Adaptive Control of Nonlinear Systems Using Neural Networks

by

Sanjay Kumar Mazumdar, B.E.(Hons)

A thesis submitted in fulfilment of the requirement for the degree of

Doctor of Philosophy

SUB CRUCE LUMEN

The University of Adelaide

Faculty of Engineering

Department of Electrical and Electronic Engineering

March 1995

Awarded 1995

Gayatri Mantra

Om Bhur Bhuvah Svah

Tat Savitur Varenyam

Bhargo Devasya Dhimahi

Dhiyo Yo Nah Prachodayat

Meaning: Let us meditate on the glory of Ishwar (the Lord), Who has created this universe, Who is fit to be worshipped, Who is the embodiment of knowledge and light, Who is the remover of all sins and ignorance. May He enlighten our intellect.

Contents

Abstract

In contrast to linear adaptive control, adaptive design techniques for nonlinear systems have yet to be established for a general class of nonlinear structure. Most of the current approaches to nonlinear adaptive control, such as sliding control, input-output linearisation and the popular feedback linearisation, primarily deal with systems where the uncertainty is due to unknown parameters which appear linearly with respect to the known nonlinearities. Artificial neural networks have offered an alternative approach to solve a more general class of nonlinear problems. In particular, it is their ability to form an arbitrarily close approximation of any continuous nonlinear function and their inherent adaptivity, that has generated much of the research into the use of neural networks for the identification and control of nonlinear systems.

This thesis is concerned with the development of a stable neural network based adaptive control scheme for discrete-time nonlinear systems. The scheme is based on the model reference adaptive control design methodology with a multi-layered neural network generating the model reference control. The neural adaptive control framework is developed for arguably the least analytically tractable nonlinear system, namely general multi-input multi-output non-affine discrete-time dynamic systems with unknown structure. The relative degree and order of the system and the maximum lag in the plant input and plant output terms are the only *a priori* knowledge assumed.

Critical to any model reference adaptive control approach is the convergence of the tracking error and the stability of the closed-loop system. Therefore, an enhancement is proposed to the model reference neural adaptive control scheme which enables the derivation of sufficient conditions to guarantee the convergence of the tracking error between the controlled output and the desired response. Lyapunov theory is used to guarantee the stability of the closed-loop system.

Simulation studies undertaken demonstrate the effectiveness of the proposed scheme in controlling discrete-time nonlinear systems which may consist of non-idealities such as nonminimum phase or marginally stable behaviour, as well as dynamic, sensor or load disturbances. The robustness of the new neural adaptive control scheme to dynamic variations and uncertainties is also demonstrated. The practical feasibility of the new approach is investigated through its application to an automobile anti-skid brake system. Despite the highly nonlinear and time-varying dynamics of the vehicle/brake system, the simulation study results indicate that the proposed neural network based anti-skid brake system can provide effective braking performance even under severe variations in environmental conditions.

From the research presented in the thesis, it is concluded that the use of artificial neural networks in the adaptive control of nonlinear systems indicates much promise for the future. Furthermore, the results of the work provide a basis for the development of practical neural adaptive controllers.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I give consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying.

Signed:... ....Date:.........24/3/95................

Acknowledgements

I gratefully acknowledge my supervisor, Dr. C. C. Lim for his guidance, encouragement and assistance throughout the course of this research. I would like to thank him for his many useful suggestions and ideas, and his thorough and detailed reviews of my work.

I also wish to express my gratitude to Dr. D. Nandagopal for his many thoughtful comments, particularly when he was acting as my supervisor during Dr. Lim's study leave.

Many thanks must go to the staff and students of the Department of Electrical and Electronic Engineering at the University of Adelaide for providing an enjoyable and stimulating work environment. In particular, my appreciation goes to John M^{c}Pheat and my fellow inhabitants of room N238, Jonathan Main and Derek Rogers, for their constant encouragement and friendship.

I would also like to take this opportunity to thank Dr. C. Coleman, Head of the Guidance and Control Group of the Weapon Systems Division, DSTO, for allowing the time off from work to complete the final stages of this thesis.

Finally, I am deeply indebted to my parents for all of their support and guidance during the period of this study. I would also like to thank my father for his thorough review of this thesis.

This work was supported by an Australian Postgraduate Research Priority Award from the Australian Research Council, which I gratefully acknowledge.

S.K.M

List of Publications

The following is a list of publications which are related to the research reported in this thesis.

S.K. Mazumdar and C.C.Lim, "Adaptive Controller for Marginally Stable Nonlinear Systems Using Neural Networks", *TENCON '92: IEEE Region 10 International Conference*, pp 535–539, November 1992.

S.K. Mazumdar and C.C. Lim, "A Stable Neural Controller for Nonminimum Phase", *Fourth Australian Conference on Neural Networks*, pp 138–141, February 1993.

S.K. Mazumdar and C.C. Lim, "Investigation of Stability and Convergence Issues for an Enhanced Model Reference Neural Adaptive Control Scheme (Invited Paper)", *ETD 2000 - Electronics Technology Directions to the Year 2000*, 1995.

S.K. Mazumdar and C.C. Lim, "Investigation of the Use Neural Networks for Anti-Skid Brake System Design", *Submitted*, February 1995.

S.K. Mazumdar and C.C. Lim, "An Enhanced Neural Adaptive Control Scheme for Discrete-Time non-Affine Nonlinear Systems", *Submitted*, March 1995.

S.K. Mazumdar and C.C. Lim, "A Neural Network Based Anti-Skid Brake System", *Submitted*, March 1995.

List of Principal Symbols

$\mathbb{N}$	set of natural numbers or positive integers
$\mathbb{R}$	set of real scalars
$\mathbb{R}^n$	set of real n-dimensional vectors
$\mathbb{Z}^+$	set of positive integers
$\Re[a, b]$	set of real numbers $x : a \leq x \leq b$
$f : \mathbb{R}^n \rightarrow \mathbb{R}^m$	function f maps $x \in \mathbb{R}^n$ onto $f(x) \in \mathbb{R}^m$
A^T	transpose of matrix A
$\mid . \mid$	vector norm
$\parallel . \parallel$	Euclidean norm or l_2-norm
α	momentum rate
β	steepness parameter (neural network context)
β	stability constant for SISO system
γ	relative degree
δ	effective error for neural network layer
ε_I	identification error tolerance
ε_T	tracking error tolerance
η	learning rate
θ	vector of unknown parameters
θ	angle of incline of the road (ABS context)
θ_c	controller parameter vector
λ	eigenvalues of a matrix A
λ	wheel slip (ABS context)
λ_{max}	maximum eigenvalue of A

$\lambda_{\mu_{\max}}$	value of wheel slip for maximum adhesion coefficient
Λ	diagonal matrix with eigenvalues of A on main diagonal
μ	neuron threshold
μ	coefficient of braking friction (ABS context)
π	estimated plant parameters
σ	normalisation parameter (basis width)
ϕ	radial basis function
Φ	diffeomorphism function
ω_v	angular speed of a free spinning wheel
ω_w	wheel angular speed
$\Omega^3_{2,10,5,1}$	3 layered neural network consisting of 2 inputs, 1 output and 10 and 5 nodes in the hidden layers

$c(.)$	residual
d	desired network response
e_c	controller error
e_I	identification error
e_T	tracking error
$f(.), g(.), h(.)$	smooth nonlinear functions (control context)
$f_m(.)$	smooth (usually linear) reference model function
$g(.)$	activation function (neural network context)
g	gravitational acceleration constant
h	sampling period
h	weighted summed input to a network node
k	time sample index for discrete-time systems
l	maximum lag in output terms
m	maximum lag in control terms
n	output vector dimension
n_w	number of wheels
p	number of output layer weights
q	shift operator

r	reference input vector
r	control vector dimension
t	time index for continuous-time systems
u	plant control vector
v	transformed input vector
$v(.)$	desired response
x	neural network input vector
x	system state vector
y	neural network output
y_d	desired trajectory
y_m	reference model output vector
y_p	plant output vector
$\hat{y}_p$	neural network output estimate vector
z	transformed state vector
z_d	desired trajectory vector
z_1	vector of transformed states
z_2	vector of internal dynamics state variables
A	stability matrix
B_w	viscous friction of a wheel
B_v	viscous friction of a vehicle
E	cost function
F_θ	force applied to the car due to road gradient
F_t	tyre friction force
I	identity matrix
J_w	rotational inertia of a wheel
$L_f h$	Lie derivative of $h(.)$ with respect to $f(.)$
M	number of network layers
M_v	vehicle mass
N_c	neural network controller
N_f	neural network approximating function $f(.)$

N_g	neural network approximating function $g(.)$
N_p	neural network emulating the plant dynamics
N_v	normal force at the tyre
R_w	wheel radius of free rolling tyre
S	matrix of eigenvectors of A
T_b	braking torque at the wheel
T_t	torque generated due to tyre-road friction
$V(.)$	Lyapunov function
V_v	vehicle linear speed
W	weight matrix
ΔW	change in weight matrix
W_c	controller neural network weight matrix
ΔW_c	change in controller neural network weight matrix
W_{ij}^m	weight for the connection from the jth node in layer $m-1$ to the ith node in layer m

List of Abbreviations

ANN	Artificial Neural Network
ABS	Anti-skid Brake System
BP	Backpropagation
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
LTI	Linear Time-Invariant
MIMO	Multi-Input Multi-Output
MLP	Multi Layer Perceptron
MRAC	Model Reference Adaptive Control
NN	Neural Network
NN-ABS	Neural Network Anti-Skid Brake System
ODS	Optimal Decision Strategy
PID	Proportional plus Integral plus Derivative
RBF	Radial Basis Function
SISO	Single-Input Single-Output
STR	Self-Tuning Regulator
TDNN	Time Delay Neural Network

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Linear Control Theory

It is well known that linear control is an established field for which there exists both a sound theoretical background and a substantial record of successful industrial applications. There are two main approaches for the study of linear control systems, namely classical control and state-space (modern) control. Classical control approaches were developed mainly between the 1930's and the 1950's and they are based on frequency response techniques using, for example, root locus and bode plots. A typical classical controller which is still used extensively in industry is the proportional plus integral plus derivative (PID) controller. More sophisticated controller design methods such as linear quadratic regulators (LQR), linear quadratic Gaussian (LQG) controllers and estimated-state feedback controllers have emerged since the 1960's. These state-space design methods have arisen because of the ability to express the solutions to some optimal control problems in the form of a feedback law or controller. Linear control theory is generally restricted by the assumption that the plant and controller are linear time-invariant (LTI) systems. These systems can be represented either in the state-space form

$$\begin{array}{rclcrcl} \dot{x}(t) & = & Ax(t)+Bu(t) & \text{or} & x(k+1) & = & \Phi x(k)+\Gamma u(k) \\ y(t) & = & Cx(t)+Du(t) & & y(k) & = & Cx(k)+Du(k) \\ & \text{continuous-time} & & & & \text{discrete-time} & \end{array} \tag{1.1}$$

where $x \in \mathbf{R}^n$ is the state vector, $u \in \mathbf{R}^r$ is the input vector, $y \in \mathbf{R}^m$ is the output vector, A is an $n \times n$ matrix, B is an $n \times r$ matrix, C is an $m \times n$ matrix, D is an $m \times r$ matrix, Φ is an $n \times n$ matrix and Γ is an $n \times r$ matrix, or as input-output transfer functions

$$\begin{array}{ccc} G(s) = \frac{N(s)}{D(s)} & & H(z) = \frac{S(z)}{R(z)} \\ \text{continuous-time} & & \text{discrete-time} \end{array} \tag{1.2}$$

where $N(s)$ and $D(s)$ are the numerator and denominator polynomials in the s-domain, and $S(z)$ and $R(z)$ are the corresponding polynomials in the z-domain. In the "real world" the concept of a linear time-invariant system is, however, a mathematical abstraction. All physical systems are nonlinear to some degree. Nonlinearities can be classified as either inherent (natural) nonlinearities which arise from the system behaviour or the hardware used, or intentional (artificial) nonlinearities which are artificially introduced into the system. Some typical examples of inherent nonlinearities resulting predominantly from the hardware in the system are saturations, hysteresis and deadzones. Drag on a vehicle, gravitational forces on satellites and the frictional effects of a road surface on a vehicle/brake system are examples of naturally occurring nonlinear system behaviour. Examples of intentional nonlinearities are adaptive control laws, robust control laws, and bang-bang optimal control laws.

Regardless of the exact nature of the nonlinearities, there is one fact that is quite apparent - the behaviour of nonlinear systems is far more complex than linear systems. In particular, many of the inherent properties of linear systems, which greatly simplify the solution for this type of system, are not valid for nonlinear systems. For example, the principle of superposition, which is a fundamental property of linear systems and, in fact, is also the basis of the definition of a linear system, does not apply to nonlinear systems. Therefore, several mathematical procedures used in linear systems cannot be used in nonlinear systems. Moreover, unlike their linear counterparts, the stability of nonlinear systems is not just a simple function of the location of its eigenvalues. Instead, it depends on the initial conditions and the nature of the input signal, as well as the system parameters. In fact, a nonlinear system that exhibits a stable response for one type of input may not have a stable response for another type of input. The rich and complex behaviour of nonlinear systems has given rise to a number of stability concepts such as asymptotic stability, exponential stability and global asymptotic stability. Furthermore, a nonlinear system excited by a periodic signal may not result in steady state outputs of the same frequency as the input. Higher harmonics, sub-harmonics or even chaotic behaviour (continuous spectra) can occur in the output of the nonlinear system. Also, an unforced nonlinear system may also display limit cycle behaviour (periodic oscillations of a fixed frequency and amplitude) which does not exist in stable linear systems [1].

The above and many other properties demonstrate that the analysis of nonlinear systems is far more complex than that of linear systems. Unlike their linear counterparts, nonlinear equations cannot, in general, be solved analytically. Therefore a complete understanding of the behaviour of nonlinear systems is difficult to obtain. Furthermore, mathematical tools such as the Laplace transform method, the transfer function approach, and the state-space formulation commonly utilised in linear control theory, cannot be used for nonlinear systems. Consequently, there are no conventional methods for analysing, or systematic procedures for designing, *general* nonlinear systems.

The common approach to deal with mild nonlinear systems which have sufficiently smooth, continuously differentiable nonlinearities is to use a linear approximation model. This is obtained by linearising the system about a known nominal solution or operating point. In this approach, it is assumed that a nonlinear system behaves similarly to its linearised approximation for a small range around the nominal solution, thus allowing linear control approaches to be used. This reasoning is the principle justification for using linear control theory in practice. The above approach is often referred to as Jacobian linearisation, small-signal theory or theory of small perturbations and can be illustrated by the following equations:

[1]Similar oscillations can be found in marginally stable linear systems. However, the amplitude of the oscillations in a marginally stable linear system is dependent on the initial conditions, whereas the amplitude of the limit cycle oscillations is independent of the initial conditions. Furthermore, limit cycles are generally robust to parameter changes, whereas marginally stable linear systems can be made stable or unstable via small changes in the system parameters.

Consider a general continuous-time nonlinear system described by the state equation

$$\dot{x} = f(x, u, t) \tag{1.3}$$

where $f(.)$ is a continuously differentiable nonlinear function.

Suppose a nominal solution $\{x_n(t), u_n(t)\}$ is known. The difference between the nominal vector functions and some slightly perturbed functions $x(t)$ and $u(t)$ can be defined as

$$\begin{aligned} \delta x &= x(t) - x_n(t) \\ \delta u &= u(t) - u_n(t) \end{aligned} \tag{1.4}$$

Therefore, equation (1.3) can be written as

$$\begin{aligned} \dot{x}_n + \delta\dot{x} &= f(x_n + \delta x, u_n + \delta u, t) \\ &= f(x_n, u_n, t) + \left[\frac{\partial f}{\partial x}\right]_n \delta x + \left[\frac{\partial f}{\partial u}\right]_n \delta u + \text{higher order terms} \end{aligned} \tag{1.5}$$

where $[\,.\,]_n$ indicates that the derivatives are evaluated at the nominal solution $\{x = x_n, u = u_n\}$.

Since the nominal solution satisfies equation (1.3), the first terms in the above Taylor series expansion cancel. For small perturbations $(\delta x, \delta u)$ (hence the name "theory of small perturbations") the higher order terms can be neglected. Therefore equation (1.5) becomes

$$\delta\dot{x} = A\delta x + B\delta u \tag{1.6}$$

where $A = \left[\frac{\partial f}{\partial x}\right]_n$ denotes the Jacobian matrix of f with respect to x at $\{x = x_n, u = u_n\}$ and $B = \left[\frac{\partial f}{\partial u}\right]_n$ denotes the Jacobian of f with respect to u at the same point. The above system (1.6) is the *linear approximation* of the original nonlinear system (1.3) at the nominal solution.

If $x_n(t)$ is a constant x_e and if $u_n(t) = \delta u(t) = 0$, then the stability of the equilibrium point x_e is governed by

$$\delta\dot{x} = A\,\delta x \tag{1.7}$$

where $\delta x = x - x_e$ and the Jacobian matrix A is a constant. The relationship between the stability of the above linear system (1.7) and that of the original nonlinear system

(1.3) is elucidated by Lyapunov's linearisation method.

THEOREM 1.1.1 Lyapunov's Linearisation Method

- *If the linearised system is strictly stable (i.e., if all the eigenvalues of A are in the left half of the s-plane), then the equilibrium point x_e is asymptotic stable (for the actual nonlinear system),*
- *If the linearised system is unstable (i.e., if at least one eigenvalue A is strictly in the right-half of the s-plane), then the equilibrium point is unstable (for the nonlinear system),*
- *If the linearised system is marginally stable (i.e., at least one eigenvalue of A is on the $j\omega$ axis and the others are in the the left-half of the s-plane), then no conclusion about the stability of the equilibrium point can be obtained from the linear model (i.e., x_e may be stable, asymptotically stable or unstable for the nonlinear system),*

where the following definitions apply:

DEFINITION 1.1.1 *A state x^* is an equilibrium state (or point) of a dynamic system if once the state vector x equals x^*, it remains equal to x^* for all future times, i.e., given the system $\dot{x} = f(x,t)$, then the equilibrium state x^* satisfies $f(x^*,t) = 0 \ \forall t$.*

DEFINITION 1.1.2 *The origin ($x = 0$) is a stable equilibrium point if, for any $R > 0$, there exists $r > 0$, such that if $\|x(0)\| < r$, then $\|x(t)\| < R$ for all $t \geq 0$. Otherwise, the origin is said to be an unstable equilibrium point.*

DEFINITION 1.1.3 *The origin is an asymptotically stable equilibrium point if it is stable and if in addition there exists some $r > 0$ such that $\|x(0)\| < r$ implies that $x(t) \to 0$ as $t \to \infty$.*

Theorem 1.1.1 is very useful because it demonstrates that stable design by linear control guarantees the local stability of the original physical system [218]. The above Jacobian linearisation approach has been successfully used in many applications such as missile autopilots [25, 211], ship autopilots [97] and weapon systems [17]. Furthermore, the use of a linearisation approach for applications such as satellite attitude control, aircraft autopilots and engine control are considered in [54].

Given the many practical successes of applying linear control theory to nonlinear systems via linearisation approaches, the question which is often posed is "What is the motivation behind the extensive research recently conducted into nonlinear control theory?"

A number of reasons can be provided to answer this question. Firstly, the Jacobian linearisation approach described above relies on the assumption that the region of operation is small in order that the linear model is valid. If this is not the case, then the inherent nonlinearities in the system may result in a poorly performing or even unstable linear controller. In contrast, a nonlinear controller would most probably be able to deal with the nonlinearities over a large region of operation. Secondly, in the linear control approach to nonlinear systems, it is usually assumed that the system model is indeed linearisable, i.e., the nonlinearities are continuously differentiable. Often this is not the case, particularly if the system includes discontinuous nonlinearities such as dead-zones, hysteresis and saturation. As hard linearities such as these often result in instabilities or limit cycle behaviour, nonlinear control methods must be employed to compensate for them. Thirdly, in linear control it is generally assumed that the system parameters are invariant and well known. However, many systems are subject to parametric or dynamic uncertainties and/or variations. These may arise because of a slow time variation of the parameters (e.g., parts fatiguing, corroding or otherwise deteriorating with time) or abrupt changes in parameters or dynamics (e.g., a change in road friction when a vehicle moves from one road surface to another). A linear controller based on such a system may perform poorly or even be unstable. Nonlinear control approaches such as robust control, adaptive control and neural control, by their very nature, are able to tolerate or compensate for such disturbances. Finally, as most physical systems are inherently nonlinear by nature, it is more intuitive and even natural to design nonlinear controllers for them. In addition to the above limitations of linear control approaches (and advantages of nonlinear control methodologies), traditional concerns about the computational complexity and intensiveness of nonlinear control techniques have been overcome with the advent of powerful and inexpensive digital signal processing (DSP) chips and microprocessors. Hence the above factors plus the need for more accurate and reliable control techniques have made nonlinear control an extremely dynamic area of research and development in recent years. This is evidenced by the substantial research work done in this area such as in [22, 69, 90, 108, 112, 168, 176, 218], and in recent American Control Conference Proceedings, IFAC World Congress Proceedings or IEEE Transactions on Automatic Control.

1.1.2 Nonlinear Control Theory

The conventional design approaches for nonlinear controllers can be categorised into a number of classes, namely robust control [47, 218], gain scheduling [9, 199], feedback linearisation [90, 173] and adaptive control [167, 207].

In robust control approaches such as sliding control and H_∞, the controller is designed to effectively account for parameter uncertainty and the presence of unmodelled dynamics. The typical structure of a robust controller consists of a nominal part, similar to a feedback linearising law or an inverse control law and an additional term which aims to deal with the model uncertainty. Robust control has been an active area of research and has proved to be very effective in a number of practical control problems, including aircraft control [1], power system stabilisers [121], ship tracking [149], vehicle traction control systems [230], and robotics [114, 224, 257]. For a more detailed discussion of robust control techniques the books by Slotine and Li [218] and Dorato *et al.* [47] and the collection of papers edited by Dorato *et al.* [46, 48] are recommended.

Gain scheduling is an attempt to apply linear control approaches to the control of nonlinear systems. The central idea is to select a family of operating points which cover the range of operation for the nonlinear system. For each of these operating points an approximate LTI model of the nonlinear plant is obtained, usually via the Jacobian linearisation procedure. A linear controller is then designed for each of the linearised models. Based on the measured signals, an estimate is made of the operating point closest to the current plant state and the corresponding linear controller is scheduled. The overall nonlinear controller for the nonlinear plant is obtained by interpolating or "scheduling" the gains of the local operating point designs via a selection algorithm. Gain scheduling is a conceptually simple approach which has been applied to a number of applications such as ship autopilots [98] and flight control systems [172, 193, 249]. Several other applications of gain scheduling such as pH control, engine control, and the control of rolling mills are briefly discussed in [9]. One advantage of gain scheduling is that it allows the incorporation of linear robust control approaches into nonlinear control design. However, gain scheduling is computationally intensive due to the need to compute many linear controllers. It also requires the plants to be linearisable and has limited theoretical stability guarantees. Some of the other potential hazards of gain scheduling and their possible solutions are addressed in [210]. Further details on this nonlinear control approach can be found in [9, 199].

Adaptive control is an extremely active area of applied and pure research, as evidenced by the large volume of literature in this area, for example [63, 69, 111, 167, 168, 176, 207]. It is also the framework chosen for many neural network based control approaches, including the model reference based neural adaptive control scheme proposed in this thesis. Therefore, it is pertinent to include a brief review of adaptive control techniques, and in particular model reference adaptive control. Similarly, significant theoretical and practical advances have been made in the area of geometric approaches to the control of nonlinear system, i.e., feedback linearisation, input-output linearisation and output feedback controllers. These approaches are arguably the most popular techniques for controlling nonlinear systems, a fact supported by the significant body of work in this area, for example [22, 35, 52, 56, 90, 135, 173, 218]. Furthermore, from a neural control perspective, these techniques are also very relevant because many neural network based control schemes are structured around a feedback linearisation framework [30, 31, 186, 234, 235]. Therefore, geometric approaches to the control of nonlinear systems will also be reviewed and discussed in detail below. Furthermore, many of the definitions provided in the review of these two nonlinear control disciplines will be used throughout the thesis.

Adaptive Control Theory

A common practical problem for researchers in the area of control systems is how to develop improved control approaches for systems which encompass constant or slowly varying uncertain parameters or nonlinearities. Typically such systems are difficult to model. This has resulted in the development of an inherently nonlinear design approach known as adaptive control. The basic concept of adaptive control theory is to estimate the uncertain plant parameters and often the corresponding controller parameters on-line based on measured values of the system output. These estimates are then used to estimate the parameters of the control scheme to generate the appropriate control signal. A block diagram of a generic adaptive control scheme is shown in Figure 1.1.

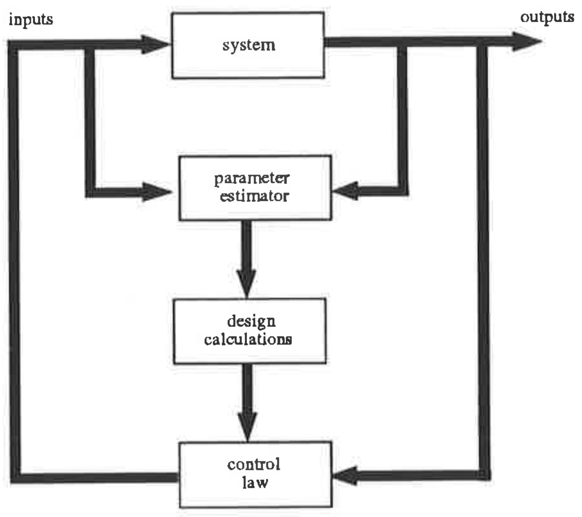


Figure 1.1: Block diagram of a generic adaptive control system

The field has progressed significantly over the past few years, both theoretically [4, 5, 66, 112, 163, 207] and from an application viewpoint [22, 35, 37, 56, 69, 176]. The growth of the adaptive control discipline is highlighted by many books recently published in this area [9, 90, 118, 167, 173, 218] and the growing number of adaptive control streams in recent conferences such as the IFAC World Congress, American Control Conference and IEEE Conference on Decision and Control.

Traditionally, adaptive control systems have been applied to linear systems in which there is parametric uncertainty [4, 7, 39, 60, 66, 180]. However, more recently adaptive approaches have been developed for nonlinear systems [12, 92, 106, 137, 206, 237]. Adaptive control approaches based on geometric concepts for nonlinear systems will be discussed in the next section. Adaptive control methodologies for nonlinear systems based on neural networks will be discussed in the remaining chapters of this thesis.

In the case of adaptive control methods applied to linear systems, the two most commonly used approaches are model reference adaptive control (MRAC) and self-tuning regulators (STR). These two approaches will be discussed in some detail as they form the basis of some of the neural controllers considered in the following chapters.

(i) Model Reference Adaptive Control

The model reference adaptive control method was originally developed by a group of researchers at MIT in 1958. A block diagram of such a system is shown in Figure 1.2. The aim of the MRAC approach is to design a controller to generate the control variables such that the output of the plant tracks the reference model output for a given bounded reference input. This is achieved by adjusting the parameters of the controller via the adjustment mechanism so as to minimise the error between the reference model and the system. Briefly, the model reference model can be posed as follows:

Consider a plant P with the input-output pair $\{u(k), y_p(k)\}$, where $u(k)$ is the control vector and $y_p(k)$ is the $\overset{\text{plant output}}{\underline{\text{state}}}$ vector. Consider a stable reference model M given by the input-output pair $\{r(k), y_m(k)\}$, where $r(k)$ is the bounded reference input vector and the reference model output, $y_m(k)$ is the desired output for the plant. The aim is to design a controller to produce a control input $u(k)$, such that the asymptotic tracking error, $e_T(k) = y_m(k) - y_p(k)$, is finite, i.e.,

$$\lim_{k \to \infty} | y_m(k) - y_p(k) | \leq \varepsilon_T \tag{1.8}$$

where $\varepsilon_T \geq 0$ is a prespecified tolerance.

The reference model is chosen to reflect the ideal response of the system being controlled. The choice of the reference model is part of the adaptive control system design. Generally, it is chosen to reflect the desired closed-loop dynamics of the system, usually in terms of performance specifications such as rise time, settling time, overshoot or frequency domain characteristics. However, the structure of the reference model is also constrained

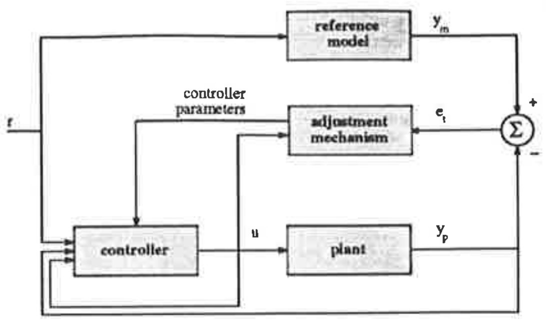


Figure 1.2: Block diagram of a model reference adaptive control system

by concepts such as its order and relative degree which are dependent on the assumed structure of the plant. These constraints are placed to ensure that the desired response is achievable by the plant.

There are two philosophically different approaches to model reference adaptive control: direct and indirect control. In the direct control approach, the control parameters are directly adjusted to meet the control requirements without the need to estimate the plant parameters. In contrast, in the indirect control approach, the plant parameters are estimated on-line and the control parameters are adjusted based on these estimates.

As mentioned earlier, the adaptation mechanism is used to adjust the parameters of the controller to ensure that the tracking error converges to zero. Therefore, the main emphasis of adaptive control design such as MRAC is to synthesise a parameter adjustment scheme which guarantees the stability of the control system and convergence of the tracking error. Much research has been conducted into developing appropriate adaptation schemes for MRAC approaches [66, 88, 118, 156, 180]. In the original development of MRAC, a parameter adjustment scheme known as the MIT rule was developed. The MIT rule can be considered to be a gradient descent scheme where the controller parameter vector θ_c is adjusted as follows:

$$\frac{d\theta}{dt} = k\ e\ \nabla_{\theta_c} e \tag{1.9}$$

where e denotes the model error, k is a parameter which determines the adaptation rate and the components of the vector $\nabla_{\theta_c} e$ are the sensitivity derivatives of the error with respect to the controller parameters. The main problem with the MIT rule approach is that it is not possible in general to prove closed-loop stability or convergence of the tracking error to zero. However, it has been shown empirically and later proven analytically by Mareels *et al.* [134] that if the parameter k is small and the magnitude of the reference input is small, then the MIT rule performs well.

The landmark paper by Parks [180] demonstrated that the system could be made more stable by using a design procedure based on Lyapunov stability methods. This paper eventually resulted in a shift away from gradient based schemes to adaptive control design schemes based on stability methods [156, 165, 180]. Since then there have been several papers on the design of MRAC schemes for linear systems [59, 116, 117, 150, 157, 164] and, more recently, nonlinear systems [12, 162, 166, 214, 217]. Several survey papers [4, 5, 66, 163] and books [9, 63, 118, 167, 168, 205] deal with the issues associated with model reference adaptive control in more detail.

(ii) Self-Tuning Regulators

The self tuning approach to adaptive control was originally proposed by Kalman [100] and later expanded upon by Åström and Wittenmark [7] and Clarke and Gawthrop [38]. As opposed to model reference adaptive control, which evolved from deterministic servomechanism problems, self-tuning regulators (STR) arose in the context of stochastic regulation problems. A STR system is shown in the block diagram given in Figure 1.3.

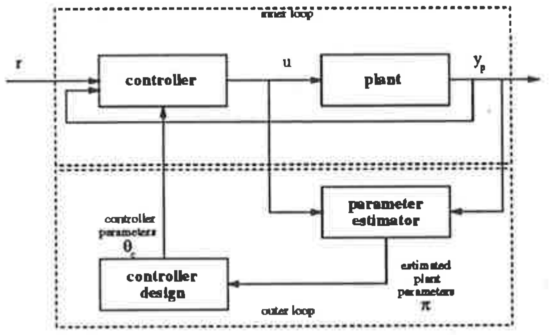


Figure 1.3: Block diagram of the self-tuning regulator scheme

The self-tuning regulator can be thought of as having two loops: an inner loop consisting of the plant and a conventional controller with varying parameters and an outer loop which is composed of a recursive parameter estimator and a controller design block which adjusts the controller parameters. The underlying design problem (represented by the controller design block) is to determine a relationship between the plant parameters and the controller parameters on-line. However, as the plant parameters are unknown, they are estimated by a recursive parameter estimation algorithm represented by the parameter estimator block. The controller parameters are then obtained from the estimates of the plant parameters as if they are the true parameters. Such an approach is known as the certainty equivalence principle. As the parameters of the plant are estimated prior to obtaining the controller parameters, the STR scheme can be classified as an indirect control approach. Such a scheme is also often referred to as an explicit STR. Implicit self-tuning regulators are based on an implicit estimation of the system and a direct update of the controller parameters. Therefore, implicit STR's are closely related to a direct model reference adaptive control scheme. The relationship between the various schemes has been investigated thoroughly in [50, 118].

The popularity of the self-tuning regulator approach extends from its flexibility with regards to the choice of controller design methodology and recursive parameter estimation scheme. As a result there have been numerous papers which deal with extensions of the original STR approach of Åström and Wittenmark [8, 23, 39, 182, 188]. In particular, controller designs based on techniques such as pole-placement, minimum variance and LQG and recursive identification schemes based on the well known least squares, extended Kalman filters and maximum likelihood, have been used.

Research on self-tuning regulators in recent year has continued on many fronts. The use of STR's in practical applications such as the ship steering problem [98, 127], industrial and biotechnical process control [24, 45, 49, 107], power systems [123, 236] and automotive control [233], to name but a few, has been investigated in recent years. As with most adaptive control approaches, stability of the control scheme is of major concern. Therefore, significant research has been directed towards addressing stability issues for STR's. It has been shown that for the ideal case the self-tuning regulator algorithm is globally convergent [60]. More recently, Ydstie [252] has investigated the stability of a direct STR in which unmodelled dynamics and disturbances are present. Another area in which considerable effort has been directed is the use of self-tuning regulators for the adaptive control of nonlinear systems [2, 28, 37, 256]. Of particular interest is the work by Chen [28, 29, 30, 31] in which neural networks are used to identify the nonlinear system and are then used within an STR approach to regulate the system. This will be discussed in more detail in the next chapter.

Geometric Approaches to the Control of Nonlinear Systems

(i) Feedback Linearisation

The theoretical developments in the design of feedback control schemes for nonlinear systems can be classified as either asymptotic or geometric approaches. Geometric approaches [90, 173, 218] are restrictive but exact, whereas the asymptotic approaches [108, 113] are less restrictive but approximate. However, as shown by Kokotovic [110], the similarity of the approaches are such that the asymptotic approaches can be presented in a geometric framework. In this review, the emphasis will be on the geometric approaches to the control of nonlinear systems [52, 90, 173, 218].

Research into the design of adaptive control schemes for nonlinear systems using geometric concepts has been extremely active since the early 1980's. The results obtained so far differ in the conditions they impose on the growth of the nonlinearities and/or the dependence of the system on the unknown parameters [105] and the assumptions they make about the system. The two most common assumptions are linear parameterisation and full-state feedback. The linear parameterisation assumption, which has been adopted by virtually all of the researchers in the field, requires that the unknown parameters appear linearly with respect to the known nonlinearities. In the continuous-time case the system considered is typically of the the form

$$\begin{aligned} \dot{x} &= f_0(x) + [g_0(x) + g(\theta, x)]u + f(\theta, x) \\ y &= h(x) \end{aligned} \tag{1.10}$$

where

$$\begin{aligned} f(\theta, x) &= \sum_{i=1}^{p} \theta_i f_i(x) \\ g(\theta, x) &= \sum_{i=1}^{p} \theta_i g_i(x) \end{aligned} \tag{1.11}$$

where $x \in \mathbf{R}^n$ is the state, $u \in \mathbf{R}$ is the control, $\theta = [\theta_1, \ldots, \theta_p]$ is the vector of unknown constant parameters belonging to Ω, a closed subset of $\mathbf{R}^p$, $f : \mathbf{R}^n \times \mathbf{R}^p \rightarrow \mathbf{R}^n$ and

$g : \mathbf{R}^n \times \mathbf{R}^p \rightarrow \mathbf{R}^n$ are smooth functions, $h : \mathbf{R}^n \rightarrow \mathbf{R}$ is the output function, f_0, g_0, f_i, g_i are smooth vector fields on $\mathbf{R}^n$ with $g_0(x) \neq 0 \; \forall \; x \in \mathbf{R}^n$ and $f_i(0) = 0$ for $0 \leq i \leq p$.

Note that such systems are affine in control, u. As the unknown parameters θ are linearly parameterized, they can be estimated using a standard recursive estimation algorithm such as the recursive least squares algorithm [130, 221]. The linearising feedback controller is then constructed in terms of the estimated parameters.

The second major assumption made in most of the geometric approaches developed so far is that the full state variable is available to design the appropriate control. This is known as the full-state feedback assumption. More recently approaches based on output feedback only have been developed [104, 138, 139]. These will be discussed in more detail later on.

The aim of state feedback approaches is to find a set of transformations of the form

$$\begin{aligned} u &= u(x,v) = \alpha(x) + \beta(x)v \\ z &= \Phi(x) \end{aligned} \tag{1.12}$$

where $x \in \mathbf{R}^n$ is the new state vector and $v \in \mathbf{R}^p$ is the new input, such that the nonlinear system (1.10) is transformed into a linear system of the form

$$\dot{z} = Az + Bv \tag{1.13}$$

in some region Ω.

Therefore, the problem is one of transforming the state space model of the nonlinear system (1.10) by a suitable coordinate transformation and state feedback such that the resultant system is (fully or partially) linear. This approach is known as feedback linearisation [90, 173, 218]. As it is difficult to achieve this transformation over the entire state space $\mathbf{R}^n$, such linearisations are usually only locally valid over some neighbourhood of the origin. If such transformations exist, the plant is called feedback linearisable and linear control techniques can be used to control and stabilise the system.

In order for the above transformation to be valid, the function $\Phi(.)$ must be a diffeomorphism, defined as follows:

DEFINITION 1.1.4 *A function $\Phi : \mathbf{R}^n \to \mathbf{R}^n$ defined in the region Ω which represents a neighbourhood of the region of interest (usually the origin) is called a* <u>*diffeomorphism*</u> *if it is smooth and its inverse exists and is smooth.*

This condition ensures that one can pass between the state vectors x and z without ambiguity. The existence of such transformations are dependent upon a number of necessary and sufficient conditions. However, prior to stating these conditions it is necessary to consider the following definitions.

DEFINITION 1.1.5 *Let $h : \mathbf{R}^n \to \mathbf{R}$ be a smooth scalar function and $f : \mathbf{R}^n \to \mathbf{R}^n$ be a smooth vector field on $\mathbf{R}^n$, then the* <u>*Lie derivative*</u> *of h with respect to f is a scalar function defined by $L_f h = \nabla h f$, indicating that the Lie derivative $L_f h$ is simply the directional derivative of h along the direction of the vector f. Furthermore, $L_{f^0} h = h, \ldots, L_{f^i} h = L_f(L_{f^{i-1}} h) = \nabla(L_{f^{i-1}} h) f$ for $i \in \mathbf{N}$.*

DEFINITION 1.1.6 *Let f and g be two vector fields on $\mathbf{R}^n$. The* <u>*Lie bracket*</u> *of f and g is a third vector field defined by*

$$[f, g] = \nabla g \, f - \nabla f \, g = ad_f g \tag{1.14}$$

where $ad_{f^0} g = g, \ldots, ad_{f^i} g = [f, ad_f^{i-1} g]$ for $i \in \mathbf{N}$.

DEFINITION 1.1.7 *A linearly independent set of vector fields $\{f_1, f_2, \ldots, f_m\}$ is said to be* <u>*involutive*</u>*, if and only if, there are scalar functions $\alpha_{ijk} : \mathbf{R}^n \to \mathbf{R}$ such that*

$$[f_i, f_j](x) = \sum_{k=1}^{m} \alpha_{ijk}(x) f_k(x) \qquad \forall i, j, \text{and } m < n \tag{1.15}$$

i.e., if one forms the Lie bracket of any pair of vector fields from the set $\{f_1, \ldots, f_m\}$ then the resulting vector field can be expressed as a linear combination of the original set of vector fields.

Therefore the system (1.10) can be linearised by state feedback and coordinate transform if and only if the following conditions hold [218, 227]:

1. the controllability matrix $[g, ad_{f^1}g, \ldots, ad_{f^{n-1}}g]$ has rank n in Ω, i.e., the set of vector fields $\{g, ad_{f^1}g, \ldots, ad_{f^{n-1}}g\}$ is linearly independent in Ω.

2. the set of vector fields $\{g, ad_{f^1}g, \ldots, ad_{f^{n-2}}g\}$ is involutive in Ω.

Hunt *et al.* [81] have combined the above local feedback linearisation conditions with global inverse function theorems and partial differential equation techniques to prove global theorems for transforming a nonlinear system of the form (1.10) to a linear system of the form (1.13). Dayawansa *et al.* [42] also derive global feedback linearisation conditions.

The feedback linearisation of discrete-time systems is addressed in the papers by Monaco *et al.* [154], Jakubczyk [92], and Grizzle *et al.* [61]. In [92] local necessary and sufficient feedback linearisation conditions for a nonlinear discrete-time system are derived. It is shown that the conditions are analogous to the continuous-time conditions given in [227] except that there are no involutiveness requirements. The papers [61, 154] deal with the feedback linearisation of sample-$\overset{\text{data}}{\text{date}}$ systems. Grizzle *et al.* show by way of an example that although a continuous-time system may be feedback linearisable, the sample-data version may not be. A technique to overcome this using multi-rate sampling is discussed. A good summary of feedback linearisation techniques for discrete-time systems can be found in [173].

Since the work of Su [227], Hunt *et al.* [81] and Dayawansa *et al.* [42], a number of papers have been published which address issues relating to the practical implementation of feedback linearisation techniques. One of the major drawbacks with the above theory is that it relies on the exact cancellation of the nonlinear terms in order to get state-space linearisation. Consequently, if there are errors or uncertainties in the model of the nonlinear terms or unknown parameters, then the cancellation is no longer exact. Hence the practical implementation of such schemes requires that further restrictions be imposed either on the location of the unknown parameters or on the type of nonlinearities. According to these additional restrictions, the adaptive scheme can be categorised as either uncertainty constrained schemes or nonlinearity constrained schemes [112].

As the name suggests uncertainty constrained schemes impose restrictive conditions, known as matching conditions, on the location of the unknown parameters, but they can handle all types of nonlinearities. Many papers which deal with uncertainty constrained schemes attempt to relax the restrictive feedback linearisation conditions which exist. One method is suggested by Kokotović in [110]. In this approach, the conditions are only required to be satisfied for a reduced model. Such models allow dynamic, but exclude parametric uncertainties. In the paper by Taylor *et al.* [231], an adaptive update law is proposed to counteract parametric uncertainties in the case where the unknown plant parameters appear linearly. The adaptive control is designed for a reduced order model. Assumptions about parametric uncertainties, linear parameterisation and feedback linearisation are made for this reduced order model. Only the states appearing in the reduced order model are available for measurement. This is in contrast to the work of Nam and Arapostathis [162] and Sastry and Isidori [207] which also address parametric uncertainty, but assume that full state information is present. Furthermore, Taylor *et al.* derive robustness properties with respect to the unmodelled dynamics, something which had not been established for other nonlinear adaptive schemes [206].

The results of [231] are used by Marino *et al.* [136] to address the adaptive tracking problem for feedback linearisable systems with parametric uncertainty. Strict matching conditions are assumed so that the results of [231] can be utilised.

DEFINITION 1.1.8 *Strict matching conditions:*

$$f_i, g_i \in span\{g_0\} \tag{1.16}$$

Further restrictions are placed on the system by assuming the extended matching conditions [102] defined below are also met.

DEFINITION 1.1.9 *Extended matching conditions (EMC):*

$$\begin{aligned} f_i &\in span\{g_0, ad_{f_0}g_0\} \\ g_i &\in span\{g_0\} \end{aligned} \tag{1.17}$$

For full-state feedback linearisable systems, the extended matching condition is a necessary and sufficient condition for the existence of a parameter independent diffeomorphism $z = \Phi(x)$.

The work by Kanellakopoulos *et al.* [103] can be seen to be an extension of the results of [231]. The authors present a direct adaptive control scheme which satisfies the extended matching conditions of their earlier work [102] and is shown to be robust with respect to unmodelled dynamics.

The main advantage of EMC based schemes is that their stability properties can be established independently of the type of nonlinearities [112]. Although the extended matching condition is quite restrictive, it is satisfied by many systems of practical importance, such as most types of electric motors [103]. Furthermore, as shown in [103], the robustness of EMC based schemes to unmodelled dynamics can be exploited to extend their applicability.

Nonlinearity constrained schemes do not restrict the location of unknown parameters, but instead impose restrictions on the growth of the nonlinearities of the original system [112]. Papers which can be classified under this approach are [162, 187, 207, 232].

In the paper by Nam and Arapostathis [162], the issue of parametric uncertainties in the system are addressed. The authors restrict their attention to a class of "well structured" nonlinear systems called pure-feedback system. These systems were first defined by Su and Hunt [228] and they have the desirable feature that the linearising map is straightforward to construct. They present matching conditions which allow them to transform the system into the pure-feedback form. This allows the implementation of an adaptive algorithm which updates the estimates of the feedback and coordinate transformation required to linearise the system, as well as to meet their objective of constructing a model reference controller.

More recent work by Kanellakopoulos *et al.* [106] has resulted in an adaptive control scheme in which the growth of the nonlinearities is not constrained. Instead they require that the nonlinear system be transformed into a parametric pure-feedback form (different to the pure-feedback form of [162, 228]) in which the new system depends only on the feedback state variables. These pure-feedback systems are shown to remove the limitations of both uncertainty constrained schemes and nonlinearity constrained schemes. Therefore, it is argued that they represent the broadest class of nonlinear systems for which adaptive controllers can be designed without imposing constraints on the growth of the system nonlinearities.

In recent years full-state feedback linearisation techniques have been applied to practical problems such as the control of stepper motors [22], induction motors [35], shunt DC motors [36], synchronous generators [56], and robotics applications [135, 41].

(ii) Input-Output Linearisation

The aim of full-state feedback linearisation is to find a state transformation $z = \Phi(x)$ and an input transformation $u = u(x, v) = \alpha(x) + \beta(x)v$ such that the nonlinear system dynamics are transformed into an equivalent linear system $\dot{z} = Az + Bv$. Standard linear control techniques are then applied to the transformed system to design v. In contrast, in input-output linearisation, the aim is to solve a tracking control problem for plants of the form

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{1.18}$$

where $x \in \mathbf{R}^n$, f, g and h are smooth nonlinear functions.

