These effects can be prevented though the use of weight decay, this ensures the weights are kept small and prevents either case occurring.

These three methods have all been used in Chapter 5 in order to ensure not only a successful reduction in the training cost but also networks that work successfully as non-linear controllers for the quarter vehicle problem.

10.3 H_{norm} Optimisation

The H_{norm} optimisation routine is not, as it stands, a viable training routine for the development of costate estimators for non-linear control problems. Although successful implementation has been achieved for both a four state linear and a single degree of freedom non-linear problem, as the complexity is increased it becomes more difficult to obtain the correct answer. This is due to two factors; the form of the cost function – there is no direct relationship between the costates and the H_{norm} cost function - and the flexibility of neural networks, allowing solutions that minimise the H_{norm} cost but do not correctly estimate the costates. Compared to the supervised learning cost function, if the generation of the original training data is ignored, the H_{norm} cost is extremely computationally expensive. The network derivative in particular adds considerably to the time taken to perform each epoch. For the supervised learning approach, the derivative is calculated only when the cost gradient with respect to the weights is required. With the H_{norm} approach, the network derivative must be calculated at every cost calculation and this is very expensive.

10.4 Neural Network Training for the H_{norm} Problem

The influence of the multiple solutions on the H_{norm} optimisation process at every calculation point can be reduced using two approaches, network shaping and pre-training. Combined with techniques already introduced for supervised learning, these significantly increase the probability of a satisfactory training run.

Shaping

Shaping is a powerful technique used to ensure that prior knowledge of the control surface's form is taken into consideration during the training process. For the H_{norm} problem two techniques have been established to utilise prior knowledge; Firstly, the creation of the cubic neuron, its shape closely resembles the basic form of the state, costate surface. Secondly, the Riccati matrix solution to the quadratic component of the cost function is enforced at the origin, to give the network the correct basic shape. This is done using the radial propagation neuron developed in this thesis to deactivate parts of a network in respect to the input space. Weight decay can then be used to ensure no sudden changes, steps occur in the mapping surface, hopefully ensuring the influence of the Riccati matrix beyond the region defined by the radial propagation neuron.

Pre-Training

In order to create the correct initial shape, from which to begin training it has been found useful to carry out some pre-training. This pre-training uses, supervised learning to costate data created using the Riccati matrix solution to a quadratic cost function. The coefficients of the quadratic cost function should be selected to achieve the best approximation to the non-quadratic cost function over the expected operational range. However, it is import when pre-training not to over train the network to the point where the input, output relationship becomes linear. This limits the networks ability to adapt to the non-linear relationship and reduces the influence of the shaping process.

10.5 Further Opportunities for improving H_{norm} Optimisation

There are a number of features about the H_{norm} optimisation cost function that distinguish it from other neural network optimisation problems. In particular the use of the network derivative in the cost function, but also the multi-element form of the cost, h_i. For these reasons, it maybe possible to develop an optimisation routine that uses these differences to its advantage. Some work was done to investigate a mini-max approach to the problem, but no significant improvement was established over the Levenberg Marquardt approach.

When neurons saturate large network derivative values can be generated, as a result of which the optimisation process maybe jeopardised. Steps have already been taken to avoid this using weight decay, however this may still not be sufficient. It may be possible using alternative methods either; for the calculation of the network derivative (e.g. via numerical differentiation, or via a comparative approach), or the correction of saturating neurons, to reduce the influence of irregular derivative calculations on the H_{norm} cost.

Even though the H_{norm} is a totally off-line training procedure, it has been possible - with no simulations to establish a relationship between the states and the costates that allows stable control of the quarter vehicle model. It would appear that without further manual intervention it is unlikely that a neural network will capture more accurately the solution. Whilst it is important that in the long term any solution can be generalised to cover other problems, at this point any further attempts to improve performance is likely to be particular to the quarter vehicle problem.

10.6 Low Frequency Disturbance Induced Suspension Drift

Three simple road inputs have been defined to highlight the problems of offset and drift in active suspension systems. Two control schemes have been presented which successfully overcome these problems. LGF was based on the estimation of v_l, the low-frequency component of the road input equivalent to estimating the local road gradient. IFC is more simply based on the feedback of an integrated suspension deflection signal.

While both approaches work well, LGF gives faster dynamic response and a reduced resonance peak at the body bounce frequency. On the other hand, IFC gives somewhat tighter control of mean suspension deflection.

The development process was performed using linear feedback control, however non-linear and adaptive schemes can also suffer from suspension drift at low frequencies. In this case, the LGF approach can be implemented quite simply via the revised velocity estimates of Equation (7.13), as was achieved in Section 7.6 where IFC may require a substantial revision of the base algorithm.

Attention has been restricted to the quarter vehicle model. There may be practical advantages in using a pitch-plane Kalman filter model, where a combination of longitudinal and vertical accelerometers and gyroscope measurements could provide additional information about the local road gradient.

One related issue not considered in Chapter 7 is the influence of static load changes on the active suspension. IFC removes the resulting offset completely, while LGF reduces the effect by estimating a spurious offset in the steady state value of v_l. Ideally, one might extend the Kalman filtering further to estimate such load changes. Alternatively the two methods could be combined; LGF can be implemented together with feedback of integrated suspension deflections, but now with a very small gain, and hence a very slow time constant, this is however an ad hoc approach.

10.7 Summary of Conclusions

- Non-linear control is beneficial to the control of vehicle suspension.
- Neural networks can be used to map the relationship between system states and the optimal control force or the costates of non-linear control strategies using supervised learning.
- To obtain a successful result using supervised learning, careful consideration of the distribution or weighting of points in the state operational range must be taken.

- Weight decay and bias cancelling are important if a neural network controlled system is to settle correctly.
- The Levenberg Marquardt optimisation approach has considerable advantages over basic line search and gradient descent routines, but systems with large numbers of coefficients can lead to memory problems.
- The H_{norm} approach can, under the right circumstances, generate a stable neural network costate estimator, Obtaining an accurate costate estimator is difficult and research in this field is hampered by the computational expense of the H_{norm} algorithm.

10.8 Further Research

This thesis has been based on neural networks and costate estimation. Once a successful method has been established for the generation of neural network costate estimators, work to investigate their successful integration with both state estimators and real systems can be performed. Supervised learning already provides a means of generating working neural network costate estimators. This will enable the field to be analysed without further development of the H_{norm} optimisation process. It also provides opportunities for work to be done that will further investigate the principles of an integrated control scheme for a full vehicle using costate estimators as defined by Gordon [39].

Neural networks could be perceived to be too flexible for the H_{norm} problem. When a cost function has multiple solutions, it is possible for the network to capture the correct solution in one part of the operational space but not in another. A less flexible approximation technique, where its flexibility can be controlled may prevent this. One such technique that has been proposed is Chebychef polynomials.

In order to make improvements in neural networks, the research and development of computer technology must continue. The work presented here has pushed the limits of current PC power, despite the expectation that the modern PC was capable of such intensive work. The most significant improvements in neural networks will only be made as computing power becomes sufficient to do so.

Some of the networks trained during this PhD have training times that can be measured in days. Only when this is reduced can meaningful research be done. Parallel processing technology already offers significant increases in performance for neural networks, however currently it doesn't have the same accessibility that PCs have achieved and requires special knowledge and equipment.

The systematic use of weighting functions is another area requiring future development. Although weighting on the basis of state magnitude has been reasonably successful, not all points with a high state magnitude have a high associated cost (supervised or H_{norm}). For example, a point at the limit of the suspension workspace but with hub and body velocities that are in a beneficial direction does not incur a large cost. These points suffer under the simple magnitude-weighting scheme, a more advanced approach capable of making an allowance based on predicted cost would be advantageous.

During this thesis, the networks have been trained to learn the control force for a uniform distribution of input states, hyper-spheres and quadroids, based on the premise that before the controller is implemented its operating range cannot be defined. However, if the likely operating range could be defined then the number of points for which the network had to learn the control force mapping would be significantly reduced. This would mean smaller networks could be used and therefore the training time and effort could be reduced. It is also likely that very high cost conditions would not occur and so the influence of high value points in the training cost function would be reduced.