The primary objective is to make the output $y(t)$ track a desired trajectory $y_d(t)$ whilst ensuring that the whole state is bounded. The difficulty with this problem is that the output y is indirectly related to the input u through the state variable x and the nonlinear state equation (1.18). Hence the intuitive basis of input-output linearisation is to find a direct and simple relationship between the system output y and the control input u.

Consider the above system again. Differentiating y with respect to time yields

$$\dot{y} = L_f h(x) + L_g h(x) u \tag{1.19}$$

where $L_f h(x) : \mathbf{R}^n \to \mathbf{R}$ and $L_g h(x) : \mathbf{R}^n \to \mathbf{R}$ are the Lie derivatives of h with respect to f and g, respectively (see Definition (1.1.5)). If $L_g h(x)$ is bounded away from zero for all x (i.e., $L_g h(x) \neq 0 \ \forall x \in \mathbf{R}^n$), the state feedback law

$$u = \frac{1}{L_g h(x)}(-L_f h(x) + v) \tag{1.20}$$

results in the linear system

$$\dot{y} = v \tag{1.21}$$

The above control results in the previous $n-1$ states of the system being unobservable.

If $L_g h(x) = 0 \; \forall x \in \mathbf{R}^n$, one differentiates $\dot{y}$ to obtain

$$\ddot{y} = L_f^2 h(x) + L_g L_f h(x) u \tag{1.22}$$

If $L_g L_f h(x) = 0$ then one differentiates again and again, until for some integer γ, $L_g L_f^{\gamma-1}$ is bounded away from zero, and the control law

$$u = \frac{1}{L_g L_f^{\gamma-1} h(x)} (-L_f^{\gamma} h(x) + v) \tag{1.23}$$

applied to

$$y^{\gamma} = L_f^{\gamma} h(x) + L_g L_f^{\gamma-1} h(x) u \tag{1.24}$$

yields the simple linear relation

$$y^{\gamma} = v \tag{1.25}$$

Such a system has strong relative degree γ and the control (1.23) renders the $n-\gamma$ states of (1.18) unobservable. Note that if $\gamma = n$ then input-output linearisation results in full-state linearisation previously described. This results in the following definition.

DEFINITION 1.1.10 *The SISO system (1.18) is said to have relative degree γ in $\mathbf{R}^n$ if $\forall x \in \mathbf{R}^n$*

$$\begin{aligned} L_g L_f^i h(x) &= 0, \qquad 0 \le i < \gamma - 1 \\ L_g L_f^{\gamma-1} h(x) &\neq 0 \end{aligned} \tag{1.26}$$

It can be shown [90, 206, 207, 218] that there exists a local diffeomorphism $z = \Phi(x)$ such that the system (1.18) is transformed into the following form

$$\begin{aligned} \dot{z}_{11} &= z_{12} \\ \dot{z}_{12} &= z_{13} \\ &\vdots \\ \dot{z}_{1\gamma} &= f_1(z_1, z_2) + g_1(z_1, z_2) u \end{aligned}$$

$$\begin{aligned}\dot{z}_{1\gamma+1} &= q_{\gamma+1}(z_1, z_2) \\ &\vdots \\ \dot{z}_{1n} &= q_n(z_1, z_2)\end{aligned} \tag{1.27}$$

where $z_{11} = h(x)$, $z_{12} = L_f h(x)$,..., $z_{1\gamma} = L_f^{\gamma-1}h(x)$

$f_1(z_1, z_2) = L_f^{\gamma}h(x)$

$g_1(z_1, z_2) = L_g L_f^{\gamma-1}h(x)$

$z_1 = (z_{11}, \dots, z_{1\gamma}) = (y, \dot{y}, \dots, y^{(\gamma-1)})$

$z_2 = (z_{1\gamma+1}, \dots, z_{1n})$

$\dot{z}_2 = q(z_1, z_2)$

$z = (z_1, z_2)^T$

and the output is defined as

$$y = z_{11}. \tag{1.28}$$

The linearising control is

$$u = \frac{1}{g_1(z_1, z_2)}(-f_1(z_1, z_2) + v) \tag{1.29}$$

The equations (1.27) are referred to as the normal form of the nonlinear equation. The state variables $z_{1\gamma+1}, \dots, z_{1n}$ represent the internal dynamics of the system, as they are unobservable from the input-output dynamics $y^{(\gamma)} = f_1(z) + g_1(z)u$.

Due to the linear relationship of the transformed system (1.25), it is easy to design the input v so that the output y behaves as desired. However, this does not ensure that the internal dynamics will also behave well, i.e., remain bounded. Therefore, the stability of the internal dynamics must also be addressed as well. This leads to the concept of zero dynamics [90, 218].

The term zero dynamics refers to the case where the motion of the system is restricted to the $n - \gamma$ dimensional smooth surface M_0 defined by $z_1 = 0$. If the initial state of the system $x(0)$ is on this surface and the input u is of the form

$$u_0 = \frac{-L_f^{\gamma}h(x)}{L_g L_f^{\gamma-1}h(x)} \tag{1.30}$$

resulting in $y^{\gamma}(t) = 0$ then the system is operating in zero dynamics. This leads to the following definition

DEFINITION 1.1.11 *The zero dynamics of the nonlinear system (1.18) are*

$$\dot{z}_2 = q(0, z_2) \tag{1.31}$$

Furthermore, the nonlinear system (1.18) is said to be minimum phase if the zero dynamics are asymptotically stable.

REMARK 1.1.1 *Discrete-time versions of the concepts of minimum phase and zero dynamics have been proposed by Monaco and Normand-Cyrot in [153].*

As mentioned earlier, the aim of input-output linearisation schemes is to solve the tracking problem. Consider the desired trajectory $z_m = [y_m, \dot{y}_m, \ldots, y_m^{(\gamma-1)}]^T$ and define the tracking error as

$$e(t) = z_1(t) - z_m(t) \tag{1.32}$$

then the following theorem applies [90]:

THEOREM 1.1.2 *Assume that the system (1.18) has relative degree γ, its zero dynamics are asymptotically stable, z_m is smooth and bounded, and that the solution Ψ_m of the equation*

$$\dot{\Psi}_m = q(z_{1m}, z_{2m}) \qquad\qquad z_m(0) = 0 \tag{1.33}$$

exists and is bounded and asymptotically stable. Choose constants k_i such that the polynomial

$$K(p) = p^\gamma + k_{\gamma-1}p^{\gamma-1} + \ldots + k_1 p + k_0 \tag{1.34}$$

has all of its roots strictly in the left half plane. Then by using the control law

$$u = \frac{1}{L_g L_f^{\gamma-1} h(x)}[-L_f^\gamma h(x) + v] \tag{1.35}$$

where

$$v = y_m^{(\gamma)} - k_{\gamma-1}e_\gamma - \ldots - k_0 e_1 \tag{1.36}$$

the whole state remains bounded and the tracking error converges to zero.

It should be noted that this scheme requires full state information, x and that the output variables $y, \ldots, y^{(\gamma-1)}$ are measurable. The proof of the above theorem is given in [90].

The input-output linearisation of MIMO systems has been addressed by Isidori [90], Sastry and Bodson [205] and Slotine and Li [218].

Issues dealing with the practical implementation of adaptive input-output linearisation schemes have been addressed in [36, 69, 155, 206, 207, 216]. From these papers it is apparent that the major drawback with the practical implementation of input-output linearisation schemes is that if there is uncertainty in the nonlinear function, then the cancellation of the nonlinear terms is not exact and the resulting input-output equation is not linear. Sastry *et al.* [206, 207] suggest the use of adaptive control to obtain asymptotically exact cancellation.

In this approach, a SISO system of the form (1.18) is considered with

$$\begin{aligned} f(x) &= \sum_{i=1}^{n1} \theta_i^1 f_i(x) \\ g(x) &= \sum_{j=1}^{n2} \theta_j^2 g_j(x) \end{aligned} \tag{1.37}$$

where θ_i^1 $i = 1, \ldots, n1$ and θ_j^2 $j = 1, \ldots, n2$ are unknown parameters and $f_i(x)$ and $g_j(x)$ are known functions.

The estimates of the functions f and g at time t are

$$\begin{aligned} \hat{f}(x) &= \sum_{i=1}^{n1} \hat{\theta}_i^1(t) f_i(x) \\ \hat{g}(x) &= \sum_{j=1}^{n2} \hat{\theta}_j^2(t) g_j(x) \end{aligned} \tag{1.38}$$

where $\hat{\theta}_i^1(t)$, $\hat{\theta}_j^2(t)$ are estimates of θ_i^1 and θ_j^2 at time t.

The control law u becomes

$$u = \frac{1}{\widehat{L_g h}}(-\widehat{L_f h} + v) \tag{1.39}$$

and $\widehat{L_g h}$ and $\widehat{L_f h}$ are estimates of the Lie derivative based on (1.38), i.e.,

$$\begin{aligned} \widehat{L_f h} &= \sum_{i=1}^{n1} \hat{\theta}_i^1 L_{f_i} h \\ \widehat{L_g h} &= \sum_{j=1}^{n2} \hat{\theta}_j^2 L_{g_j} h \end{aligned} \tag{1.40}$$

It can be shown [205] that if the control law for tracking is given by

$$v = \dot{y}_m + \alpha(y_m - y) \tag{1.41}$$

and y_m is bounded and $\widehat{L_g h}$ is bounded away from zero, then the parameter update law

$$\dot{\phi} = -(y - y_m)W \tag{1.42}$$

yields bounded $y(t)$ which asymptotically converges to $y_m(t)$, where $\phi = \theta - \hat{\theta}$ is the parameter error with $\theta = [\theta^{1T}, \theta^{2T}]$, and $W \in \mathbf{R}^{n1+n2}$ is the concatenation of

$$\begin{aligned} W_1 &= \begin{bmatrix} L_{f_1} h \\ \vdots \\ L_{f_{n1}} h \end{bmatrix} \\ &\text{and} \\ W_2 &= \begin{bmatrix} L_{g_1} h \\ \vdots \\ L_{g_{n2}} h \end{bmatrix} \left(\frac{-\widehat{L_f h} + v}{\widehat{L_g h}} \right) \end{aligned} \tag{1.43}$$

The above input-output linearisation scheme of Sastry and Isidori [207] belongs to the class of nonlinearity constrained schemes. The main constraint employed in this scheme is that the nonlinear functions f, g and h are globally Lipschitz in x. The input-output linearisation approach of Teel *et al.* [232] similarly considers global Lipschitz conditions. The primary difference between these approaches is that Sastry and Isidori employ parameter update laws analogous to those used in indirect adaptive linear control, whilst Teel *et al.* combine parameter estimation elements from both direct and indirect control approaches.

DEFINITION 1.1.12 *A function $f(x,t)$ is said to be globally Lipschitz if for any x_1 and x_2 in the state space the Lipschitz condition*

$$||f(x_2,t) - f(x_1,t)|| \leq L||x_2 - x_2|| \tag{1.44}$$

is satisfied where L is a strictly positive constant known as the Lipschitz constant and $t \in [t_0, t_0 + T]$ where T is a strictly positive constant.

Recently a number of researchers have implemented input-output linearisation techniques similar to the above scheme using neural networks [28, 29, 30, 31, 234, 235]. These techniques will be discussed in the next chapter.

(iii) Output Feedback Controllers

Two of the main assumptions made in the aforementioned schemes are that full state information is available and the unknown parameters appear linearly in the system equation. Recently a great deal of work has been undertaken to relax these assumptions. In particular, considerable research has been addressed at designing adaptive control schemes for nonlinear systems assuming that only the output is available [105, 104, 101, 137, 138, 139].

The earlier work of Kanellakopoulos *et al.* [104, 105] involves introducing a set of input and output matching conditions which results in the existence of a diffeomorphism to transform a nonlinear system of the form (1.10) into the following input-output description

$$A(D)y = B(D)\{p_0(y) + p^T(y)\theta + [q_0(y) + q^T(y)\theta]u\} \tag{1.45}$$

where D denotes the differentiation operator, the coefficients $\alpha_0, \ldots, \alpha_{n-1}$ of the denominator polynomial $A(D) = D^n + \alpha_{n-1}D^{n-1} + \ldots + \alpha_0$ are unknown, θ is the l-dimensional vector of unknown parameters, $B(D) = b_m D^m + \ldots + b_0$ is known and Hurwitz, $p(y) = [p_1(y), \ldots, p_l(y)]$, $q(y) = [q_1(y), \ldots, q_l(y)]$ and $p_i(y)$, $q_i(y)$ $i = 1, \ldots, l$ are smooth nonlinearities.

Furthermore, the nonlinearities are restricted to the following form (sector-type nonlinearities)

$$\begin{aligned} |\, p_0(y) \,| &\leq \kappa + \kappa \,|\, y \,| \\ \|p(y)\| &\leq \kappa + \kappa \,|\, y \,| \\ \|q(y)\| &\leq \kappa \end{aligned} \tag{1.46}$$

where $|\,.\,|$ is the scalar norm, $\|.\|$ is the vector norm, and κ denotes a positive constant and some *a priori* information about the unknown parameters is assumed.

An indirect adaptive control scheme is then presented in which the unknown parameters are estimated using a prediction-error type estimator. These estimates are then used to design a certainty equivalence control law.

More recent work by Kanellakopoulos *et al.* [101] is aimed at removing conditions (1.46), thus removing the need for the nonlinearities to satisfy any growth conditions, whilst ensuring that all stability and tracking results are global.

Extensive work in output-feedback control has also been conducted by Marino and Tomei [137, 138, 139]. In the paper [137], Marino and Tomei have presented conditions to globally transform a nonlinear system of the form (1.10) into a linear minimum phase and observable system, making use of filtered output feedback transformations. Such transformations are shown to consist of a state space change of coordinates and a output feedback control both driven by asymptotically stable linear filters whose inputs are nonlinear functions of the outputs. They have shown that the system is globally output feedback stabilisable. In the paper [138], Marino and Tomei present sufficient conditions for the existence of adaptive output feedback control. The control strategy presented is not subject to the matching condition restrictions or sector-type nonlinearities as in [104, 105]. In their companion paper [139], Marinó and Tomei present a global output feedback control strategy for a nonlinear system in which the unknown quantities may be nonlinearly parameterized, i.e., the assumption of linear parameterisation which is one of the key assumptions of all of the above methods is removed.

1.1.3 Neural Networks in Control

The nonlinear control approaches described above generally apply to a limited class of nonlinear systems. For example, the feedback linearisation approaches primarily deal with systems where the uncertainty is due to unknown parameters which appear linearly with respect to the known nonlinearities. Furthermore, such systems are generally linear in the control (affine). Therefore, in contrast to the adaptive control of linear systems, adaptive design techniques for nonlinear systems have yet to be established for a general class of nonlinear structure.

The emergence of artificial neural networks as a method of forming an arbitrarily close approximation to any continuous nonlinear function [40, 74, 55] has offered an alternative approach to solve a more general class of nonlinear problems. As a result of their approximation abilities as well as their inherent adaptivity, artificial neural networks have generated a great deal of interest in the control community. Furthermore, the parallel nature of neural networks and their fast adaptability has provided additional incentive for the investigation of their use in the identification and control of complex nonlinear systems. Consequently, several neural network based control architectures have been proposed in recent years [189, 171, 169, 28, 204]. A neural network is trained to learn the inverse plant dynamics in the scheme suggested by Psaltis *et al.* [189]. This neural network is then used as an open-loop feedforward controller. Nguyen and Widrow [171] have proposed a scheme in which a multilayered neural network, known as the emulator, is trained off-line to identify the system dynamics. A controller neural network is then trained to control the emulator. The trained controller is then applied to the actual system. The papers by Narendra and Parthasarathy [169] and Chen [28] are amongst the first to utilise neural networks in a more traditional control framework. In the approach presented in [169], neural networks are used in a model reference adaptive control environment. The scheme is shown to be effective for a wide range of nonlinear systems. On the other hand, Chen combines multilayered neural networks with self-tuning adaptive control techniques to control single-input single-output feedback linearisable systems. More recently, Sanner and Slotine [204] have presented a direct adaptive tracking control procedure in which the weights of the Gaussian radial basis function network are updated by a stable adjustment scheme based on Lyapunov theory.

As the above issues are central to the research presented in this thesis they are discussed in greater detail in Chapter 2.

1.2 Outline of the Thesis

This thesis is primarily concerned with the development of a neural network based model reference adaptive control scheme for discrete-time non-affine nonlinear systems. The aims of this study are therefore:

- to *explore* the issues associated with using neural networks in the modelling and control of nonlinear dynamic systems;

- to *establish* a framework for the use of neural networks in the model reference adaptive control of nonlinear systems;
- to *develop* the theory to ensure the convergence and stability properties of the closed-loop system, and
- to *investigate* the effectiveness of the proposed stable neural adaptive control scheme through simulation studies of arbitrary and "real world" systems.

In Chapter 2, an introduction to artificial neural networks is provided. A brief discussion of various neural network architectures commonly used in the neural control area is provided and some of the implementation issues such as the network structure and learning algorithms are considered. The use of artificial neural networks in the modelling of dynamic systems is discussed. In particular, time delay neural networks (TDNN), which are used in the proposed neural control scheme, are analysed in detail. Correlation tests are used to determine the validity of a particular TDNN model for a given nonlinear dynamic system. An overview of the major existing neural control schemes is also provided in Chapter 2. The limitations and strengths of the approaches considered are also discussed.

A neural network based model reference adaptive control scheme for discrete-time non-affine nonlinear systems is presented in Chapter 3. The approach combines the forward modelling scheme of Jordan [95] with the model reference neural adaptive control approach of Narendra and Parthasarathy [166]. In this approach the nonlinear system is treated as a "black box" with the only *a priori* knowledge assumed being the relative degree and order of the system and the maximum lag of plant input and plant output terms. Therefore, unlike other approaches, there is no need to assume knowledge of the separability or otherwise of the control terms and the output variable terms, nor their respective nonlinear functions, and the inverse of the nonlinear functions do not need to be explicitly identified. Furthermore, the approach can also deal with nonlinear systems which are non-affine in control and where the control is heavily embedded within the nonlinearities of the system dynamics. This is in contrast to existing geometric control approaches for nonlinear system and many other neural adaptive control approaches. In the approach, time delay neural networks based on the multilayer perceptron are used to determine the plant Jacobian and synthesise the reference model control. The neural adaptive control approach is presented in a multi-input multi-output framework. The implementation of the scheme in both an off-line and on-line environment is discussed. Simulation examples are provided to highlight the effectiveness of the scheme in both an off-line and on-line environment for single-input single output systems and multi-input multi-output system. The ability of the proposed approach to deal with disturbances such as dynamic plant noise, sensor noise and load disturbance is also considered. The advantages of the on-line approach, particularly as a result of performing both identification and control simultaneously and in a closed-loop environment, are demonstrated via a simulation study of a marginally stable nonlinear system. Finally, the limitations of the scheme are discussed which gives rise to the issues addressed in the next chapter.

An enhanced neural network based model reference control scheme is proposed in Chapter 4. As with the scheme discussed in the previous chapter, the enhanced neural control scheme is formulated for a general discrete-time multi-input multi-output nonlinear system. Furthermore, the general nonlinear systems considered are non-affine in control and the control may be heavily embedded in the nonlinearities of the system. Weak assumptions regarding the order, relative degree and number of delay terms in the plant output and control variable are made. Output feedback is also assumed. The enhancements derived allow the issues of stability of the closed-loop system and convergence of the tracking error to be addressed. An enhanced reference model is introduced which allows the derivation of sufficient conditions to guarantee the convergence of the tracking error. Lyapunov theory is used to guarantee the stability of the closed-loop system. The basic strategy of the proposed scheme is to generate a control input via the neural network controller such that the plant output is nearest, in some norm sense, to a desired plant output generated by the enhanced reference model. A modified controller neural network weight update equation is proposed to achieve the desired control. Simulation studies demonstrate the effectiveness of the new approach for both single-input single-output systems and multi-input multi-output systems. The performance of the system in the presence of dynamic plant noise, sensor noise, load disturbance and plant uncertainty and/or variations are investigated. The ability of the proposed neural control scheme to handle non-ideal plant dynamics such as nonminimum phase systems and marginally stable systems is also examined.

The practical feasibility of the enhanced neural control scheme is investigated in Chapter 5. The proposed approach is applied to the anti-skid brake system problem. This problem represents a difficult challenge for any control scheme because of the highly nonlinear and time-varying dynamics of the vehicle–brake system. In this chapter, a model of the dynamics is first derived. The effects of environmental conditions such as road surface are incorporated into the dynamics via a highly nonlinear relationship. A study of existing approaches to this problem is also conducted to identify the practical difficulties associated with an anti-skid brake system. A neural network based anti-skid brake system is proposed which is demonstrated to provide effective braking performance even under harsh environmental conditions.

From the extensive study described in this thesis, general conclusions regarding the development of the new neural network based model reference adaptive control strategies for discrete-time non-affine nonlinear systems of unknown structure are drawn in Chapter 6. The principal features of the neural control approach developed in this thesis are highlighted. From these conclusions and the experience and knowledge gained throughout this research, a number of recommendations are made for the future development of neural adaptive control schemes.

1.3 Major Contributions of the Thesis

The principal contributions made in this thesis are as follows:

- A critical review of existing geometric approaches to the adaptive control of nonlinear systems.
- A critical review of the use of artificial neural networks for the modelling and control of nonlinear dynamic systems.
- The development of a neural network based model reference adaptive control scheme for general multi-input multi-output non-affine discrete-time nonlinear dynamic systems of unknown structure.

- The development of an enhanced model reference neural control scheme for which proofs of the convergence of the tracking error and stability of the closed-loop system are furnished.
- The derivation of a modified controller neural network weight update equation.
- The investigation of the performance of the neural control system in the presence of disturbances, dynamic variations and uncertainties and non-ideal dynamic behaviour such as marginal stability and nonminimum phase behaviour.
- The application of the proposed neural control system to a vehicle anti-skid brake system.

Chapter 2

Neural Networks for the Control of Dynamical Systems

2.1 Introduction and Overview

The birth of artificial neural networks is usually credited to M$^{\text{c}}$Cullogh and Pitts [146] who outlined the first model of an elementary computing neuron. Research into neural networks continued strongly during the 1950's and 60's. However, towards the end of the 1960's problems arose due to the relatively modest computational resources available to undertake research on neural networks, the lack of efficient learning schemes for the networks and doubts about the potential of layered learning networks. Neural network research entered a stagnation phase during the 1970's and beginning of the 1980's with only a handful of researchers continuing in the area.

However, since the early 1980's there has been a rapid growth in theoretical and applied research related to artificial neural networks. Much of this growth can be attributed to the advances made in the computer technology in the 1980's, which enabled neural network researchers to simulate and test their ideas in a thorough manner. The resurgence in interest in neural networks was also due in part to a number of seminal papers that significantly furthered the potential of the area [72, 73, 128, 179, 200]. In particular, the papers of Hopfield [72, 73], in which a recurrent neural network for associative memory was introduced, resulted in an explosion of activity into the computational properties of fully connected networks. Arguably a more important development occurred in the mid 1980's. The backpropagation algorithm was developed independently by Parker [179] and Rumelhart *et al.* [200][1]. In Lippman's often cited paper [128], several different neural network models are discussed, such as the Hopfield network, the multilayer perceptron and self-organising maps and these still remain the foundation of most of the current neural network research.

Applied research related to neural networks has also continued unabated since the early 1980's. The application of neural networks to image recognition, optimisation, pattern recognition, speech recognition and character recognition have been well studied (see for example IEEE Transactions on Neural Networks, Proceedings of the International Neural Networks Conference, Neural Computation etc). One application which has received a great deal of attention in recent years is the use of neural networks in the control of dynamic systems. The explosion of activity in this area can be attested to by the recent special issues of the IEEE Control Systems Magazine (1988, 1989, 1990, 1992) [84, 85, 86, 87] and the many papers on the subject in both control journal and conferences and neural networks journals and conferences. Furthermore, a number of recent books and collection of papers have been produced in this area [64, 151, 242].

From the control theory viewpoint, the emergence of neural networks as a method of forming an arbitrarily close approximation to any continuous nonlinear function has provided control researchers with an alternative approach to the traditional schemes to deal with a general class of complex nonlinear control problems. The traditional control methods developed for nonlinear systems, such as feedback linearisation, input-output linearisation and output feedback control require a number of assumptions to be made about the system to be controlled and subsequently, can only be applied to certain classes of systems (see Chapter 1). As a result, adaptive design techniques have yet to be established for a general class of nonlinear structure. Artificial neural network based control schemes offer promise in this area. Apart from the approximation abilities of neural networks, their inherent adaptivity has also generated a great deal of interest in

[1]The backpropagation (BP) algorithm was first developed by Werbos in 1974 [244] and its first practical application was for estimating a dynamic model to predict nationalism and social communications. However, the work of Werbos was virtually unknown in the scientific community

the control community. Furthermore, the parallel nature of neural networks and their fast adaptability has provided additional incentive for the investigation of their use in the identification and control of complex nonlinear structures.

In this chapter, a very brief exposition of the biological motivations and foundations of artificial neural networks is provided. An overview of some of the major neural network architectures, particularly the multilayer perceptron, and learning algorithms, such as backpropagation, is also provided. Some of the major concepts in the modelling of dynamic systems using neural networks are then discussed. Finally, an overview of the main neural control schemes is provided.

2.2 Basic Concepts of Artificial Neural Networks

2.2.1 Biological Foundations

Artificial neural networks were first derived as an attempt to model the networks of the biological neurons in the brain. Although artificial neural networks are highly simplified models of their human equivalent, they still provide an insight into the principles of biological computation.

The human brain consists of approximately 10^{11} elementary nerve cells called neurons. These elements form the fundamental building block of the biological neural network. Each of these neurons is connected to approximately 10^4 other neurons, resulting in a biological neural network of about 10^{15} connections. A diagram of a biological neuron is given in Figure 2.1.

A typical cell has three main regions: the cell body, the axon and the dendrites. The cell nucleus is located in a region called the cell body or soma. Around the body of the neuron is a tree-like network of nerve fibres called dendrites. Dendrites receive information from other neurons through the axon. The axon is a long fibre extending from the cell body which branches or arborizes into a series of strands and substrands, at the end of which are the interconnections to other neurons, known as the synapses. Thousands of such connections are made with other neurons. These connections are generally not physical

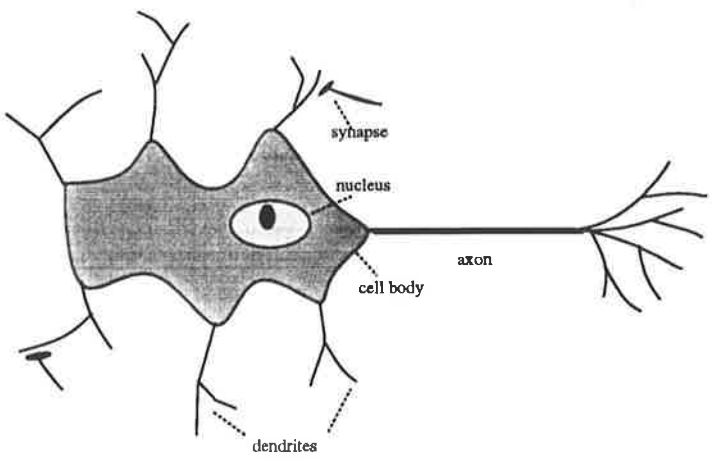


Figure 2.1: A typical biological neuron

connections as the axon and dendrites do not touch. The very small gap between the "connected" neurons is known as the synapse gap or synapse cleft.

Signals are transmitted from one cell to another at the synapse via a complex chemical process in which chemical molecules, called neurotransmitters, are released. These transmitters diffuse across the synapse gap to the dendrites at the other side of the synapse and modify the potential of the cell membrane. Depending upon the type of neurotransmitter generated, the cell potential is either increased (excitatory synapse) or decreased (inhibitory synapse). The signals received from each connecting neuron are then aggregated over a short time interval known as the period of latent summation. If the resultant potential exceeds a certain threshold, the neuron "fires" and a pulse of fixed strength and duration is sent down the axon where it branches out to other neurons. This voltage pulse is known as the action potential. After carrying a pulse, the neuron cannot fire for a certain time called the refractory period, even if it receives a very large excitation. Although the refractory period is not uniform over the cells, it is generally of the order of 3-4 milliseconds. Further details on the operation of biological neuron can be found in [3, 258].

2.2.2 Artificial Neural Network Models

As research in this area continues to develop and evolve, new and extended definitions of artificial neural networks continue to be generated. However, the formal definition of artificial neural networks proposed by Żbikowski and Gawthrop [254] is a succinct and relevant definition:

DEFINITION 2.2.1 An *artificial neural network* or *connectionist model* is characterised by

- *parallel architecture: it is composed of many self-contained, parallel interconnected processing elements or neurons;*
- *similarity of neurons: each basic processor is described by a standard nonlinear algebraic or differential equation;*
- *adjustable weights: there are multiplicative parameters each associated with a single interconnection and they are adaptive.*

Alternative and equally valid definitions have been proposed by Hecht-Nielsen [70], Hertz *et al.* [71] and Zurada [258].

The basic unit in the above definition is the neuron. The first formal definition of an artificial neuron model based on the highly simplified biological model presented above was formulated by M^{c}Cullogh and Pitts in 1943 [146]. A diagram of this model is given in Figure 2.2.

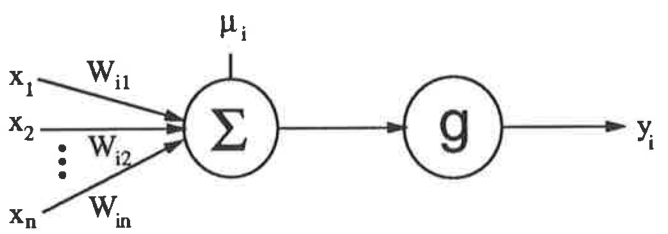


Figure 2.2: McCullogh-Pitts neuron model

In this model, a neuron is represented as a binary threshold unit, i.e., the neuron can be in only one of two possible states. In particular, the artificial neuron performs a weighted summation of its inputs from other units and generates a one or zero depending on whether the sum is above a below a given threshold. This can be represented by the following equation:

$$y_i = g(\sum_{j=1}^{n} W_{ij}x_j - \mu_i) \tag{2.1}$$

where y_i is either 1 or 0, and represents either the neuron firing or not firing,

W_{ij} represents the synaptic strength of the connection from neuron j to neuron i and is greater than zero for an exhitory synapse and less than zero for an inhibitory synapse,

μ_i is the threshold for neuron i which must be exceeded for the neuron to fire, and

$g(h)$ is the Heaviside function given by

$$g(h) = \begin{cases} 1 & h \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Although the M$^{\text{C}}$Cullogh and Pitts neuron model is very simplistic, it is still a computationally powerful device capable of performing basic logical operations such as NOT, OR and AND, provided its weights and thresholds (bias) are *selected* properly. One of the major limitations is that these values are fixed for a particular operation and are not adaptive. A great deal of research has been conducted into developing appropriate techniques to adapt the weights and thresholds, and these will be discussed in later sections.

A more commonly used neuron model utilises a sigmoid nonlinearity instead of the hard-limiting nonlinearity described above. A sigmoid function may be loosely defined as a continuous, real valued function whose derivative is always positive and whose range is bounded. The most commonly used sigmoidal activation function is the logistic function,

$$g(h) = \frac{1}{1 + e^{-2\beta h}} \tag{2.3}$$

where β is the steepness parameter, usually $\frac{1}{2}$ or 1. Other sigmoid functions such as the hyperbolic tangent are often used, i.e.,

$$g(h) = \tanh(\beta h) = \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{2.4}$$

One advantage of these types of functions is that their derivatives are easily found

$$g'(h) = \begin{cases} 2\beta g(h)(1 - g(h)) & \text{for (2.3)} \\ \beta(1 - g^2(h)) & \text{for (2.4)} \end{cases} \tag{2.5}$$

This property proved to be of major benefit in deriving gradient based learning algorithms for the multilayer perceptron. The sigmoidal nonlinearity is also popular in applications which require continuous-valued outputs rather than the binary output produced by the hard limiter. Control applications fall into this category.

The function $g(.)$ is often referred to as the *activation function* as it represents the threshold which exists within a biological neuron that must be exceeded in order for the neuron to activate. Although the above logistic activation function (2.3) or tanh activation function (2.4) are most commonly used, basically any monotonic, continuously differentiable thresholding function may be used for $g(.)$. Recently, B. L. Kalman and Kwasny [99] have made a strong case for the use of the hyperbolic tangent activation function. They show that tanh is the only activation function which satisfies four properties described as being necessary for the effective training of the neural network.

2.2.3 Network Topology

An artificial neural network consists of many interconnected neurons. The way in which these neurons are interconnected classifies the network into two major categories, namely feedforward neural networks, also known as static networks, and feedback neural networks also known as recurrent or dynamic networks.

Feedforward networks are systems in which the input and intermediate signals are always propagated forwards. Therefore, the flow of information is from the input of the network and is directed to the output of the network with no returning paths being allowed. A block diagram of a layered feedforward neural network is given in Figure 2.3a. In this network there is a set of input terminals whose role is to distribute the input signal to the rest of the network. This is followed by (possibly) several intermediate layers of neurons and then an output layer from which the network output is obtained. A feedforward neural network in which there are no connections from neurons in one layer to neurons in previous layers, the same layer or to neurons more than one layer ahead is known as a strictly feedforward artificial neural network. A feedforward neural network is often referred to as a static network because it implements a static mapping from its input space to its output space.

As the name suggests, feedback (or recurrent or dynamic) artificial neural networks involve either the states or the outputs being fed back. Therefore, feedback neural networks consist of interconnections between multiple neurons (and often onto themselves). One of the key features of recurrent neural networks is that because feedback is present, the current values of the network are influenced by the past values. Therefore, recurrent neural networks are dynamic systems and are generally described by nonlinear difference or differential equations. A block diagram of a recurrent neural network is given in Figure 2.3b. A special case of a recurrent network is one in which the output of every neuron is fed back with varying non-zero weights to the inputs of all neurons. Such a network is called a fully connected neural network. Recurrent neural networks are inherently dynamic and usually only consist of one layer because their complex nonlinear dynamics provide them with powerful representation capabilities. The ability of dynamic neural networks to model nonlinear systems will be considered in Section 2.3.

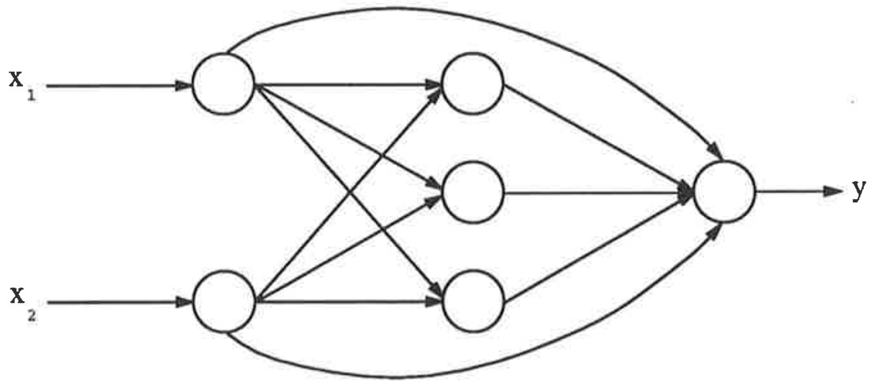


Figure 2.3a: Feedforward artificial neural network

2.2.4 Learning Algorithms

As defined earlier, artificial neural networks consist of adjustable weights which are adaptive multiplicative parameters associated with each neuron interconnection. This prop-

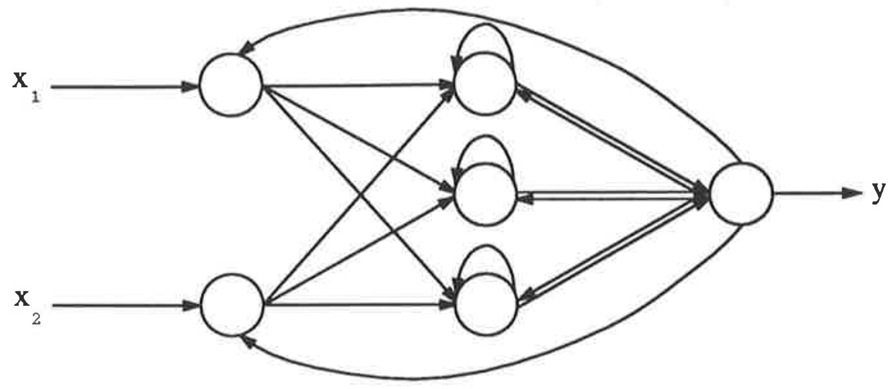


Figure 2.3b: Feedback artificial neural network

erty of artificial neural networks is known as learning. The ability to learn is one of the most attractive features of neural networks and one of the main reasons for their popularity. Learning is achieved by iteratively adjusting the weights (W_{ij}'s) of a network so as to improve its performance, as indicated by some performance or cost function. There are two general learning paradigms, namely *supervised* and *unsupervised* learning.

In supervised learning, one considers a training pair consisting of an input vector and a desired output vector. The input vector is applied to the network and then the resultant output vector is compared with the desired output vector. The difference is then used to modify the weights W_{ij} so that the output error is reduced via some learning algorithm. This scheme is sometimes called learning with a teacher, where the teacher provided the desired outputs. Supervised learning is well suited to model reference control and self-tuning regulators as the desired response is available. A special case of supervised learning is reinforcement learning. In this case the teacher provides information about whether the output is right or wrong. Therefore, an explicit desired response is not present in reinforcement learning. Reinforcement learning is sometimes called learning with a critic.

In unsupervised learning, no target vector is present. Therefore, explicit error information cannot be used to improve the network behaviour. Instead, the network must attempt to classify the output according to patterns, features, correlations or regularities in the input data. Such networks must thus self-organise so that a consistent output is produced for a given input vector. This learning is often referred to as self-organisation. Unsupervised learning is only useful when there is a certain degree of redundancy in the input data. Without redundancy, the input data would seem like random noise, thus making it impossible to detect any patterns or features. Examples of self-organising networks are the adaptive resonance theory (ART) networks [27] and Kohonen's self-organising feature maps [109].

2.2.5 Feedforward Artificial Neural Network Models

The first learning feedforward artificial neural network was developed by Rosenblatt in 1958 [196]. Rosenblatt proposed numerous variations of the perceptron, but the most commonly referred to consists of three layers and is often called the elementary perceptron. A diagram of this perceptron is given in Figure 2.4.

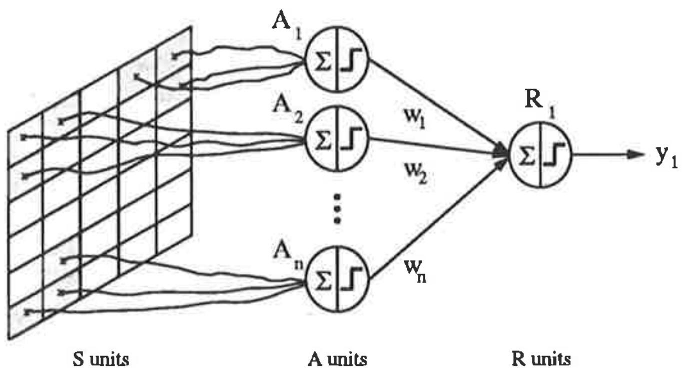


Figure 2.4: Rosenblatt's elementary perceptron

The first layer is an input retina, which consists of a series of sensory units or S-units. It is connected to a second layer composed of what he called associations units (A-units) which act as feature detectors. Finally the units in the output response layer (R-units) produce an output that is a threshold weighted sum of their inputs. As shown in the above figure, the original perceptron proposed by Rosenblatt used a hard-limiting nonlinearity. This was modified in subsequent incarnations of the perceptron. Another feature worth noting is that the weights connecting the input layer to the middle layer are fixed. Thus, although the perceptron model has three layers, it is actually more like a two layer network. Rosenblatt proposed a supervised learning rule to update the weights connecting the A-units to the R-units. A special case of Rosenblatt's perceptron model is the single layer perceptron. Rosenblatt was able to prove the convergence of a proposed learning algorithm to iteratively adjust the weights of a single layer perceptron [197]. This proof is known as the Perceptron Convergence Theorem [258]. Minsky and Papert [152] have analysed the perceptron in detail in their book *Perceptrons* and discovered several limitations with the perceptron learning theorem. One of the most important limitations proved by Minsky and Papert was that the single layer perceptron can only solve problems that are linearly separable. However, many interesting problems are not linearly separable. The simplest example is the exclusive-or (XOR) problem. Very similar networks called adalines were developed around the same time by Widrow and Hoff [246, 247].

Rosenblatt also studied structure with multiple layers of neurons and believed they could overcome the limitations of the simple perceptron. However, at the time there was no reliable algorithm which could be used to train the network. Minsky and Papert [152] doubted that a suitable algorithm could be found and as a result the interest in neural networks was diverted to other areas such as artificial intelligence until the mid 1980's.

Multilayer Perceptron

Interest in the multilayer perceptron (MLP) waned after the publication of the book by Minsky and Papert. However, there was a resurgence of interest in the mid-1980's thanks to the derivation of the backpropagation algorithm [179, 200]. A multilayer perceptron is formed by cascading two or more layers of neurons together. The neurons used in the MLP generally employ the sigmoidal nonlinearity instead of the hard-limiter of Rosenblatt's perceptron. A diagram of a fully connected multilayer perceptron is given in Figure 2.5.

This network is described as a three layered MLP as the input nodes do not comprise a layer. This is because they do not perform any computation. Instead they distribute the input vector to the internal (hidden) layers of the network. A *hidden layer* is defined as a layer of neurons which is not accessible from outside the MLP. Therefore the network shown has two hidden layers and one output layer. The output from neuron i in layer m

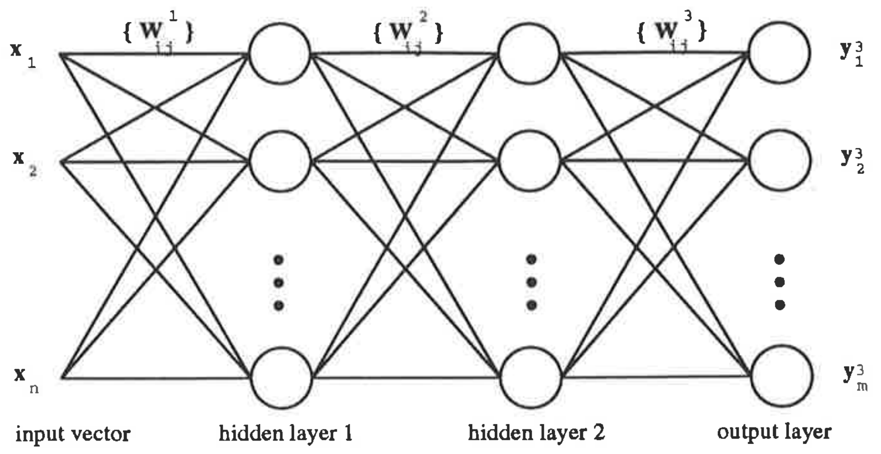


Figure 2.5: A multilayer perceptron

is given by

$$y_i^m = g(\sum_{j=1}^{m} W_{ij}^m y_j^{m-1}) \tag{2.6}$$

where W_{ij}^m is the weight from neuron j in layer $m-1$ to neuron i in layer m and $y_j^0 = x_j$. The convenient notation introduced in [169] to represent a multilayered perceptron will be adopted in this thesis. Thus a 3 layered network with 4 inputs, 1 output and 20 and 10 nodes in the respective hidden layers will be denoted $\Omega^3_{4,20,10,1}$.

Although in equation (2.6) it is assumed the output nodes have the same sigmoidal activation function as the hidden layer nodes, this is not always the case. Many applications use the linear output function $g(x) = x$ as the activation function for the output layer. The primary advantage with using linear activation functions is that the output is not constrained to the limits of the sigmoid, e.g. $[0,1]$. This is particularly important in time-series prediction and functional approximation, and therefore in control applications.

As mentioned earlier, the capabilities of a simple perceptron are limited to problems which consist of linear decision boundaries and simple logic functions. However, multilayer perceptrons are capable of nonlinear partitioning of the input space and thus it is possible to implement complex decision boundaries and arbitrary logic functions [82]. Furthermore, as will be discussed later, multilayer perceptrons are capable of forming an arbitrarily close approximation to any smooth, continuous nonlinear function.

Backpropagation Learning Algorithm

The backpropagation learning algorithm was originally proposed by Werbos in 1974 [244]. However, his work was not widely known in the scientific community. It was not until the rediscovery of the algorithm by Parker [179] and particularly Rumelhart *et al.* [200] that interest in the MLP was rekindled. To this day, it arguably remains the most commonly used supervised learning algorithm. The backpropagation algorithm is essentially an extension of the gradient descent algorithm to the multilayer perceptron. One of the primary stumbling blocks with developing a learning algorithm for the original formulation of the MLP [196] was that the derivative for a hard-limiting activation function is zero almost everywhere. However, the nonlinearity needed to be present otherwise the network would merely be implementing a linear transformation and therefore could be reduced to a single layer network. The introduction of a sigmoidal activation function made it possible to derive a gradient descent algorithm.

The backpropagation algorithm essentially provides a means of adapting the weights W_{ij} in a MLP to learn the training pair $\{x_j, d_i\}$ in a minimum least squares sense. A derivation of the algorithm is provided in Appendix A.

The backpropagation algorithm can be summarised as follows :

1. Randomly initialise the weights such that, for example, $W_{ij}^m \in \Re[-0.5, 0.5]$

2. Apply the inputs to the network $y_i^0 = x_i$

3. Generate the outputs for the following layers (forward pass)

$$y_i^m = g(h_i^m) = g(\sum_j W_{ij}^m y_j^{m-1}) \tag{2.7a}$$

4. Compute the delta for the output layer

$$\begin{aligned} \delta_i^M &= g'(h_i^M)[d_i - y_i^M] \\ &= g'(\sum_j W_{ij}^M y_j^{M-1})[d_i - y_i^M] \qquad i = 1, \ldots, N \end{aligned} \tag{2.7b}$$

5. Propagate the δ's backwards to get the the δ's for the inner layers

$$\delta_i^{m-1} = g'(h_i^{m-1}) \sum_i W_{ij}^m \delta_i^m \qquad m = M, \ldots, 1 \tag{2.7c}$$

6. Calculate the change in weights

$$\Delta W_{ij}^{m}(k) = \eta \delta_i^m y_j^{m-1} \tag{2.7d}$$

7. Calculate the new weights

$$W_{ij}^{m}(k+1) = W_{ij}^{m}(k) + \Delta W_{ij}^{m}(k) \tag{2.7e}$$

8. Go back to step 2 and repeat for the next time step.

where the neural network has M layers, $m = 1, \ldots, M$ and

y_i^m	is the output of neuron i in layer m,
y_i^0	is the ith input $= x_i$,
W_{ij}^m	is the weight for the connection from the jth node in layer $m-1$ to the ith in layer m,
d_i	is the desired response of the ith output node, and
N	is the number of nodes in the output layer.