References

1. P.J.TH.Venhovens, A.C.M.V.D. Knaap, and H.B. Pacejka, *Semi-Active Attitude and Vibration Control.* Vehicle System Dynamics, 1993. **22**: p. 359-381.

2. Williams, R.A., *Automotive Active Suspensions Part 2: Practical Considerations.* Proceedings of the Institute of Mechanical Engineers, Part D: Journal of Automobile Engineering, 1997. **211**(6): p. 427-444.

3. Baker, A., *Lotus Active Suspensions.* Automotive Engineer, 1984(Feb./Mar): p. 56-57.

4. Strensson, A., C. Asplund, and L. Karlsson, *The Nonlinear Behaviour of a MacPherson Strut Wheel Suspension.* Vehicle System Dynamics, 1994. **23**: p. 85-106.

5. Gordon, T.J., *Non-linear optimal control of a semi-active vehicle suspension system.* Chaos Solutions & Fractals, 1995. **5**(9): p. 1603-1617.

6. Gordon, T.J., C. Marsh, and M.G. Milsted, *A Comparison of Adaptive LQG and Non-linear Controllers for Vehicle Active Suspension Systems..* Vehicle System Dynamics, 1991. **20**: p. 321-340.

7. Williams, D.A. and P.G. Wright, *Vehicle Suspension System.* 1986, Group Lotus Public Limited Company,Norwich, England: United States Patent.

8. Mercedes, *Road Test Number 4445, Mercedes-Benz CL500,* in *Autocar.* 14/6/00. p. 56-59.

9. Darling, J., R.E. Dorey, and T.J. Ross-Martin, *A low cost active anti-roll suspension for passenger cars.* Journal of Dynamic Systems Measurement & Control-Transactions of the ASME, 1992. **114**(4): p. 599-605.

10. LandRover, *Road Test Number 4346 Land Rover Discovery,* in *Autocar.* 14/10/98. p. 50-55.

11. Bender, E.K., *Some Fundamental Limitations of Active And Passive Vehicle-Suspension Systems.* SAE Paper No. 680750, 1968.

12. Karnopp, D.C. and A.K.Trikha, *Comparative Study of Optimization Techniques for Shock and Vibration Isolation.* Transactions ASME, Journal of Engineering Industry, 1969: p. 1128-1132.

13. Gillespie, T.D., *Ride,* in *Fundamentals of Vehicle Dynamics.* 1992, Society of Automotive Engineers, Inc. p. 125-189.

14. Davis, L.I., et al., *A Benchmark Problem for Neural Control: Quarter-Car Active Suspension.* Intelligent Engineering Systems Through Artificial Neural Networks, 1993. **3**: p. 581-586.

15. Thompson, A.G. and C.E.M. Pearce, *Physically realisable feedback controls for a fully active preview suspension applied to a half-car model.* Vehicle System Dynamics, 1998. **30**(1): p. 17-35.

16. Esmailzadeh, E. and F. Fahimi, *Optimal adaptive active suspensions for a full car model.* Vehicle System Dynamics, 1997. **27**(2): p. 89-107.

17. Williams, R.A., *Active Suspensions Classical or Optimal.* Vehicle System Dynamics, 1985. **14**(1-3): p. 127-132.

18. Hac, A., *Adaptive Control of Vehicle Suspension.* Vehicle System Dynamics, 1987. **16**: p. 57-74.

19. Kim, C. and P.I. Ro, *A sliding mode controller for vehicle active suspension systems with nonlinearities.* Proceedings of the Institue of Mechanical Engineers, Part D, 1998. **212**(D2): p. 79-92.

20. Kurimoto, M. and T. Yoshimura, *Active suspension of passenger cars using sliding mode controllers (based on reduced models).* International Journal of Vehicle Design, 1998. **19**(4): p. 402-414.

21. Yokoyama, M., K.J. Hedrick, and S. Toyama. *A Model Following Sliding Mode Controller for Semi-Active Systems with MR Dampers.* in *American Control Conference.* 2001. Arlington, VA.

22. A.S.Cherry and R.P.Jones, *Fuzzy Logic Control of Automotive Suspension System.* IEE Proceedings: Control Theory & Applications, 1995. **142**(2): p. 149-160.

23. Kuo, Y. and T. Li, *GA-based fuzzy PI/PD controller for automotive active suspension system. IEEE Transactions on Industrial Electronics.* IEEE Transactions on Industrial Electronics, 1999. **46**(6): p. 1051-6.

24. Vallurupalli, S.S., R.V. Dukkipati, and M.O.M. Osman, *Adaptive Active Suspension for a Half Car Model with a Stochastic Dirt Road Input.* AVEC, 1994: p. 367-372.

25. Henson, M.A. and D.E. Seborg, *Feedback Linearizing Control,* in *Nonlinear Process Control,* M.A. Henson and D.E. Seborg, Editors. 1997. p. 149-232.

26. Buckner, G.D., K.T. Schuetze, and J.H. Beno. *Active Vehicle Suspension Control Using Intelligent Feedback Linearization.* in *American Control Conference.* 2000.

27. Gobbi, M., G. Mastinu, and C. Doniselli, *Optimizing a Car Chassis.* Vehicle System Dynamics, 1999. **32**(2): p. 149-170.

28. Giacomin, J., *Neural Network Simulation of an Automotive Shock Absorber.* Engineering Applications of Artificial Intelligence, 1991. **4**(1): p. 59-64.

29. Pandya, A.S. and R.B. Macy, *Pattern Recognition with Neural Networks in C++.* 1995: IEEE Press.

30. Arditi, D., F.E. Oksay, and O.B. Tokdemir, *Predicting the Outcome of Construction Litigation using Neural Networks.* Computer Aided Civil and Infrastructure Engineering, 1998. **13**(2): p. 75-81.

31. Fernandez-Rodriguez, B., N. Jaramillo, and N. Pandya, *Neural Network Controllers: A Literature Review.* Intelligent Engineering Systems Through Artificial Neural Networks, 1996. **6**: p. 533-542.

32. Narendra, K.S. and S. Mukhopadhyay, *Adaptive Control Using Neural Networks and Approximate Models.* IEEE Transactions on Neural Networks, 1997. **8**(3): p. 475-485.

33. Jeong-Woo, L. and O. Jun-Ho, *Time Delay Control of Nonlinear System with Neural Network Modelling.* 1997.

34. Moran, A., T. Hasegawa, and M. Nagai. *Continuously controlled Semi-Active Suspension using Neural Networks.* in *AVEC 94.* 1994.

35. Moran, A. and M. Nagai. *Optimal Preview Control of Rear Suspension Using Nonlinear Neural Networks.* in *AVEC.* 1992.

36. Watanabe, Y. and R.S. Sharp, *Neural Network Learning Control of Automotive Active Suspension Systems.* International Journal of Vehicle Design, 1999. **21**(2/3): p. 124-147.

37. Hampo, R.J. and K.A. Marko. *Neural Network Architectures for Active Suspension Control.* in *International Conference on Neural Networks.* 1991. Seattle: IEEE.

38. Hampo, R.J. and K.A. Marko. *Investigation of the Application of Neural Networks to Fault Tolerant Control of an Active Suspension System.* in *American Control Conference.* 1992. Chicago: ACC.

39. Gordon, T.J., *An Integrated Strategy for the control of a Full Vehicle Active suspension System.* Vehicle System Dynamics Supplement, 1996. **25**: p. 229-242.

40. Banks, S.P., *Control Systems Engineering.* 1986: Prentice / Hall International.

41. Anderson, B.D.O. and J.B. Moore, *The Riccati Equation*, in *Optimal Control, Linear Quadratic Methods*. 1990, Prentice Hall. p. 370-374.

42. Sharp, R.S. and D.A. Crolla, *Road Vehicle Suspension Design - A Review.* Vehicle System Dynamics, 1987. **16**(3): p. 167-192.

43. Milliken, W.F. and D. L.Milliken, *Race Car Vehicle Dynamics*. 1995: SAE.

44. Dixon, J.C., *Tires, Suspension and Handling*. 2nd ed. 1996: SAE.

45. Williams, R.A., *Automotive Active Suspensions, Part 1: Basic Principles.* Proceedings of the Institute of Mechanical Engineers, Part D: Journal of Automobile Engineering, 1997. **211**(6): p. 415-426.