The *learning rate* (η) is related to the size of the step taken along the error surface when adjusting the weights. A small learning rate means that only small steps are taken down the error surface, resulting in a smooth, virtually continuous path of descent. However, with a small learning rate it is possible to become stuck in a local minima, giving a sub-optimal solution. With a larger learning rate, bigger steps are taken and thus the time taken to reach a minima is smaller. However, with a larger learning rate, it is also possible to step over a narrow valley that may contain the global minima, but on the other hand it is also possible to step out of a valley containing a local minima. Hence, a trade-off between speed of convergence and optimality of the solution must be considered in setting the learning rate.

As shown above, the weights are usually initialised to small random values. This generally ensures that the search is started in a relatively safe position [82]. Nguyen and Widrow [170] have suggested an alternative method for choosing in the initial values of the adaptive weights. Rather than randomly initialising all of the weights over a particular region say $\{-0.5, 0.5\}$, Nguyen and Widrow suggest randomly initialising the weights of each hidden node over a distinct, but slightly overlapping smaller linear region. This is shown to result in a large reduction in learning time.

A complaint which is often made about the backpropagation algorithm is that learning can be very slow [77]. The reason for this is that the error surface for the MLP has a number of flat spots. As the step size for each update is given by $\eta \Delta W$, with the learning rate η usually fixed, then the actual steps taken along the error surface are determined by the gradient at that point. This means that on the commonly occurring flat portions of the surface, the gradient is small and thus the gradient search will move slowly along these regions. Therefore, learning is also slow. One method to speed up the learning process is to adapt the learning rate according to the local curvature of the surface [91, 200]. The simplest method is to add a momentum term of the form $\alpha(W_{ij}^m(k) - W_{ij}^m(k-1))$ to each weight update, so that equation (2.7e) becomes

$$W_{ij}^m(k+1) = W_{ij}^m(k) + \Delta W_{ij}^m(k) + \alpha(W_{ij}^m(k) - W_{ij}^m(k-1)) \qquad (2.8)$$

where $0 < \alpha \leq 1$ is the momentum constant and is typically chosen as 0.9. The momentum term makes the current search direction an exponentially weighted average of past directions. This is essentially a low pass filter applied to the search direction, so that rapid fluctuations are filtered out and thus the remaining trend will be towards a more global minimum. Furthermore, the addition of a momentum term helps with the problems associated with travelling along flat regions of the error surface after descending from steep portions. It has been shown that a momentum term helps to speed up convergence quite significantly [91, 258].

A primary reason for using a MLP in control and signal processing is its ability to implement nonlinear transformations for functional approximation problems. Training a neural network using input-output data can be considered a nonlinear functional approximation problem [80].

It has been shown by Cybenko [40], Hornik *et al.* [74] and Funahashi [55] that multilayered neural networks with an arbitrarily large number of nodes in the hidden layer can approximate any continuous function on a compact subset of the input space to a desired degree of accuracy. In particular, a continuous function can be approximated by a feedforward network with only a single internal hidden layer, where each node has a sigmoidal nonlinearity. Other forms of nonlinearities were also considered. Funahashi's theorem on the existence of an approximation function is given below.

THEOREM 2.2.1 ***Funahashi's Theorem***

Let $\phi(x)$ be a nonconstant, bounded and monotonically increasing continuous function. Let K be a compact subset of $\mathbf{R}^p$ and $f(x_1, \ldots, x_p)$ be a real valued continuous function on K. Then for any $\varepsilon > 0$, there exists an integer N and real constants $c_i, \sigma_i (i = 1, \ldots, N)$ and $W_{ij} (i = 1, \ldots, N; j = 1, \ldots, p)$ such that

$$\hat{f}(x_1, \ldots, x_p) = \sum_{i=1}^{N} c_i \phi(\sum_{j=1}^{p} W_{ij} x_j - \sigma_i) \tag{2.9}$$

satisfies $\max_{x \in K} \mid f(x_1, \ldots, x_p) - \hat{f}(x_1, \ldots, x_p) \mid < \varepsilon$ where $\sigma_i^{\prime}s$ are bias weights.

This theorem, and the equivalent theorems provided by Cybenko [40] and Hornik *et al.* [74], are existence theorems. They provide no information about the number of nodes required to approximate a nonlinear function. In general, the appropriate size of a network for a given problem is difficult to establish. Furthermore, because of the unique demands for the networks that each problem imposes, this problem is unlikely to be solved for a general case. Baum and Haussler [16] have found that if the network is too small it will not be able to form a good model for the problem. On the other hand, if the network is too big, it may be able to form a number of solutions that are consistent with the training data, but most of which provide only a poor approximation to the problem. Generally the number of nodes in the network is chosen using heuristic rules based upon past experience or some specific knowledge of the structure of the problem. For the case of a two-layer network (i.e., one hidden layer and one output layer), Huang and Huang [76] have derived a least upper bound for the functional approximation problem. They found that the number of hidden nodes should be one less than the number of training samples. In practice, however, it is desirable that the number of hidden nodes is much less than the number of training samples in order to prevent the network from simply memorising the training samples [83].

On the topic of the number of hidden layers required to approximate a function, Cybenko, Funahashi and Hornik *et al.* [40, 55, 74] have mathematically proved the approximation capabilities of multilayered neural networks with as few as one hidden layer. However, Chester [34] has given theoretical support to the empirical observation that networks with two hidden layers perform better in terms of accuracy and generalisation capabilities than a network with one hidden layer. Furthermore the overall number of processing units is less with two hidden layers. Blum and Li [21] have shown that there is a class of piecewise constant functions which cannot be implemented by a two-layer McCullogh-Pitts network. They show that three-layer networks are, in general, required to perform simple function approximation. Whilst Blum and Li, and Chester demonstrate the need for 2 hidden layers in terms of numerical accuracy and total number of neurons, Sontag [222, 223] has shown that nonlinear control systems can be stabilised using 2 hidden layers, but not, in general, using one. His reasoning is based upon the fact that control laws for nonlinear systems require the use of discontinuous mappings, which cannot be approximated well by single hidden-layer networks. In practice, the question of how many hidden layers would be best for an approximation problem is similar to the issue of the number of hidden layer nodes, in that the answer depends largely on the problem being considered.

Other Learning Algorithms

Apart from the backpropagation algorithm, a number of other supervised learning approaches have been recently developed [11, 32, 131, 198, 213]. In [11], a learning process for MLP's based on the recursive least squares algorithm is developed by Azimi. The method iteratively minimises the global sum of the squared errors between the actual and the desired output values. The weights in the networks are updated by recursively solving a system of normal equations. An analog of the backpropagation algorithm is used to determine the desired target values for the hidden layers. Simulations results indicate a very fast convergence behaviour for the proposed algorithm.

Chen *et al.* [32] have developed a recursive prediction error algorithm for the training of a MLP. The algorithm is based on the standard recursive prediction error algorithm provided in [130] and modified so that it can be integrated into the parallel structure of the network. The main drawback with the algorithm is that it is computationally more complex than the backpropagation algorithm, but it is claimed that it is likely to be more efficient in terms of convergence.

In the paper by Loh and Fong [131], the backpropagation algorithm is re-cast as a generalised least squares algorithm. The proposed method is an iterative procedure developed for a nonlinear identification problem. By using the generalised least squares algorithm, the learning rate is replaced by a general gain matrix derived from all the previous data. Simulation results are used to show its rapid rate of convergence compared with the backpropagation algorithm. However, as with [32], this is also achieved with an increase in computational complexity.

Singhal and Wu [213] have suggested the use of the extended Kalman filter to train the weights in a multilayer perceptron. A standard Kalman filter attempts to estimate the state of a system that can be modelled as a linear system driven by additive white Gaussian noise. The measurements available of linear combinations of the system states corrupted by additive white Gaussian noise. In the approach presented in [213], the weights of the MLP are the states that the Kalman filter attempts to estimate and the output of the network is the measurement used by the Kalman filter. As the multilayer perceptron is being considered, the extended Kalman filter is used. This algorithm uses the Gauss-Newton search direction in which the negative gradient is multiplied by the inverse of an approximate Hessian matrix of the given criterion . This is claimed to be a more efficient search direction than the steepest-descent approach of backpropagation and thus it results in a significant improvement of the convergence performance. However, it also results in an increase in computational complexity and the weight update requires a centralised processing and so the parallel structure of the MLP is not exploited. Ruck *et al.* [198] have performed a comparative analysis of backpropagation and the extended Kalman filter approaches and found that (1) the backpropagation algorithm is a degenerate form of the extended Kalman filter, (2) the backpropagation algorithm sacrifices a great deal of information about the weights that the extended Kalman filter uses, but (3) the backpropagation is computationally more efficient, achieves comparable classification performance and is a superior training algorithm in terms of accuracy as a function of computational cost over the extended Kalman filter.

Radial Basis Function

Another commonly used feedforward artificial neural network is the radial basis function (RBF) network [26, 158, 185, 226]. A RBF network consists of two layers in which the hidden nodes consist of basis (or kernel) functions which produce a localised response to the input vector, i.e., they produce a significant response when the input vector is within a small localised (radial) region of the input space defined by the basis function. The output nodes perform a simple linear combination of the basis function outputs. Due to the localised nature of the basis functions, RBF networks are often referred to as localised receptive field networks [158]. A block diagram of an RBF network is shown in Figure 2.6.

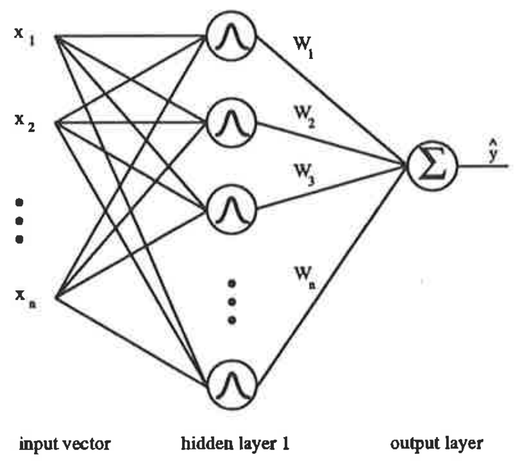


Figure 2.6: A radial basis function network

Radial basis functions get their names from the fact they employ Gaussian kernels that are radially symmetric. Therefore each node produces an identical output for inputs that are at a fixed radial distance from the centre of the kernel. The output from each node and the subsequent network output is given by

$$\phi_i(x) = \exp\left(-\frac{(x-x_i)^T(x-x_i)}{2\sigma_i^2}\right) \qquad i = 1,\ldots,N_h \tag{2.10a}$$

$$y = \sum_{i=1}^{N} W_i\phi_i(x) \tag{2.10b}$$

where ϕ_i is the output from the ith node in the hidden layer,

x is the input vector,

x_i is the centre of the basis function of hidden unit i,

σ_i is the normalisation parameter of the ith node, also called the width,

N_h is the number of hidden layer nodes,

W_i are the weights in the output layer, and

N is the number of output nodes.

The radial basis function can be considered to be a hybrid network, consisting of an unsupervised learning part (learning in the hidden layer) and a supervised learning part (learning in the output layer). The unsupervised part of learning is the determination of the basis centres x_i and widths σ_i. An approach for determining these values is the K-means clustering algorithm. This approach involves firstly randomly choosing the initial cluster (basis) centres x_i. Usually these are set to the first N_h training samples. Then all of the training patterns are grouped together with their closest cluster centre. For each group of samples, the new cluster is calculated as the mean of the sample values. The process is then continued until there is no change in the centre values. Once the clustering algorithm is complete, the normalisation parameters (or basis width) σ_i are then calculated. These are often chosen using the P-nearest neighbour heuristic [158]. The $P = 2$ nearest neighbour heuristic calculate the Euclidean distance between each cluster centre and its two nearest neighbours [226]

$$\sigma_i = \frac{1}{\sqrt{2}} \langle (x_i - x_\beta)^2 \rangle_p^{\frac{1}{2}} \tag{2.11}$$

where $\langle . \rangle_p$ is the average over the P nearest neighbours.

Moody and Darken [158] use a global value of $\sigma = \sigma_i$, which is the average of (2.11) over all cluster centres. Once the basis centres and widths have been determined they are usually held fixed. By doing this, the RBF network can be considered a special two-layer network which is linear in the parameters W_i. Thus the hidden layer performs a fixed nonlinear transformation with no adjustable parameters and it maps the input space into a new space with the only adjustable parameters being the weights of the linear combination given by equation (2.10b). These parameters can be determined using the linear least mean squares method, which is an important advantage of this approach. This represents the supervised learning stage of this approach. The weight update equation is given by

$$W_i(k+1) = W_i(k) - \eta(y_i - d_i)\phi_i(x) \tag{2.12}$$

More recently Chen *et al.* [33] have developed a systematic approach to the selection of RBF centres based on the orthogonal least squares algorithm. This approach is shown to be far superior than a random selection of RBF centres. Furthermore, in this approach, the selection of the centres is directly linked to the reduction of the error signals of the network.

As with the multilayer perceptron, the radial basis function network has been shown to be able to form an arbitrarily close approximation of any continuous nonlinear function [68, 178]. Therefore RBF networks are also used in time series prediction problems [33, 93, 158, 148] and identification and control [51, 79, 186, 202, 203, 204, 234, 235].

2.3 Dynamic Systems Modelling Using Artificial Neural Networks

Many forms of neural networks are used in the modelling of nonlinear dynamic systems, including modified versions of static networks such as the multilayer perceptron and the radial basis function, and dynamic networks such as the continuous-time and discrete-time recurrent neural networks. In this section the use of the major networks in dynamic systems modelling is discussed.

2.3.1 Time Delay Neural Networks

A time delay neural network (TDNN) is a modification of a static network such as the multilayer perceptron or the radial basis function which is used to model dynamic systems. It is formed by feeding the input sequence into a tapped delay line of a specified length and then feeding the taps from the delay line into the static neural network. A diagram of a TDNN is shown in Figure 2.7.

Such a network is capable of modelling nonlinear systems in which the output has a finite

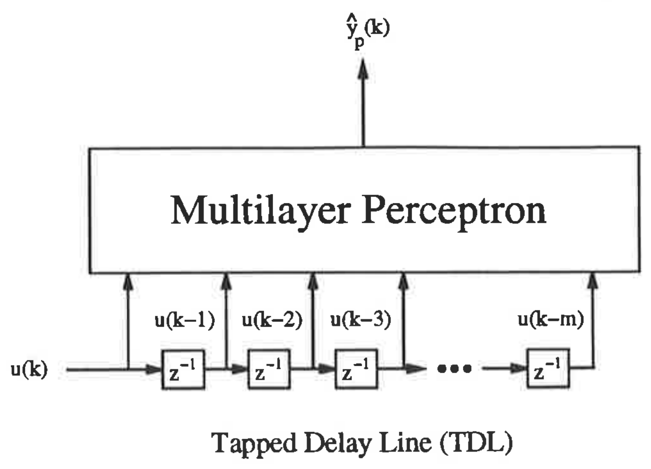


Figure 2.7: Time delay neural networks

temporal dependence on the input [83]:

$$y_p(k) = f(u(k), u(k-1), \ldots, u(k-m)) \tag{2.13}$$

These networks can be trained using the conventional backpropagation algorithm. TDNN have been widely applied to the problem of speech recognition [119, 129, 239, 240], time series prediction [93, 94, 120, 148] and identification and control [28, 169, 166].

Feedback can be incorporated into the above structure by simply feeding back the output of the network through a second tapped delay line. The architecture is considered by Narendra and Parthasarathy in [169] for use in the identification and control of nonlinear systems. A block diagram of such a system is given in Figure 2.8.

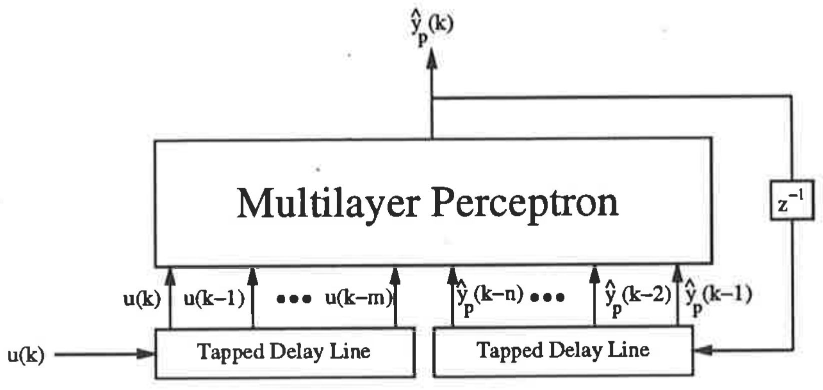


Figure 2.8: TDNN with output feedback

The architecture is very general and can be used to model nonlinear systems of the form

$$y_p(k) = f(y_p(k), \ldots, y_p(k-n), u(k), \ldots, u(k-m)) \tag{2.14}$$

The architecture shown in Figure 2.8 is often referred to as a parallel model because the previous outputs of the identification model (namely $\hat{y}_p(k-1), \ldots, \hat{y}_p(k-n)$) are fed back to generate the current model output. In practice, a series-parallel model is generally used. In a series-parallel model, the actual output of the plant, rather than the neural estimate of the plant output, is fed back into the neural network approximating the plant dynamics. Reasons for choosing a series-parallel model over a parallel identification model are provided in the next chapter.

If time delay neural networks are to be used for the identification and control of nonlinear systems (2.14), then the number of delayed values of plant input (m) and plant output (n) need to be known *a priori*. This knowledge enables the required number of taps for the network to be determined to achieve a good approximation of the nonlinear system. Over-parameterisation (choosing too many taps) or under-parameterisation (too few taps) often results in the model being very sensitive to noise dynamics or grossly inaccurate. Furthermore, as shown by Baum and Haussler [16], a neural network with excess degrees of freedom has bad generalisation performance. The adequacy of a particular model is also affected by such factors as insufficient hidden nodes, poor choice of learning rate and noisy data. In the work presented in this thesis, these values are assumed to be known. However, if the number of taps cannot be known *a priori*, then several approaches can be used to determine the appropriateness of a particular model. One may use the magnitude of the identification error as a guide to validate the neural network identification model. However, this is unreliable as a poorly chosen model may produce an equally good identification error as the correct model for a particular data set. A more quantitative method of model validation, suggested by Billings *et al.* , is that if a model of a system is adequate then the identification error should be unpredictable from all linear and nonlinear combinations of past inputs and outputs. Therefore, the following correlation conditions should hold if a particular model is valid:

$$\begin{aligned} \phi_{\epsilon\epsilon}(\tau) &= E[\epsilon(k-\tau)\epsilon(k)] = \delta(\tau) && \text{(2.15a)} \\ \phi_{u\epsilon}(\tau) &= E[u(k-\tau)\epsilon(k)] = 0 \qquad \forall\tau && \text{(2.15b)} \end{aligned}$$

$$\phi_{\acute{u^2}\epsilon}(\tau) = E[(u^2(k-\tau)-\bar{u}^2(k))\epsilon(k)] = 0 \qquad \forall\tau \qquad (2.15c)$$

$$\phi_{\acute{u^2}\epsilon^2}(\tau) = E[(u^2(k-\tau)-\bar{u}^2(k))\epsilon^2(t)] = 0 \qquad \forall\tau \qquad (2.15d)$$

$$\phi_{u(\epsilon u)}(\tau) = E[(\epsilon(k-\tau)\epsilon(k-1-\tau)-u(k-1-\tau))] = 0 \qquad \tau \geq 0 \quad (2.15e)$$

where $\acute{u^2}$ is the signal u^2 with the mean level $\bar{u}^2$ removed and the correlation function between two sequences ψ_1 and ψ_2 is

$$\phi_{\psi_1\psi_2}(\tau) = \frac{\sum_{k=1}^{N-\tau}\psi_1(k)\psi_2(k+\tau)}{\left[\sum_{k=1}^{N}\psi_1^2(k)\sum_{k=1}^{N}\psi_2^2(k)\right]^{\frac{1}{2}}} \qquad (2.16)$$

The correlations are normalised such that $-1 \leq \phi_{\psi_1\psi_2}(\tau) \leq 1$. For a large data set N, the standard deviation of the correlation estimate is $\frac{1}{\sqrt{N}}$ and therefore the 95% confidence limits are approximately $\frac{1.96}{\sqrt{N}}$. These limits are used to indicate if the estimated correlations are significant or not.

For linear systems, well-established tests are available for validating the estimated model. In particular, the well-known covariance tests, which consist of calculating the autocorrelation of the residuals (2.15a) and the cross-correlation between the residuals and the input (2.15b), are commonly used. However, as shown by Billings and Voon in [19], the covariance tests are not sufficient to validate a model of a nonlinear system. As a result additional tests (2.15c – 2.15e) are derived for a particular class of analytic nonlinear systems [19]. Neural networks can be used to model a wider range of nonlinear systems. Therefore, it is impossible to state that the correlation tests will be able to detect all of the possible nonlinear terms that the neural network represents. However the simulation results considered here and by Billings *et al.* [18] indicate that these tests are useful tools in the analysis of neural networks for the modelling of nonlinear systems.

Model validity tests such as this are applied to detect if there are any unmodelled terms in the residuals which will result in biased estimates when they are omitted from the model. However, the above tests can only be applied when a noise process is also present. The tests can be applied to systems in which the noise is additive at the output, such as sensor or measurement noise, or where it enters the system internally and thus is coloured by the dynamics of the system, known as dynamic noise. In the tests, it is assumed that worst possible combinations of signal properties exist, i.e., the input $u(.)$ and noise $v(.)$ are independent zero mean processes, $v(.)$ is white and $u(.)$ may be white.

The following example is used to highlight the effectiveness of a correctly chosen time delay neural network in modelling a nonlinear dynamic system and the usefulness of the model validity tests.

EXAMPLE 2.3.1 Consider the nonlinear system defined by the difference equation

$$\begin{aligned} \acute{y}(k+1) &= \frac{\acute{y}^3(k) + 0.5u(k) + 0.5u(k-1)}{1 + \acute{y}^2(k)} \\ y_p(k+1) &= \acute{y}(k+1) + n(k+1) \end{aligned} \tag{2.17}$$

The plant is nonlinear in output and control, and the nonlinearities are not separable. The plant output $\acute{y}(k+1)$ is corrupted by a white measurement noise $n(k+1)$, thus resulting in the measured plant output of $y_p(k+1)$. The input-output pair $\{u(.), y_p(.)\}$ is used to train the neural network.

A correctly parameterised model for this system consists of a neural network with the input vector $x(k) = [y_p(k), u(k), u(k-1)]^T$ and thus can be represented by the identification model

$$y_c(k+1) = N_1[y_p(k), u(k), u(k-1)] \tag{2.18}$$

An over-parameterised model is represented by the equation

$$y_{op}(k+1) = N_2[y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1), u(k-2)] \tag{2.19}$$

and an under-parameterised model by

$$y_{up}(k+1) = N_3[y_p(k), u(k)] \tag{2.20}$$

The neural networks N_1, N_2 and N_3, which are of the form $\Omega^3_{3,20,10,1}$, $\Omega^3_{6,20,10,1}$ and $\Omega^3_{2,20,10,1}$, respectively, are trained with a zero mean random input signal $u(k) \in \Re[-1,1]$ and a noise process $n(k) \in \Re[-0.1, 0.1]$ for 10,000 iterations with a learning rate of $\eta = 0.1$ and a momentum rate of $\alpha = 0.9$. The resultant prediction errors are $\epsilon_1 = 0.0065$, $\epsilon_2 = 0.0066$ and $\epsilon_3 = 0.0573$. The identification responses for the three models are given in Figures 2.9a, 2.9b and 2.9c.

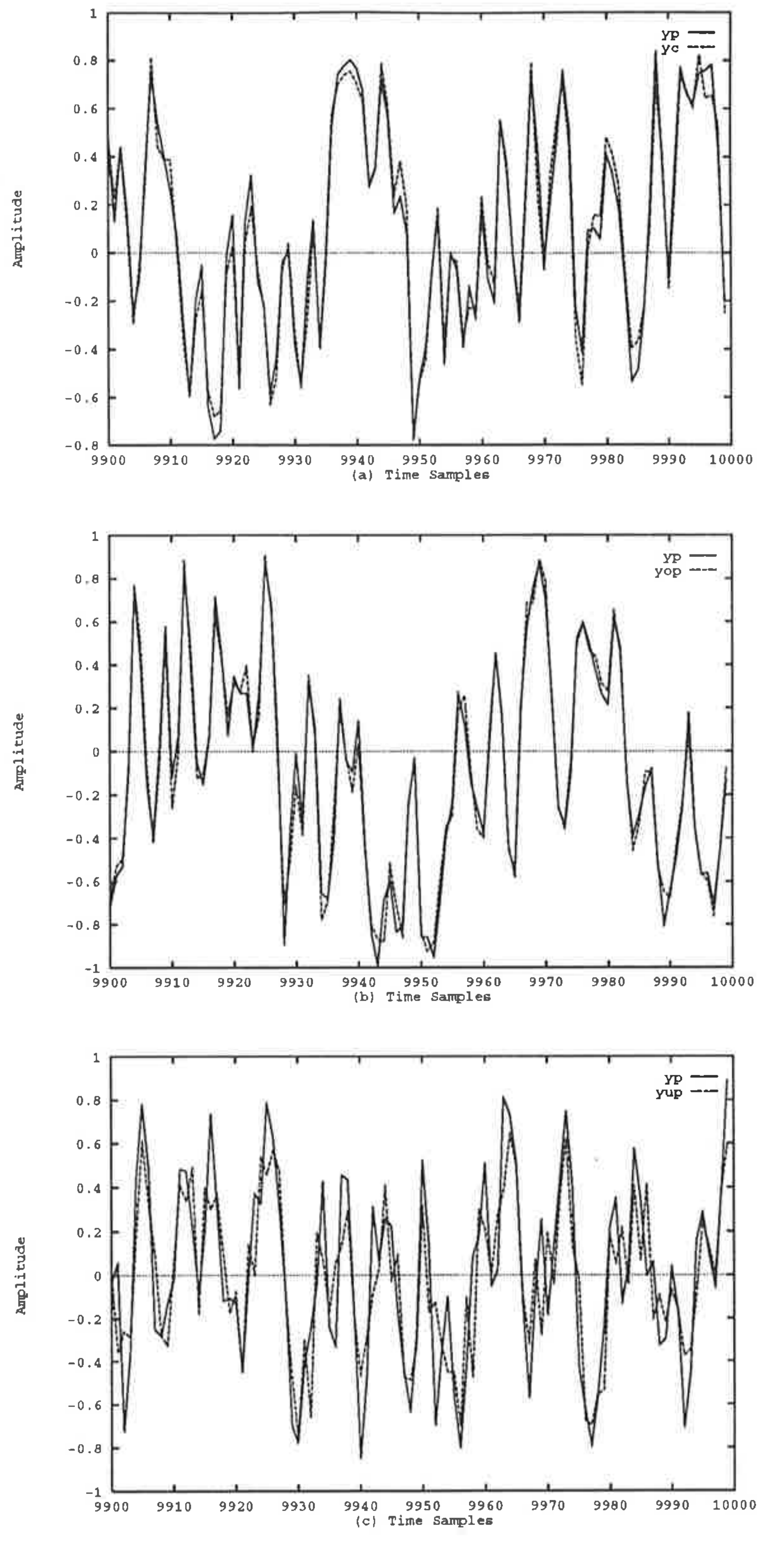


Figure 2.9: Response of the plant and identification model for Example 2.3.1 with (a) correctly parameterised (b) over-parameterised and (c) under-parameterised time delay neural network

As can be seen from these figures, the identification results are very good for the correctly parameterised case and the over-parameterised case, but are very poor for the under-parameterised case. The poor performance of the under-parameterised case is primarily due to the fact that the $u(k-1)$ term is neglected in the identification model, but it has a significant effect on the plant response. These figures also highlight the deficiency in using the identification error to try and determine the appropriate neural network parameterisation. In this example, the over-parameterised model consists of three extra terms $(y_p(k-1), y_p(k-2), u(k-2))$, yet it virtually has an equivalent identification error as the correctly parameterised model. Not only does the over-parameterised model require additional weights and thus greater computation time than the correctly parameterised case, for an alternative data set it is likely to perform significantly worse.

The model validity tests are then carried out for each model with a zero mean uniformly distributed white noise input $u(k)$. The results are shown in Figures 2.10a, 2.10b and 2.10c.

The results for the correctly parameterised model given in Figure 2.10a illustrate that all of the validity tests, except $\phi_{u^{2\prime}\epsilon}$ are satisfied. However as shown in [20], if a additive noise process is being considered then the correlation terms $\phi_{\epsilon\epsilon}$, $\phi_{u\epsilon}$ and $\phi_{\epsilon(\epsilon u)}$ are of principle importance in determining the validity of a model, whereas the remaining terms $\phi_{u^{2\prime}\epsilon}$ and $\phi_{u^{2\prime}\epsilon^2}$ help to determine the validity of the model when dynamic or internal noise is also present. Therefore, for the system considered, the correlation results given in Figure 2.10a verify that the model considered is the correct model.

The results given in Figure 2.10b indicate that several of the correlation terms are outside the 95% confidence limits for the over-parameterised case. In particular, the terms $\phi_{u\epsilon}$ and $\phi_{\epsilon(\epsilon u)}$ indicate that the model is deficient in some way. Therefore, the results confirm that the correlation tests are a better method of determining the validity of a particular model than comparing identification errors. However, a point which must be noted for both the correctly parameterised model and the over-parameterisation model is that although $\phi_{\epsilon\epsilon} \approx \delta$, this autocorrelation will never exactly be the delta function because both the system inputs and outputs have been used as input nodes to the network.

The model validity test for the under-parameterised case are provided in Figure 2.10c.

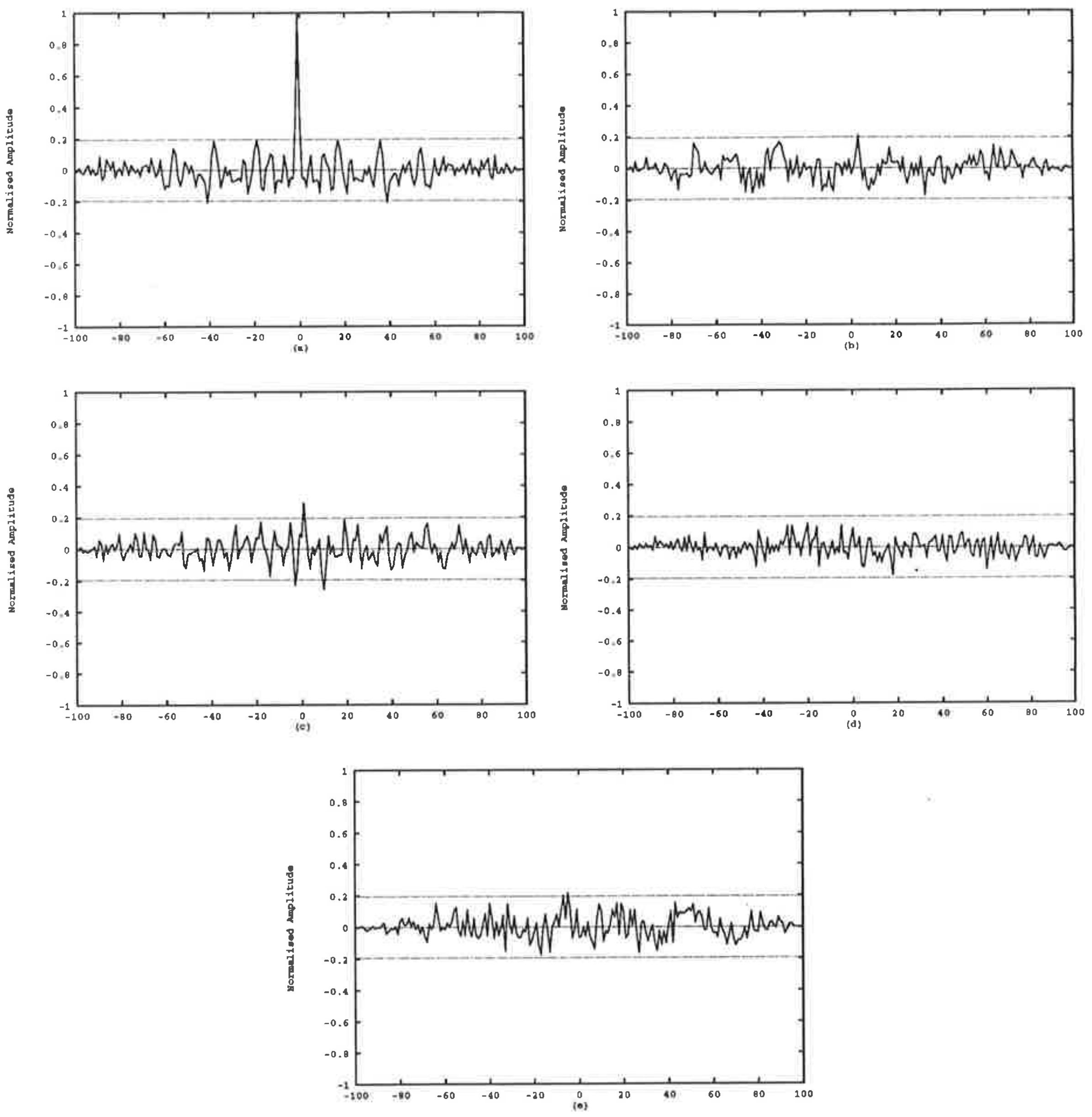


Figure 2.10a: Model validity tests for a correctly parameterised identification model: (a) $\phi_{\epsilon\epsilon}$, (b) $\phi_{u\epsilon}$, (c) $\phi_{u^{2'}\epsilon}$, (d) $\phi_{u^{2'}\epsilon^2}$, (e) $\phi_{u(\epsilon u)}$

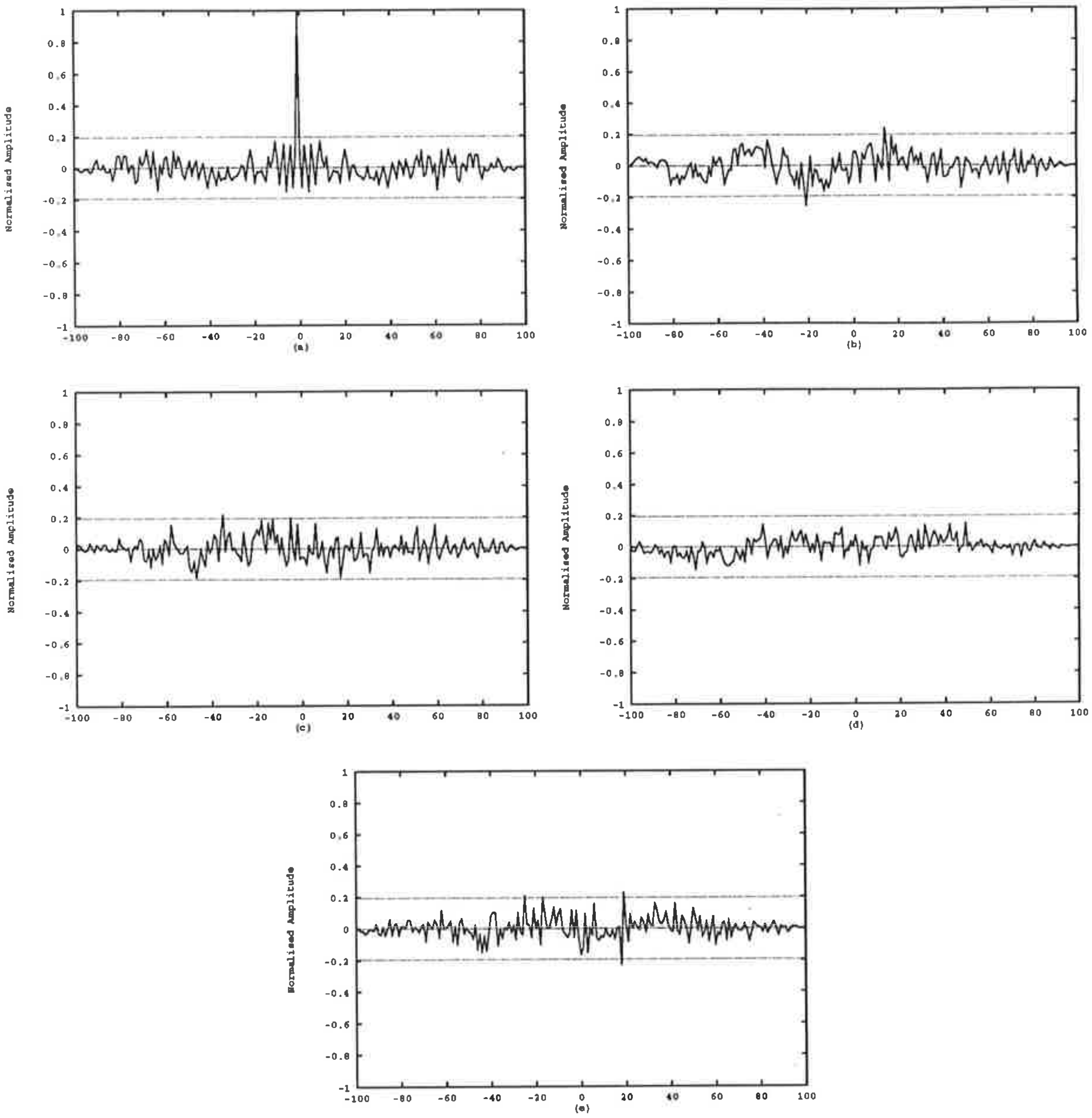


Figure 2.10b: Model validity tests for an over-parameterised identification model: (a) $\phi_{\epsilon\epsilon}$, (b) $\phi_{u\epsilon}$, (c) $\phi_{u^{2'}\epsilon}$, (d) $\phi_{u^{2'}\epsilon^2}$, (e) $\phi_{u(\epsilon u)}$

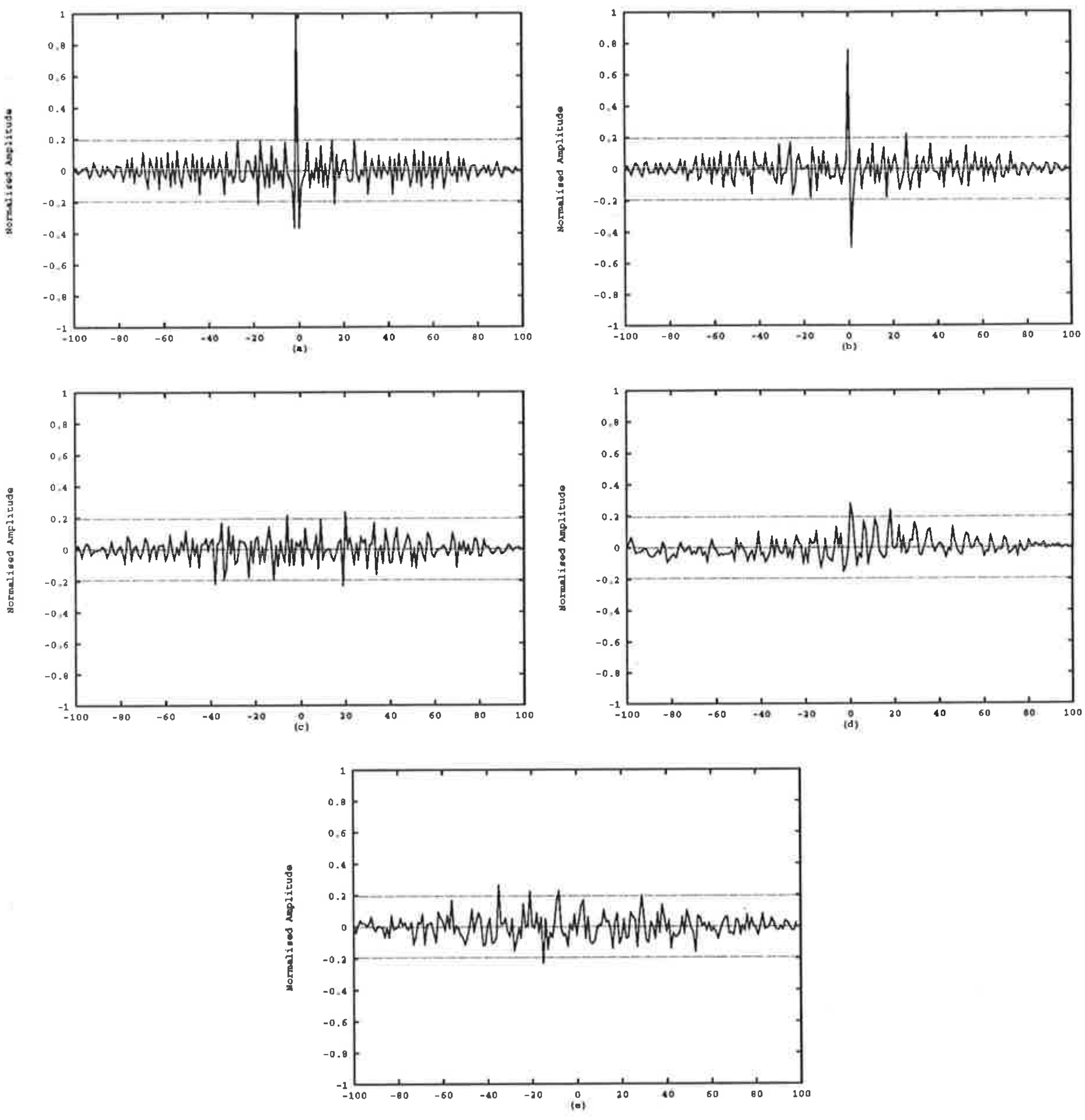


Figure 2.10c: Model validity tests for an under-parameterised identification model: (a) $\phi_{\epsilon\epsilon}$, (b) $\phi_{u\epsilon}$, (c) $\phi_{\acute{u^2}\epsilon}$, (d) $\phi_{\acute{u^2}\epsilon^2}$, (e) $\phi_{u(\epsilon u)}$

These results show that all of the correlations are well outside the residual noise correlation $\phi_{\epsilon\epsilon}$ and the cross-correlation $\phi_{u\epsilon}$, confirming that the under-parameterised model is severely deficient for the given system.

Hence, if prior knowledge of the number of delayed values of plant input and plant output is not available, then the main conclusions which could be drawn from the above results are that the correct model for the system considered would be the first one analysed and that the remaining two models are deficient in some way.

In the examples considered here, uncorrelated white noise is added to the output as sensor noise. However, if coloured noise (either coloured sensor noise or dynamic noise) is present then the noise process would induce bias in the neural estimate. This bias means that although the neural network provides good prediction over the data used in training, it is valid over that data set and may not perform as well for different sets. Therefore, the neural network performs a curve-fitting role, rather than modelling the underlying system dynamics. This defeats the purpose of using neural networks to model nonlinear systems. It is therefore very important to eliminate this bias. This can be achieved if the residual becomes uncorrelated with respect to past measurements of the system response. One way to do this is to model the noise as well. For an additive coloured noise source, the neural network identification model can also be used to model the noise process by incorporating extra input terms to represent the noise process, e.g., $x(k) = [y_p(k), u(k), u(k-1), n(k), \ldots, n(k-p)]^T$, where p is the maximum lag of the noise process. This issue is dealt with in more detail in [18].

2.3.2 Dynamic Neural Networks

Dynamic or recurrent neural networks can offer great advantages over feedforward neural networks in certain problems. Dynamic networks are particularly appropriate for identification, control and filtering applications. However, one difficulty with these networks is developing learning algorithms, particularly gradient search based algorithms. This is because the output of the nodes in a recurrent network is a recursive function of the output of the nodes from the previous time step and thus the calculation of the gradient is also a recursive computation. This makes the learning algorithm far more complex.

A model of a continuous-time recurrent network [71, 79, 184] is given in Figure 2.11. This network consists of a single layer of nodes which are fully interconnected. The dynamics of the network are described by the following differential equation

$$\tau_i \dot{y}_i(t) = -y_i(t) + g(\sum_{j=1}^{N} W_{ij} y_j(t) + u_i) \quad i = 1, 2, \ldots, N \tag{2.21}$$

where τ_i is a positive constant,

y_i is the state of the ith node,

W_{ij} is the weight connecting node j to node i,

u_i is the input to the ith node, and

$g(.)$ is the nonlinear (sigmoidal) activation function.

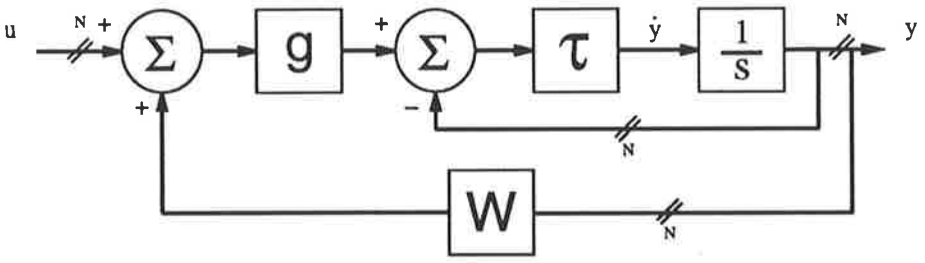


Figure 2.11: Continuous-time recurrent neural network

The discrete-time version of the continuous-time recurrent neural network, called the discrete-time recurrent neural network [194, 250], is shown in Figure 2.12 and is described by the following difference equation

$$y_i(k+1) = g(\sum_{j=1}^{N} W_{ij} y_j(k) + u_i(k)) \tag{2.22}$$

Recurrent neural networks possessing the same structure can exhibit different dynamic behaviour, depending on the learning algorithm used. Therefore, recurrent neural networks are defined not only by their structure, but also by the learning rules used. There are two general approaches to the training of recurrent neural networks, fixed point learning and trajectory learning. Fixed point learning is aimed at making the network reach a desired equilibrium or fixed point. This requires that the transients die away and

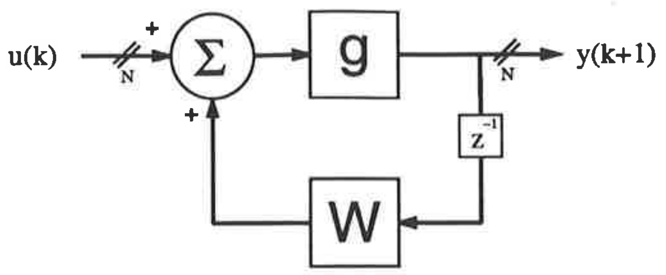


Figure 2.12: Discrete-time recurrent neural network

that the fixed point is a stable attractor. In trajectory learning, the network is trained to follow the desired trajectory in time. Trajectory learning can be considered to be a generalisation of fixed point learning as when $t \to \infty$, the trajectory will also reach a prescribed steady state.