46. Thompson, A.G., *An Active Suspension with Optimal Linear State Feedback.* Vehicle System Dynamics, 1976. **5**: p. 187-203.

47. Wilson, D.A., R.S. Sharp, and S.A. Hassan, *The Application of Linear Optimal Control Theory to the Design of Active Automotive Suspension.* Vehicle system Dynamics, 1986. **15**: p. 105-118.

48. Marsh, C., *A Nonlinear Control Design Methodology for Computer-Controlled Vehicle Suspension Systems*, in *Department of Aeronautical and Automotive Engineering*. 1992, Loughborough University: United Kingdom.

49. Bryson, A., E, Jr. and Y.-C. Ho, *Applied Optimal Control, Optimization, Estimation and Control.* 1975.

50. Demuth, H. and M. Beale, *Neural Network Toolbox User's Guide*. 4th ed. Matlab User Guides. 1992: The Math Works, Inc.

51. Hagan, M.T., H.B. Demuth, and M. Beale, *Neural Network Design*. 1996, Boston; London: PWS Publishing.

52. Hertz, J., A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*. Vol. 1. 1991: Addison-Wesley Publishing Company.

53. M.Nørgaard, et al., *3.2 Direct Inverse Control*, in *Neural Networks for Modelling and Control of Dynamics Systems*. 2000, Springer.

54. Psaltis, D., A. Sideris, and A.A. Yamamura, *A Multilayer Neural Network Controller.* IEEE Control Systems Magazine, 1988: p. 17-21.

55. Wu, Q.M.J., K. Stanley, and C.W.d. Silva, *4.4 Neurocontroller-Design Methods*, in *Intelligent Adaptive Control, Industrial Applications*, L.C. Jain and C.W.d. Silva, Editors. 1999, CRC Press.

56. El-Gindy, M. and L. Palkovics, *Possible Application of Artificial Neural Networks to Vehicle Dynamics and Control: A Literature Review.* International of Vehicle Design, 1993. **14**(5/6): p. 592-614.

57. Hrycej, T., *Chapter 8 Learning and Adaptiveness in Neurocontrol, Towards an Industrial Control Methodology.* 1997, John Wiley & Sons. p. 223-241.

58. Werbos, J.P., *Backpropagation Through Time: What it Does and How to do it.* Proceedings of the IEEE, 1990. **78**(10): p. 1550-1560.

59. Nguyen, D.H. and B. Widrow, *Neural Networks for self-learning control systems.* Control Systems Magazine, 1990. **10**(3): p. 18 - 23.

60. Prokhorov, D.V., *Adaptive Critic Designs and their Applications,* in *Department of Electrical Engineering.* 1997, Texas Tech University.

61. Saini, G. and S.N.Balakrishnan. *Adaptive Critic Based Neurocontroller for Autolanding of Aircrafts with Varying Glidslopes.* in *IEEE International Conference on neural networks.* 1997. USA: IEEE.

62. Anderson, C.W., *Learning to Control an Inverted Pendulum Using Neural Networks.* Control Systems Magazine, 1989. **9**(3): p. 31-37.

63. Shimizu, K., Y. Ishisuka, and M. Ohtani. *Optimal State Feedback Control Law for Nonlinear Systems and Its Approximation by A Neural Network.* in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics.* 1999. Tokyo Jpn.

64. kim, H. and Y.-S. Yoon, *Neuro Controlled Active Suspension with Preview for Ride Comfort.* SAE, 1993(931969): p. 2133-2141.

65. Thorpe, C., et al., *Toward autonomous driving: The CMU Navlab--I: Perception.* IEEE Expert, 1991. **6**(4): p. 31-42.

66. Macadam, C.C. and G.E. Johnson, *Application of elementary Neural Networks and Preview Sensors for Representing Driver Steering Control Behaviour.* Vehicle System Dynamics, 1996: p. 3-30.

67. Press, W.H., et al., *Numerical Recipes - The Art of Scientific Computing.* 2nd ed. 1992: Cambridge University Press.

68. Press, W.H., et al., *9.2 Secant Method, False Position Method and Ridders' Method,* in *Numerical Recipes.* 1992, Cambridge University Press. p. 347-352.

69. Chong, E.K.P. and S.H. Zak, *Section 11, Quasi-Newton Methods,* in *An Introduction to Optimization.* 1996, Wiley - Interscience.

70. Narendra, K.S. and K. Parthasarathy, *Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks.* IEEE Transactions on Neural Networks, 1991. **2**(2): p. 252-262.

71. Nakanishi, H., T. Kohda, and K. Inoue. *Design Method of Optimal Control System by use of Neural Network.* in *IEEE International Conference on Neural Networks.* 1997. Houstan, Texas, USA.

72. Haykin, S., *Section 5.5, Regularization Theory,* in *Neural Networks, A comprehensive Foundation.* 1999, Prentice Hall. p. 267-270.

73. Catala, A. and C. Angulo, *Comparison between the Tikhonov and Baysian approaches to calculate regularisation matrices.* Neural Processing Letters, 2000. **11**(3): p. 185-195.

74. Fu, L.M., *Neural Networks in Computer Intelligence.* 1994: McGraw - Hill. 93.

75. Reed, R., *Pruning Algorithms - A Survey.* IEEE Transactions on Neural Networks, 1993. **4**(5): p. 740-747.

76. Krogh, A. and J.A. Hertz, *A Simple Weight Decay can Improve Generalization.* Advances in Neural Information Processing Systems, 1995. **4**: p. 950-957.

77. Sharma, k., D.A. Crolla, and D.A. Wilson. *The Design of A Fully Active Suspension System Incorporating a Kalman Filter for State Estimation.* in *Control 94.* 1994: IEE.

78. Fairgrieve, A. and T.J. Gordon, *On-line Estimation of Local Road Gradient for Improved Steady State Suspension Deflection Control.* Vehicle System Dynamics Supplement, 1999. **33**: p. 590-603.

79. Karnopp, D.C., M.J. Crosby, and R.A. Harwood, *Vibration Control Using Semi-Active Force Generators.* ASME Journal of Engineering for Industry, 1974. **96**(2): p. 612-626.

80. Best, M.C., *On the Modelling Requirements for a Practical Implementation of Advanced Vehicle Suspension Control,* in *Department of Aeronautical and Automotive Engineering.* 1995, Loughborough University: Loughborough.

81. Thompson, A.G. and B.R. Davis, *Optimal Linear Active Suspension with Derivative Constraints and Output Feedback Control.* Vehicle System Dynamics, 1998. **17**: p. 179-192.

82. Elmadany, M.M., *Optimal Linear Active Suspension with Multivariable Integral Control.* Vehicle System Dynamics, 1990. **19**: p. 313-329.

83. ElMadany, M.M., *Integral and State Variable Feedback Controllers for Improved Performance in Automotive Vehicles.* Computers and Structures, 1992. **42**(2): p. 237-244.

84. Hecht-Nielsen, R., *Chapter 3, Learning Laws: Self-Adaptation Equations*, in *Neurocomputing*. 1990, Addison-Wesley. p. 48-49.

85. Bertsekas, D.P., *Constrained Optimization and Lagrange Multiplier Methods*. Computer Science and Applied Mathematics. 1982: New York, London Academic Press.

86. Rowland, J.R., *8.2.3 The Linear Quadratic Control Problem*, in *Linear Control Systems*. 1986, John Wiley & Sons. p. 469-470.

87. Kailath, T., *3.4.3 The Algebraic Riccati Equation*, in *Linear Systems*. 1980, Prentice Hall. p. 230-237.

88. Werbos, P.J., *Neural Control and Supervised Learning*, in *Handbook of Intelligent Control*, S.D.A. White David A, Editor. 1992, Van Nostrand Reinhold: New York. p. 65-89.

89. Kreyszig, E., *1.9 Existence and Uniqueness of Solutions Picard Iteration*, in *Advanced Engineering Mathematics*. 1999. p. 52-61.