Fixed Point Learning

It can be seen from equation (2.21) that a fixed point for the network (i.e., where $\dot{y} = 0$) is given by

$$\xi_k = g(\sum_{j=1}^{N} W_{kj} y_j(t) + u_k) \tag{2.23}$$

The error measure for the fixed point is defined as

$$E = \frac{1}{2} \sum_k E_k^2 \tag{2.24}$$

where

$$E_k = \begin{cases} \xi_k - y_k & \text{if } k \text{ is an output unit} \\ 0 & \text{otherwise} \end{cases} \tag{2.25}$$

During fixed point learning, the network does not receive any external inputs. It is excited by the initial conditions and evolves with ξ_k as a constant reference signal. This is achieved using recurrent backpropagation [184].

Trajectory Learning

Trajectory learning is a more appropriate scheme for dynamical systems modelling as it involves training a recurrent neural network to follow a desired trajectory (or input-output sequence) in time. For this case the error for the recurrent network defined by the differential equation (2.21) is given by

$$E = \frac{1}{2}\int_{t_0}^{t_1}[(d(\tau) - y(\tau))^T(d(\tau) - y(\tau))]d\tau \tag{2.26}$$

where $d(\tau)$ is a vector of the desired trajectories and t_1 is a constant for an off-line scheme or a variable for an on-line scheme. The discrete-time equivalent of the system (2.22) is given by

$$E = \frac{1}{2}\sum_{t_0}^{t_1}[(d(\tau) - y(\tau))^T(d(\tau) - y(\tau))] \tag{2.27}$$

In order to train a recurrent neural network to minimise the above functions, a new learning rule is required. Conventional backpropagation cannot be used as it does not allow for modifiable recurrent connections. Recurrent backpropagation will not suffice either as it assumes constant inputs and an approach to a fixed point or attractor, whereas the interest here is in input-output sequences. The simplest method of achieving trajectory learning is a scheme called backpropagation through time which was originally suggested by Minsky and Papert [152], combined with backpropagation by Rumelhart *et al.* [200] and elaborated by Werbos [245]. The basis of the scheme is to replace a one-layer recurrent neural network with a feedforward one with t_1 layers. This is often referred to as "unfolding the network through time". The resultant unfolded network is strictly feedforward and can be trained using standard backpropagation. Further details can be found in [79, 80, 245]. A continuous version of backpropagation through time is introduced by Pearlmutter in [181] and can be applied to the system described by equation (2.21) and error function (2.26). One problem with backpropagation through time is that it is memory exhaustive as it requires the storage of several copies of the network. This memory requirement grows as the length of the training sequence grows. Another approach commonly used is real-time recurrent learning [194, 250]. Whereas backpropagation through time is inherently an off-line technique, real-time recurrent learning is essentially an on-line algorithm in which the gradient is calculated recursively. The primary advantage with this method is that it is more memory efficient than backpropagation through time, however it is computationally expensive because of its recursive nature. Further details about real-time recurrent learning can be found in [80, 194, 250, 254].

2.4 Neural Adaptive Control Architectures

Some of the most popular nonlinear control schemes are the geometric approaches such as feedback linearisation, input-output linearisation and output feedback control. However, as shown in Chapter 1, these methods require a number of assumptions to be made about the system. In particular, such approaches assume the following continuous-time canonical model

$$\dot{x} = f(x) + g(x)u \tag{2.28}$$

which, although fairly general, is not universal. The great diversity of nonlinear systems is the main reason that adaptive design techniques for nonlinear systems have yet to be established for a general class of systems. This is in contrast to linear system theory where control schemes are well established for a general class of plants [167, 205].

From the control point of view, artificial neural networks offer an alternative to the conventional approach. In particular, it is their ability to represent nonlinear mappings and hence to model nonlinear systems which can be utilised in the synthesis of nonlinear controllers. As a result, a number of neural network based control approaches have been developed in recent years. These can be broadly categorised into four major approaches: supervised control, inverse control, adaptive critic and neural adaptive control.

The basic idea of supervised control is to train a neural network to mimic the actions of an existing controller. The earliest example of a supervised control system is the broom balancing system of Widrow and Smith [248]. In this system the teaching controller, on which the neural network is trained, implements a linear switching surface. A similar approach is adopted by Guez ad Selinsky [62]. In this paper a neural network is trained to mimic a "teacher" which controls a cart-pole system. Three main types of "teacher" are considered: explicit linear control law, explicit nonlinear control law or a human operator. Hecht-Nielsen [70] also considers the issue of training a neural network to control a cart-pole system using a human teacher.

The inverse control approach was first suggested by Psaltis *et al.* [189]. The basic idea is that a neural network is trained to emulate an inverse system model, and then this inverse model is cascaded with the plant such that the combined system results in an identity mapping between the desired response (reference input to the controller) and the controlled system output. This approach has several limitations when nonlinear systems are being considered. Firstly, the inverse of a nonlinear system may not exist, and if it does, it may not be unique. Secondly, such an approach is limited to systems with equal number of inputs and outputs. Finally, the approach is an open-loop feedforward control scheme and thus the robustness of such a scheme can be poor. This lack of robustness can be attributed primarily to the absence of feedback. However, this problem can be overcome to a certain extent by using an on-line scheme, such as the specialised learning architecture [189] or the forward modelling approach [96]. One problem with such schemes is that the weight update equation for the controller network is difficult to derive. This is because the plant is located between the controller network and the output error. Therefore, in order to obtain the appropriate error for the controller output it is necessary to find the Jacobian of the plant. Psaltis *et al.* [189] consider a first order approximation of the plant Jacobian. Jordan and Jacobs [96] have shown that the plant Jacobian can be obtained by backpropagating the output error through a neural network which identifies the plant. As a neural network is used to model the plant, this scheme is often referred to as forward modelling or indirect inverse control. These issues will be discussed in more details in the next chapter.

The adaptive critic approach is an extension of the reinforcement learning method. Unlike other neural control approaches, where the aim is to determine the control outputs from desired plant responses, the adaptive critic scheme involves determining the control that would lead to an increase in a measure of the plant performance. This measure is not necessarily defined in terms of the desired response. This approach generally consists of two neural networks: a critic network and a controller network. The critic evaluates the plant performance and then generates an evaluation signal which is used in conjunction with a reinforcement learning algorithm to train the controller network. Such a scheme is used by Barto *et al.* [15] for a cart-pole system.

The final major neural control approach is neural adaptive control. This approach is perhaps the most popular neural control scheme, particularly amongst the control community. This is because neural adaptive control involves the incorporation of artificial neural networks into conventional control frameworks such as model reference adaptive control, self-tuning regulators and feedback linearisation. In the remainder of this section, some of the recent neural adaptive control approaches will be reviewed.

One of the seminal publications in neural adaptive control is the paper by Narendra and Parthasarathy [169]. The paper demonstrates that the use of artificial neural networks in the identification and control of a wide range of non-trivial nonlinear systems is feasible. Four different classes of discrete time single input-single output (SISO) systems are considered. These can be described by the following nonlinear difference equations :

- Model I

$$y_p(k+1) = \sum_{i=0}^{n-1} a_i y_p(k-i) + g[u(k), u(k-1), \cdots, u(k-m+1)] \tag{2.29}$$

- Model II

$$y_p(k+1) = f[y_p(k), y_p(k-1), \cdots, y_p(k-n+1)] + \sum_{i=0}^{m-1} b_i u(k-i) \tag{2.30}$$

- Model III

$$y_p(k+1) = f[y_p(k), y_p(k-1), \cdots, y_p(k-n+1)] + g[u(k), u(k-1), \cdots, u(k-m+1)] \tag{2.31}$$

- Model IV

$$y_p(k+1) = f[y_p(k), y_p(k-1), \cdots, y_p(k-n+1); u(k), u(k-1), \cdots, u(k-m+1)] \tag{2.32}$$

where $f(.)$ and $g(.)$ are nonlinear functions, $u(k) \in \mathbf{R}$ is the scalar control and $y_p(k) \in \mathbf{R}$ is the scalar output, $m \leq n$, a_i and b_i are scalar constants and the plants are minimum phase with known order and relative degree. In Model I, the plant is linear in output, but nonlinear in control, whilst in Model II, the opposite is the case. In Model III, the plant is nonlinear in output and control, but the respective nonlinear functions are separable, whilst in Model IV they are not. Model IV clearly encompasses the other three models and is thus the most general model.

In the paper, Narendra and Parthasarathy successfully implement a neural network emulator for plants belonging to each of the four models. However, in models where linear terms exist, they assume that these are known and only identify the nonlinear parts of the plant. In the case of Model III, where two nonlinear functions exist, they implement two neural emulators, one to identify the nonlinear function $f(.)$ and the other to identify the nonlinear function $g(.)$. A multi input-multi output (MIMO) system is also successfully identified.

The problem of designing a controller, such that the output of the plant tracked the output of a stable reference model, is then considered. In designing the controller, Narendra and Parthasarathy assume that control and output terms are separable and that the linear terms present are known or can be separately identified. Furthermore, when controlling plants belonging to Models I and III, they assume that the inverses of the operators on the control $u(k)$ exist and can be approximated. A control structure for a plant belonging to Model IV is not considered as it is regarded as being analytically intractable. In spite of its limitations, the results provided show that the method is very effective, in terms of good tracking and identification.

The approach proposed in [169] is best highlighted by the following example.

EXAMPLE 2.4.1 Consider the plant described by the following difference equation

$$y_p(k+1) = \frac{y_p(k)}{1+y_p^2(k)} + u^3(k) \tag{2.33}$$

The plant equation is nonlinear in output and control, and their respective nonlinear functions are separable. It thus belongs to Model III. In [169] two neural network emulators, N_f and N_g are used to emulate the separable nonlinear functions $f(y_p(k))$ and $g(u(k))$. The identification model used is

$$\hat{y}_p(k+1) = N_f[y_p(k)] + N_g[u(k)] \tag{2.34}$$

Both neural networks belonged to the class $\Omega^3_{1,20,10,1}$ and are trained over the interval $[-10, 10]$ and $[-2, 2]$, respectively. The reference model used is of the form

$$y_m(k+1) = \alpha_m y_m(k) + r(k) \tag{2.35}$$

and thus the control input is given by

$$u(k) = N_{g^{-1}}[-N_f[y_p(k)] + \alpha_m y_p(k) + r(k)] \tag{2.36}$$

where $N_{g^{-1}}$ is a neural network of the class $\Omega^3_{1,20,10,1}$ trained to estimate of the inverse of the nonlinear function in control, $g(.)$ and α_m is a prespecified constant. $N_{g^{-1}}$ is trained such that $N_g[N_{g^{-1}}[r]] \approx r$ over the interval $r \in [-4, 4]$. Good tracking results are achieved for $r(k) = \sin(\frac{2\pi k}{25}) + \sin(\frac{2\pi k}{10})$

REMARK 2.4.1 *Control laws of the form of equation (2.36) will be referred to as explicit control laws as they are analytical equations for the control input which rely on knowledge of the plant structure. For example, in equation (2.36) the control law is generated by incorporating the neural estimate of the nonlinear function in the output and the neural estimate of the inverse of the nonlinear function in control into a difference equation whose form is governed by knowledge that the plant output is simply the sum of the outputs from these nonlinear functions.*

This example highlights how the approach presented in [169] relies on the assumption that any separability of control and output data is known and can be utilised and that the inverse of a function exists and it can be identified.

Another neural adaptive control proposed by F-C. Chen [28, 30, 31] uses neural networks to approximate the Lie derivatives of a plant of the form (1.18) and then a feedback linearizing control is generated to ensure the output of the plant tracks the desired signal. More specifically, Chen and his co-authors consider a SISO γ-th order system with a normal form[2] as follows:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_\gamma &= f(x) + g(x)u \\ y &= h(x) = x_1 \end{aligned} \tag{2.37}$$

where $x \in \mathbf{R}^\gamma$ and $f(.) = L_f^\gamma h(x)$ and $g(.) = L_g L_f^{\gamma-1} h(x)$ are smooth functions, where $h(x) = x_1$ in [31]. As shown in Chapter 1, differentiating the output y with respect to

[2]Refer to Chapter 1 for a detailed discussion of the normal form of a nonlinear system and input-output linearisation techniques.

time γ times leads to the input-output form of the above normal system

$$y^\gamma = L_f^\gamma h(x) + L_g L_f^{\gamma-1} h(x) u \tag{2.38}$$

for which the control

$$u = \frac{1}{L_g L_f^{\gamma-1} h(x)} (-L_f^\gamma h(x) + r) \tag{2.39}$$

results in the simple input-output linearised system

$$y^\gamma = v \tag{2.40}$$

The aim of the control approach presented in [31] is for the plant output y to track a reference trajectory y_m. In [31] the the reference input is defined as

$$r = y_m^{(\gamma)} - k_{\gamma-1} e_\gamma - \ldots - k_0 e_1 \tag{2.41}$$

where the constants k_i are chosen such that the polynomial

$$K(p) = p^\gamma + k_{\gamma-1} p^{\gamma-1} + \ldots + k_1 p + k_0 \tag{2.42}$$

has all of its roots strictly in the left half plane. Thus the control input (2.39) is shown to result in error equation

$$e^\gamma + k_{\gamma-1} e^{\gamma-1} + \ldots + k_1 e \tag{2.43}$$

where e is the tracking error. A local convergence theorem for the above tracking error is then provided in [30, 31] which relies on the assumptions that (1) the Lie derivative $L_f^\gamma h(x)$ vanishes at the origin, (2) $L_g L_f^{\gamma-1} h(x)$ is bounded away from zero, (3) the system is minimum phase, and (4) that the Lie derivatives can be exactly represented by a multi-layered neural network without bias weights.

The above two schemes are limited by the fact that an artificial neural network can only approximate the plant over a specified finite region. Recently, Sanner and Slotine [202, 203, 204] have proposed adding a sliding control term to the control law. The sliding control term takes over from the neural adaptive component when the state moves outside of the region on which the network approximation is valid. When the state of the plant returns to the region where the neural network provides a good approximation, then the sliding control is turned off. The control law thus has a dual character acting as either a sliding controller or a neural controller depending on the position of the plant state. To avoid control chattering (high control activity) occurring along the boundary between the two types of operation, a modulation function is incorporated to control the degree to which each component contributes to the control law. This ensures a smooth transition between the adaptive and sliding control strategy.

A similar scheme is adopted by Tzirkel-Hancock and Fallside in [235]. A sliding control term is used in conjunction with a neural control approach proposed in [234]. As with [202, 203, 204], the sliding control compensates for uncertainties in the plant nonlinearities outside the state region in which the network approximation is used. The neural control method is based on neural network approximation of the Lie derivatives of the continuous-time nonlinear system. Adaptation of the network weights is based on Lyapunov stability results. An input-output linearisation scheme is used to make the system output track a desired reference signal.

Polycarpou and Ioannou [186] similarly consider a stability based network adaptation scheme. However, the neural networks are embedded in a certainty equivalence based control scheme in which the neural networks are used to model the nonlinear functions $f(.)$ and $g(.)$ of a first order continuous-time system of the form (1.18) rather than their Lie derivatives. The weights of the neural networks are adapted by laws derived from Lyapunov stability analysis and the use of a projection algorithm. The control objective of this scheme is to force the plant output to follow a reference model. As with [202, 235], a sliding controller term is used in conjunction with the certainty equivalence control law. It can be easily shown that this approach is equivalent to the approach described in [234, 235], except that in [186], the output variable is the state variable and the unity relative degree is considered.

The schemes presented by Chen and his co-authors, Tzirkel-Hancock and Fallside and Polycarpou and Ioannou are essentially based on feedback/input-output linearisation approaches, and as a result are subject to the restrictions imposed on such schemes (as discussed in Chapter 1). In particular, they are restricted to a system which is affine in control. Although, such systems are fairly general, they are not universal and thus these approaches can only be used on a limited class of system. Furthermore, the approaches presented in [31, 186, 234, 235] deal with continuous-time nonlinear systems only[3]. One of the problems with discrete-time nonlinear systems is the complexities which arise when discretising a continuous-time nonlinear system. In particular, the discretisation of a continuous affine-in-control nonlinear system results in a non-affine discrete-time system which are difficult to control using geometric approaches such as feedback linearisation.

2.5 Conclusions

This chapter highlights some of the major issues and concepts associated with the use of artificial neural networks in the control of nonlinear dynamical systems.

For the sake of completeness, several basic concepts of artificial neural networks are presented in Section 2.2. In particular, a brief exposition of some of the biological foundations and motivations of artificial neural networks is provided. A definition of an artificial neural network is given and the main network topologies and learning schemes are discussed. As multilayer perceptrons trained with the backpropagation learning algorithm are used in the neural controller presented in subsequent chapters of this thesis, these concepts are also discussed in detail. A major factor in the use of artificial neural networks in control is their ability to approximate any continuous nonlinear function. Therefore, this ability is examined in some depth. Implementation issues, such as the number of hidden layers, the number of processing nodes and the type of nonlinear activation function are also discussed.

One area of interest with artificial neural networks is their ability to model nonlinear dynamical systems. This issue is of particular importance from a control context as such systems are usually considered. Therefore, a number of artificial neural network architectures for dynamic systems modelling are considered in Section 2.3. In particular, the use of time delay neural networks for the modelling of nonlinear dynamic systems is investigated thoroughly. The problems associated with incorrectly choosing the number of taps in the tapped delay line used in these networks are discussed. The use of correlation tests

[3]An input-output linearisation approach for discrete-time affine-in-control nonlinear systems is provided in [28, 30]

to determine the validity of a given neural network model is investigated. These tests are shown to be able to distinguish between a correctly parameterised neural network model for a discrete-time nonlinear system and an incorrectly parameterised model, such as an over-parameterised or under-parameterised model.

Finally, a detailed discussion of existing neural control schemes is provided. In particular, the four major classes of schemes, namely supervised control, inverse control, adaptive critics and neural adaptive control are considered. A critical review of the major neural adaptive control papers is presented. In particular, some of the limitations and strengths of these schemes are discussed. This will facilitate the comparison of the neural control scheme presented in this thesis with the existing approaches.

The neural adaptive control approaches reviewed in this chapter are, in general, limited by the type of system that they can handle. In particular, the approaches proposed by Chen *et al.*, Tzirkel-Hancock and Fallside and Polycarpou and Ioannou are essentially feedback/input-output linearisation approaches which require that the system is affine (linear) in control. Furthermore, the approach presented by Narendra and Parthasarathy utilises knowledge of the separability of the control and output terms to design the explicit control law. Plants in which such knowledge cannot be utilised are considered difficult to control. In the next chapter, a neural adaptive control procedure will be presented which provides a unified approach to the control of a general class of nonlinear system. More specifically, the approach will be shown to be able to deal with nonlinear systems which are non-affine in control and where the control is heavily embedded within the nonlinearities of the system, including systems where the control and output terms are not separable.

Chapter 3

Adaptive Neural Controller

3.1 Introduction and Overview

A great deal of research has recently been conducted into the development of neural network based control schemes. As shown in the previous chapter, these schemes can be broadly categorised into four major approaches. In the control community, arguably the most commonly used approach is neural adaptive control. This is primarily because in this approach neural networks are used within traditional and well understood control frameworks such as model reference control. The analysis of the major neural adaptive control schemes provided in Chapter 2 shows that most of the approaches developed so far are limited to certain classes of systems or are subject to a number of assumptions about the system.

The primary motivation for the neural controller design presented in this chapter is to relax the assumptions made in the paper by Narendra and Parthasarathy [169], namely that (*i*) control and output terms are separable, (*ii*) the separate nonlinear functions $f(.)$ and $g(.)$ can be independently identified, and (*iii*) the inverse of the nonlinear functions $f(.)$ and $g(.)$ can be *explicitly* identified. However, to ensure the existence of a control, it is assumed that the plants are completely controllable[1] and observable[2]. The end

[1]In this context the notion of controllability means that a control input u exists which can transfer the systems from one state to another in a finite number of states.

[2]Observability is concerned with the problem of determining the state of a dynamic system from

result is that a unified method of the control of nonlinear systems has been developed, in which plants belonging to Models I–IV[3] are treated exactly the same. This is achieved because the plant is treated as a "black box"[4] with little *a priori* knowledge of the system required. In fact, in terms of the plant structure and dynamics, the only *a priori* knowledge required is the number of delayed values of plant input and plant output, and the order and relative degree of the system. This knowledge is necessary to determine the number of taps required for the time delay neural networks used in the approach. Hence because of the limited amount of system information required in this approach, a wide range of nonlinear systems can be handled, including those in which the output and control terms are not necessarily separable.

Time delay neural networks are employed in the approach to emulate the plant dynamics and to synthesise the reference model control. The neural controller is designed to ensure the output from a nonlinear plant is made to track the output of a stable reference model which represents the desired closed-loop dynamics of the system.

Model reference adaptive control is chosen as the control framework into which neural networks are incorporated because of its suitability for use in a supervised learning neural network scheme. The reference model provides the desired output which is used by the supervised learning scheme to generate an output error to modify the neural network weights. Furthermore, MRAC allows the system designer to incorporate the desired closed-loop dynamics of the system into a structured form, i.e., the reference model. This provides an additional degree of freedom to the system designer as the structure of the reference model can be altered to meet different performance specifications.

The basic structure of the scheme presented here is described in detail in the next section. The problem to be addressed is formulated for a general multi-input multi-output system. The implementation of the scheme in an off-line and on-line environment is also discussed. Simulation examples are then provided to highlight the effectiveness of the scheme. The limitations of the scheme are then discussed which gives rise to the solutions presented

observations of the output and control vectors in a finite number of sampling periods.

[3]Refer to Chapter 2 for a definition of the nonlinear systems belonging to Model I - Model IV.

[4]In the same way that transfer functions provide a generic input-output representation for linear black box models, neural networks potentially provide a generic representation for nonlinear black box models.

in the next chapter.

3.2 Basic Structure of the Control Scheme

The neural adaptive control method presented in this chapter is a combination of the model reference neural adaptive control approach of Narendra and Parthasarathy [169] and the forward modelling approach of Jordan [95].

In the approach presented in Narendra's paper, multilayer neural networks are used to identify the unknown nonlinearities of the system and then the neural estimates are used in an explicit control law to ensure that the output of the plant tracks the output of the reference model. As shown in the example given in Chapter 2, this approach utilises any partial linearity or separability in the system to design the control.

In the forward modelling approach of Jordan, and similarly the specialised learning approach of Psaltis *et al.* [189], a neural network is used as the controller and it is trained to ensure that the output of the plant tracks a desired response which is the input to the controller. Therefore, this approach is an inverse control scheme. As discussed in the previous chapter, a problem with forward modelling schemes is that the plant is located between the output of the controller network and the output error. This provides a difficulty in obtaining the controller error and thus deriving a weight update equation for the controller neural network. It is shown in [189] that in order to obtain the appropriate error for the controller output, it is necessary to find the Jacobian of the plant. In the forward modelling approach of Jordan [95], it is shown that the plant Jacobian can be obtained from a neural network which identifies the plant.

Hence, the aim of the approach considered here is to design a control such that the output of the nonlinear system tracks the output of the reference model as suggested by Narendra and Parthasarathy [169]. However, unlike [169], a neural controller is trained to achieve the desired control rather than using an explicit control law. As in Jordan's scheme [95], the weights of the controller neural network are modified by a weight update equation which utilises the plant Jacobian obtained from a neural network model of the plant. However, unlike [95, 171, 189], the neural controller does not learn an inverse model of the the system. Instead it can be thought of as learning a "detuned" inverse of the system with the amount of "detuning" dependent upon the reference model chosen. The approach further differs from the scheme presented by Narendra and Parthasarathy [169] in that the neural network emulator models the entire input-output dynamics of the system, rather than just the nonlinearities in the output or control terms.

As the neural adaptive control scheme considered here is based on the well established model reference adaptive control methodology, it is subject to the design requirements of this scheme. Firstly, the choice of the reference model is extremely important to achieve good control. The reference model is chosen to reflect the desired closed-loop dynamics of the system, usually in terms of performance specifications such as rise time, settling time, overshoot or frequency domain characteristics. However, the structure of the reference model is also constrained by its order and relative degree, given *a priori* knowledge of the order of the plant. These constraints are placed to ensure that the desired response is achievable by the plant. Generally, the reference model is chosen to be linear and stable and its order is chosen to be equal or greater than the order of the plant to be controlled.

In an MRAC scheme, the controller is designed to generate control variables such that the output of the plant tracks the reference model output for a given bounded reference input. This is achieved by adjusting the parameters of the controller via an adjustment mechanism so as to minimise the error between the reference model and the system. In the neural network MRAC scheme considered here, the parameters of the controller can be considered to be the weights of the controller neural network and the adjustment mechanism is the weight update equation. For this neural network based model reference control problem, the above scheme can be mathematically expressed for a general multi-input multi-output system as follows:

Consider a plant governed by the following nonlinear difference equation

$$\begin{aligned} y_p(k+1) &= f(y_p(k), \cdots, y_p(k-l+1); u(k), \cdots, u(k-m+1)) && (3.1) \\ y_p(0) &= y_{p_0} \qquad \forall k \in \mathbf{N} && (3.2) \end{aligned}$$

where $y_p \in \mathbf{R}^n$ is the output vector, $u \in \mathbf{R}^r$ is the control vector, $f : \mathbf{R}^{n \times l} \times \mathbf{R}^{r \times m} \longrightarrow \mathbf{R}^n$ is a smooth nonlinear function, y_{p_0} is the initial output vector, k is the time index, $\mathbf{N}$ is the set of natural numbers and m and l are the number of delayed values of plant input and plant output, respectively.

The control strategy is to find a feasible control input

$$u(k) = g(y_p(k), \cdots, y_p(k-l+1); u(k-1), \cdots, u(k-m+1); r(k); W_c) \tag{3.3}$$

where $g : \mathbf{R}^{n \times l} \times \mathbf{R}^{r \times m} \longrightarrow \mathbf{R}^r$, is a neural network parameterised by the weights W_c, such that for a stable reference model governed by

$$\begin{aligned} y_m(k+1) &= f_m(y_m(k), y_m(k-1), \cdots, y_m(k-d+1); r(k)) \qquad (3.4)\\ y_m(0) &= y_{m_0} \qquad (3.5) \end{aligned}$$

where $y_m \in \mathbf{R}^n$ is the reference model output vector, $r \in \mathbf{R}^n$ is the reference input, $f_m : \mathbf{R}^{n \times d} \times \mathbf{R}^r \longrightarrow \mathbf{R}^n$ is usually a linear function, $d \geq l$ and y_{m_0} is a given initial output vector for the reference model. In this case, the absolute tracking error satisfies the following relationship

$$\begin{aligned} \|e_T(k+1)\| &= \|y_m(k+1) - y_p(k+1)\| \leq \varepsilon_T \qquad (3.6)\\ e_T(0) &= e_{T_0} = y_{m_0} - y_{p_0} \qquad (3.7) \end{aligned}$$

where $\varepsilon_T \geq 0$ is a predefined tracking error tolerance, e_{T_0} is the initial tracking error vector and $\|.\|$ is the vector norm.

REMARK 3.2.1 *The above scheme is formulated for discrete-time nonlinear systems represented by difference equations of the form (3.1). This equation represents the dynamical relationship between the input and output of the plant and is often referred to as the input-output equation or observer's equation. However, many nonlinear systems are expressed in the state space form*

$$\begin{aligned} x(k+1) &= f^o(x(k), u(k))\\ y(k+1) &= h(x(k)) \qquad (3.8) \end{aligned}$$

where $x \in \mathbf{R}^p$ are the states of the system which may be be observable and $f^o : \mathbf{R}^p \times \mathbf{R}^r \longrightarrow \mathbf{R}^n$ and $h : \mathbf{R}^p \longrightarrow \mathbf{R}^n$ are unknown smooth functions. Constructing adaptive controllers for this form of general nonlinear system is a more difficult task. Therefore, systems which can be expressed in the observer's equation form are desirable. Formal methods of *constructing the observer's equation form of the nonlinear system from the state space representation have been proposed by Goh and Noakes [57]. Hence, the observer's equation will be used in this thesis to describe the nonlinear systems, bearing in mind that this form can be obtained from the state space representation.*

Whilst the inverses of operators do not need to be explicitly modelled in this approach, the existence of inverse operators still needs to be assumed in order to ensure the existence of a control vector $u(k)$ to achieve the desired objective. However, rather than the function $g(.)$ representing the the true inverse of equation (3.1), i.e., $f^{-1}(.)$, as is the case in the inverse control approach [95, 189], in this MRAC scheme it represents a "detuned" inverse of the plant. Furthermore, it is assumed that for any set of values $u(k)$ in a compact region of $\mathbf{R}^n$, a solution to the problem does indeed exist.

Apart from the assumptions about the controllability and observability of the systems under study, the following definitions and assumptions are also central to the neural adaptive control scheme presented in this chapter.

DEFINITION 3.2.1 *A plant of the form (3.1) has a* <u>*relative degree*</u> $\{\gamma_1, \gamma_2, \ldots, \gamma_r\}$, *where* $\gamma_i \in \mathbf{N}$, $i = 1, \ldots, r$ *if*

$$\frac{\partial y_{p_i}(k+1)}{\partial u_j(k-\kappa_{ij}+1)} = 0 \qquad \forall \kappa_{ij} < \gamma_i \tag{3.9}$$

where $i = 1, \ldots, n$, $j = 1, \ldots, r$ *and there exists at least one* j *such that*

$$\frac{\partial y_{p_i}(k+1)}{\partial u_j(k-\gamma_i+1)} \neq 0 \tag{3.10}$$

for all points in the neighbourhood of the reference point $v^0 = (y^0, y^0 \ldots, y^0, u^0, u^0, \ldots, u^0)$.

ASSUMPTION 3.2.1 *For the sake of simplicity, all of the plants considered in this thesis are assumed to be of relative degree unity,* $\gamma_1 = \gamma_2 = \ldots = \gamma_r = 1$, *i.e., the input at* k *affects the output at* $k+1$. *However, after suitable modifications in the control strategy, a similar procedure can be adopted when the relative degree is greater than unity.*

DEFINITION 3.2.2 *The system is* <u>*minimum phase*</u> *if the zero dynamics*

$$0 = f(0, 0, \ldots, 0; u(k), u(k-1), \ldots, u(k-m+1)) \tag{3.11}$$

where $0 \in \mathbf{R}^n$ *and* $u \in \mathbf{R}^r$*, is asymptotically stable to the reference point* u^0 *when the plant output is kept at* 0*, i.e., no unbounded inputs lie in the null space of the operator representing the plant. The zero dynamics represent the internal dynamics of the system when the system output is kept at zero by the input.*

ASSUMPTION 3.2.2 *Unless otherwise specified, all of the plants are assumed to be minimum phase.*

As the nonlinear function $f(.)$ is unknown, the plant needs to be identified. This is achieved by training a neural network parameterised by the weights W_I to emulate the plant dynamics such that the predicted plant output is given by

$$\hat{y}_p(k+1) = \hat{f}(y_p(k), \cdots, y_p(k-l+1); u(k), \cdots, u(k-m+1); W_I) \quad (3.12)$$
$$\hat{y}_p(0) = 0 \quad (3.13)$$

and the absolute identification error $\|y_p(k+1) - \hat{y}_p(k+1)\|$ is finite, i.e.,

$$\|e_I(k+1)\| = \|y_p(k+1) - \hat{y}_p(k+1)\| \leq \varepsilon_I \quad (3.14)$$
$$e_I(0) = e_{I_0} = y_{p0} \quad (3.15)$$

where $\varepsilon_I \geq 0$ is a predefined error tolerance and e_{I_0} is the initial identification error vector.

REMARK 3.2.2 *It has been shown by Cybenko [40], Funahashi [55] and Hornik et al. [74] that multi-layered neural networks with an arbitrarily large number of nodes in the hidden layers can approximate any real valued continuous function* $f(.)$ *on a compact subset of* $\mathbf{R}^p = \mathbf{R}^{n \times l} \times \mathbf{R}^{r \times m}$ *to a desired degree of accuracy, i.e., equation (3.14) is satisfied.*

The identification model used here is known as a series-parallel identification model as the output of the plant (y_p) rather than the neural estimate of the plant output ($\hat{y}_p$) is fed back into the neural network approximating the plant dynamics. Therefore the identification model has the form given in equation (3.12). The alternative identification model often used is a parallel model. In the parallel model the previous outputs of the identification model are fed back to generate the current model output. A parallel model is governed by the following difference equation

$$\hat{y}_p(k+1) = \hat{f}(\hat{y}_p(k), \hat{y}_p(k-1), \cdots, \hat{y}_p(k-l+1); u(k), u(k-1), \cdots, u(k-m+1)) \quad (3.16)$$

The series-parallel model has several advantages over the parallel model. Firstly, if the plant is bounded-input bounded-output (BIBO) stable[5], then all of the inputs to the neural network emulator are bounded. The same cannot be said for a parallel model as the stability of the neural network identification model cannot be guaranteed. Hence if a parallel model is used, it cannot be guaranteed that the model parameters (weights) will converge or that the identification error will decrease to zero. Furthermore, static backpropagation can be used to adjust the weights of the network as no feedback loop exists in the model.

The control procedure presented here is often referred to as an indirect control approach [166, 169] because an identification model (NN emulator) is used to identify the input-output behaviour of the plant. Using this identification model, the controller parameters are then adjusted. However, unlike the traditional definition of an indirect control scheme given in Chapter 1, it is difficult to derive an expression for the controller parameters in terms of the parameters of the identification model. This is because a neural network is a nonparametric identifier for which it is not possible to derive a simple relationship between the learned weights of the network and the parameters of the plant. Hence the definition of an indirect control procedure in the context of a neural network control scheme is slightly different to the traditional case. Due to the nonlinear nature of both the plant and controller, it has not been until very recently that methods for directly controlling the parameters of the controller have been derived [186, 204, 235]. These schemes do not employ neural networks in an identification role.

Theorem 2.2.1 (Funahashi [55]) and the equivalent theorems provided by Cybenko [40] and Hornik *et al.* [74] are, in fact, existence theorems and thus provide no information about the number of nodes required to approximate a nonlinear function, the required values of the weights nor how to choose the weights. Therefore a number of factors are

[5]Most of the simulation examples considered are BIBO stable. However, this is not the case for the marginally stable systems considered in this thesis.

taken into account in deciding the structure and type of neural networks used in this approach.

Firstly, because of the dynamic nature of the systems under study, the neural networks used in this approach are time delay neural networks with output feedback of the form shown in Figure 2.8. The number of taps in the tapped delay line is determined by the number of delayed values of plant input (m) and plant output (l). The number of inputs to the multi-layer perceptron is also dependent upon the output vector dimension (n) and control vector dimension (r). Therefore the total number of inputs to the MLP is ($n \times k + m \times r$). This information represents the only necessary *a priori* knowledge about the structure of the system.

Furthermore, as multi-layered perceptrons are used in this approach, the backpropagation algorithm is used to train networks and their structure is chosen based on the considerations presented in the previous chapter. In particular, in accordance with the discussion presented by Sontag [223], 2 hidden layers are utilised. The number of hidden layer nodes is chosen using heuristic rules based upon past experience.

The problem formulated above can be addressed using either an off-line approach, in which the controller and emulator neural networks are trained separately and off-line, and an on-line approach, in which the neural networks are trained simultaneously whilst the system is running.

3.2.1 Off-line Approach

The off-line approach to the neural network based model reference control problem is essentially a two step procedure. The first is the identification phase and it involves training a neural network using the backpropagation learning algorithm to approximate the input-output dynamics of the plant to a sufficient degree of accuracy. This utilises the functional approximation results discussed in Remark 3.2.2. A block diagram of the identification process is shown in Figure 3.1a.

The emulator is typically trained with a random input signal $u(k)$ uniformly distributed in the input space $\mathbf{R}^r$ until the mean square identification error is less than a predefined

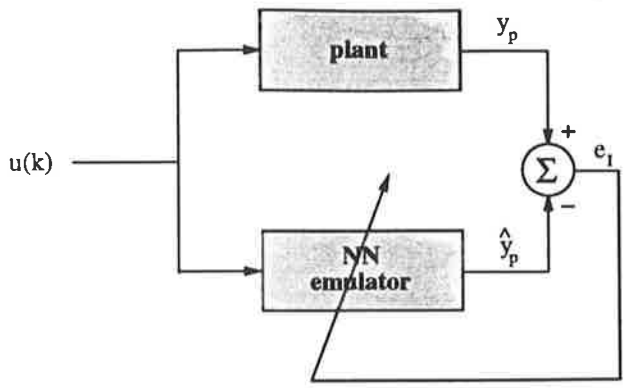


Figure 3.1a: Identification stage for the off-line approach

error tolerance. Such an input signal ensures that the training set is representative of the entire class of possible inputs for the system. This enables the system to respond in the desired fashion even when it is subject to an input not in the training set. This input signal is often referred to as persistently exciting.

DEFINITION 3.2.3 *A vector U is persistently exciting if there exist positive constants t_0, T_0, α such that*

$$\int_t^{t+T_0} U(\tau)U^T(\tau)d\tau \geq \alpha I \quad \forall t \geq t_0 \tag{3.17}$$

Once the plant has been learned sufficiently well, the neural controller is then trained so that the plant output tracks the desired output from the reference model. A block diagram of the controller learning stage is given in Figure 3.1b. As with the identification stage, the controller is trained with a random reference input signal $r(k)$ uniformly distributed in the input space $\mathbf{R}^r$ (i.e., a persistently exciting signal) until the mean square tracking error is less than a predefined tolerance. The weight vector of the controller neural network is modified by a weight update equation which utilises the plant Jacobian. To achieve this, the tracking error is backpropagated through the emulator neural network as is shown in Figure 3.1b. A detailed account of the controller weights update procedure is given in Section 3.2.3. As with the identification stage, this process is undertaken off-line.

Once the controller has been trained sufficiently well off-line, the training process is discontinued and the system is re-connected. The system is then subjected to the oper-

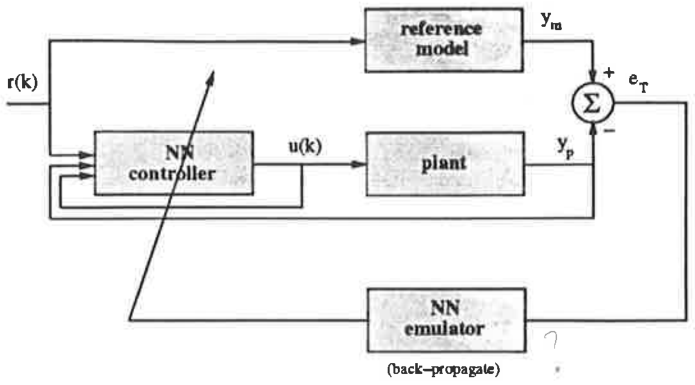


Figure 3.1b: Controller training stage for the off-line process

ational reference input, which is assumed to be part of the the training sequence of the controller neural network. If this is not the case then, provided the reference input is still assumed to be part of the input space of the network, an input-output relationship is extrapolated from the training examples to achieve the desired control. This is often referred to as the generalisation capability of a neural network. The training process is disconnected during the control stage to ensure that the network weights continue to map the input-output relationship over the entire input space, rather than only a localised subset of the input space represented by the actual operating conditions.

3.2.2 On-line Approach

In the on-line approach to the model reference control of a nonlinear system using neural networks shown in Figure 3.2, the plant identification and control synthesis are done simultaneously and in a closed-loop environment. This removes the need to separately identify the system off-line, which can often be a very time consuming, costly and impractical task particularly if it involves taking a complex system off-line for a long period of time. One of the main disadvantages with this approach is that because the controller and emulator neural networks are trained whilst the system is operating under actual working conditions, the training is very localised. This is because the actual input signals represent a small subset of the input space of the system and thus networks are trained over this smaller, localised input space. However, the on-line approach relies on the inherent adaptivity of the neural networks and thus any changes in the operational input space results in a modification of the network weights. It is therefore a truly adaptive system.

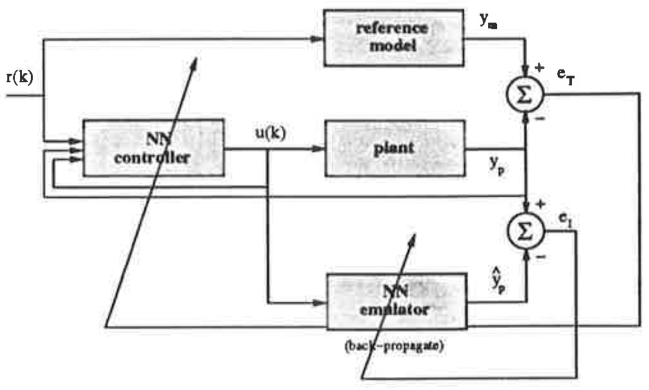


Figure 3.2: Block diagram of the on-line identification and control scheme

Such an approach is particularly important for systems which cannot be identified using open-loop techniques, such as marginally stable systems[6]. These systems often arise in practice. Examples are ship steering [127], missile guidance [78] and the position control of a servo system [10].

As with the off-line approach, the backpropagation learning algorithm is used to modify the weights of the controller and emulator networks. Furthermore, the method proposed by Jordan is also used to determine the plant Jacobian. However, as both the controller and emulator neural networks are trained on-line, the plant Jacobian approximate will initially be poor. As a result, the tracking performance for the initial few samples will also be poor. In the situation that the tracking error becomes so large that the plant leaves the usual region of operation, Tzirkel-Hancock and Fallside [234, 235] and Sanner and Slotine [202, 203, 204] have suggested the use of a sliding control to force the plant back into this region. An alternative solution would be to employ a short period of off-line identification prior to the on-line operation. This would provide the neural network emulator with some initial knowledge about the system and thus the Jacobian approximate would initially be more accurate. For the systems considered in the simulation studies undertaken in this research, the above limitation with the scheme did not prove to be a problem. However, currently there are no theoretical results to always guarantee this.

[6]A discussion of the definition of a marginal stable nonlinear system is provided in Example 3.3.11.

3.2.3 Weight Update Equations

For both the off-line approach and the on-line approach the emulator neural network is trained using the standard backpropagation learning algorithm. The measurable identification error, $e_I(k+1) = y_p(k+1) - \hat{y}_p(k+1)$, is backpropagated to modify the weight of the network, where $y_p, \hat{y}_p, e_I \in \mathbf{R}^n$.

However, in order to train the controller using a supervised learning scheme such as backpropagation, the error $u_d(k) - u(k)$ in the controller output is required. The term $u_d(k)$ is the desired control which would produce $y_m(k+1)$ if applied to the plant. However, as $u_d(k) \in \mathbf{R}^r$ is not known, an approximation of the controller error denoted by e_c, must be generated. A method for generating such an error is derived for a general multi-input multi-output system below.

The cost function that is to be minimised is given by

$$E = \frac{1}{2}\sum_{k=0}^{N-1}(y_m(k+1) - y_p(k+1))^T(y_m(k+1) - y_p(k+1)) \tag{3.18}$$

where N is the number of samples, $y_m = [y_{m_1}, \ldots, y_{m_n}]^T$ and $y_p = [y_{p_1}, \ldots, y_{p_n}]^T$.

E is minimised by performing a gradient descent in $E(W_c(k))$, where $W_c(k)$ are the controller weights:

$$\begin{aligned}\frac{\partial E}{\partial W_c(k)} &= \left[\left(\frac{\partial E}{\partial u(k)}\right)^T \left(\frac{\partial u(k)}{\partial W_c(k)}\right)^T\right]^T \\ &= \frac{\partial u(k)}{\partial W_c(k)}\frac{\partial E}{\partial u(k)}\end{aligned} \tag{3.19}$$

where $u = [u_1, u_2, \ldots, u_r]^T$.

By performing a gradient descent in $E(u(k))$, the term $\frac{\partial E}{\partial u(k)}$ can be computed:

$$\begin{aligned}\frac{\partial E}{\partial u(k)} &= \left[\left(\frac{\partial E}{\partial y_p(k+1)}\right)^T \left(\frac{\partial y_p(k+1)}{\partial u(k)}\right)^T\right]^T \\ &= \frac{\partial y_p(k+1)}{\partial u(k)}\frac{\partial E}{\partial y_p(k+1)}\end{aligned} \tag{3.20}$$

Therefore, the gradient descent in $E(W_c(k))$ is given by

$$\frac{\partial E}{\partial W_c(k)} = \frac{\partial u(k)}{\partial W_c(k)}\frac{\partial y_p(k+1)}{\partial u(k)}\frac{\partial E}{\partial y_p(k+1)} \tag{3.21}$$

where $\frac{\partial E}{\partial W_c(k)}$ is a $p \times 1$ vector[7], $\frac{\partial E}{\partial u(k)}$ is a $r \times 1$ vector, $\frac{\partial u(k)}{\partial W_c(k)}$ is a $p \times r$ matrix, $\frac{\partial y_p(k+1)}{\partial u(k)}$ is a $r \times n$ matrix, and $\frac{\partial E}{\partial y_p(k+1)}$ is a $n \times 1$ vector.

Equation (3.21) involves the Jacobian of the plant $\frac{\partial y_p(k+1)}{\partial u(k)}$, which is unknown. By backpropagation through the neural network which identifies the plant, an approximate of the Jacobian $\frac{\partial \hat{y}_p(k+1)}{\partial u(k)}$, can be obtained.

The term $\frac{\partial E}{\partial u(k)}$ represents the change in the cost function as a result of a change in the controller neural network output. As shown in [95], it can be obtained by backpropagating the tracking error through to the inputs of the emulator neural network corresponding to the control u. As these inputs are the outputs of the controller neural network, the term $\frac{\partial E}{\partial u(k)}$ is related to the error in the output of the controller neural network. Since the backpropagation algorithm performs a search along the negative gradient of the error surface, the controller error is therefore defined as $e_c(k) = -\frac{\partial E}{\partial u(k)}$. Hence, the weight update equation is given by

$$\begin{aligned} W_c(k+1) &= W_c(k) - \eta \frac{\partial E}{\partial W_c(k)} \\ &= W_c(k) - \eta \frac{\partial u(k)}{\partial W_c(k)} \frac{\partial E}{\partial u(k)} \\ W_c(k+1) &= W_c(k) + \eta \frac{\partial u(k)}{\partial W_c(k)} e_c(k) \end{aligned} \tag{3.22}$$

where the controller error is an r-dimensional vector $e_c = [e_{c_1}, e_{c_2}, \ldots, e_{c_r}]$ given by

$$\begin{aligned} e_c(k) &= -\frac{\partial \hat{y}_p(k+1)}{\partial u(k)} \frac{\partial E}{\partial \hat{y}_p(k+1)} \\ &= -\begin{bmatrix} \frac{\partial \hat{y}_{p1}}{\partial u_1} & \cdots & \frac{\partial \hat{y}_{pn}}{\partial u_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_{p1}}{\partial u_r} & \cdots & \frac{\partial \hat{y}_{pn}}{\partial u_r} \end{bmatrix} \begin{bmatrix} -(y_{m_1} - y_{p_1}) \\ \vdots \\ -(y_{m_n} - y_{p_n}) \end{bmatrix} \\ &= \begin{bmatrix} (y_{m_1} - y_{p_1})\frac{\partial \hat{y}_{p1}}{\partial u_1} + (y_{m_2} - y_{p_2})\frac{\partial \hat{y}_{p2}}{\partial u_1} + \cdots + (y_{m_n} - y_{p_n})\frac{\partial \hat{y}_{pn}}{\partial u_1} \\ (y_{m_1} - y_{p_1})\frac{\partial \hat{y}_{p1}}{\partial u_2} + (y_{m_2} - y_{p_2})\frac{\partial \hat{y}_{p2}}{\partial u_2} + \cdots + (y_{m_n} - y_{p_n})\frac{\partial \hat{y}_{pn}}{\partial u_2} \\ \vdots \\ (y_{m_1} - y_{p_1})\frac{\partial \hat{y}_{p1}}{\partial u_r} + (y_{m_2} - y_{p_2})\frac{\partial \hat{y}_{p2}}{\partial u_r} + \cdots + (y_{m_n} - y_{p_n})\frac{\partial \hat{y}_{pn}}{\partial u_r} \end{bmatrix} \end{aligned} \tag{3.23}$$

[7]The dimension p is the number of output layer weights = the number of output nodes $\times$ the number of nodes in the last hidden layer.

Therefore for the g-th controller output, the network output error is

$$e_{c_g}(k) = (y_{m_1} - y_{p_1})\frac{\partial \hat{y}_{p1}}{\partial u_g} + (y_{m_2} - y_{p_2})\frac{\partial \hat{y}_{p2}}{\partial u_g} + \cdots + (y_{m_n} - y_{p_n})\frac{\partial \hat{y}_{p_n}}{\partial u_g} \tag{3.24}$$

where the time index k is neglected for the sake of simplicity.

The above controller error can be derived by backpropagating the tracking error through the neural network which identifies the plant. In the following derivation, the results are expressed in terms of the neural network system equations for a multi-input multi-output system and can therefore be regarded as an extension of the work of Jordan [95].

The derivation of the controller error is as follows:

Consider a two layered neural network[8] with a inputs, b hidden units and n outputs. The state vector $y_p = [y_{p_1}, y_{p_2}, \ldots, y_{p_n}]^T$ is of dimension n and the control vector $u = [u_1, u_2, \ldots, u_r]^T$ is of dimension r, where $r \leq a$. The neural network output vector is denoted $\hat{y}_p = [\hat{y}_{p_1}, \hat{y}_{p_2}, \ldots, \hat{y}_{p_n}]^T$. The individual neural network outputs are given by

$$\begin{aligned}
\hat{y}_{p_1}(k+1) &= g(\sum_{j=1}^{b} W^2_{1j} g(\sum_{k=1}^{a} W^1_{jk} x_k)) \\
\hat{y}_{p_2}(k+1) &= g(\sum_{j=1}^{b} W^2_{2j} g(\sum_{k=1}^{a} W^1_{jk} x_k)) \\
&\vdots \\
\hat{y}_{p_n}(k+1) &= g(\sum_{j=1}^{b} W^2_{nj} g(\sum_{k=1}^{a} W^1_{jk} x_k))
\end{aligned} \tag{3.25}$$

Let the gth control input $u_g(k)$ be the lth network input, x_l. Thus

$$\begin{aligned}
\frac{\partial \hat{y}_{p_1}(k+1)}{\partial u_g(k)} &= \frac{\partial \hat{y}_{p_1}(k+1)}{\partial x_l} = g'(\sum_{j=1}^{b} W^2_{1j} g(\sum_{k=1}^{a} W^1_{jk} x_k)) \sum_{j=1}^{b} (W^2_{1j} g(\sum_{k=1}^{a} W^1_{jk} x_k) W^1_{jl}) \\
&\vdots \\
\frac{\partial \hat{y}_{p_n}(k+1)}{\partial u_g(k)} &= g'(\sum_{j=1}^{b} W^2_{nj} g(\sum_{k=1}^{a} W^1_{jk} x_k)) \sum_{j=1}^{b} (W^2_{nj} g(\sum_{k=1}^{a} W^1_{jk} x_k) W^1_{jl})
\end{aligned} \tag{3.26}$$

Now consider the effective output error δ^2

$$\delta^2_1 = g'(\sum_{j=1}^{b} W^2_{1j} g(\sum_{k=1}^{a} W^1_{jk} x_k))(y_{m_1}(k+1) - y_{p_1}(k+1))$$

[8]A two layered neural network is considered in the above proof for space reasons. However, it is a straightforward task to extend the proof to an arbitrary number of layers.

$$\vdots$$

$$\delta_n^2 = g'(\sum_{j=1}^{b} W_{nj}^2 g(\sum_{k=1}^{a} W_{jk}^1 x_k))(y_{m_n}(k+1) - y_{p_n}(k+1)) \tag{3.27}$$

where $y_m = [y_{m_1}, y_{m_2}, \ldots, y_{m_n}]^T$ is the n dimensional desired output vector.

Backpropagating the above output errors to the hidden layer gives

$$\delta_j^1 = g'(\sum_{k=1}^{a} W_{jk}^1 x_k) \sum_{s=1}^{n} W_{sj}^2 \delta_s^2 \tag{3.28}$$

Backpropagating to the input layer gives

$$\delta_k^0 = \sum_{j=1}^{b} W_{jk}^1 g'(\sum_{k=1}^{a} W_{jk}^1 x_k) \sum_{s=1}^{n} W_{sj}^2 g'(\sum_{j=1}^{b} W_{sj}^2 g(\sum_{k=1}^{a} W_{jk}^1 x_k))(y_{m_s} - y_{p_s}) \tag{3.29}$$

where the time index k is neglected for simplicity sake.

As $u_g(k)$ is the lth network input,

$$\begin{aligned}
\delta_l^0 &= \{\sum_{s=1}^{n} W_{sj}^2 g'(\sum_{j=1}^{b} W_{sj}^2 g(\sum_{k=1}^{a} W_{jk}^1 x_k))\} \sum_{j=1}^{b} W_{jl}^1 g(\sum_{k=1}^{a} W_{jk}^1 x_k)(y_{m_s} - y_{p_s}) \\
&= g'(\sum_{j=1}^{b} W_{1j}^2 g(\sum_{k=1}^{a} W_{jk}^1 x_k)) \sum_{j=1}^{b} (W_{1j}^2 g(\sum_{k=1}^{a} W_{jk}^1 x_k) W_{jl}^1)(y_{m_1} - y_{p_1}) \\
&+ g'(\sum_{j=1}^{b} W_{2j}^2 g(\sum_{k=1}^{a} W_{jk}^1 x_k)) \sum_{j=1}^{b} (W_{2j}^2 g(\sum_{k=1}^{a} W_{jk}^1 x_k) W_{jl}^1)(y_{m_2} - y_{p_2}) \\
&\vdots \\
&+ g'(\sum_{j=1}^{b} W_{nj}^2 g(\sum_{k=1}^{a} W_{jk}^1 x_k)) \sum_{j=1}^{b} (W_{nj}^2 g(\sum_{k=1}^{a} W_{jk}^1 x_k) W_{jl}^1)(y_{m_n} - y_{p_n}) \\
&= (y_{m_1} - y_{p_1}) \frac{\partial \hat{y}_{p_1}(k+1)}{\partial u_g(k)} + (y_{m_2} - y_{p_2}) \frac{\partial \hat{y}_{p_2}(k+1)}{\partial u_g(k)} + \ldots + (y_{m_n} - y_{p_n}) \frac{\partial \hat{y}_{p_n}(k+1)}{\partial u_g(k)}
\end{aligned} \tag{3.30}$$

The controller error $e_c(k) = [e_{c_1}(k), \ldots, e_{c_g}(k), \ldots, e_{c_r}(k)]^T$ is a vector of dimension r. Hence, the output error for the g-th output node of the controller neural network is given by

$$e_{c_g}(k) = (y_{m_1} - y_{p_1}) \frac{\partial \hat{y}_{p_1}(k+1)}{\partial u_g(k)} + \ldots + (y_{m_n} - y_{p_n}) \frac{\partial \hat{y}_{p_n}(k+1)}{\partial u_g(k)} \tag{3.31}$$

where the output vector of the controller neural network is the r-dimensional control vector $u = [u_1, \ldots, u_g, \ldots, u_r]^T$.

■ ■ ■

The weight update equation (3.22) is then used in conjunction with the backpropagation algorithm to train and synthesise the controller. As is apparent from the above derivation, the identification of the plant is only necessary to provide a means of generating the plant Jacobian and thus allows adjustment of the control parameters W_c.

A number of alternative schemes have been suggested for obtaining an approximate of the plant Jacobian. Saerens and Soquet [201] have suggested the use of the sign of the Jacobian rather than its actual value to train the neural controller. However, in order for the sign of the Jacobian to be known, *a priori* information about the orientation in which the control parameters influence the output of the plant is required.

Numerical differentiation methods can also be used to obtain an approximation of the Jacobian. For example a first order approximation is given by

$$\frac{\partial y_p(k+1)}{\partial u(k)} \approx \frac{y_p(k+1) - y_p(k)}{u(k) - u(k-1)} \tag{3.32}$$

However, such a scheme would suffer from the large errors associated with most numerical differentiation approaches. Furthermore, problems arise when the control input is relatively constant. Such a method is suggested by Psaltis *et al.* [189]. A perturbation scheme in which the approximate Jacobian can be determined by changing each input to the plant slightly at the operating point and measuring the change in output is also suggested in [189].

3.3 Simulation Examples

Currently most of the studies done in the area of neural adaptive control have been empirical based on computer simulations [28, 79, 169, 171]. This is because of the difficulty in deriving theoretical results for neural network based control schemes. It has only been recently that work has been done in determining theoretical properties like controllability, observability and, most importantly, stability of neural control schemes [126, 186, 204, 235]. However, a great deal of information and insight can still be gained from empirical studies. Furthermore, such studies will undoubtedly generate more problems and theoretical issues which will need to be addressed.

Therefore, in this section several examples will be considered to highlight the effectiveness of the proposed method for both the off-line approach and the on-line approach. In particular, single-input single-output and multi-input multi-output systems belonging to Models III and IV will be considered. Plants from these classes of nonlinear systems are considered primarily because they are the most general class of models and are the most difficult to control because of the nonlinearities in the control variable. The examples belonging to Model IV, a class of systems considered difficult to control [169], will highlight the fact that the method considered here provides a unified approach to the control of nonlinear systems. A marginally stable system will also be considered for the on-line case to highlight one of the main advantages with this approach.

3.3.1 Off-line Learning

EXAMPLE 3.3.1 The example from Model III considered here is taken directly from [169], and thus also allows direct comparison with the method proposed by Narendra and Parthasarathy. The plant equation is given by

$$y_p(k+1) = \frac{y_p(k)}{1+y_p^2(k)} + u^3(k) \tag{3.33}$$

Narendra and Parthasarathy [169] use two neural network , N_f and N_g, to emulate the functions $f(y_p(k))$ and $g(u(k))$. The explicit control law used is given by

$$u(k) = \hat{g}^{-1}[-\hat{f}[y_p(k)] + \alpha_m y_p(k) + r(k)] \tag{3.34}$$

where $\hat{g}^{-1}$ is the estimate of the inverse of the nonlinear function in control and α_m is a prespecified constant. In [169] the function $\hat{g}^{-1}$ is emulated by a neural network of the class $\Omega^3_{1,20,10,1}$. In the method presented in this chapter, the plant is treated as a "black box" and thus a single neural network is used to emulate the plant. A neural network belonging to the class $\Omega^3_{2,20,10,1}$ is used, i.e.,

$$\hat{y}_p(k+1) = N_p[y_p(k), u(k)] \tag{3.35}$$

This neural network is trained with a random input signal $u(k) = \Re[-1,1]$.

Identification is carried out for $100,000$ iterations with a learning rate of 0.1, resulting in a mean square error of 0.0049. The output from the plant and the identification model for an input $u(k) = 0.5(\sin(\frac{2\pi k}{25}) + \sin(\frac{2\pi k}{10}))$ is shown in Figure 3.3. As can be seen there is virtually no discernible difference between the plant and emulator responses for this input.

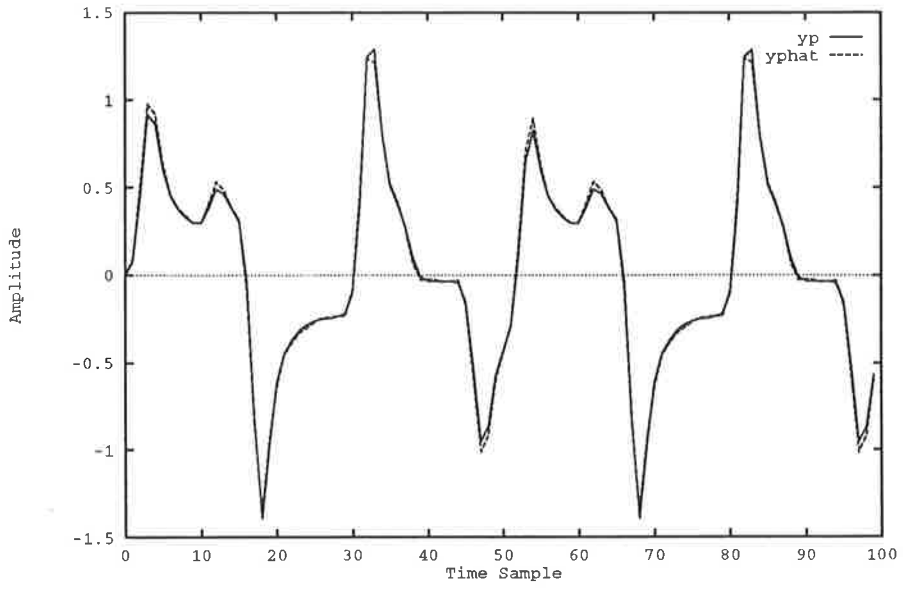


Figure 3.3: Response of the plant (y_p) and identification model ($\hat{y}_p$) for Example 3.3.1 with $u(k) = 0.5(\sin(\frac{2\pi k}{25}) + \sin(\frac{2\pi k}{10}))$

The controller is then implemented. A neural network of the class $\Omega^3_{2,20,10,1}$ is used to implement the model reference control and it can be represented as follows

$$u(k) = N_c[y_p(k), r(k)] \tag{3.36}$$

The following first order reference model is used

$$y_m(k+1) = 0.2y_m(k) + r(k) \tag{3.37}$$

To train the network, a random input $r(k) = \Re[-1, 1]$ is used. Training is carried out for 100,000 iterations, with a learning rate of 0.1. The resultant tracking mean square error is 0.0045. The reference model and plant outputs are shown for reference signals $r(k) = 0.5(\sin(\frac{2\pi k}{25}) + \sin(\frac{2\pi k}{10}))$ and $r(k) = \Re[-1, 1]$ in Figures 3.4a and 3.4b, respectively.

The plant response in these figures is virtually indistinguishable from the reference model response. The excellent tracking results highlight the effectiveness of the controller for plants belonging to Model III, even with an uncorrelated, random input. The control

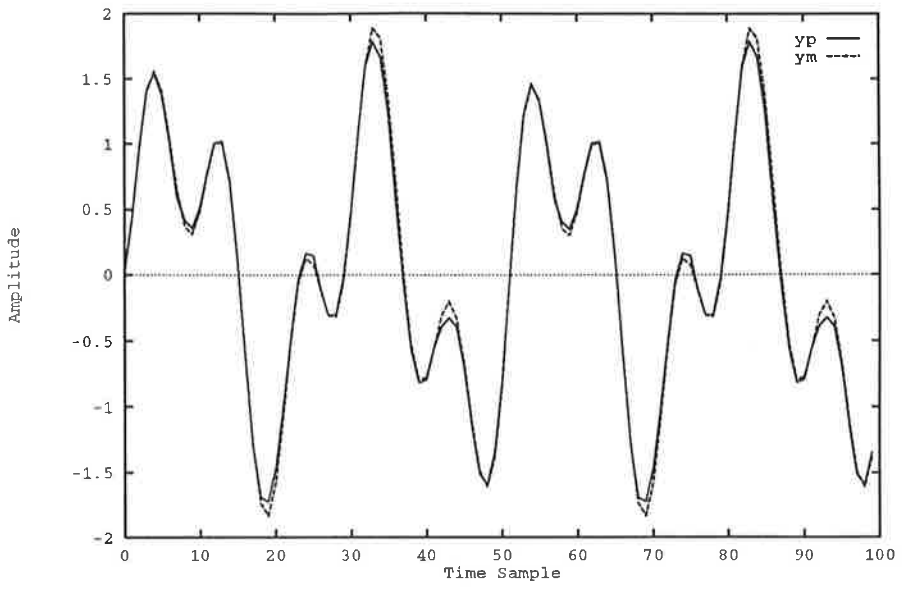


Figure 3.4a: Response of the plant (y_p) and reference model (y_m) for Example 3.3.1 with $r(k) = 0.5(\sin(\frac{2\pi k}{25}) + \sin(\frac{2\pi k}{10}))$

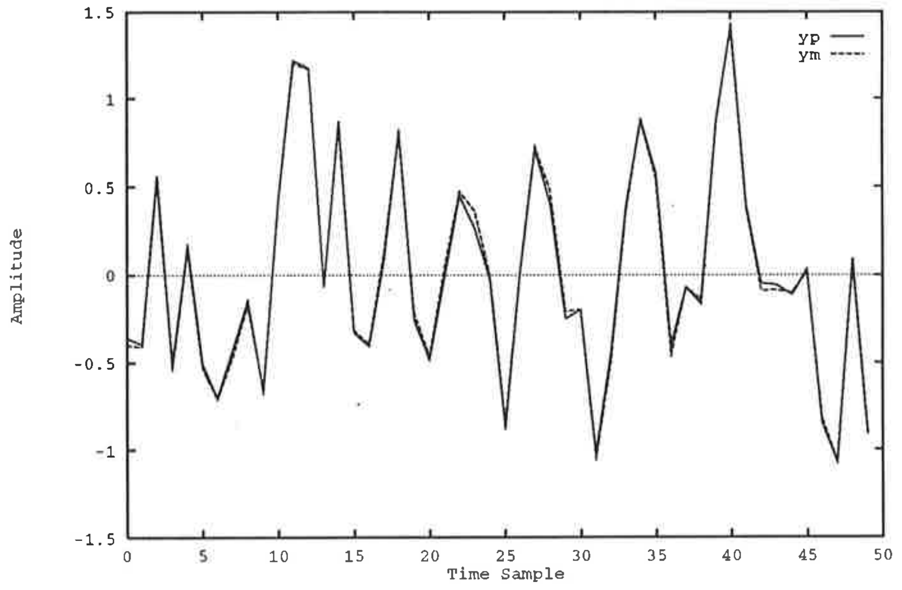


Figure 3.4b: Response of the plant (y_p) and reference model (y_m) for Example 3.3.1 with $r(k) = \Re[-1, 1]$

scheme is designed with the system treated as a "black box", in which the neural network emulator models the input-output mapping of the plant. As a result, knowledge about the separability or otherwise of the nonlinearities in the output and control terms is not required. Therefore, unlike [169] there is no need to explicitly model the function $g^{-1}(.)$, which is the inverse of the nonlinear function in control, or the function $f(.)$, which is the nonlinear function in the output terms. However, this means that an explicit control law as in [169] cannot be stated, but as shown by the results provided, the scheme is equally effective in this example.

REMARK 3.3.1 *Although the tracking error converges to a neighbourhood of zero, this does not in general guarantee that the network weights will converge. Linear theory results imply that convergence will not be obtained unless the signals used are persistently exciting [167, 204]. However, currently there are no methods of characterising persistently exciting inputs which will guarantee the convergence of the network weights.*

EXAMPLE 3.3.2 In this example a SISO plant belonging to Model IV is considered. Such plants are difficult to control as the control term u is heavily embedded in the nonlinearities of the system. Therefore it is difficult to find an inverse operator in the control and consequently an explicit control law is very difficult to obtain. It is because of these reasons these systems are considered the least tractable analytically and consequently, not discussed in many approaches. The plant equation considered in this example is given by

$$y_p(k+1) = \frac{y_p(k)y_p(k-1)u(k) + u^3(k) + 0.5y_p(k-1)}{1 + y_p^2(k) + y_p^2(k-1)} \tag{3.38}$$

The reference model considered is given by (3.37).

The plant equation is nonlinear in output and control, and their nonlinear terms are not separable. Thus, this plant belongs to Model IV. A neural network belonging to the class $\Omega^3_{3,20,10,1}$ is used to identify the plant. The identification model used is

$$\hat{y}_p(k+1) = N_p[y_p(k), y_p(k-1), u(k)] \tag{3.39}$$

The network is trained with an input signal $u(k) = \Re[-1,1]$, with a learning rate of 0.1. Training is carried out for 100,000 iterations, resulting in a mean square identification error of 0.0041. The outputs for the plant and the identification model for an input $u(k) = \sin(\frac{2\pi k}{250})$ are shown in Figure 3.5.

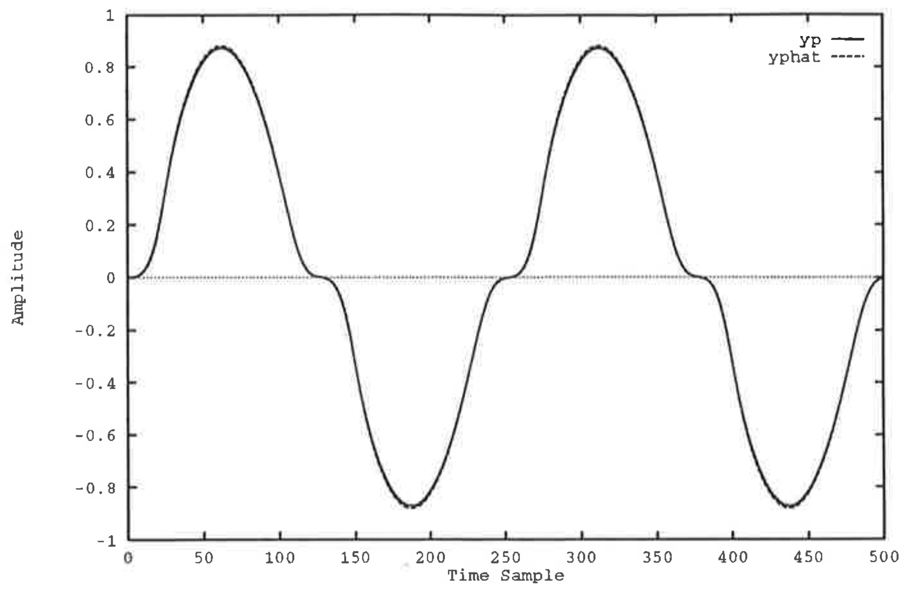


Figure 3.5: Response of the plant (y_p) and identification model ($\hat{y}_p$) for Example 3.3.2 with $u(k) = \sin(\frac{2\pi k}{250})$

A neural network of the class $\Omega^3_{3,20,10,1}$ is used to implement the controller. The control law to be approximated is given by

$$u(k) = N_c[y_p(k), y_p(k-1), r(k)] \tag{3.40}$$

The controller is trained with a reference signal $r(k) = \Re[-1,1]$ for 100,000 iterations with a learning rate of 0.1. The resultant mean square tracking error is 0.0084. The outputs from the reference model and the plant are given for reference inputs

$$r(k) = \begin{cases} \sin(\frac{2\pi k}{250}) & k \leq 500 \\ \sin(\frac{2\pi k}{250}) + \sin(\frac{2\pi k}{100}) & k > 500 \end{cases} \tag{3.41}$$

and $r(k) = \Re[-1,1]$ in Figures 3.6a and 3.6b, respectively.

In [169] the control of a plant belonging to Model IV is regarded as analytically the least tractable and thus not attempted. This example indicates that the method presented here can be used to control plants of the form of Model IV as effectively as plants from other models. However, Figure 3.6a also indicates one of the main problems with an

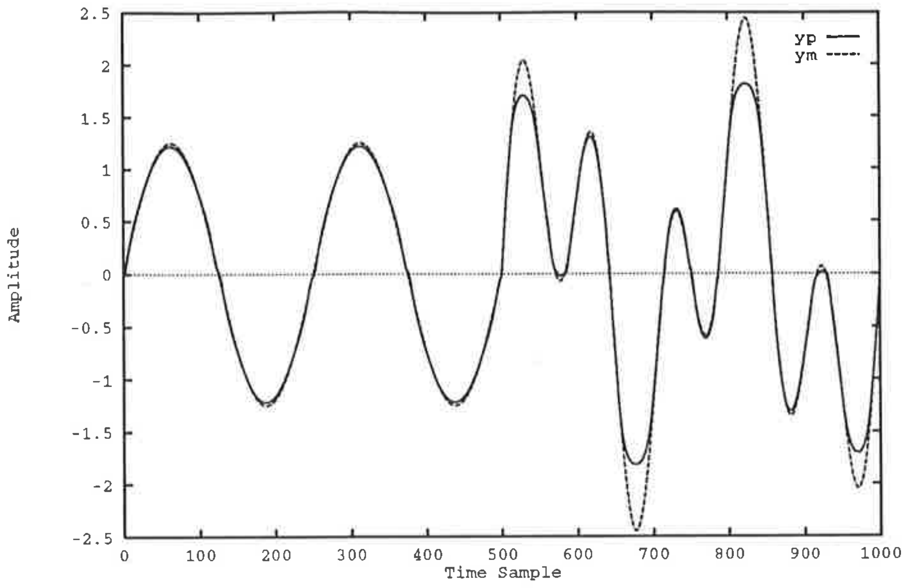


Figure 3.6a: Response of the plant (y_p) and reference model (y_m) for Example 3.3.2 with $r(k) = \sin(\frac{2\pi k}{250})$ for $k \leq 500$ and $r(k) = \sin(\frac{2\pi k}{250}) + \sin(\frac{2\pi k}{100})$ for $k > 500$

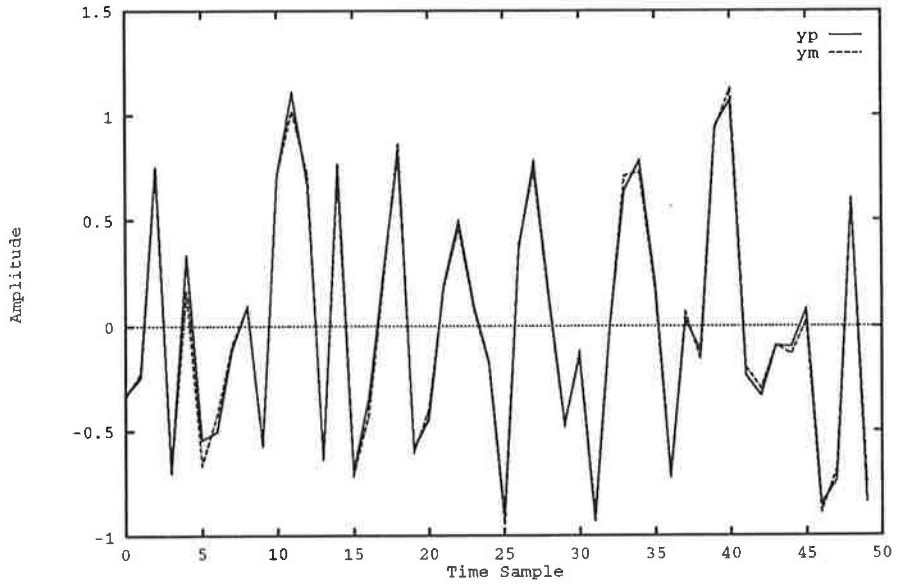


Figure 3.6b: Response of the plant (y_p) and reference model (y_m) for Example 3.3.2 with $r(k) = \Re[-1, 1]$

off-line neural control scheme. The controller neural network is trained over the region $r(k) = \Re[-1,1]$ and when the operational reference input is within this region, as is the case for $r(k) = \sin(\frac{2\pi k}{250})$ $k \leq 500$, the performance of the controller is very good. However, when the reference input is outside of the region on which the controller is trained, as is the case for $r(k) = \sin(\frac{2\pi k}{250}) + \sin(\frac{2\pi k}{100})$ for $k > 500$, then the controller performance deteriorates. This can be overcome if weight adaptation is switched on whilst the system is operational and thus the system will learn to control the plant for inputs outside the off-line training region. This is effectively an on-line training scheme. However, this results in localised learning as the neural control learns to control the plant for the system acting under the operating reference input rather than the general training reference input $r(k) = \Re[-1,1]$. This results in the neural controller "forgetting" about the training responses. Whilst such an approach is useful particularly when operating in a highly adaptive environment, it does make the computationally intensive and relatively time consuming training process slightly redundant. A better solution would be to train the system off-line for a short period of time to provide the controller neural network and emulator neural network some initial idea about the system and then implement an on-line approach.

EXAMPLE 3.3.3 In this example a two-input two-output system belonging to Model IV is considered. The plant equation is given by

$$\begin{aligned} y_{p_1}(k+1) &= \frac{0.5y_{p_1}(k) + 0.4u_1(k) + 0.6u_2(k)}{1 + y_{p_1}^2(k)} \\ y_{p_2}(k+1) &= \frac{0.5y_{p_2}(k) + 0.6u_1(k) + 0.4u_2(k)}{1 + y_{p_2}^2(k)} \end{aligned} \tag{3.42}$$

The plant equation is nonlinear in output and control, and their corresponding terms are not separable. Apart from the complex nonlinearity of the system, such a plant is difficult to control because of the cross-coupling of the control inputs, i.e., the plant outputs y_{p_1} and y_{p_2} are functions of both control inputs u_1 and u_2. This form of cross-coupling occurs commonly in practical systems. A classic example is in the dynamic equations for a guided missile or an aircraft in which significant cross-coupling exists between the pitch dynamics, the yaw dynamics and the roll dynamics.

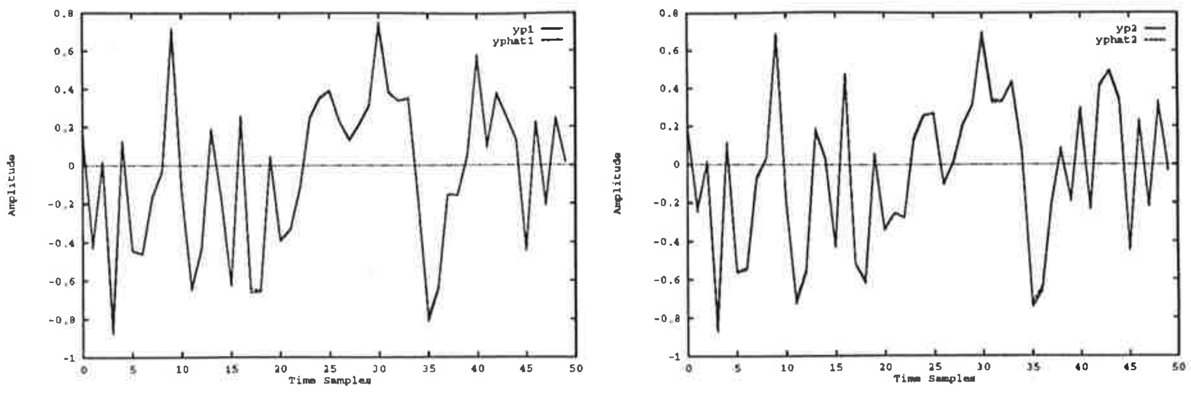


Figure 3.7: Response of the plant and identification model for Example 3.3.3 with $u_1(k) = \Re[-1,1]$ & $u_2(k) = \Re[-1,1]$

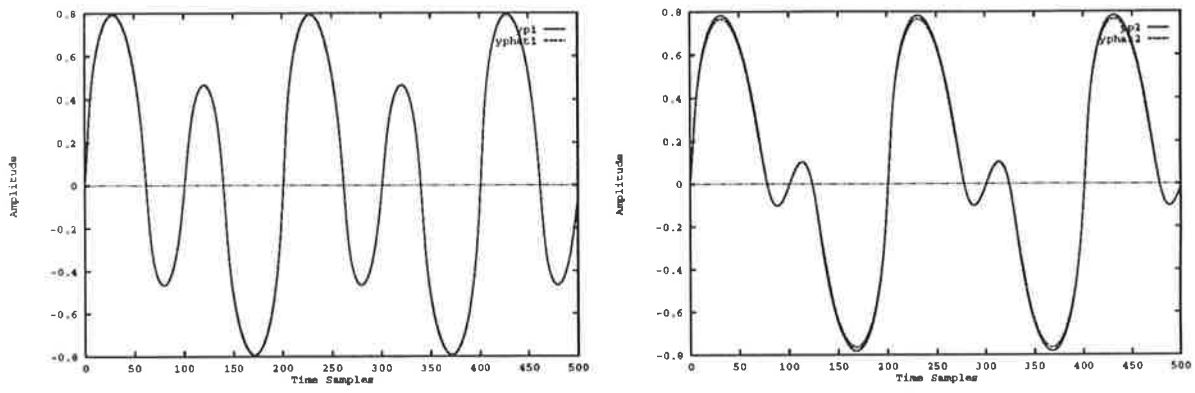


Figure 3.8: Response of the plant and identification model for Example 3.3.3 with $u_1(k) = \sin(\frac{2\pi k}{200})$ & $u_2(k) = \sin(\frac{2\pi k}{100})$

A single neural network belonging to the class $\Omega^3_{4,10,10,2}$ is used to identify the plant. The identification model is

$$\hat{y}_p(k+1) = [\hat{y}_{p_1}(k+1), \hat{y}_{p_2}(k+1)]^T = N_p[y_{p_1}(k), y_{p_2}(k), u_1(k), u_2(k)] \tag{3.43}$$

The network is trained with input signals $u_1(k) = \Re[-1,1]$ and $u_2(k) = \Re[-1,1]$ with a learning rate of 0.1 and a momentum rate of 0.9. Training is carried out for approximately 50,000 iterations, resulting in a mean square identification error of 0.001. The plant and identification model outputs are shown for input signals $u_1(k) = \Re[-1,1]$ & $u_2(k) = \Re[-1,1]$ and $u_1(k) = \sin(\frac{2\pi k}{200})$ & $u_2(k) = \sin(\frac{2\pi k}{100})$ in Figures 3.7 and 3.8, respectively.

These figures highlight the ability of neural networks to approximate the dynamics of a complex multi-input multi-output nonlinear system to an excellent degree of accuracy.

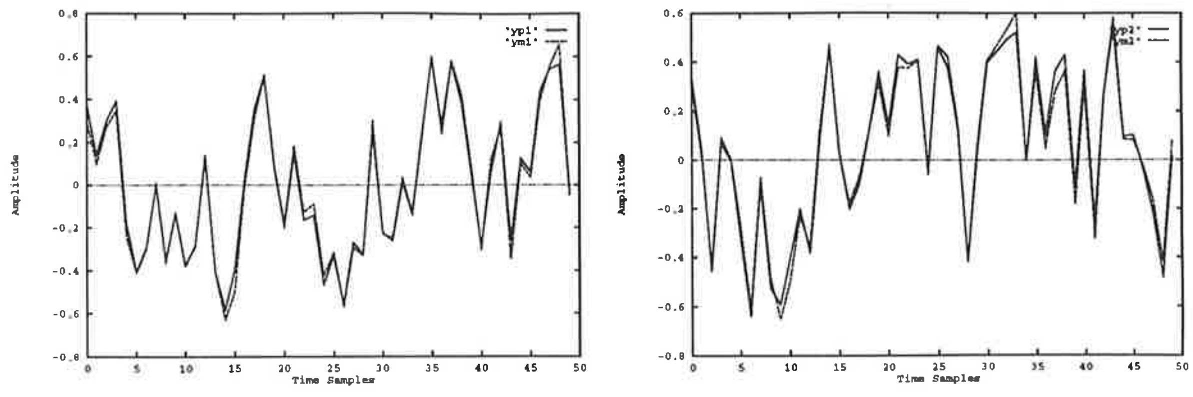


Figure 3.9: Response of the plant and reference model for Example 3.3.3 with $r_1(k) = \Re[-1,1]$ & $r_2(k) = \Re[-1,1]$

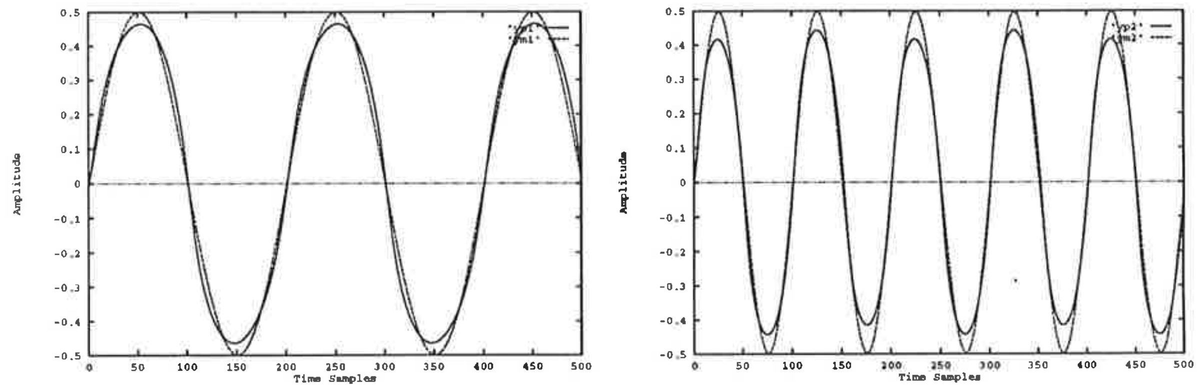


Figure 3.10: Response of the plant and reference model for Example 3.3.3 with $r_1(k) = \sin(\frac{2\pi k}{200})$ & $r_2(k) = \sin(\frac{2\pi k}{100})$

A single neural network of the class $\Omega^3_{4,20,10,2}$ is used to implement the neural controller. The inputs to the controller neural network are $[y_{p_1}(k), y_{p_2}(k), r_1(k), r_2(k)]$. The controller is trained with reference signals $r_1(k) = \Re[-1,1]$ and $r_2(k) = \Re[-1,1]$ to track the output of a reference model given by

$$\begin{aligned} y_{m_1}(k+1) &= 0.5y_{m_1}(k) + 0.5r_1(k) \\ y_{m_2}(k+1) &= 0.5y_{m_2}(k) + 0.5r_2(k) \end{aligned} \tag{3.44}$$

Training is carried out for 100,000 iterations resulting in a mean square tracking error of 0.0026. The outputs for the reference model and the plant are given for reference inputs $r_1(k) = \Re[-1,1]$ & $r_2(k) = \Re[-1,1]$ and $r_1(k) = \sin(\frac{2\pi k}{200})$ & $r_2(k) = \sin(\frac{2\pi k}{100})$ in Figures 3.9 and 3.10, respectively.

The theoretical results provided in Section 3.2.3 are derived for a general multi-input multi-output. This simulation example confirms that the neural network control scheme presented in this chapter can be used to control a MIMO system. This is an important result because most practical systems consist of multiple inputs and multiple outputs. Furthermore, the approach presented here does not require any additional information to control MIMO systems as opposed to SISO systems. Also, a single neural network is used to model the entire plant dynamics and an additional single neural network is used to implement the controller as opposed to an individual neural network for each system output term and each control term. Whilst from a control perspective having individual neural networks for each control and output variable may be equally effective, from an implementation perspective it is very memory and computation intensive.

The Effect of Disturbances on Off-Line Learning

The effect of three types of disturbances on the neural control system presented in this chapter will now be considered, namely load changes, sensor noise and dynamic plant noise. These forms of disturbances commonly arise in practical systems and thus the performance of the controller in their presence is an important issue.

(i) Load Disturbance Compensation

The first type of disturbance considered is load disturbances. They can represent disturbance forces in a mechanical system, such as waves acting on the hull of a ship or load changes on a motor or, as in process control, they may represent variations in feed flow [10]. One technique for compensating for such disturbances is feedforward control. The basic principle of this technique is to measure or estimate the disturbances as they occur and make adjustments in the manipulated variable so as to prevent them from upsetting the controlled variable [43]. The bias compensation technique employed here is based on this approach and the load disturbances are assumed unknown and not measurable.

Load disturbances are operational disturbances in that they occur whilst the system is under normal operating conditions. Therefore, they do not occur during the training process in which the system is taken off-line and subjected to a persistently exciting training input. Thus, as the controller is trained in a disturbance free environment, there is a need to compensate for the bias to ensure that the the disturbance-corrupted plant output, $y_p(k+1)$, tracks the output of the reference model, $y_m(k+1)$. The unknown bias is estimated by subtracting output of the neural network emulator, $\hat{y}(k+1)$, which is trained to approximate the plant in a disturbance free environment, from the bias corrupted plant output, $y_p(k+1)$. Low pass filtering is undertaken to eliminate the initial transients. The estimated bias term is given by

$$\hat{b}(k+1) = \rho\hat{b}(k) + (1-\rho)[y_p(k+1) - \hat{y}_p(k+1)] \tag{3.45}$$

where $\hat{y}_p(k+1)$ is the output from the NN emulator and $0 < \rho < 1$.

The controller has been trained such that the plant output tracks the output of a linear reference model with gain G. However if the uncompensated reference input is applied to the controller the resultant output from the plant is given by

$$y_p(k+1) = Gr(k) + b \tag{3.46}$$

whereas the output from the reference model is given by

$$y_m(k+1) = Gr(k) \tag{3.47}$$

Thus it is necessary to scale the reference input by the term $\frac{b}{G}$ to ensure that correct model following occurs. A diagram of the method employed is shown in Figure 3.11.

EXAMPLE 3.3.4 The plant considered to highlight the bias compensation technique is given by

$$y_p(k+1) = 0.2y_p(k) + 0.2y_p(k-1) + 0.5u^3(k) \tag{3.48}$$

This plant belongs to the Model II class of systems which are linear in output, but nonlinear in control. The reference model used is given by

$$y_m(k+1) = 0.2y_m(k) + r(k) \tag{3.49}$$

The bias term used is shown with its estimate in Figure 3.12a. A variable bias is used to demonstrate the effectiveness of the bias compensation technique under different load (bias) conditions. As can be seen from the plant and reference model responses given in

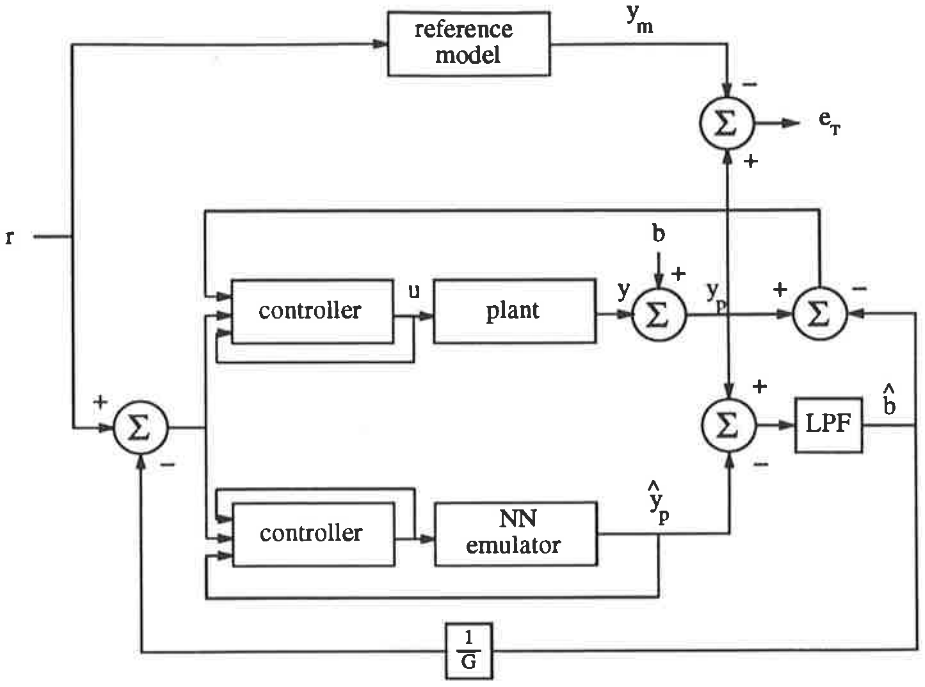


Figure 3.11: Block diagram of the bias compensated controller system

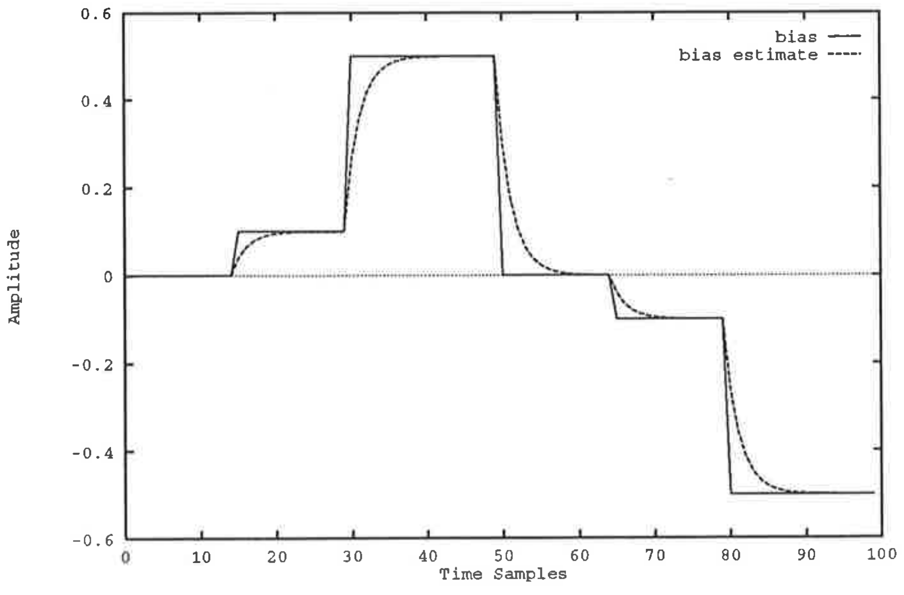


Figure 3.12a: Variable load disturbance (b) and its estimate ($\hat{b}$)

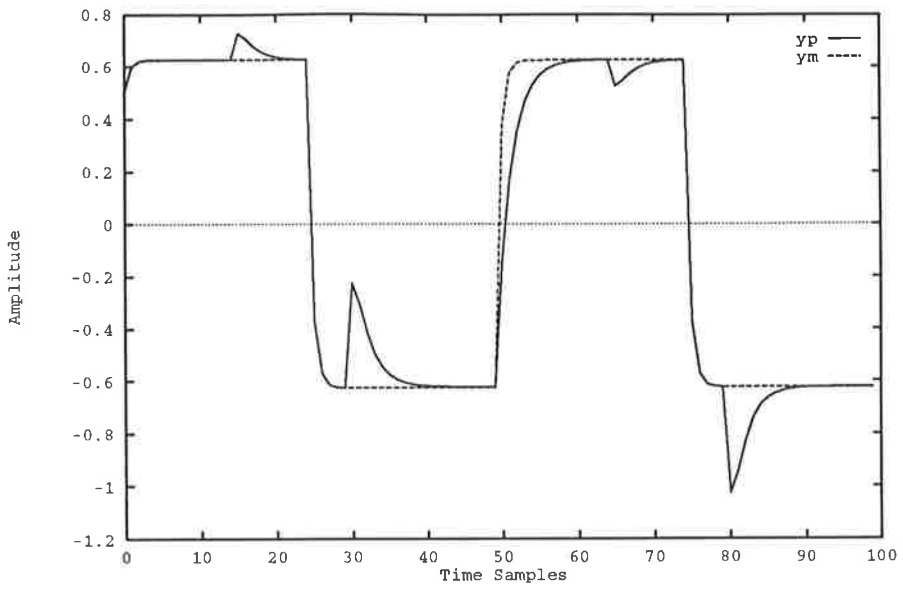


Figure 3.12b: Response of the plant (y_p) and the reference model (y_m) to the bias term (b)

Figure 3.12b, the proposed technique successfully compensated for the load disturbances considered.

An alternative to this scheme would be to retrain the emulator and controller neural networks in real time to offset the bias. However, the problem with this approach is that it will result in localised training around a particular operating point. The off-line training which is undertaken over the entire input space therefore becomes redundant. As the off-line training stage is computationally expensive, this would be a waste of resources. If very little or no training is performed prior to operation, then this approach is feasible. As will be shown in the next section, on-line training is capable of compensating for DC load disturbances.

(ii) Measurement and Sensor Noise

Sensor noise enters the system because of the imperfect measurement of the output of the plant. In practice it can be caused by random transducer and sensor errors, transmission noise and high frequency load disturbance. The noise is generally considered to consist of high frequency components and is thus commonly represented by a white noise process.

In examining the effect of sensor noise on the controller, it is necessary to train the emulator and controller in a noisy environment. This is because sensor noise represents the noise inherent to the measuring device and thus each measurement of the output, $y_p(k+1)$, results in the introduction of a noise term $v(k)$. Therefore, sensor noise is present even during training and so it cannot be assumed that training occurs in a noiseless environment.

The method used to train the emulator and controller is the same as before, except that the output of the plant is corrupted by noise.

The output from the noise-corrupted plant is given by

$$\begin{aligned} y_n(k+1) &= y_p(k+1) + v(k) \\ &= f[y_p(k), \ldots, y_p(k-n+1); u(k), \cdots, u(k-m+1)] + v(k) \end{aligned} \quad (3.50)$$

where $v(k)$ is the zero mean, random sensor noise.

The output from the emulator is given by

$$\hat{y}_n(k+1) = N_e[y_n(k), y_n(k-1), \ldots, y_n(k-n+1); u(k), u(k-1), \cdots, u(k-m+1)] \quad (3.51)$$

where N_e represents the emulator neural network.

As the emulator is modelling the noise-corrupted plant, viz the plant output plus sensor noise, the mean square error is given by

$$e_{I_{ms}} = \frac{1}{N} \sum_{k=0}^{N-1} [y_n(k+1) - \hat{y}_n(k+1)]^2 \quad (3.52)$$

As $v(k)$ is uncorrelated white noise, intuitively one can expect that in order to achieve the same mean square error as for the noiseless case, far more training iterations will be required.

Once again the same procedure as the noiseless case is used to train the controller, except that the plant output is corrupted by the white sensor noise.

The control signal is given by

$$u(k) = N_c[y_n(k), y_n(k-1), \ldots, y_n(k-n+1); u(k-1), \cdots, u(k-m+1); r(k)] \quad (3.53)$$

where N_c represents the controller neural network, with the noise corrupted output as a feedback variable

Two examples are considered to highlight the effectiveness of the control scheme when sensor noise is present.

EXAMPLE 3.3.5 The same plant and reference model as Example 3.3.1 are used. The emulator is trained with a random signal $u(k) = \Re[-0.5, 0.5]$, with a random sensor noise signal $v(k) = \Re[-0.06, 0.06]$ present. This results in an output signal-to-noise ratio of 20dB. Training is carried out for 100,000 iterations with a learning rate of 0.1, resulting in a mean square error of 0.034. The outputs for the noise-corrupted plant and identification model for an input signal $u(k) = 0.5\sin(\frac{2\pi k}{250})$ are shown in Figure 3.13a.

The controller is trained with a reference signal $r(k) = [-0.5, 0.5]$ for 100,000 iterations with a learning rate of 0.1, resulting in a mean square tracking error of 0.0152. The plant and reference model outputs for reference inputs $r(k) = 0.5\sin(\frac{2\pi k}{250})$ and $r(k) = 0.5sgn[\sin(\frac{2\pi k}{250})]$ are shown in Figures 3.13b and 3.13c, respectively.

EXAMPLE 3.3.6 In this example, the same plant and reference model as Example 3.3.2 are used. The emulator is trained with a random signal $u(k) = \Re[-1, 1]$, with a random sensor noise signal $v(k) = \Re[-0.08, 0.08]$ present. This once again results in an output signal-to-noise ratio of 20dB. Training is carried out for 100,000 iterations with a learning rate of 0.1, resulting in a mean square error of 0.0079. The outputs for the noise-corrupted plant and identification model for an input signal $u(k) = \sin(\frac{2\pi k}{250})$ are shown in Figure 3.14a. As can be seen from this figure, the NN emulator output resembles a filtered version of the actual noisy plant output.

The controller is trained with a reference signal $r(k) = [-0.5, 0.5]$ for 100,000 iterations with a learning rate of 0.1, resulting in a mean square tracking error of 0.0197. The plant and reference model outputs for reference inputs $r(k) = 0.5\sin(\frac{2\pi k}{250})$ and $r(k) = 0.5sgn[\sin(\frac{2\pi k}{250})]$ are shown in Figures 3.14b and 3.14c, respectively.

The above results highlight that for the cases considered measurement noise of the order of 10% (20dB) does not seem to present a practical problem. In both cases tracking

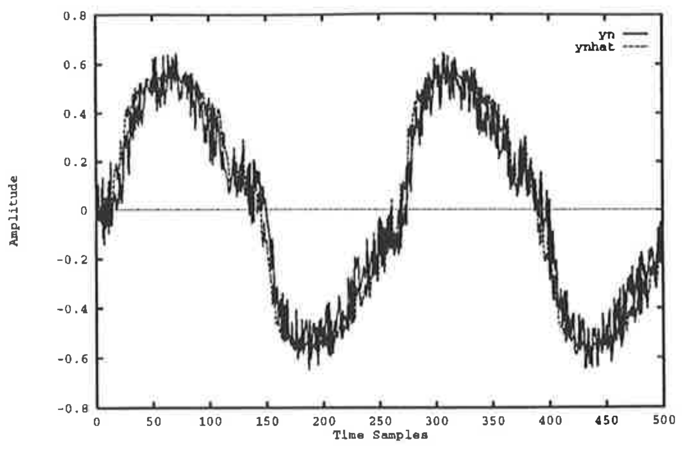


Figure 3.13a: Response of the noise-corrupted plant (y_n) and identification model ($\hat{y}_n$) for Example 3.3.5 with $u(k) = 0.5\sin(\frac{2\pi k}{250})$

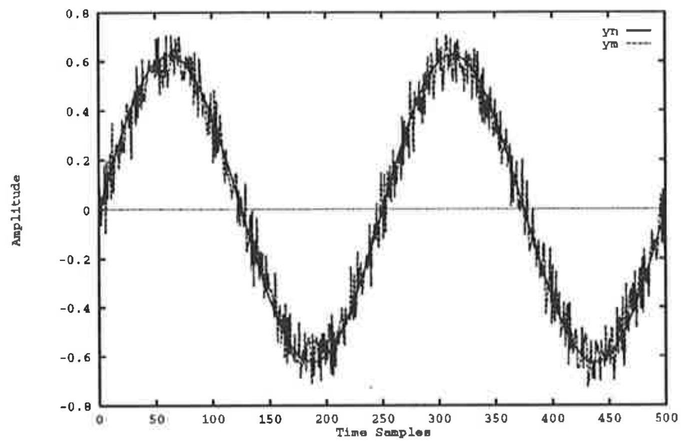


Figure 3.13b: Response of the noise-corrupted plant (y_n) and reference model (y_m) for Example 3.3.5 with $r(k) = 0.5\sin(\frac{2\pi k}{250})$

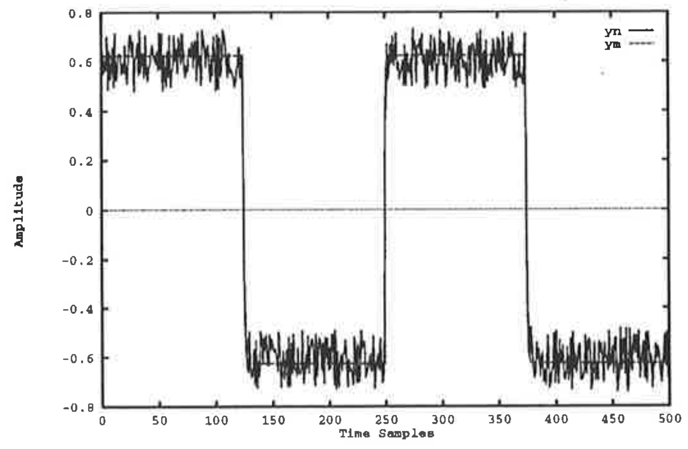


Figure 3.13c: Response of the noise-corrupted plant (y_n) and reference model (y_m) for Example 3.3.5 with $r(k) = 0.5sgn[\sin(\frac{2\pi k}{250})]$

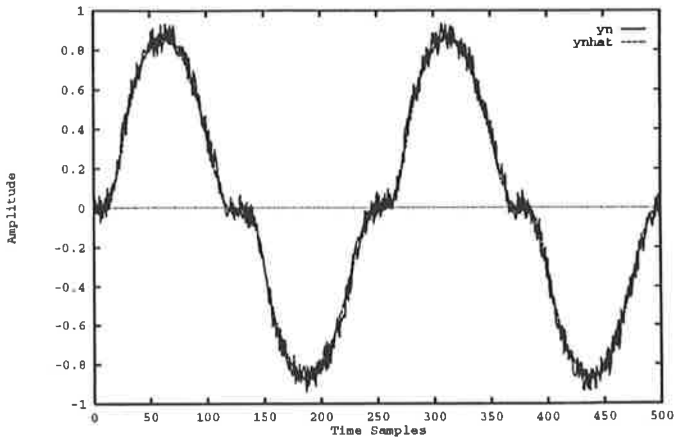


Figure 3.14a: Response of the noise-corrupted plant (y_n) and identification model ($\hat{y}_n$) for Example 3.3.6 with $u(k) = \sin(\frac{2\pi k}{250})$

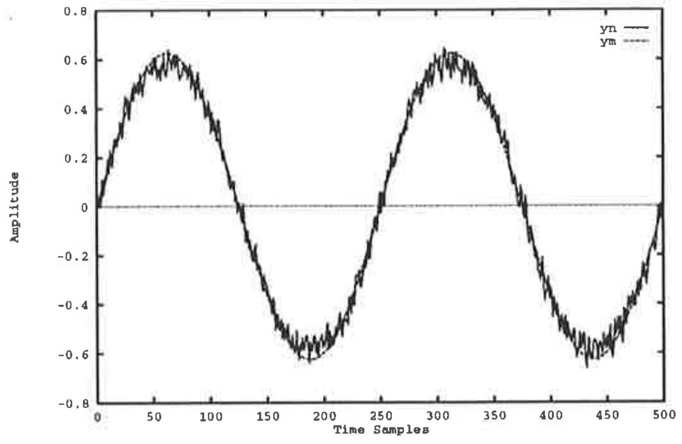


Figure 3.14b: Response of the noise-corrupted plant (y_n) and reference model (y_m) for Example 3.3.6 with $r(k) = 0.5\sin(\frac{2\pi k}{250})$

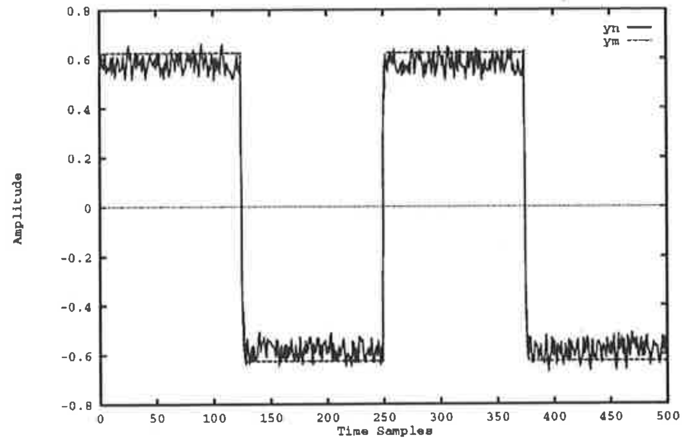


Figure 3.14c: Response of the noise-corrupted plant (y_n) and reference model (y_m) for Example 3.3.6 with $r(k) = 0.5sgn[\sin(\frac{2\pi k}{250})]$

errors of similar order to the noiseless case are obtained. Once control is initiated, the mean of the output of the noise-corrupted plant tracked the output of the reference model and the noise component is attenuated effectively. An observation that is made is that the output of the emulator trained to approximate the noise-corrupted nonlinear plant tracked the mean of the noisy plant output.

(iii) Dynamic Plant Noise

The ability to deal with external stochastic disturbances is of major concern in all control systems. If the unwanted disturbance is of sufficient magnitude, it behaves as an equivalent input signal to the plant and thus adversely affects the performance of the system. Common examples of such disturbances are a gust of wind on an airplane, waves on a ship and internal noise from the control system components. The effect of these disturbances acting on the plant is to a large extent subject to the inherent dynamics and characteristics of the plant. In the study considered, the dynamic effect of the plant noise is simulated by adding a white noise sequence to the input of the plant. Therefore, the noise component of the output of the plant, $y_n(k+1)$, has been coloured by the plant dynamics and is thus no longer separable from the actual plant output, as is the case for the sensor noise. Hence, the disturbance appears at the output of the plant as a coloured noise process.

As with the sensor noise case, training of the emulator and controller is undertaken in a noisy environment. This is because dynamic plant noise is often due to noise from internal components in the control system and thus will be present even when the system is taken off-line. For the case of dynamic noise produced solely by operational disturbances such as wind or waves, a noise-free environment may be assumed during training. However, this is rarely the case.

The procedures adopted for training the emulator and controller are the same as for the noiseless case, except that the input to the plant, $u'(k)$, is a noise-corrupted input.

The output from the noise-corrupted plant is given by

$$y_n(k+1) = f[y_p(k), y_p(k-1), \ldots, y_p(k-n+1); u'(k), u'(k-1), \cdots, u'(k-m+1)] \quad (3.54)$$

where $u'(k) = u(k) + v(k)$ and $v(k)$ is Gaussian white noise sequence.

The output from the emulator is given by

$$\hat{y}_n(k+1) = N_e[y_n(k), y_n(k-1), \ldots, y_n(k-n+1); u(k), u(k-1), \cdots, u(k-m+1)] \quad (3.55)$$

where N_e represents the emulator neural network.

The control signal is given by

$$u(k) = N_c[y_n(k), y_n(k-1), \ldots, y_n(k-n+1); u(k-1), \cdots, u(k-m+1); r(k)] \quad (3.56)$$

where N_c represents the controller neural network.

The same two examples used in the sensor noise case are considered to highlight the effectiveness of the control scheme when dynamic plant noise is present.

EXAMPLE 3.3.7 The same values used for the corresponding Model III sensor noise example are used except that a input signal $u(k) = \Re[-1,1]$ and noise signal $v(k) = \Re[-0.1, 0.1]$ is used, resulting in an input signal-to-noise ratio of 20dB. Once again, training is carried out for 100,000 iterations with a learning rate of 0.1, resulting in a mean square error of 0.0328. The outputs for the noise-corrupted plant and identification model for an input signal $u(k) = \sin(\frac{2\pi k}{250})$ are shown in Figure 3.15a.

The controller is trained with a reference signal $r(k) = \Re[-1,1]$ for 100,000 iterations with a learning rate of 0.1, resulting in a mean square tracking error of 0.0541. The plant and reference model outputs for reference inputs $r(k) = \sin(\frac{2\pi k}{250})$ and $r(k) = sgn[\sin(\frac{2\pi k}{250})]$ are shown in Figures 3.15b and 3.15c, respectively.

EXAMPLE 3.3.8 In this example, the same plant and reference model as Example 3.3.2 are used. The emulator is trained with a random signal $u(k) = \Re[-1,1]$, and with a random noise signal $v(k) = \Re[-0.1, 0.1]$ present. This once again results in an input signal-to-noise ratio of 20dB. Training is carried out for 100,000 iterations with a learning rate of 0.1, resulting in a mean square error of 0.0129. The outputs for the noise-corrupted plant and identification model for an input signal $u(k) = \sin(\frac{2\pi k}{250})$ are shown in Figure 3.16a.

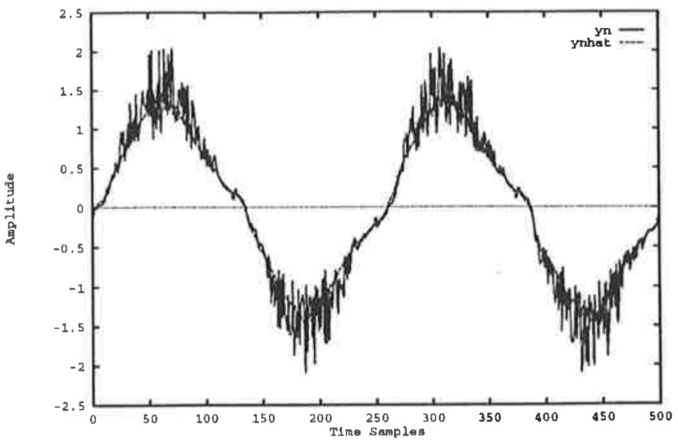


Figure 3.15a: Response of the noise-corrupted plant (y_n) and identification model ($\hat{y}_n$) for Example 3.3.7 with $u(k) = \sin(\frac{2\pi k}{250})$

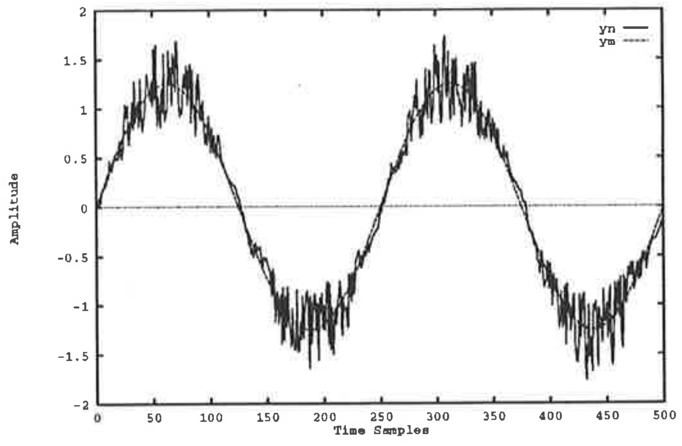


Figure 3.15b: Response of the noise-corrupted plant (y_n) and reference model (y_m) for Example 3.3.7 with $r(k) = \sin(\frac{2\pi k}{250})$

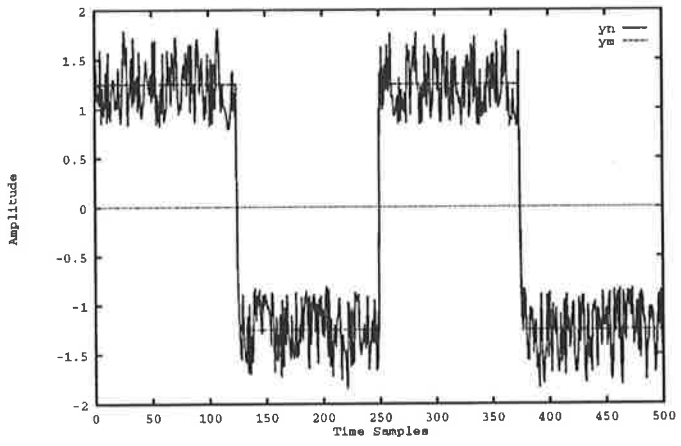


Figure 3.15c: Response of the noise-corrupted plant (y_n) and reference model (y_m) for Example 3.3.7 with $r(k) = sgn[\sin(\frac{2\pi k}{250})]$

The controller is trained with a reference signal $r(k) = \Re[-1,1]$ for 100,000 iterations with a learning rate of 0.1, resulting in a mean square tracking error of 0.0238. The plant and reference model outputs for reference inputs $r(k) = 0.5\sin(\frac{2\pi k}{250})$ and $r(k) = 0.5sgn[\sin(\frac{2\pi k}{250})]$ are given in Figures 3.16b and 3.16c, respectively.

The above results once again indicate that the controller performance is still very good even when trained in a noisy environment. As with the sensor noise case, the emulator response resembled a filtered version of the noise corrupted output. The level of the noise at the output of the plant is dependent upon the structure of the plant. In the examples considered, the output signal-to-noise ratio for the Model III plant which contained a $u^3(k)$ term is worse than the Model IV plant which contained the term $\frac{u^3(k)}{1+y_p^2(k)+y_p^2(k-1)}$. However in both cases the output noise is attenuated during the control stage. Apart from the presence of noise, the control performance is satisfactory, which is a promising result considering the plant noise considered is probably the worst case scenario, as the noise is coloured by the plant dynamics, and is thus not separable from the plant output.

3.3.2 On-line Learning and Control

EXAMPLE 3.3.9 The plant considered here is used in [169] and also is considered in the off-line approach in Example 3.3.1. The plant equation is given by

$$y_p(k+1) = \frac{y_p(k)}{1+y_p^2(k)} + u^3(k) \tag{3.57}$$

As is the case for Example 3.3.1, the reference model used is given by

$$y_m(k+1) = 0.2y_m(k) + r(k) \tag{3.58}$$

Three types of reference inputs are considered. These are

$$r(k) = \sin(\frac{2\pi k}{25}) \tag{3.59a}$$

$$r(k) = \begin{cases} \sin(\frac{2\pi k}{25}) & k < 920 \\ 0.5sgn[\sin(\frac{2\pi k}{25})] & k \geq 920 \end{cases} \tag{3.59b}$$

$$r(k) = sgn[\sin(\frac{2\pi k}{300})] \tag{3.59c}$$

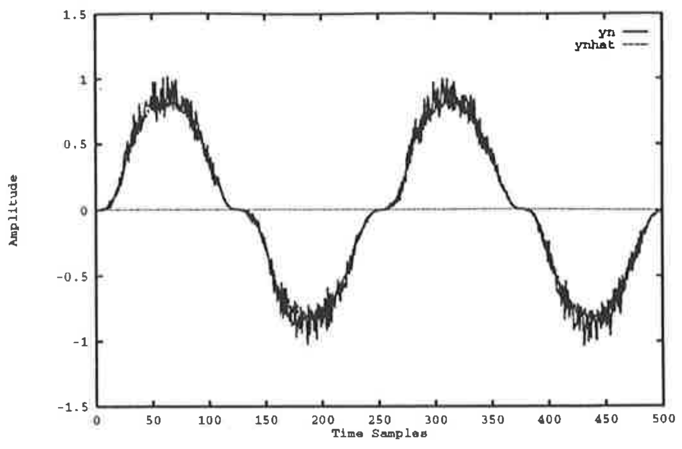


Figure 3.16a: Response of the noise-corrupted plant (y_n) and identification model ($\hat{y}_n$) for Example 3.3.8 with $u(k) = \sin(\frac{2\pi k}{250})$

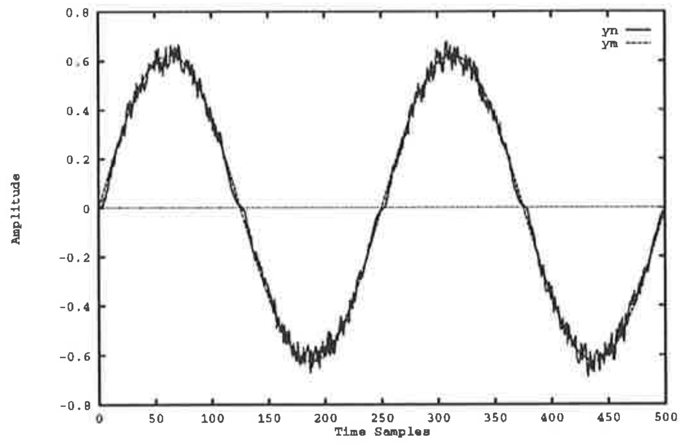


Figure 3.16b: Response of the noise-corrupted plant (y_n) and reference model (y_m) for Example 3.3.8 with $r(k) = 0.5\sin(\frac{2\pi k}{250})$

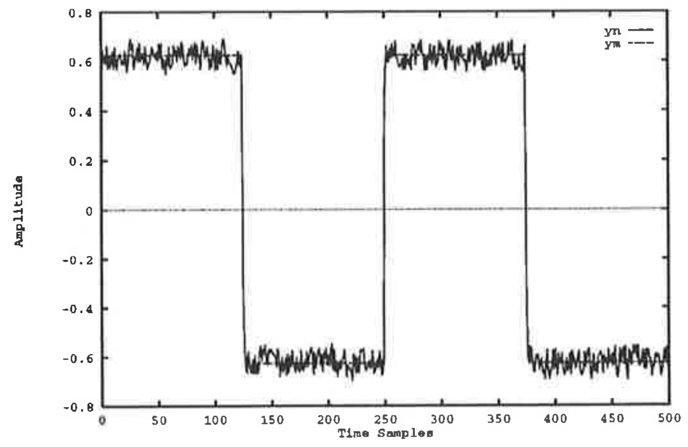


Figure 3.16c: Response of the noise-corrupted plant (y_n) and reference model (y_m) for Example 3.3.8 with $r(k) = 0.5sgn[\sin(\frac{2\pi k}{250})]$

The disturbance-free plant and reference model responses for these inputs are given in Figures 3.17a–3.18b. Neural networks consisting of 2 inputs, 2 hidden layers with 20 and 10 nodes, respectively, and 1 output are used for the controller and emulator. A learning rate of $\eta = 0.1$ is used for both networks. On average, these parameters provided the best results.

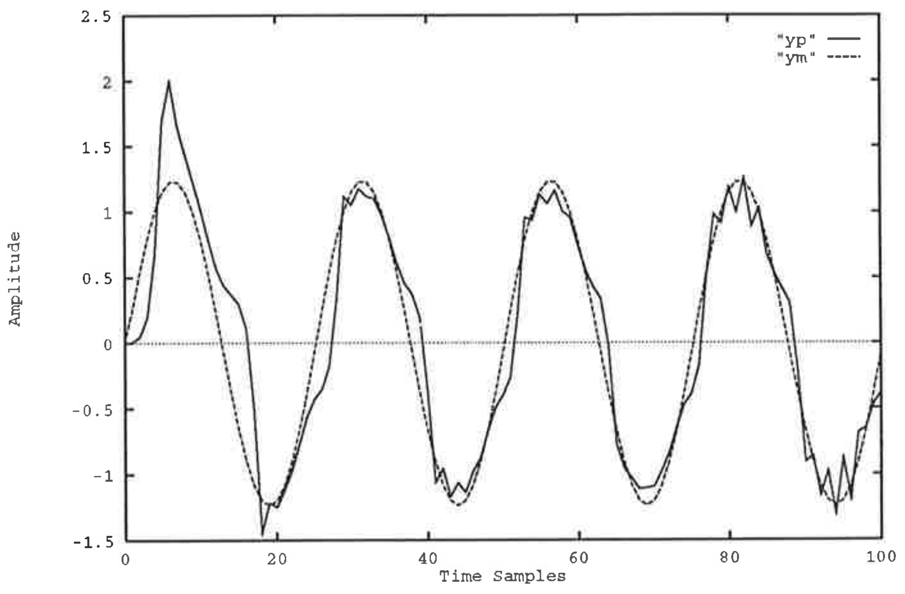


Figure 3.17a: Response of the plant (y_p) and reference model (y_m) for Example 3.3.9 with $r(k) = \sin(\frac{2\pi k}{25})$ for $k = 0 \ldots 100$

Figure 3.17a shows that for the reference input given by equation (3.59a) the initial tracking performance is not very good, as the emulator knows little about the plant. However as Figure 3.17b shows, the performance improves with time.

Figure 3.18a shows that the effect on the tracking performance of a change in reference input at $k = 920$. At the instant the signal changes, the tracking performance deteriorates greatly, but as the neural network adapts to the new input, the performance improves rapidly. Figure 3.18b shows the control input required to produce the plant response shown in Figure 3.18a. As can be seen, significant control effort is required at the instant the input changes and also at the transition in states of the plant output.

Figure 3.19a shows the effect the variable load disturbances shown in Figure 3.19b on the on-line neural control scheme. In the off-line approach, training is undertaken in an

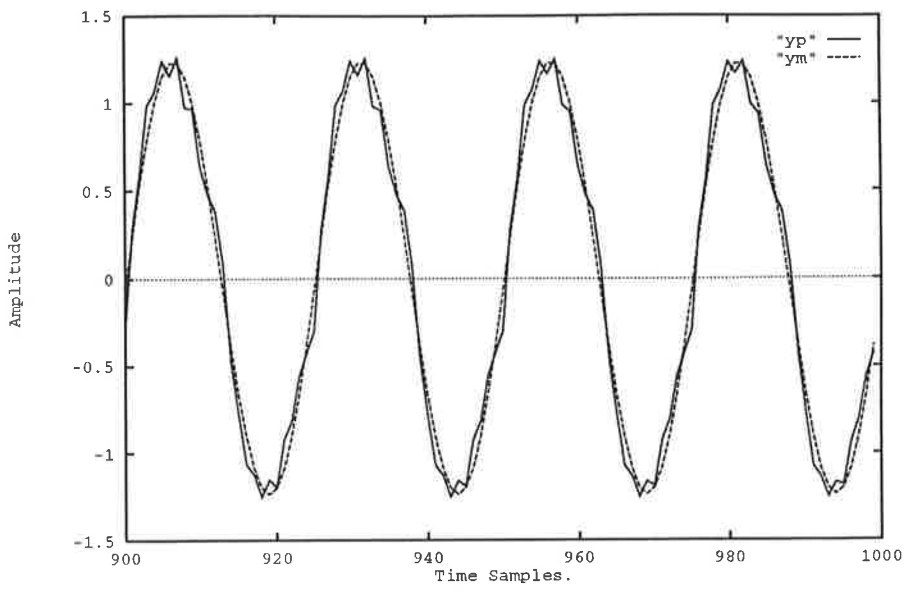


Figure 3.17b: Response of the plant (y_p) and reference model (y_m) for Example 3.3.9 with $r(k) = \sin(\frac{2\pi k}{25})$ for $k = 900 \ldots 1000$

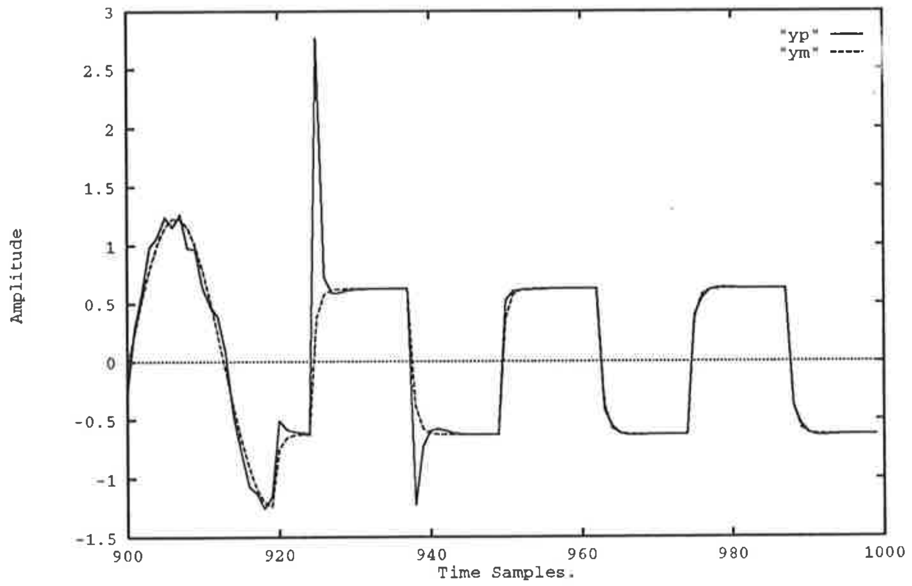


Figure 3.18a: Response of the plant (y_p) and reference model (y_m) for Example 3.3.9 with $r(k) = \sin(\frac{2\pi k}{25})$ for $k < 920$ and $r(k) = 0.5sgn[\sin(\frac{2\pi k}{25})]$ for $k \geq 920$

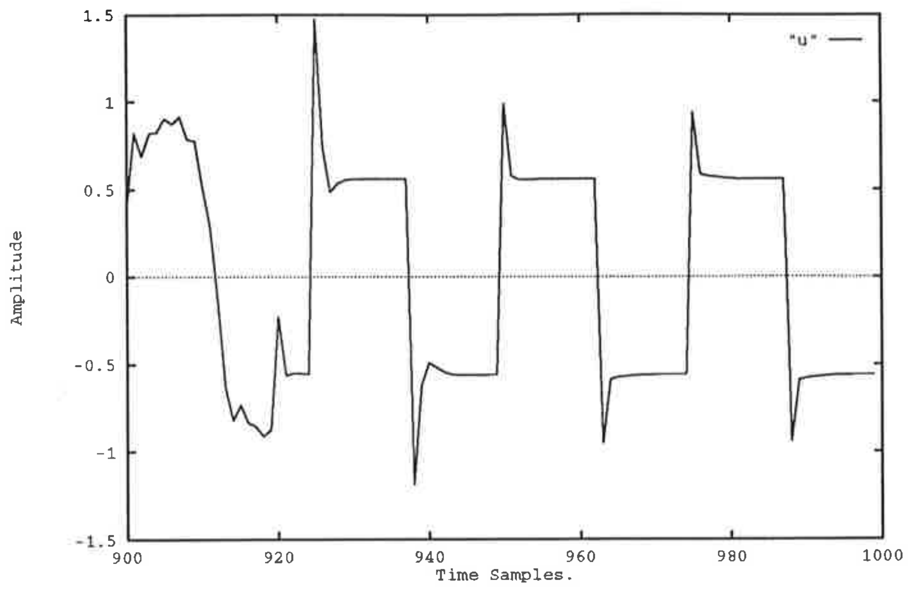


Figure 3.18b: Control input $u(k)$ for Example 3.3.9 with $r(k) = \sin(\frac{2\pi k}{25})$ for $k < 920$ and $r(k) = 0.5sgn[\sin(\frac{2\pi k}{25})]$ for $k \geq 920$

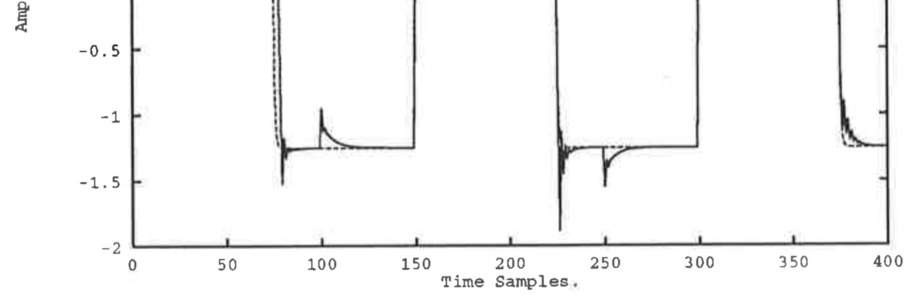


Figure 3.19a: Response of the plant (y_p) subjected to a load disturbance and reference model (y_m) for Example 3.3.9 with $r(k) = sgn[\sin(\frac{2\pi k}{150})]$

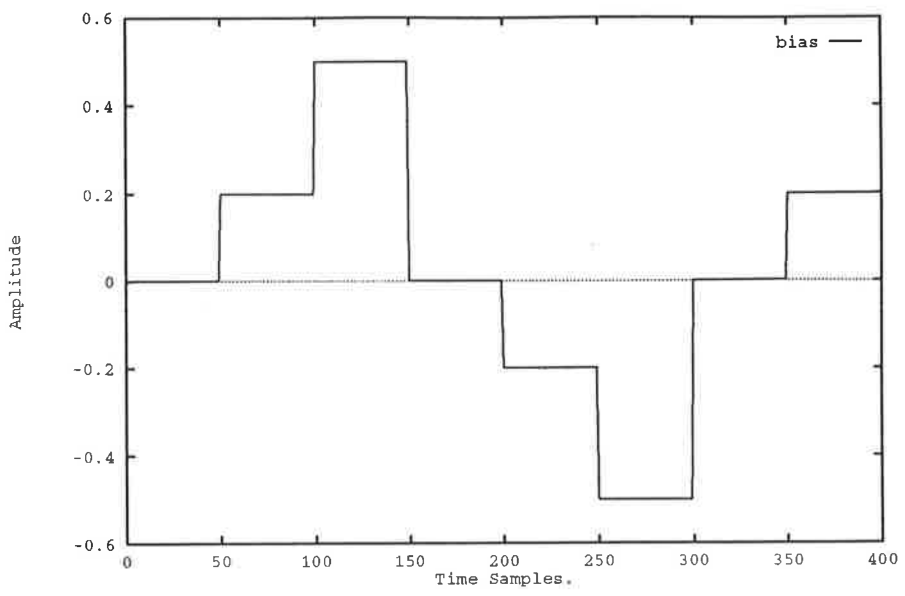


Figure 3.19b: Load disturbance used for Example 3.3.9

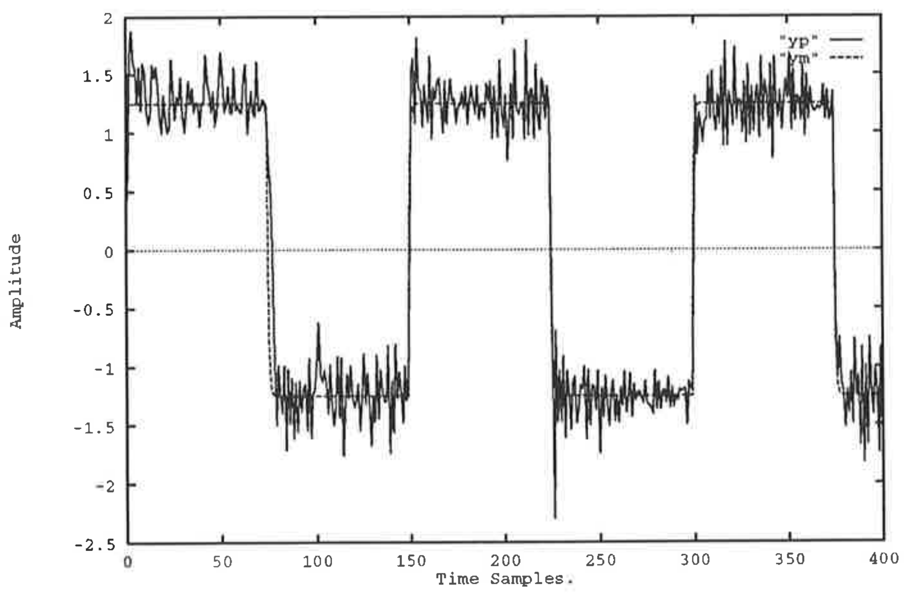


Figure 3.19c: Response of the plant (y_p) and reference model (y_m) for Example 3.3.9 with $r(k) = sgn[\sin(\frac{2\pi k}{150})]$ and with dynamic plant noise, sensor noise and a load disturbance present.

environment not subject to operational load disturbances, and thus a bias compensation needs to be employed. Whereas in the on-line scheme, training is performed on-line with the load disturbance present. Therefore, the controller is trained to compensate for the DC bias and so there is no need to implement a bias compensation scheme. As can be seen, the on-line scheme is capable of dealing with such disturbances very well with only small changes to the tracking performance occurring at the instant at which the load disturbance is changed.

Figure 3.19c shows the effect of dynamic plant noise of the order of 20dB signal-to-noise ratio applied to the control input, sensor noise also of the order of 20dB signal-to-noise ratio and the load disturbance shown in Figure 3.19b. The combination of these disturbances represents a typical practical situation in which noise inherent to the measuring device (sensor noise), noise due to internal components or environmental factors (dynamic noise) and changes in load conditions are present. The results provided in Figure 3.19c show that other than the presence of noise, which can generally be minimised through filtering, the control performance appears satisfactory.

As shown in the above figures, the tracking errors are quite large in the initial phase of operation, but reduce to the order of 5 to 10% after only a few iterations. Thus the performance of the control scheme is quite good despite the fact that a seemingly non-persistent excitation is used in most cases, except perhaps where dynamic (white) noise is added to the input. However, the concept of persistent excitation is not well understood for nonlinear systems. In fact it is stated by Narendra and Annaswamy [167] that the conventional definitions of persistent excitation may be neither necessary nor sufficient to assure asymptotic stability of nonlinear systems. Therefore, unlike linear systems, the success of the adaptive control process does not always rely on a persistently excited input $u(k)$.

Simulation results also indicate that the on-line controller performance can be improved, particularly in the initial phase, by preceding the on-line control stage by a short identification phase in which some persistently excited input is injected into the plant. However, this assumes that the system can be taken off-line. In addition, the initial tracking error can be reduced by introducing a sliding control term [202, 203, 204, 234, 235] which can come into operation when the system leaves a certain prespecified region.

EXAMPLE 3.3.10 A two-input two-output plant belonging to class IV is considered in this example. The system is nonlinear in output and control and the nonlinearities are not separable. The plant consists of cross-coupling in the output parameters.

$$\begin{aligned} y_{p_1}(k+1) &= \frac{y^3_{p_1}(k) + 0.5u_1(k) + 0.5u_3(k) + 0.1y_{p_2}(k)}{1 + y^2_{p_1}(k)} \\ y_{p_2}(k+1) &= \frac{y^3_{p_2}(k) + 0.5u_2(k) + 0.5u_3(k) + 0.1y_{p_1}(k)}{1 + y^2_{p_2}(k)} \end{aligned} \tag{3.60}$$

A stable linear reference model given by the following difference equation is used

$$\begin{aligned} y_{m_1}(k+1) &= 0.5y_{m_1}(k) + 0.5u_1(k) + 0.5u_3(k) \\ y_{m_1}(k+1) &= 0.5y_{m_2}(k) + 0.5u_2(k) + 0.5u_3(k) \end{aligned} \tag{3.61}$$

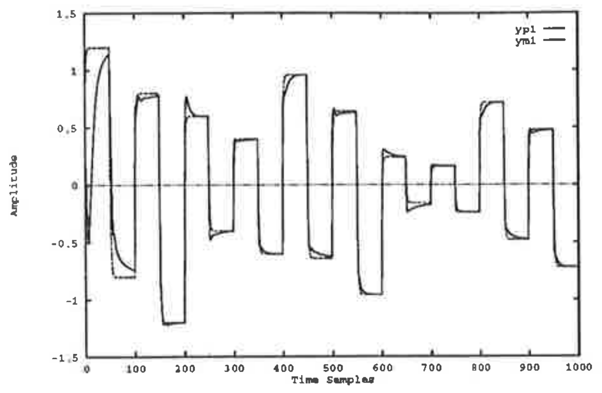
A neural network consisting of 5 inputs $[y_{p_1}, y_{p_2}, u_1, u_2, u_3]$, 2 hidden layers with 10 nodes each, and 3 outputs is used to implement the controller, i.e., $\Omega^3_{5,10,10,3}$. The emulator network belonged to the class $\Omega^3_{5,10,10,2}$. A learning rate of $\eta = 0.2$ and momentum rate of $\alpha = 0.1$ is used for both networks.

The ability of the controller to adapt to a changing input is investigated by considering the following reference input

$$\begin{aligned} r_1(k) &= a(k/200)sgn[\sin(\frac{2\pi k}{100})] \\ r_2(k) &= a(k/200)sgn[\sin(\frac{2\pi k}{200})] \\ r_3(k) &= 0.2a(k/200)sgn[\sin(\frac{2\pi k}{200})] \end{aligned} \tag{3.62}$$

where $a(k/200)$ is a random variable in $\Re[0,1]$ whose value is changed after every 200 samples. The various responses for this input are given in Figures 3.20 and 3.21.

As can be seen, the tracking performance of the on-line control approach is very good for this example, despite the variable nature of the input amplitudes and the complex nonlinearities of the system. Due to the cross-coupling in output, the tracking response for y_{p_2} reflects the changes in the more frequently changing response y_{p_1}. These are observed as small overshoots in the plant response at every 100 samples. However, the



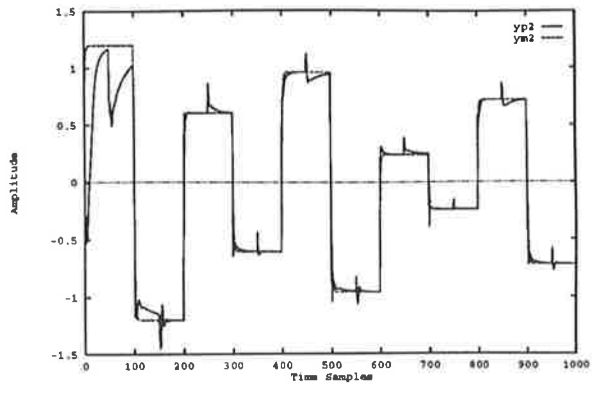
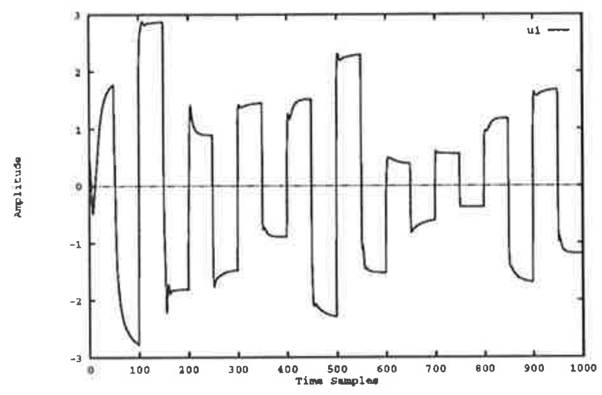
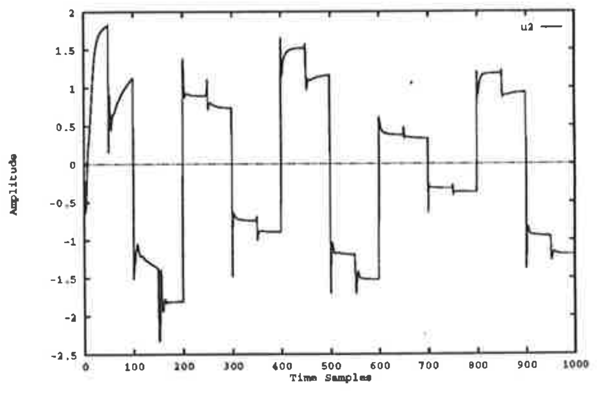


Figure 3.20: Response of the plant ($y_p = [y_{p_1}, y_{p_2}]^T$) and reference model ($y_m = [y_{m_1}, y_{m_2}]^T$) for Example 3.3.10





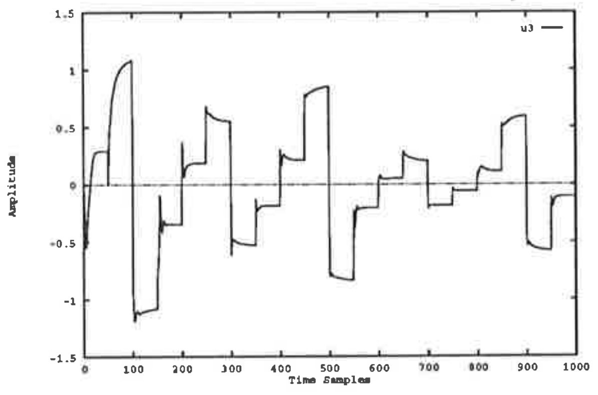


Figure 3.21: Control input ($u = [u_1, u_2, u_3]^T$) for Example 3.3.10

control scheme quickly accounts for this cross-coupling effect by a change in amplitude of control signals u_2 and u_3. As mentioned earlier, cross-coupling of output and/or control variables occurs commonly in practice and thus the ability to compensate for such complexities is an important property of this control approach. Furthermore, these results highlight the general nature of the control approach and demonstrate its ability to handle difficult MIMO systems without any additional information.

EXAMPLE 3.3.11 In this example a marginally stable nonlinear system is considered. Mathematically speaking marginal stability is generally a property of a linear system in the sense that by definition it implies a pole on the imaginary axis of the s-plane or on the unit circle of the z-plane. However, it is known that nonlinear system which are inherently open-loop marginally stable commonly occur in practice. Well known examples are the ship steering problem [127] and the homing missiles guidance problem [78]. Lyapunov theory is often employed to study the stability of nonlinear systems. In particular, Lyapunov's direct method is commonly used. This method involves the construction of a Lyapunov function $V(x)$ for which a number of properties are verified, particularly concerning the rate of change of the function, i.e., $\Delta V(x)$. Essentially, if $V(x)$ is positive definite and $\Delta V(x)$ is negative definite (semidefinite), then the system is asymptotically stable (stable). The closest analogy to marginal stability is the condition for a stable system. For a linear system the stable case above corresponds to a system whose discrete-time eigenvalues are all ≤ 1, whereas the asymptotic stability case above corresponds to a system with all eigenvalues strictly within the unit circle. However, the problem is that the choice of Lyapunov function can affect the result, i.e., the system may be stable according to Lyapunov's direct method when it is actually asymptotically stable [108, 125]. Therefore, a formal definition of a marginally stable *nonlinear* system is still rather elusive. For the purpose of this simulation study, such systems can be considered to be a stable system cascaded by an integrator. This is expressed as follows:

$$\begin{aligned} y_p'(k+1) &= \left(\frac{1}{1-q^{-1}}\right) y_p(k+1) \\ &= y_p'(k) + y_p(k+1) \qquad (3.63a) \end{aligned}$$

where y_p' is the output of the augmented plant which is nonlinear and marginally stable,

y_p is the output of the stable system and q is the shift operator. The stable plant is defined by the following difference equation

$$y_p(k+1) = \frac{y_p(k)y_p(k-1)u(k) + u^3(k) + 0.5y_p(k-1)}{1 + y_p^2(k) + y_p^2(k-1)} \tag{3.64}$$

The on-line neural control procedure presented in this chapter has several advantages over off-line neural network control procedures for controlling marginally stable systems. Firstly, it does not employ a separate, open-loop identification stage, which requires that the system is strictly stable. Secondly, identification of the plant and the control are done simultaneously and in a closed-loop environment. Therefore, the closed-loop control system can compensate for any tendency towards instability exhibited by the plant. These facts suggest that an on-line neural control system is capable of controlling marginally stable systems.

The reference model considered is chosen as

$$y_m(k+1) = 0.2y_m(k) + 0.2y_m(k-1) + r(k) \tag{3.65}$$

The performance of the controller is investigated by observing the step responses of the plant and reference model. The ability of the controller to adapt to a changing operating environment is also investigated by changing the reference input. Therefore the following reference input is used to demonstrate the effectiveness of the controller.

$$r(k) = \begin{cases} s(k-25) & k < 600 \\ \sin(\frac{2\pi k}{100}) & k \geq 600 \end{cases} \tag{3.66}$$

where $s(k)$ is a step at $k = 0$.

The plant and reference model responses for this input are given in Figure 3.22a. Neural networks consisting of 3 inputs, 2 hidden layers with 20 and 10 nodes, respectively, and 1 output are used to implement the controller and emulator.

Figure 3.22a shows the transient response for the neural controller with a learning rate of $\eta = 0.025$. The transient is initially quite oscillatory, but as the plant is a marginally stable one, this is not unexpected. The rise time and settling time of the response are comparable with the specified reference model dynamics. Figure 3.22a also highlights

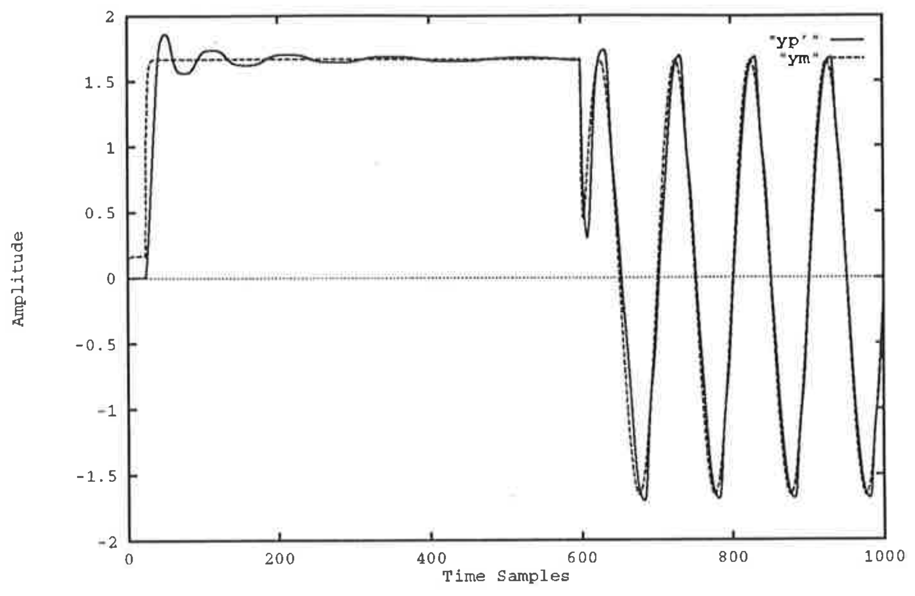


Figure 3.22a: Response of the plant (y_p') and reference model (y_m) for the marginally stable system (3.63a) with $r(k) = s(k-25)$ for $k < 600$ and $\sin(\frac{2\pi k}{100})$ for $k \geq 600$

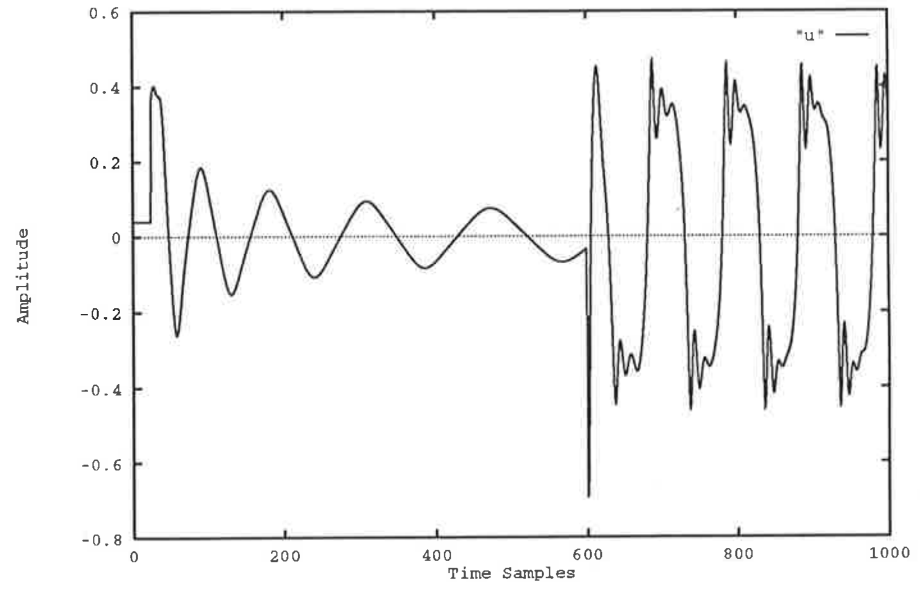


Figure 3.22b: Control input $u(k)$ for the marginally stable system (3.63a) with $r(k) = s(k-25)$ for $k < 600$ and $0.5\sin(\frac{2\pi k}{100})$ for $k \geq 600$

the fact that the system can quickly adapt to a change in its environment, represented here by a change in reference input occurring at $k = 600$. In fact after an initial period in which the tracking performance deteriorates, the system quickly adapts to the new input by adjusting its weights, resulting in an improved tracking performance.

Figure 3.22b shows the control effort required to obtain the responses given in Figure 3.22a. It is worth noting that when the reference input is changed, a great deal of control effort is required .

Whilst investigating the performance of the controller, it is observed that the learning rate used in the backpropagation algorithm had a significant influence on the performance of the system. In particular, it is found that the system performance is relatively insensitive to the choice of learning rate of the emulator (η_i), whilst the controller learning rate (η_c) affects the performance significantly. Several simulations are carried out to investigate the effect of the learning rates. The system considered is the plant given in equation (3.63a). The reference model given in equation (3.65) is used. The step responses of the system for various learning rates are given in Figures 3.23a, 3.23b, 3.23c and 3.23d.

These results highlight the fact that the controller learning rate affects the level of damping of the system. In particular, the results show that as the learning rate is increased, the level of damping decreases and consequently the transient response becomes more oscillatory, but the rise time and settling time are reduced. A learning rate of $\eta_c = 0.001$ represents a system which is overdamped, while a learning rate of $\eta_c = 0.05$ represents an underdamped system. It is found that the system became unstable when the controller learning rate is increased to 0.06.

To explain, consider the physical significance of the learning rate. The learning rate (η) is related to the size of the step taken along the error surface when adjusting the weights. A small learning rate means only small steps are taken down the error surface, resulting in a smooth, virtually continuous path of descent. Therefore, with small learning rates the change in the values of the weights and output from the network are smooth and non-oscillatory. This is reflected in Figure 3.23(a). With larger learning rates, bigger steps are taken and thus the time to reach a minima is smaller. However, with a large learning rate, the corrections to the weights are more severe, resulting in a greater oscillations in

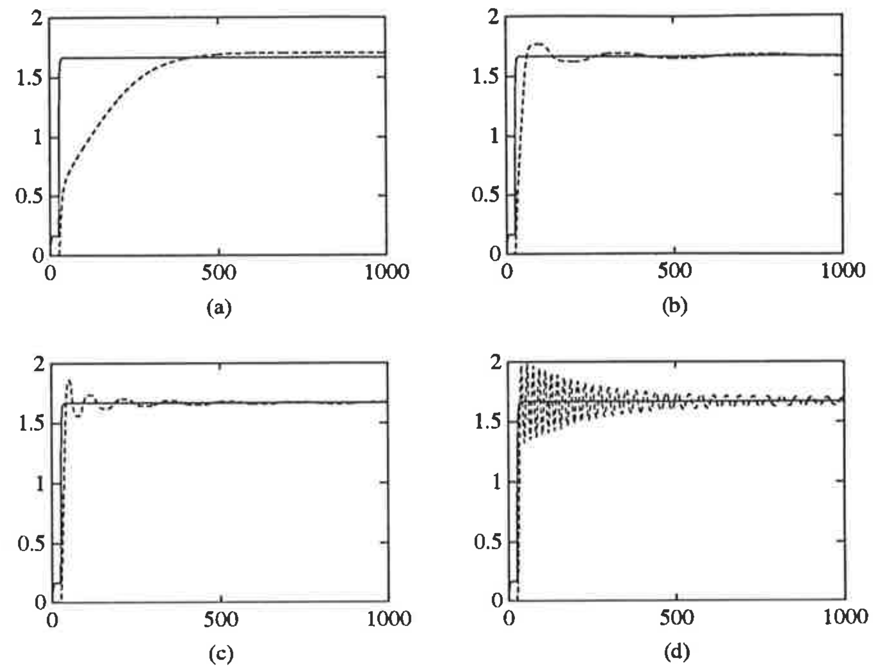


Figure 3.23: Response of the plant (y_p) and reference model (y_m) with $r(k) = s(k-25)$, and (a) $\eta_c = 0.001$ & $\eta_i = 0.05$, (b) $\eta_c = 0.01$ & $\eta_i = 0.05$, (c) $\eta_c = 0.025$ & $\eta_i = 0.05$ and (d) $\eta_c = 0.05$ & $\eta_i = 0.05$

the values of the weights, and subsequently in the network output. This is reflected in Figure 3.23(d).

3.4 Conclusions

A neural network based approach to the model reference adaptive control of nonlinear systems is presented in this chapter. The approach combines the model reference neural adaptive control scheme proposed by Narendra and Parthasarathy [169] with the forward modelling approach of Jordan [95]. A time delay multi-layered neural network is used to generate the appropriate control so that the plant output tracks the output of a reference model. The reference model is chosen to reflect the desired dynamics. A second neural network is used to generate an approximate of the plant Jacobian. It is shown that the Jacobian is necessary for updating the controller weights.

The advantages of this method are:

- Apart from the number of delayed values of plant input and plant output in the nonlinear plant equation, and the relative degree and order of the system, no other *a priori* knowledge of the plant is required, viz the plant can be treated as a "black box". Thus the requirements of [169], namely (*i*) control and output terms are separable, (*ii*) the separate nonlinear functions $f(.)$ and $g(.)$ can be independently identified, and (*iii*) the inverse of operators on control can be *explicitly* approximated, have been relaxed.

- It provides a unified approach to the control of stable nonlinear plants, in which plants belonging to Model IV are treated exactly the same as plants belonging to Models I-III.

An off-line approach and an on-line approach are discussed and the relative merits of both approaches are presented. The controller weight update equation is derived for a general multi-input multi-output system and it is demonstrated that the controller error required to modify the controller network weights can be obtained by backpropagating the tracking error through the neural network which emulates the plant.

The effectiveness of the control approach is demonstrated through a number of simulation examples. For the off-line case, single-input single-output systems belonging to Model III and Model IV are considered. The identification and control of the Model III system is shown to be possible without the need to separately model the nonlinear function in control or its inverse. This is shown to be an advantage over the approach presented in [169]. The Model IV system is also successfully controlled. This represented a significant achievement as in such systems the control is heavily embedded in the nonlinearities of the system. A multi-input multi-output system belonging to Model IV is also considered to highlight the generality of the approach. The effects of three types of disturbances are considered, namely load disturbances represented by a DC bias on the output of the plant, sensor noise represented by an additive white noise process on the output of the plant and dynamic plant noise represented by a white noise process on the input of the plant. A bias compensation scheme is developed to account for the DC bias and this is shown to be effective for varying load conditions. The control scheme is also shown to be effective in the presence of dynamic and sensor noise.

In the on-line case, the same SISO system belonging to Model III is considered. Whilst an equivalently small tracking error is not achieved, the scheme is still able to produce an effective control. The scheme is also shown to be effective in dealing with systems subject to load disturbances, dynamic plant noise and sensor noise without the need for any additional compensation schemes. A MIMO system belonging to Model IV is also considered to demonstrate the effectiveness of the approach. One of the major advantages with the on-line approach is shown to be its ability to effectively control marginally stable systems. This is a significant result, as many of the existing neural control approaches are unable to deal with such systems because they employ open-loop identification.

The two major shortcomings of the approach presented in this chapter are that (1) there is no guarantee that the tracking error converges to zero, and (2) the control scheme is not shown to be stable. These are perhaps two of the most important issues associated with any control scheme and must be addressed in order that the scheme be practically viable. Whilst the simulation results show that the tracking error tends to zero rapidly, this is, to some extent, due to a fortuitous choice of reference model, plant and reference inputs, rather than any theoretical guarantees. Therefore results which guarantee the convergence of the tracking error are desired. The problem of stability is particularly difficult when artificial neural networks are used for identification or control and the system is nonlinear. Unlike linear systems, it is difficult to derive simple algebraic conditions to ensure stability of the overall system. In the next chapter an enhancement of the neural adaptive control scheme is proposed which enables the derivation of conditions under which these issues are addressed.

Chapter 4

Stable Neural Adaptive Control

4.1 Introduction and Overview

The most important property of any control system is its stability. An unstable control system is typically useless and potentially dangerous. Qualitatively, the concept of stability deals with the effect of unknown disturbing forces on a dynamical system. If the effect of the disturbing force is insignificant, such that a system starting near its desired operating point stays at that point forever after, then the system is considered stable. Since these disturbing forces are present in most physical systems, the study of stability properties is of major theoretical and practical importance. The concept of stability has been researched extensively over the past century and so as a result, a great deal of literature dealing with this issue is available [167, 205, 218, 237]. In particular, the most commonly used approach for studying the stability of control systems is Lyapunov theory [174, 218].

Another important concept in the design of control systems is the convergence property of the output error. As discussed in detail in earlier chapters, the neural adaptive control scheme presented in this thesis is based on the model reference adaptive control methodology. The fundamental principle in MRAC schemes is to design a control such that the tracking error (the difference between the plant output and the desired reference model output) converges to zero, or at least an arbitrarily small value.

In the neural control scheme presented in the previous chapter, the tracking error seems to converge to zero for the simulation examples. However, this is, to some extent, due to fortuitous choices of reference model, reference inputs and plant structure rather than any theoretical guarantees. Therefore, theoretical results are required to ensure the convergence of the tracking error. To further ensure the practical and theoretical viability of the proposed neural control scheme, theoretical results are also required to guarantee the stability of the overall system.

However, in a control system with neural networks it is difficult to prove properties such as stability. The main reason is the mathematical difficulties associated with the use of highly nonlinear neural network controllers in complex nonlinear systems. Some progress has been made in this area and some important theoretical results are beginning to emerge, but the overall knowledge and development of stability techniques for neural control systems is still quite immature.

Amongst the prominent research done in this area is the work by Chen and Khalil [30] and Chen and Liu [31]. In both of these works, local convergence theorems are given for the tracking error. These are provided for discrete-time systems and continuous-time systems, respectively, using a linearizing feedback neural network control scheme. The issue of designing a stable neural control scheme and the convergence of the tracking error to a neighbourhood of zero has been recently addressed by Polycarpou and Ioannou [186], Sanner and Slotine [202, 203, 204], and Tzirkel-Hancock and Fallside [234, 235]. In all of these approaches, results of Lyapunov stability theory are used to adjust the neural network weights. Furthermore, a sliding control is also employed to help provide global stability. However, in these papers certain restrictive assumptions about the plant are made.

In this chapter, an enhanced neural network based model reference control scheme is proposed. As with the scheme discussed in the previous chapter, this enhanced neural control scheme is formulated for a general discrete-time multi-input multi-output nonlinear system. Furthermore, the general nonlinear systems considered are non-affine in control and the control may be heavily embedded in the nonlinearities of the system. Weak assumptions regarding the order, relative degree and number of delay terms in the plant output and control variable are made. Output feedback is also assumed. The concepts of stability and convergence of the tracking error for the neural adaptive control scheme are addressed through the introduction of an enhanced reference model. A subsequent enhancement to the model reference scheme is made which enables the derivation of sufficient conditions to guarantee the convergence of the tracking error. Furthermore, Lyapunov's direct method is used to demonstrate that the overall system is stable. The basic strategy of the proposed scheme is to generate a control input via the neural network controller such that the plant output is nearest, in some norm sense, to a desired plant output generated by the enhanced reference model. Therefore, a modified controller neural network weight update equation is proposed to achieve the desired control.

This chapter is structured as follows. A detailed discussion of the concept of an enhanced reference model is provided in Section 4.2. The origins of this approach are discussed, particularly in relation to the optimal decision control strategy of Spong *et al.* [225] and the linear programming approach of Rehbock *et al.* [190, 192]. The resultant enhancement of the neural network based model reference control scheme presented in the previous chapter is provided in Section 4.3. Two alternative sufficient conditions are derived to ensure the convergence of the tracking error. The corresponding proofs are also furnished to demonstrate the convergence results. Lyapunov's direct method is used to guarantee the stability of the closed-loop system. A modification to the weight update equation is provided to achieve the desired control. Several simulation studies are then considered in Section 4.4 to demonstrate the effectiveness of the proposed scheme. In particular the performance of the scheme for systems subject to a range of disturbances and other non-idealities is investigated.

4.2 Enhanced Reference Model

The concept of introducing a desired velocity function in a reference model was first introduced by Spong *et al.* [225]. In this paper, the problem of tracking a desired trajectory in the state space of an n-link robotic manipulator subject to bounds on the allowable input torques is considered. The controller is designed using an optimal decision strategy (ODS) which belongs to a class of pointwise optimal control strategies. The aim of the ODS approach shown in [225] is to minimise the Euclidean norm of the difference between the actual vector of instantaneous joint accelerations and a desired joint acceleration vector. As shown in [14], the ODS technique is a special case of the optimal aim strategy originally proposed by Barnard [13]. The ODS technique is claimed to have several advantages over classic optimal control techniques which are based on calculus of variations and yield a pair of control and state time histories that are optimal with respect to a performance index evaluated over a specified time interval. The solution to the classical optimal control problem requires a two-point boundary value problem to be solved. This computational complexity means that it is infeasible to implement on-line schemes based on classical optimal control for nonlinear dynamical systems. The ODS scheme is a pointwise optimisation approach which optimises the present state of the systems without regard to future events. This technique has the advantage of allowing input and/or state variable constraints and it does not require the solution of a two-point boundary problem. Therefore, due to its simplicity, on-line implementation of the ODS scheme is feasible.

The approach presented by Spong *et al.* [225] considers a nonlinear dynamic system of the form

$$\dot{y}_p(t) = f(y_p(t)) + G(y_p(t))u(t); \qquad y_p(0) = y_0 \tag{4.1}$$

subject to the constraints $u_i^{min} \leq u_i \leq u_i^{max}$, $i = 1, \ldots, m$ where $y_p(t) \in \mathbf{R}^n$ is the output vector, $u \in \mathbf{R}^m$ denotes the control vector, u_i^{min} and u_i^{max} are given scalars, and $f : \mathbf{R}^n \longrightarrow \mathbf{R}^n$ and $G : \mathbf{R}^m \longrightarrow \mathbf{R}^{nxm}$ are given smooth nonlinear functions. In addition, for any $x \in \mathbf{R}^n$, $G(x)^T G(x)$ is positive definite. Let $y_m(t)$ be the desired trajectory which the output of the dynamical system must track. Since the control vector is bounded, the domain of the actual trajectory $y_p(t)$ is restrained by the set $C(y_p)$ of velocity vectors where

$$\begin{aligned} C(y_p) &= \{z \in \mathbf{R}^n \mid z = f(y_p) + g(y_p)u, u \in \Omega\} \\ \Omega &= \{u \in \mathbf{R}^m \mid u_i^{min} \leq u_i \leq u_i^{max}, i = 1, \ldots, m\} \end{aligned} \tag{4.2}$$

It is then assumed that the function $v(t) \in \mathbf{R}^n$ is specified *a priori* as a function of $y_p(t)$ and $y_m(t)$. The function $v(t)$ is referred to as the "desired velocity in state space" or "desired velocity" for short. However, as pointed out in [225], v does not represent the actual joint velocities for a given manipulator. The ODS method is used to obtain an optimal control law $u^*(t)$ such that the corresponding instantaneous velocity $\dot{y_p}^* = f(y_p(t)) + g(y_p(t))u^*(t)$ is nearest to $v(t)$, hence the name "desired velocity". More precisely, $u^*(t)$ is chosen to minimise the performance index

$$J(y_p(t), v(t)) = \min_{u \in \Omega}\{[\dot{y_p}(t) - v(t)]^T Q[\dot{y_p}(t) - v(t)]\} \tag{4.3}$$

where Q is an $n \times n$ symmetric positive definite matrix. The validity of the scheme and the properties of the resultant closed-loop system depend on the choice of the desired velocity function $v(t)$, which is selected *a priori* by the designer. Several choices of the desired velocity function are examined in [225]. It is stated that if v is chosen to be linear in some coordinate system $\zeta = T(y)$, then a feedback linearisation approach, as discussed in Chapter 1, results. By choosing $v(y) = -y$, the optimal aim approach of Barnard [13] is obtained. If v is chosen to point to a sliding hyperplane in $\mathbf{R}^n$ then the so-called equivalent control approach is implemented. The aim of the approach presented in [225] is to align the closed-loop system with v as closely as possible in a least square sense. Thus, the desirable properties of the velocity function such as stability are translated into the overall closed-loop performance of the system. The form of the desired velocity function considered for controlling a robot manipulator is

$$v(t) = \dot{y}_m(t) + A(y_p(t) - y_m(t)) \tag{4.4}$$

where $\dot{y}_m$ and y_m are generated from the reference model

$$\dot{y}_m(t) = Ay_m(t) + Br(t) \tag{4.5}$$

in which A is an $n \times n$ Hurwitz matrix. With this form of velocity function, a general control scheme can be realised by the block diagram of Figure 4.1. An *enhanced reference model* is introduced to represent the reference model and the desired velocity function. For an ODS scheme, the control block in this figure represents the selection of a control input $u^*(t)$ such that the performance index (4.3) is minimised. The approach realised in this block diagram highlights the model following character of this choice of velocity function. This approach is also adopted by Lee *et al.* [123] for an aircraft terrain following system.

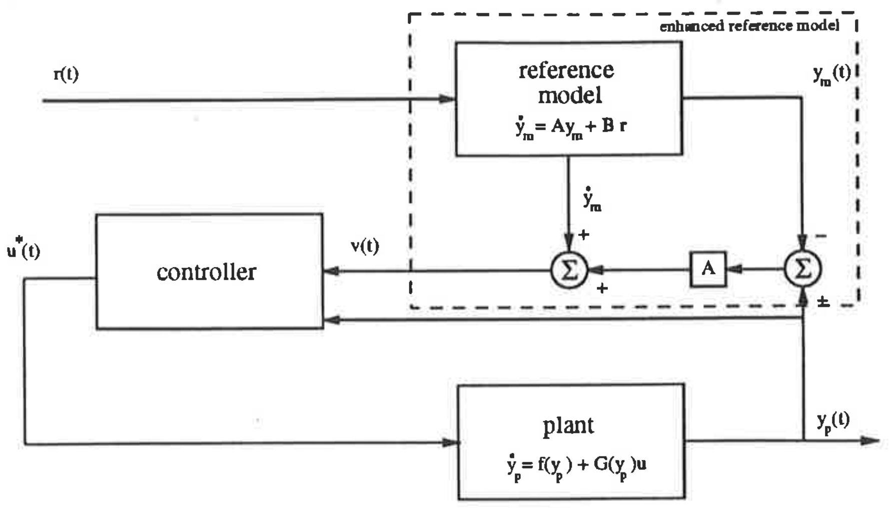


Figure 4.1: Model following scheme using an enhanced reference model

An alternative ODS method is proposed for discrete-time systems by Rehbock *et al.* [190] and for continuous-time systems by Rehbock *et al.* [192]. In the approach for a discrete-time system, the performance index $|y_p(k+1) - v(k)|_1$ is used instead of $[\dot{y}_p(t) - v(t)]^T Q[\dot{y}_p(t) - v(t)]$, where $|.|_1$ represents the l_1 norm. Thus, this approach avoids having to choose the appropriate matrix Q. Furthermore, this change allows the pointwise optimisation problem to be converted into a linear programming problem which is then solved using the well known simplex method [191]. For this case, the desired velocity function is given by

$$v(k) = y_m(k+1) + A(y_p(k) - y_m(k)) \tag{4.6}$$

and the reference model is given by

$$y_m(k+1) = s(y_m(k)) + r(k) \tag{4.7}$$

where $s : \mathbf{R}^n \longrightarrow \mathbf{R}^n$ is an appropriate function. This choice of desired velocity is shown to result in a stable model following system with or without control constraints. The approach proposed in [190] is applied to the ship steering problem and the aircraft terrain tracking problem. It is found to be very effective in controlling these systems which are nonlinear and affine in control. Note that in the discrete-time representation, it is less obvious that $y_m(k+1)$ of (4.7) represents the instantaneous velocity of the model output. Consequently, it may cause confusion when $v(k)$ is termed the desired velocity. To avoid this confusion, $v(k)$ will henceforth be denoted $y'_m(k+1)$ and will be referred to as the enhanced model output.

4.3 Enhanced Model Reference for Neural Control

In order to ensure stability and the convergence of the tracking error, an enhancement is made to the neural network based model reference adaptive control scheme which is presented in Chapter 3. The enhancement involves introducing an enhanced reference model of the form discussed above for model reference adaptive control schemes. This enhancement can be mathematically expressed as follows:

Consider a plant governed by the following nonlinear difference equation

$$\begin{aligned} y_p(k+1) &= f(y_p(k), \cdots, y_p(k-l+1); u(k), \cdots, u(k-m+1)) \\ y_p(0) &= y_{p_0} \qquad\qquad \forall k \in \mathbb{N} \end{aligned} \tag{4.8}$$

where $y_p \in \mathbf{R}^n$ is the output vector, $u \in \mathbf{R}^r$ is the control vector, $f : \mathbf{R}^{n \times l} \times \mathbf{R}^{r \times m} \rightarrow \mathbf{R}^n$ is a smooth nonlinear function, $y_{p_0} \in \mathbf{R}^n$ is the initial output vector, k is the time index, $\mathbb{N}$ is the set of natural numbers, and m and l are the number of delayed values of plant input and plant output, respectively.

Consider a stable reference model governed by

$$\begin{aligned} y_m(k+1) &= f_m(y_m(k), \cdots, y_m(k-d+1); r(k)) \\ y_m(0) &= y_{m_0} \end{aligned} \tag{4.9}$$

where $y_m \in \mathbf{R}^n$ is the reference model output vector, $r \in \mathbf{R}^n$ is the piecewise continuous and bounded reference input, $f_m : \mathbf{R}^{n \times d} \times \mathbf{R}^r \rightarrow \mathbf{R}^n$ is usually a linear function, d is the number of delayed values of reference model output with $d \geq l$ and $y_{m_0} \in \mathbf{R}^n$ is a given initial output vector for the reference model.

The control strategy is to find a feasible control input

$$u(k) = g(y_p(k), \cdots, y_p(k-l+1); u(k-1), \cdots, u(k-m+1); r(k); W_c) \tag{4.10}$$

where $g : \mathbf{R}^{n \times l} \times \mathbf{R}^{r \times m} \rightarrow \mathbf{R}^r$ is a neural network parameterised by the set of weights W_c, such that the corresponding plant output is nearest, in some norm sense, to a desired output $y'_m(k+1)$. This output variable is to be specified *a priori* as a function of the plant output and the reference model output. An appropriate way to generate $y'_m(k+1)$ is

$$y'_m(k+1) = y_m(k+1) - A(y_m(k) - y_p(k)) \tag{4.11}$$

where A is an $n \times n$ Hurwitz (stable) matrix. The combination of the reference model (4.9) and the enhanced output (4.11) is called the *enhanced reference model* in Figure 4.2.

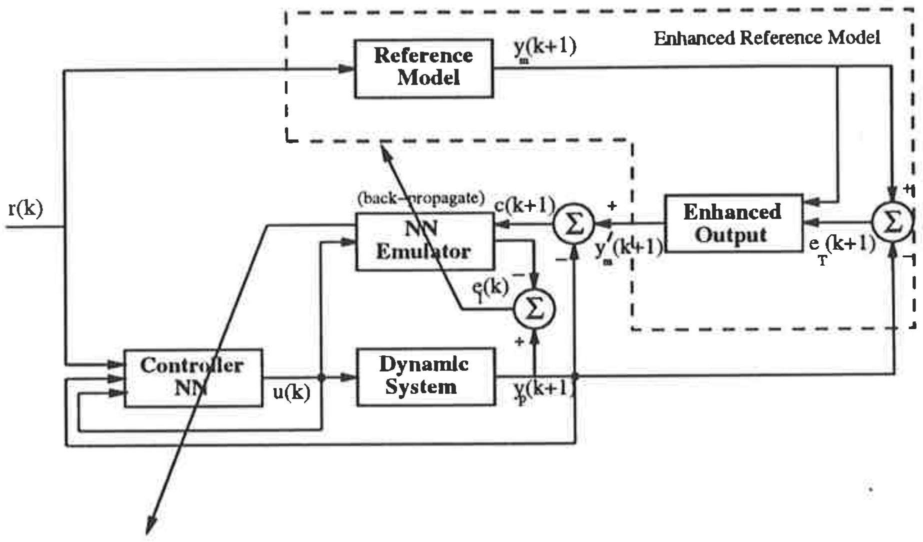


Figure 4.2: Block diagram of the neural control scheme with an enhanced reference model

4.3.1 Convergence of the Tracking Error

The aim of the proposed neural control scheme is to generate a suitable control $u(k)$ such that the cost function given by

$$E = \frac{1}{2}\sum_{k=0}^{N-1}(y'_m(k+1) - y_p(k+1))^T(y'_m(k+1) - y_p(k+1)) \tag{4.12}$$

is minimised, where N is the number of samples.

The motivation for using $y'_m(k+1)$ in the cost function is that sufficient conditions to ensure the convergence of the tracking error can be established. To demonstrate this, firstly define the tracking error as

$$\begin{aligned} e_T(k+1) &= y_m(k+1) - y_p(k+1) \\ e_T(0) &= y_m(0) - y_p(0) = e_0 \qquad \forall k \in \mathbb{N} \end{aligned} \tag{4.13}$$

A residual $c(k+1)$ which represents the difference between the plant output and enhanced model response, is then introduced, i.e.,

$$c(k+1) = y'_m(k+1) - y_p(k+1) \tag{4.14}$$

It follows that

$$\begin{aligned} e_T(k+1) &= A(y_m(k) - y_p(k)) + c(k+1) \\ &= Ae_T(k) + c(k+1) \end{aligned} \tag{4.15}$$

The principal aim of a model reference adaptive control scheme is to ensure the convergence of the tracking error. It can be shown that with the proposed enhancement, the neural network based MRAC scheme presented here will satisfy this aim under certain conditions. This issue is addressed in the following theorems.

THEOREM 4.3.1 *The tracking error $e_T(k)$ defined in (4.13) converges to zero if the residual $c(k+1)$ defined in (4.14) satisfies the following inequality*

$$\|c(k+1)\| < (1 - K\lambda_{max})\|e_T(k)\| \qquad \forall k \in \mathbb{N} \tag{4.16}$$

where $\|.\|$ is the l_2-norm, $K = \|S\|\ \|S^{-1}\|$, λ_{max} is the largest eigenvalue of A, and S is the matrix of eigenvectors of A.

Proof: Consider the matrix A. As it is stable, its eigenvalues are within the unit circle on the z-plane. Also assume that A is chosen such that it has real and distinct eigenvalues. Therefore

$$A = S\Lambda S^{-1} \tag{4.17}$$

where Λ is a diagonal matrix with the eigenvalues of A on the main diagonal and S is the corresponding matrix of eigenvectors of A.

Taking the Euclidean 2-norm, one gets

$$\begin{aligned} \|A\| &= \|S\Lambda S^{-1}\| \\ &\leq \|S\|\ \|\Lambda\|\ \|S^{-1}\| \\ &\leq \|S\|\ \|S^{-1}\|\lambda_{max} \\ &= K\lambda_{max} \end{aligned} \tag{4.18}$$

where $K = \|S\| \; \|S^{-1}\|$, and λ_{max} is the largest eigenvalue of A.

Taking the Euclidean 2-norm of the tracking error (4.15) one gets

$$\begin{aligned} \|e_T(k+1)\| &= \|Ae_T(k) + c(k+1)\| \\ &\leq \|A\| \; \|e_T(k)\| + \|c(k+1)\| \\ &\leq K\lambda_{max}\|e_T(k)\| + \|c(k+1)\| \end{aligned} \tag{4.19}$$

For the tracking error $e_T(k)$ to converge to zero as $k \to \infty$, it is sufficient that

$$\|e_T(k+1)\| - \|e_T(k)\| < 0. \tag{4.20}$$

This is achievable when the residual $\|c(k+1)\|$ satisfies the following

$$\|c(k+1)\| < (1 - K\lambda_{max})\|e_T(k)\| \tag{4.21}$$

■ ■ ■

An alternative sufficient condition on $c(k+1)$ to ensure the convergence of the tracking error is presented in the following theorem.

THEOREM 4.3.2 *The tracking error $e_T(k)$ converges to zero if the residual $c(k+1)$ satisfies the following inequality*

$$\|c(k+1)\| \leq \frac{1 - \lambda_{max}}{K} \|e_T(k)\| \qquad \forall k \in \mathbb{N} \tag{4.22}$$

where $\|.\|$ is the l_2-norm, $K = \|S\| \; \|S^{-1}\|$, λ_{max} is the largest eigenvalue of A, and S is the matrix of eigenvectors of A.

Proof: Consider the tracking error defined by equation (4.15), i.e.,

$$\begin{aligned} e_T(k+1) &= Ae_T(k) + c(k+1) \\ e_T(0) &= e_0 \end{aligned} \tag{4.23}$$

Its solution takes the form

$$e_T(k) = A^k e_0 + \sum_{i=1}^{k} A^{k-i} c(i) \tag{4.24}$$

Taking l_2-norms of both sides of equation (4.24) yields

$$\|e_T(k)\| \leq \|A^k\| \; \|e_0\| + \sum_{i=1}^{k} \|A^{k-i}\| \; \|c(i)\| \tag{4.25}$$

Assume that A is chosen such that it has real and distinct eigenvalues. Therefore

$$A^k = S\Lambda^k S^{-1} \tag{4.26}$$

where Λ is a diagonal matrix with the eigenvalues of A on the main diagonal and S is the corresponding matrix of eigenvectors of A. It follows that

$$\begin{aligned} \|A^k\| &\leq \|S\| \; \|\Lambda^k\| \; \|S^{-1}\| \\ &\leq K\lambda_{max}^k \end{aligned} \tag{4.27}$$

where $K = \|S\| \; \|S^{-1}\|$ and λ_{max} is the largest eigenvalue of A. From (4.25) and (4.27), one gets

$$\|e_T(k)\| \leq K\lambda_{max}^k \; \|e_0\| + \sum_{i=1}^{k} K\lambda_{max}^{k-i} \; \|c(i)\| \tag{4.28}$$

Consider that inequality (4.22) holds. Therefore (4.28) becomes

$$\|e_T(k)\| \leq K\lambda_{max}^k \; \|e_0\| + \sum_{i=1}^{k} m\lambda_{max}^{k-i+1} \; \|e_T(i-1)\| \tag{4.29}$$

where $m < \frac{1-\lambda_{max}}{\lambda_{max}}$. Multiplying (4.29) by λ_{max}^{-k} and defining $\gamma(k) = \lambda_{max}^{-k}\|e_T(k)\|$, yields

$$\gamma(k) \leq K \; \|e_0\| + \sum_{i=1}^{k} m\gamma(i-1) \tag{4.30}$$

Expanding (4.30), one gets

$$\begin{aligned} \gamma(k) &\leq K \; \|e_0\| + m\gamma(0) + m\gamma(1) + m\gamma(2) + \cdots + m\gamma(k-1) \\ &\leq K \; \|e_0\| + m\gamma(0) + m[K \; \|e_0\| + m\gamma(0)] + m[K \; \|e_0\| + m\gamma(0) + m\gamma(1)] + \cdots \\ &\quad + m[K \; \|e_0\| + m\gamma(0) + m\gamma(1) + \cdots + m\gamma(k-2)] \end{aligned} \tag{4.31}$$

Recursively substituting (4.30) into (4.31) yields

$$\gamma(k) \leq K \; \|e_0\| \; B(m) + m\gamma(0) \; B(m) \tag{4.32}$$

where

$$\begin{aligned} B(m) &= m^{k-1} + (k-1)m^{k-2} + \frac{(k-1)(k-2)}{2}m^{k-3} + \cdots + \binom{k-1}{i}m^{k-1-i} + \cdots \\ &\quad + \frac{(k-1)(k-2)}{2}m^2 + (k-1)m + 1 \end{aligned} \tag{4.33}$$

Using the binomial theorem, (4.32) can written as

$$\gamma(k) \le [K\ \|e_0\| + m\gamma(0)](m+1)^{k-1} \tag{4.34}$$

With $\gamma(0) = \|e_0\|$

$$\gamma(k) \le (K+m)\ \|e_0\|\ (m+1)^{k-1} \tag{4.35}$$

Multiplying (4.35) by λ_{max}^k, one gets

$$\|e_T(k)\| \le (K+m)\ \|e_0\|\ (m+1)^{k-1}\lambda_{max}^k \tag{4.36}$$

or

$$\|e_T(k)\| \le \frac{K+m}{m+1}\ \|e_0\|\ [(m+1)\lambda_{max}]^k \tag{4.37}$$

As $m < \frac{1}{\lambda_{max}} - 1$, then $(m+1)\lambda_{max} < 1$. Thus as $k \to \infty$, $\|e_T(k)\| \to 0$

■ ■ ■

REMARK 4.3.1 *Theorems 4.3.1 and 4.3.2 and their corresponding proofs provide the theoretical guarantees for the convergence of the tracking error of the proposed neural adaptive control scheme. Details on how the resultant sufficient conditions (4.16) and (4.22) are used in practice will be presented in Section 4.3.4.*

REMARK 4.3.2 *In theory the tracking error converges to zero when either condition (4.16) or (4.22) is satisfied. However, in practice, perfect tracking is unlikely due of the practical difficulty in achieving a perfect neural network approximation of the plant dynamics and the unknown control function. Therefore it will be sufficient for the tracking error to converge to ε_T, a small finite value.*

REMARK 4.3.3 *It may appear from Theorems 4.3.1 and 4.3.2 that the convergence of the tracking error is independent of the neural networks. However, a number of implicit assumptions are made which must be met in order that the sufficient conditions (4.16) and (4.22) are valid, namely (i) a solution to the control problem exists, i.e., the system is controllable and therefore a function $g(.)$ exists, (ii) the neural network controller is capable of approximating this function, and (iii) the parameters of the neural networks are chosen such that they perform the approximation role satisfactorily. If these assumptions do not hold, then the sufficient conditions (4.16) and (4.22) will not be met and convergence may not be achieved.*

In many of the simulation examples considered multi-input multi-output systems are used. For multiple-output systems, a diagonal matrix A is considered for the sake of simplicity. The following theorem provides some guidelines on how to choose the diagonal elements of A.

THEOREM 4.3.3 *For an $n \times n$ diagonal matrix A, the condition for the existence of an upper bound on the magnitude of the elements of A in order for Theorem 4.3.1 to hold is*

$$a_{ii} < \frac{1}{n} \qquad\qquad i = 1, \ldots, n \tag{4.38}$$

Proof: Recall from the proof to Theorem 4.3.1 that the stability matrix A can be represented as follows

$$A = S \Lambda S^{-1} \tag{4.39}$$

where Λ is a diagonal matrix with the eigenvalues of A on its main diagonal and S is the corresponding matrix of eigenvectors. For a diagonal matrix, S is the n-dimensional identity matrix and $\Lambda = A$. The sufficient condition given in Theorem 4.3.1 is

$$\|c(k+1)\| < (1 - K\lambda_{max})\|e_T(k)\| \tag{4.40}$$

With the Euclidean 2-norm defined as

$$\|B\| = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} B_{ij}^2 \right)^{\frac{1}{2}}, \tag{4.41}$$

one obtains $\|S\| = \sqrt{n}$ and $K = n$. Condition (4.40) therefore becomes

$$\|c(k+1)\| < (1 - n\lambda_{max})\|e_T(k)\| \tag{4.42}$$

As $(1 - n\lambda_{max}) > 0$, the maximum eigenvalue of A is $\lambda_{max} < \frac{1}{n}$. As mentioned earlier, the eigenvalues of a diagonal matrix A, denoted λ_i, are the diagonal elements a_{ii}. Hence

$$a_{ii} < \frac{1}{n} \tag{4.43}$$

■ ■ ■

REMARK 4.3.4 *The sufficient condition (4.16) places greater restrictions on the choice of the elements of the diagonal matrix A than condition (4.22), for which the only requirements are that $a_{ii} < 1$. Therefore, if the matrix A is chosen to satisfy Theorem 4.3.3, then the restrictions on A imposed by condition (4.22) will also be met.*

The above approach is derived for a general multivariable system. However, for a single-input single-output system ($n = 1, r = 1$), the stability matrix is 1×1. For SISO systems, a stability constant (β) is defined such that $A = (\beta) \in \mathbf{R}^{1\times1}$, where $0 < \beta < 1$. Therefore the tracking error becomes

$$e_T(k+1) = \beta e_T(k) + c(k+1) \tag{4.44}$$

and the stability conditions on the residual become

$$\|c(k+1)\| < (1-\beta)\|e_T(k)\| \tag{4.45}$$

or

$$\|c(k+1)\| \leq \frac{(1-\beta)}{\beta}\|e_T(k)\|, \tag{4.46}$$

respectively.

■ ■ ■

4.3.2 System Stability

Stability of the overall system is an important property which needs to be guaranteed in order that the control scheme fulfills its original aims. Lyapunov stability theory may be used to guarantee stability. This involves firstly choosing a Lyapunov function candidate, and then selecting the control strategy to ensure that the hypotheses of a particular stability theorem are satisfied. More precisely, consider the tracking error

$$e_T(k+1) = Ae_T(k) + c(k+1); \quad e_T(0) = e_0 \tag{4.47}$$

Let a possible Lyapunov function candidate be

$$V(e_T(k)) = e_T(k)^T P e_T(k) \tag{4.48}$$

where $P \in \mathbf{R}^{n\times n}$ is a positive definite real symmetric matrix. Then

$$\begin{aligned}
\Delta V(e_T(k)) &= V(e_T(k+1)) - V(e_T(k)) \\
&= e_T(k+1)^T P e_T(k+1) - e_T(k)^T P e_T(k) \\
&= [Ae_T(k) + c(k+1)]^T P [Ae_T(k) + c(k+1)] - e_T(k)^T P e_T(k)
\end{aligned}$$

$$\begin{aligned}
&= e_T(k)^T A^T P A e_T(k) + e_T(k)^T A^T P c(k+1) \\
&\quad + c(k+1)^T P A e_T(k) + c(k+1)^T P c(k+1) - e_T(k)^T P e_T(k) \\
&= e_T(k)^T [A^T P A - P] e_T(k) + 2c(k+1)^T P A e_T(k) + c(k+1)^T P c(k+1) \\
&= -e_T(k)^T Q e_T(k) + 2c(k+1)^T P A e_T(k) + c(k+1)^T P c(k+1) \qquad (4.49)
\end{aligned}$$

If

$$2c(k+1)^T P A e_T(k) + c(k+1)^T P c(k+1) < e_T(k)^T Q e_T(k) \qquad (4.50)$$

where $-Q = A^T P A - P$, then

$$\Delta V(e_T(k) < 0 \qquad \text{(a negative definite)} \qquad (4.51)$$

Thus if $c(k+1)$ satisfies the inequality (4.50), then the neighbourhood $e_T = 0$ of (4.47) is stable.

The result is stated as follows:

THEOREM 4.3.4 *There exists a Lyapunov function candidate $V(e_T(k)) = e_T(k)^T P e_T(k)$ which results in the neighbourhood $e_T = 0$ of (4.47) being stable, where P is a solution of the Lyapunov equation $A^T P A - P = -Q$; $Q \in \mathbf{R}^{n \times n}$, if the residual $c(k+1)$ satisfies the inequality*

$$2c(k+1)^T P A e_T(k) + c(k+1)^T P c(k+1) < e_T(k)^T Q e_T(k). \qquad (4.52)$$

REMARK 4.3.5 *The inequality (4.52) is also an alternative sufficient condition on the residual $c(k+1)$.*

4.3.3 Supervised Learning Scheme

A block diagram of the neural network controller proposed in this chapter is shown in Figure 4.2. In order to train the neural network controller using a supervised learning scheme such as backpropagation, the error $u_d(k) - u(k)$ in the controller output is required. The term $u_d(k)$ is the desired control which would produce $y_m(k+1)$ if applied to the plant. However, as $u_d(k) \in \mathbf{R}^r$ is not known, an approximation of the controller error denoted by e_c, must be generated. A method for generating such an error is derived for a general multi-input multi-output system below.

Firstly, consider the cost function that is to be minimised. This is given by

$$E = \frac{1}{2}\sum_{k=0}^{N-1}(y'_m(k+1) - y_p(k+1))^T(y'_m(k+1) - y_p(k+1)) \tag{4.53}$$

where N is the number of samples, $y'_m = [y'_{m_1}, \ldots, y'_{m_n}]^T$, $y_p = [y_{p_1}, \ldots, y_{p_n}]^T$, $y'_m(k+1)$ is defined by equation (4.11) and the term $y'_m(k+1) - y_p(k+1)$ is the residual $c(k+1)$.

E is minimised by performing a gradient descent in $E(W_c(k))$, where $W_c(k)$ are the controller weights:

$$\frac{\partial E}{\partial W_c(k)} = \left(\frac{\partial y'_m(k+1)}{\partial W_c(k)} - \frac{\partial y_p(k+1)}{\partial W_c(k)}\right)(y'_m(k+1) - y_p(k+1)) \tag{4.54}$$

where $\frac{\partial E}{\partial W_c(k)}$ is a $p \times 1$ vector, $\frac{\partial y'_m(k+1)}{\partial W_c(k)}$ is a $p \times n$ matrix, $\frac{\partial y_p(k+1)}{\partial W_c(k)}$ is a $p \times n$ matrix, and p is the order of the output layer weight vector = number output nodes $\times$ the number of nodes in the last hidden layer.

Calculating the partial derivative of $y'_m(k+1)$ with respect to the weight vector $W_c(k)$ results in

$$\frac{\partial y'_m(k+1)}{\partial W_c(k)} = \frac{\partial y_m(k+1)}{\partial W_c(k)} - \frac{\partial y_m(k)}{\partial W_c(k)}A + \frac{\partial y_p(k)}{\partial W_c(k)}A \tag{4.55}$$

where A is an $n \times n$ Hurwitz matrix which first appeared in (4.11).

In the above equation $\frac{\partial y_m(k+1)}{\partial W_c(k)} = 0$ and $\frac{\partial y_m(k)}{\partial W_c(k)} = 0$, so that

$$\frac{\partial y'_m(k+1)}{\partial W_c(k)} = \frac{\partial y_p(k)}{\partial W_c(k)}A \tag{4.56}$$

Now the partial derivative of $y_p(k)$ with respect to $W_c(k)$ can be expressed as

$$\frac{\partial y_p(k)}{\partial W_c(k)} = \frac{\partial u(k)}{\partial W_c(k)}\frac{\partial y_p(k)}{\partial u(k)} \tag{4.57}$$

where $\frac{\partial u(k)}{\partial W_c(k)}$ is a $p \times r$ matrix, $\frac{\partial y_p(k)}{\partial u(k)}$ is a $r \times n$ matrix, n is the order of the output vector, and r is the order of the control vector.

However, from equation (4.8) it is apparent that $y_p(k)$ is a function of $u(k-1)$ not $u(k)$. So $\frac{\partial y_p(k)}{\partial W_c(k)}$ is redefined as follows

$$\frac{\partial y_p(k)}{\partial W_c(k)} = \frac{\partial y_p(k+1)}{\partial W_c(k)}\left[\frac{\partial y_p(k+1)}{\partial y_p(k)}\right]^{-1} \tag{4.58}$$

It can be approximated that $\frac{\partial y_p(k+1)}{\partial y_p(k)} \approx I$, where I is the n-dimensional identity matrix. This is a reasonable approximation because if the sampling period h is chosen sensibly, then the change in y_p over this sample should be small. Thus (4.56) becomes

$$\frac{\partial y'_m(k+1)}{\partial W_c(k)} \approx \frac{\partial y_p(k+1)}{\partial W_c(k)} A \tag{4.59}$$

Substituting (4.59) into (4.54) results in

$$\begin{aligned}\frac{\partial E}{\partial W_c(k)} &= \left(\frac{\partial y_p(k+1)}{\partial W_c(k)} A - \frac{\partial y_p(k+1)}{\partial W_c(k)}\right)(y'_m(k+1) - y_p(k+1)) \\ &= \frac{\partial y_p(k+1)}{\partial W_c(k)}(A-I)(y'_m(k+1) - y_p(k+1)) \\ &= \frac{\partial u(k)}{\partial W_c(k)} \frac{\partial y_p(k+1)}{\partial u(k)}(A-I)(y'_m(k+1) - y_p(k+1)) \qquad (4.60)\end{aligned}$$

Now from the backpropagation algorithm, the update equation for the output layer weights of the controller neural network with learning rate η is given by

$$W_c(k+1) = W_c(k) - \eta \frac{\partial E}{\partial W_c(k)} \tag{4.61}$$

Substituting (4.60) into the above equation results in

$$W_c(k+1) = W_c(k) - \eta \frac{\partial u(k)}{\partial W_c(k)} \frac{\partial y_p(k+1)}{\partial u(k)}(A-I)(y'_m(k+1) - y_p(k+1)) \tag{4.62}$$

Equation (4.62) involves the Jacobian of the plant $\frac{\partial y_p(k+1)}{\partial u(k)}$, which is unknown. Assuming that an approximate of the Jacobian, $\frac{\partial \hat{y}_p(k+1)}{\partial u(k)}$, is available, the following weight update equation results

$$W_c(k+1) = W_c(k) + \eta \frac{\partial u(k)}{\partial W_c(k)} e_c(k) \tag{4.63}$$

where the controller error vector $e_c = [e_{c_1}, e_{c_2}, \ldots, e_{c_r}]^T$ is given by

$$e_c(k) = \frac{\partial \hat{y}_p(k+1)}{\partial u(k)}(I-A)(y'_m(k+1) - y_p(k+1)) \tag{4.64}$$

and the term $\hat{y}_p(k+1)$ represents the n-dimensional estimate of the plant output vector obtained from the neural network emulator, i.e.,

$$\begin{aligned}\hat{y}_p(k+1) &= \hat{f}(y_p(k), \cdots, y_p(k-l+1); u(k), \cdots, u(k-m+1); W_I) \qquad (4.65) \\ \hat{y}_p(0) &= 0 \qquad (4.66)\end{aligned}$$

In Section 3.2.3, it is demonstrated that the controller error, e_c, can be obtained by backpropagating the tracking error

$$e_T(k+1) = y_m(k+1) - y_p(k+1) \tag{4.67}$$

through the neural network that identifies the plant. For the enhanced scheme proposed in this chapter, it can be shown in the following derivation that the controller error governed by equation (4.64) can be obtained by backpropagating the modified error $(I-A)c(k+1) = (I-A)(y'_m(k+1)-y_p(k+1))$ through the neural network approximating the plant dynamics, where $c(k+1)$ is the residual defined in (4.14).

Firstly, consider the desired controller error

$$\begin{aligned}
e_c(k) &= \frac{\partial \hat{y}_p(k+1)}{\partial u(k)}(I-A)(y'_m(k+1)-y_p(k+1)) \\
&= \begin{bmatrix} \frac{\partial \hat{y}_{p1}}{\partial u_1} & \cdots & \frac{\partial \hat{y}_{pn}}{\partial u_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_{p1}}{\partial u_r} & \cdots & \frac{\partial \hat{y}_{pn}}{\partial u_r} \end{bmatrix} \begin{bmatrix} 1-a_{11} & \cdots & -a_{1n} \\ \vdots & \ddots & \vdots \\ -a_{n1} & \cdots & 1-a_{nn} \end{bmatrix} \begin{bmatrix} y'_{m_1}-y_{p_1} \\ \vdots \\ y'_{m_n}-y_{p_n} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial \hat{y}_{p1}}{\partial u_1} & \cdots & \frac{\partial \hat{y}_{pn}}{\partial u_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_{p1}}{\partial u_r} & \cdots & \frac{\partial \hat{y}_{pn}}{\partial u_r} \end{bmatrix} \begin{bmatrix} (1-a_{11})(y'_{m_1}-y_{p_1})-a_{12}(y'_{m_2}-y_{p_2})-\cdots-a_{1n}(y'_{m_n}-y_{p_n}) \\ -a_{21}(y'_{m_1}-y_{p_1})+(1-a_{22})(y'_{m_2}-y_{p_2})-\cdots-a_{2n}(y'_{m_n}-y_{p_n}) \\ \vdots \\ -a_{n1}(y'_{m_1}-y_{p_1})-a_{n2}(y'_{m_2}-y_{p_2})-\cdots+(1-a_{nn})(y'_{m_n}-y_{p_n}) \end{bmatrix} \\
&= \begin{bmatrix} B_1\frac{\partial \hat{y}_{p1}}{\partial u_1}+B_2\frac{\partial \hat{y}_{p2}}{\partial u_1}+\cdots+B_n\frac{\partial \hat{y}_{pn}}{\partial u_1} \\ B_1\frac{\partial \hat{y}_{p1}}{\partial u_2}+B_2\frac{\partial \hat{y}_{p2}}{\partial u_2}+\cdots+B_n\frac{\partial \hat{y}_{pn}}{\partial u_2} \\ \vdots \\ B_1\frac{\partial \hat{y}_{p1}}{\partial u_r}+B_2\frac{\partial \hat{y}_{p2}}{\partial u_r}+\cdots+B_n\frac{\partial \hat{y}_{pn}}{\partial u_r} \end{bmatrix}
\end{aligned} \tag{4.68}$$

where

$$\begin{aligned}
B_1 &= (1-a_{11})(y'_{m_1}-y_{p_1})-a_{12}(y'_{m_2}-y_{p_2})-\cdots-a_{1n}(y'_{m_n}-y_{p_n}) \\
B_2 &= -a_{21}(y'_{m_1}-y_{p_1})+(1-a_{22})(y'_{m_2}-y_{p_2})-\cdots-a_{2n}(y'_{m_n}-y_{p_n}) \\
&\vdots \\
B_n &= -a_{n1}(y'_{m_1}-y_{p_1})-a_{n2}(y'_{m_2}-y_{p_2})-\cdots+(1-a_{nn})(y'_{m_n}-y_{p_n})
\end{aligned} \tag{4.69}$$

For the sake of simplicity, assume that A is a diagonal matrix. Therefore, the elements $a_{ij} = 0 \;\; i = 1,\ldots,n \;\; j = 1,\ldots,n \;\; i \neq j$ and the diagonal elements $a_{ii} \;\; i = 1,\ldots,n$ are the eigenvalues of A and thus $0 < a_{ii} < 1$. Therefore the controller error $e_c(k)$ becomes

$$e_c(k) = \begin{bmatrix} (1-a_{11})(y'_{m_1}-y_{p_1})\frac{\partial \hat{y}_{p1}}{\partial u_1} + (1-a_{22})(y'_{m_2}-y_{p_2})\frac{\partial \hat{y}_{p2}}{\partial u_1} + \cdots + (1-a_{nn})(y'_{m_n}-y_{p_n})\frac{\partial \hat{y}_{pn}}{\partial u_1} \\ (1-a_{11})(y'_{m_1}-y_{p_1})\frac{\partial \hat{y}_{p1}}{\partial u_2} + (1-a_{22})(y'_{m_2}-y_{p_2})\frac{\partial \hat{y}_{p2}}{\partial u_2} + \cdots + (1-a_{nn})(y'_{m_n}-y_{p_n})\frac{\partial \hat{y}_{pn}}{\partial u_2} \\ \vdots \\ (1-a_{11})(y'_{m_1}-y_{p_1})\frac{\partial \hat{y}_{p1}}{\partial u_r} + (1-a_{22})(y'_{m_2}-y_{p_2})\frac{\partial \hat{y}_{p2}}{\partial u_r} + \cdots + (1-a_{nn})(y'_{m_n}-y_{p_n})\frac{\partial \hat{y}_{pn}}{\partial u_r} \end{bmatrix} \tag{4.70}$$

For this modified neural adaptive control scheme, the network output error for the g-th controller output is

$$e_{c_g}(k) = (1-a_{11})(y'_{m_1}-y_{p_1})\frac{\partial \hat{y}_{p_1}}{\partial u_g} + (1-a_{22})(y'_{m_2}-y_{p_2})\frac{\partial \hat{y}_{p_2}}{\partial u_g} + \cdots + (1-a_{nn})(y'_{m_n}-y_{p_n})\frac{\partial \hat{y}_{p_n}}{\partial u_g} \tag{4.71}$$

where the time index k on y'_m, y_p and u is neglected for brevity. Now recall from the derivation provided in the previous chapter that for a two layered neural network with a inputs, b hidden units and n outputs, the effective output error $\delta^2 = [\delta^2_1, \ldots, \delta^2_n]^T$ is given by

$$\begin{aligned} \delta^2_1 &= g'(\sum_{j=1}^{b} W^2_{1j} g(\sum_{k=1}^{a} W^1_{jk} x_k)) e_{o_1}(k+1) \\ &\vdots \\ \delta^2_n &= g'(\sum_{j=1}^{b} W^2_{nj} g(\sum_{k=1}^{a} W^1_{jk} x_k)) e_{o_n}(k+1) \end{aligned} \tag{4.72}$$

where $e_o = [e_{o_1}, \ldots, e_{o_n}]^T$ is the error to be backpropagated. For the original neural control scheme provided in Chapter 3, $e_o(k+1) = y_m(k+1) - y_p(k+1)$. For the enhanced scheme proposed in this chapter, $e_o(k+1) = (I - A)(y'_m(k+1) - y_p(k+1))$.

Backpropagating the above output errors to the input layer yields for the l-th network input

$$\begin{aligned} \delta^0_l &= \sum_{s=1}^{n} \{W^2_{sj} g'(\sum_{j=1}^{b} W^2_{sj} g(\sum_{k=1}^{a} W^1_{jk} x_k)) \sum_{j=1}^{b} W^1_{jl} g(\sum_{k=1}^{a} W^1_{jk} x_k)(1 - a_{ss})(y'_{m_s} - y_{p_s})\} \\ &= g'(\sum_{j=1}^{b} W^2_{1j} g(\sum_{k=1}^{a} W^1_{jk} x_k)) \sum_{j=1}^{b} (W^2_{1j} g(\sum_{k=1}^{a} W^1_{jk} x_k) W^1_{jl})(1 - a_{11})(y'_{m_1} - y_{p_1}) \\ &+ g'(\sum_{j=1}^{b} W^2_{2j} g(\sum_{k=1}^{a} W^1_{jk} x_k)) \sum_{j=1}^{b} (W^2_{2j} g(\sum_{k=1}^{a} W^1_{jk} x_k) W^1_{jl})(1 - a_{22})(y'_{m_2} - y_{p_2}) \\ &\vdots \\ &+ g'(\sum_{j=1}^{b} W^2_{nj} g(\sum_{k=1}^{a} W^1_{jk} x_k)) \sum_{j=1}^{b} (W^2_{nj} g(\sum_{k=1}^{a} W^1_{jk} x_k) W^1_{jl})(1 - a_{nn})(y'_{m_n} - y_{p_n}) \end{aligned} \tag{4.73}$$

Substituting (3.26) into (4.73) yields

$$\delta_l^0 = (1-a_{11})(y'_{m_1}-y_{p_1})\frac{\partial \hat{y}_{p_1}}{\partial u_g} + (1-a_{22})(y'_{m_2}-y_{p_2})\frac{\partial \hat{y}_{p_2}}{\partial u_g} + \cdots + (1-a_{nn})(y'_{m_n}-y_{p_n})\frac{\partial \hat{y}_{p_n}}{\partial u_g} \quad (4.74)$$

where the time index k is neglected in some of the terms for the sake of brevity and A is assumed to be a diagonal matrix. The above equation is equivalent to the controller error given in equation (4.71). The above results can be summarised by the following two theorems.

THEOREM 4.3.5 *Minimising the mean square residual error* $E = \frac{1}{2}\sum_{k=0}^{N-1}(y'_m(k+1) - y_p(k+1))^T(y'_m(k+1) - y_p(k+1))$ *can be achieved by utilising the following approximation of the controller error*

$$e_c(k) = \frac{\partial \hat{y}_p(k+1)}{\partial u(k)}(I - A)(y'_m(k+1) - y_p(k+1)) \quad (4.75)$$

to modify the weights of the neural network controller.

THEOREM 4.3.6 *The controller error governed by*

$$e_c(k) = \frac{\partial \hat{y}_p(k+1)}{\partial u(k)}(I - A)(y'_m(k+1) - y_p(k+1)) \quad (4.76)$$

can be obtained by backpropagating the modified error $(I - A)c(k+1)$ *through the neural network approximating the plant dynamics, where* $c(k+1) = y'_m(k+1) - y_p(k+1)$ *is the residual.*

4.3.4 Enhanced Neural Control Scheme - Practical Issues

The above theorems allow the weight update equation (4.63) to be used, in conjunction with the backpropagation algorithm, to train and synthesise the controller. As with the method presented in the previous chapter, the identification of the plant is only necessary to obtain an approximate of the plant Jacobian which is then used to update the controller weights.

In order that the residual $c(k+1)$ satisfies either sufficient condition (4.16), (4.22) or (4.52), an iterative search is conducted on the control. The search is initialised with the controller input vector at time k and the controller network weights are modified according to the weight update equation (4.61) until one of the the termination criteria are met: (*i*) the residual $c(k+1)$ satisfies (4.16), (*ii*) the residual satisfies (4.22), (*iii*) the residual satisfies (4.52), or (*iv*) the number of iterations reaches a maximum ("computational limit"). The new value of control $u(k)$ is then the value of control at the end of the iterative search. This approach is similar in spirit to the procedure presented by Hoskins *et al.* [75], except that in [75] the iteration is terminated when the predicted cost or the magnitude of the cost with respect to the control is less than a threshold. The cost function in this case is related to a Lyapunov-like function associated with a convergence model. This convergence model is similar in some respects to the enhanced reference model considered here, however the convergence and stability results provided in [75] are significantly different to the results provided in this thesis. Furthermore, the general philosophy of the approach proposed in this thesis is inspired by the ODS approaches of Spong *et al.* [225] and Rehbock *et al.* [190, 192]. The iterative approach described above is also similar in spirit to commonly used on-line optimisation schemes for control applications.

The iterative search is dependent on the plant output $y_p(k+1)$ and the plant Jacobian $\frac{\partial y_p(k+1)}{\partial u(k)}$. However, during the search routine, the iterative values of control are not applied to the plant as this would alter the state of the plant. Instead the neural network emulator is used to provide the necessary "plant output" and Jacobian. As the neural network emulator is initially not an accurate model of the plant dynamics the iterative search is generally conducted for the maximum number of iterations during the first few samples of operation. However, this problem can be easily overcome by improving the accuracy of the neural network emulator via a short period of off-line identification prior to the on-line control procedure.

In a typical practical environment, the neural controller would be implemented on a current generation floating-point DSP chip or microprocessor such as the TMS320C40, which has an instruction cycle time of 40 ns. Given a typical sampling interval of 0.1s and assuming a conservative figure of 2 cycles per instruction[1], this allows 1.25×10^6 opera-

[1]The TMS320C40 DSP chip undertakes most floating-point operations such as multiplication, addition, subtraction, etc., in 1 instruction cycle.

tions per sample. The neural network MRAC approach using a typical neural network architecture of $\Omega^3_{4,20,10,2}$ requires approximately 5000 operations. This means that the computational limit for the iterative control search is approximately 250 iterations per sample. For most of the systems considered an average of 1 to 5 iterations is required, meaning that from a practical perspective more than sufficient computational time is available between samples for the iterative search. Furthermore, the above analysis does not take into account the potential to speed up computations by exploiting the parallel nature of neural networks.

4.4 Simulation Examples

In this section, five simulation examples are considered to demonstrate the effectiveness of the enhanced neural adaptive control scheme. The first two highlight the improved performance of the proposed scheme over the on-line scheme presented in Chapter 3 for both a single-input single-output system and a multi-input multi-output system. The remaining three simulation examples demonstrate the ability of the scheme to deal with commonly occurring practical non-idealities. In particular, the effectiveness of the scheme in dealing with nonminimum phase behaviour, its robustness to dynamic uncertainties and variations, and its ability to effectively control a marginally stable nonlinear system are highlighted.

4.4.1 Single-Input Single-Output System

The first example considered is the Model III single-input single-output system studied in Chapter 3 and by Narendra and Parthasarathy [169]. This system is nonlinear in output and control with the respective nonlinearities being separable. The difficulty in controlling such systems arises because of the complex nonlinearity in control.

EXAMPLE 4.4.1 The system dynamics are described by the following difference equation

$$y_p(k+1) = \frac{y_p(k)}{1+y_p^2(k)} + u^3(k) \tag{4.77}$$

The reference model used is a stable first order discrete-time system given by

$$y_m(k+1) = 0.2y_m(k) + r(k) \tag{4.78}$$

Narendra and Parthasarathy utilised the separability of the control and output variable terms to design an explicit control law which incorporated the neural estimates of these separable nonlinear terms. The results provided in Chapter 3 demonstrated that plants such as the one described in equation (4.77) can be effectively controlled without knowledge of the separability (or otherwise) of the control and output variables. In this simulation study, the enhanced model reference neural adaptive control scheme will be shown to result in an improved tracking performance compared with the scheme presented in Chapter 3.

The enhanced reference model consists of (4.78) and

$$y'_m(k+1) = y_m(k+1) - 0.1(y_m(k) - y_p(k)) \tag{4.79}$$

The ability of the controller to adapt to a changing input is investigated by considering the following reference input

$$r(k) = a(k/150)sgn[\sin(\frac{2\pi k}{150})] \tag{4.80}$$

where $a(k/150)$ is a random variable in $\Re[0,1]$ which changes value every 150 time samples.

The various responses for this input are given in Figures 4.3a–4.3i. As the plant is first order in output and control, neural networks consisting of 2 inputs, 2 hidden layers with 20 and 10 nodes, respectively, and 1 output are used for the controller and emulator. A learning rate of $\eta = 0.1$ is used for both networks. As shown in (4.79), a stability constant of $\beta = 0.1$ is used. These parameters provided the best results.

Figure 4.3a shows the response of the plant and enhanced reference model output for the reference input given by (4.80). As can be seen, excellent tracking results are obtained. In particular, apart from the first few samples, the plant response is indistinguishable from the the reference model response and the desired response. Also the variable nature of the reference input does not prove to be a problem and the tracking error effectively remains at zero even after the various step changes.

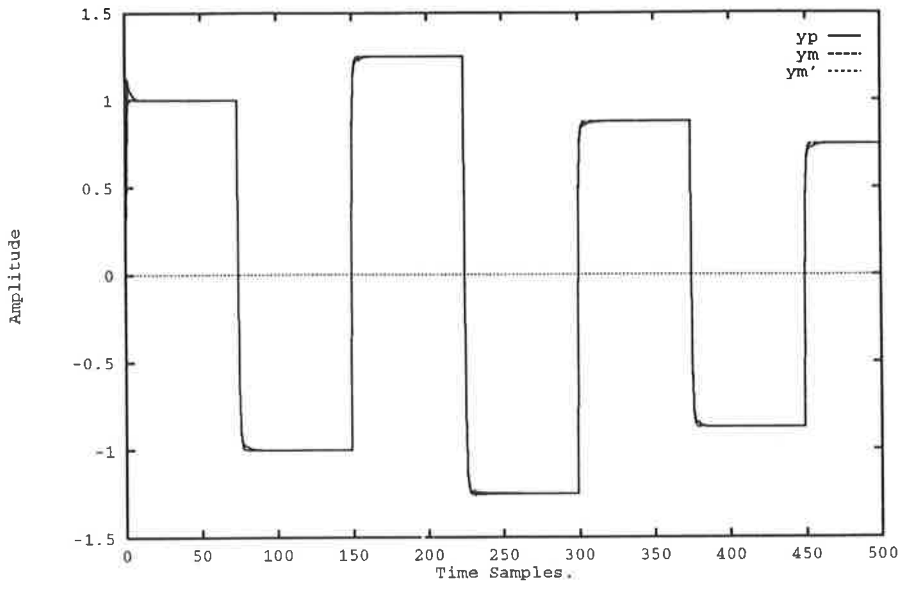


Figure 4.3a: Response of the plant (y_p), reference model (y_m) and enhanced reference model (y'_m) for Example 4.4.1 with $r(k) = a(k/150)sgn[\sin(\frac{2\pi k}{150})]$

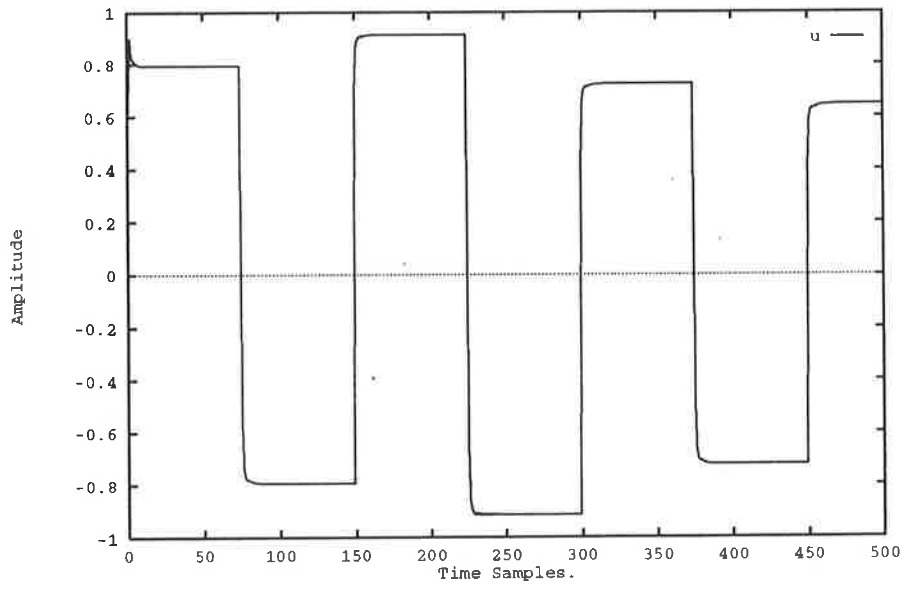


Figure 4.3b: Control input $u(k)$ for Example 4.4.1 with $r(k) = a(k/150)sgn[\sin(\frac{2\pi k}{150})]$

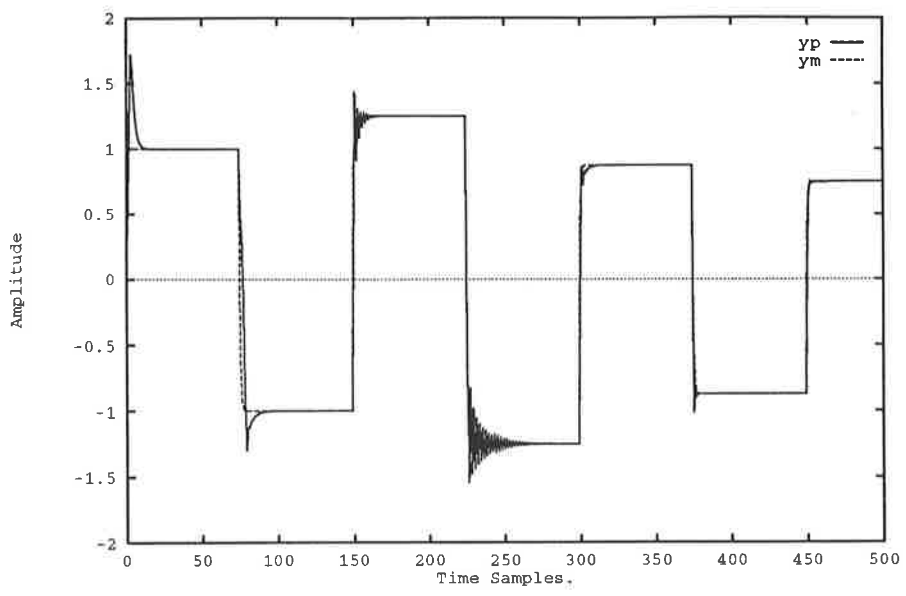


Figure 4.3c: Response of the plant (y_p) and reference model (y_m) for Example 4.4.1 *without* the proposed enhancement and with $r(k) = a(k/150)sgn[\sin(\frac{2\pi k}{150})]$

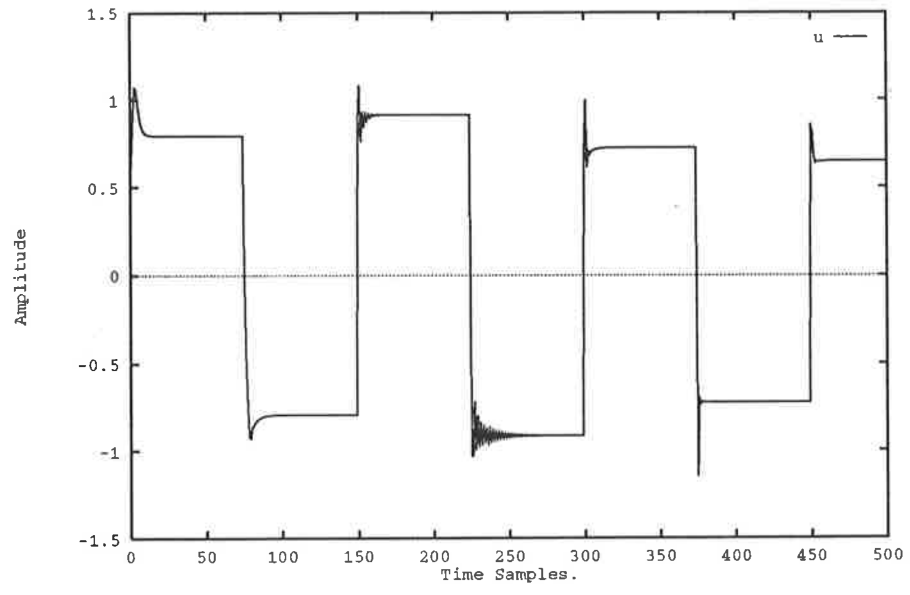


Figure 4.3d: Control input $u(k)$ for Example 4.4.1 *without* the proposed enhancement and with $r(k) = a(k/150)sgn[\sin(\frac{2\pi k}{150})]$

The results of the approach considered in Chapter 3, subject to the same reference input, are provided in Figures 4.3c and 4.3d. The plant and reference model responses indicate that the tracking performance is not as good as for the enhanced scheme. In particular, the significant overshoots which occur for the original scheme are not present in the plant response for the proposed control method. Also, the high frequency oscillations at $k = 150$ and $k = 225$ in the responses for the original scheme are not present in the responses for the enhanced scheme. Comparison of the control response obtained for the enhanced neural control approach (Figure 4.3b) with the response obtained for the original neural control approach (Figure 4.3d) shows that the new approach results in a less active control signal with no large overshoots. This is of practical benefit because a control signal which consists of large, rapid changes in magnitude results in greater demands on the various actuators, and therefore reduced life of the control mechanism and increased energy (fuel) consumption.

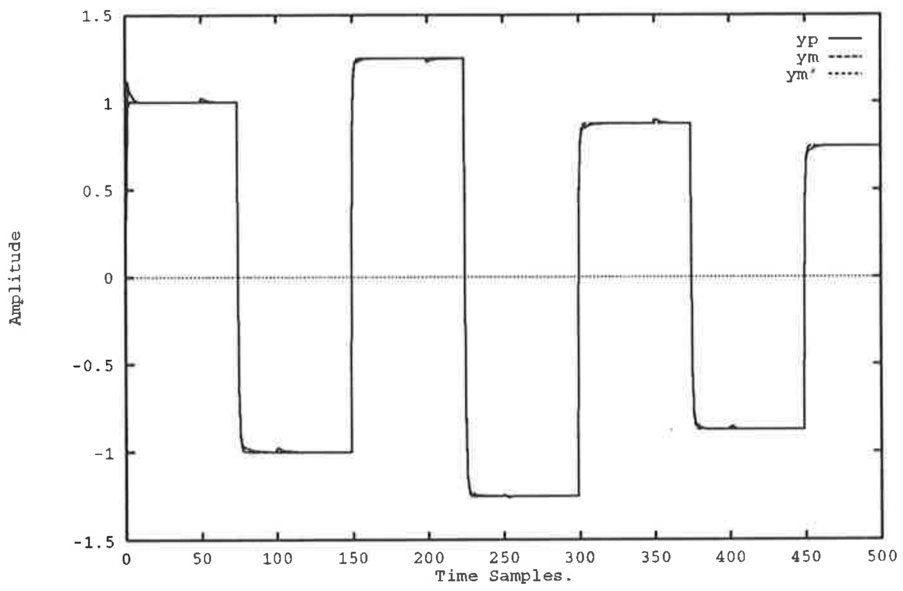


Figure 4.3e: Response of the plant (y_p), reference model (y_m) and enhanced reference model (y'_m) for Example 4.4.1 with $r(k) = a(k/150)sgn[\sin(\frac{2\pi k}{150})]$ and with a load disturbance present

Figure 4.3e demonstrates the ability of the enhanced neural controller to deal with a variable DC bias on the output of the plant. The form of the DC bias disturbance considered in this example is shown in Figure 4.3f. As explained in the previous chapter,

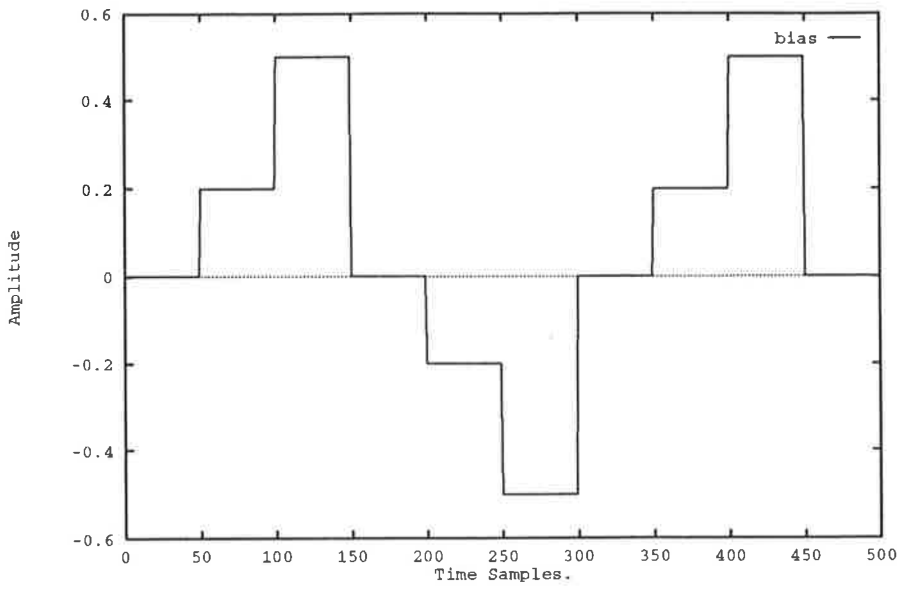


Figure 4.3f: Load disturbance for Example 4.4.1

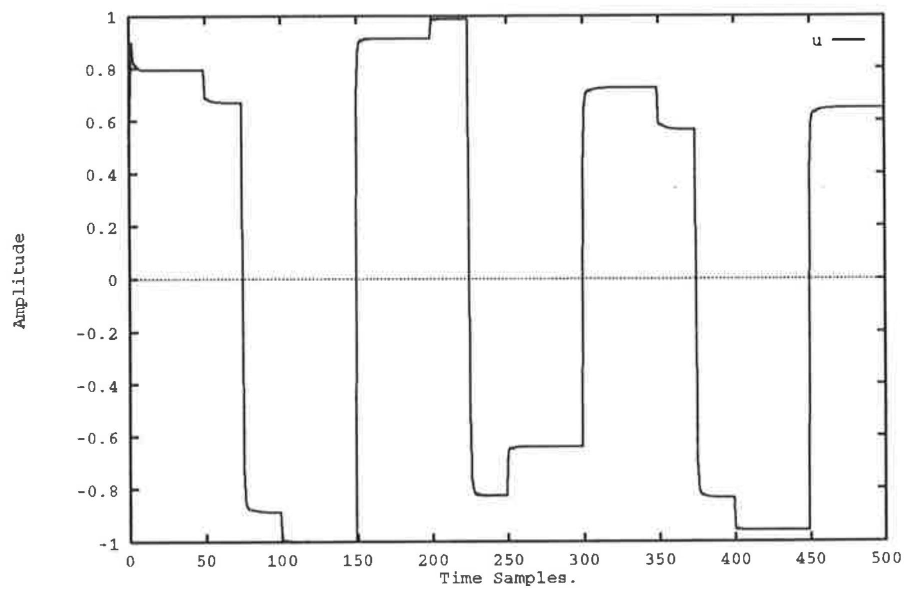


Figure 4.3g: Control input $u(k)$ for Example 4.4.1 with a load disturbance present

such disturbances arise because of load changes on the system and are therefore often called load disturbances. In the corresponding example in Chapter 3 (Figure 3.19a), the control scheme is capable of compensating for load disturbances very well. Apart from small perturbations in the plant response whenever the load is changed (Figure 3.19b), the tracking error is virtually zero. The plant and reference model responses for the scheme proposed in this chapter demonstrates that even these perturbations are reduced to the extent that perfect tracking occurs. The control necessary to achieve these excellent results is shown in Figure 4.3g. As can be seen from this figure, the magnitude of control input changes more frequently in order to compensate for the load disturbance. However, the control activity is still at a level which should not pose any problem for the control mechanism.

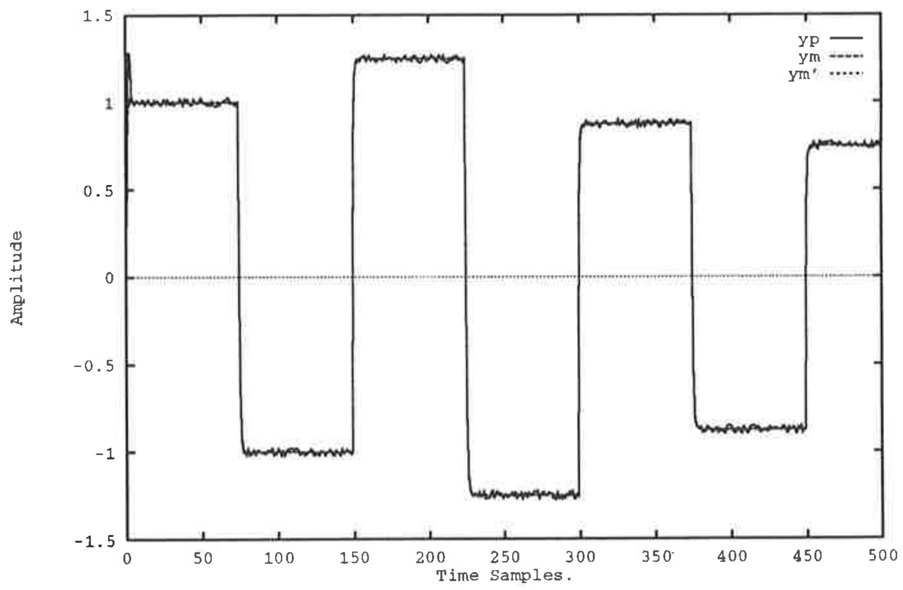


Figure 4.3h: Response of the plant (y_p), reference model (y_m) and enhanced reference model (y'_m) for Example 4.4.1 with $r(k) = a(k/150)sgn[\sin(\frac{2\pi k}{150})]$ and with dynamic plant noise, sensor noise and a load disturbance present

The final two figures for this simulation study reflect the effect of dynamic plant noise, sensor noise and load disturbances on the performance of the system. This can be considered a worst case scenario as not only is there a variable mean white noise process (load disturbance + sensor noise) on the output of the plant, but the plant response is corrupted by a coloured noise process (dynamic noise). A 20 dB signal-to-noise ratio

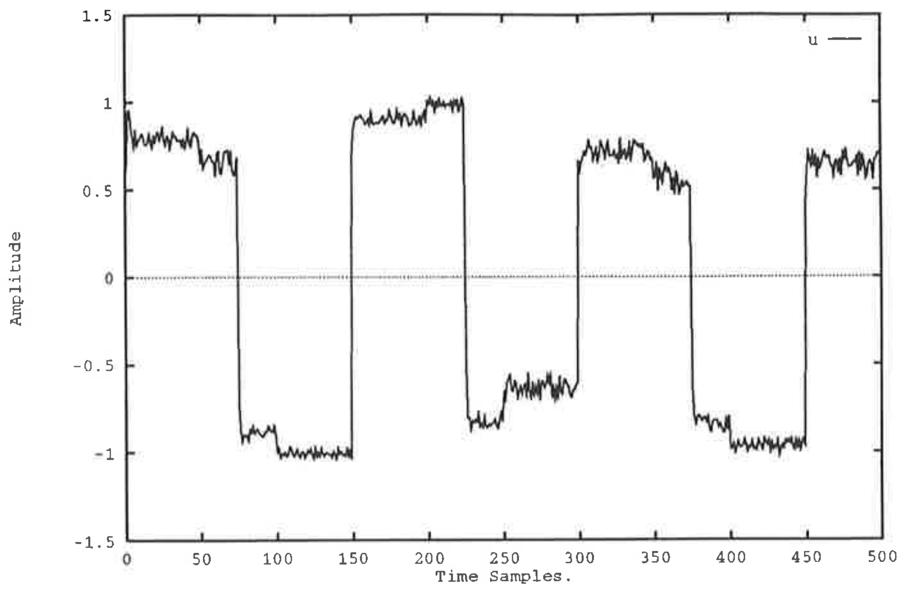


Figure 4.3i: Control input $u(k)$ for Example 4.4.1 with $r(k) = a(k/150)sgn[\sin(\frac{2\pi k}{150})]$ and with dynamic plant noise, sensor noise and a load disturbance present

is used for both noise processes. The plant, reference model and enhanced reference model responses provided in Figure 4.3h indicate that apart from the presence of noise on the plant output, the control system performs very well. Furthermore, the noise at the output is attenuated by the control system such that the output signal-to-noise ratio is significantly increased. The negative aspect of the presence of noise in the system is that a far more active control response results as shown in Figure 4.3i. However, the magnitude of the control chattering is small compared with the overall magnitude of the control.

4.4.2 Multi-Input Multi-Output System

The theory for the enhanced model reference neural adaptive control scheme furnished in Section 4.3 is derived for a general multi-input multi-output case. Therefore to demonstrate the effectiveness of the proposed scheme, a multivariable simulation study is presented. The Model IV MIMO plant used in Chapter 3 is once again considered in this example. The system is nonlinear in output variable and control and the nonlinearities are not separable. The plant consists of cross-coupling in the output variables.

EXAMPLE 4.4.2 The plant dynamics are described by the following difference equation

$$\begin{aligned} y_{p_1}(k+1) &= \frac{y_{p_1}^3(k) + 0.5u_1(k) + 0.5u_3(k) + 0.1y_{p_2}(k)}{1 + y_{p_1}^2(k)} \\ y_{p_2}(k+1) &= \frac{y_{p_2}^3(k) + 0.5u_2(k) + 0.5u_3(k) + 0.1y_{p_1}(k)}{1 + y_{p_2}^2(k)} \end{aligned} \tag{4.81}$$

A stable linear reference model given by the following difference equation is used

$$\begin{aligned} y_{m_1}(k+1) &= 0.5y_{m_1}(k) + 0.5u_1(k) + 0.5u_3(k) \\ y_{m_2}(k+1) &= 0.5y_{m_2}(k) + 0.5u_2(k) + 0.5u_3(k) \end{aligned} \tag{4.82}$$

The enhanced reference model used consisted of (4.82) and

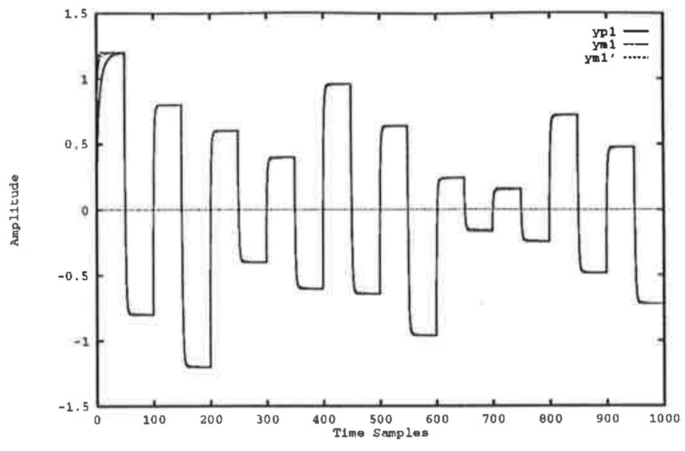
$$\begin{aligned} \begin{bmatrix} y'_{m_1}(k+1) \\ y'_{m_2}(k+1) \end{bmatrix} &= \begin{bmatrix} y_{m_1}(k+1) \\ y_{m_2}(k+1) \end{bmatrix} - A \begin{bmatrix} y_{m_1}(k) - y_p(k) \\ y_{m_2}(k) - y_p(k) \end{bmatrix} \\ &= \begin{bmatrix} y_{m_1}(k+1) \\ y_{m_2}(k+1) \end{bmatrix} - \begin{bmatrix} a_{11} & 0 \\ 0 & a_{22} \end{bmatrix} \begin{bmatrix} y_{m_1}(k) - y_p(k) \\ y_{m_2}(k) - y_p(k) \end{bmatrix} \end{aligned} \tag{4.83}$$

The elements of the 2×2 diagonal matrix A are chosen in accordance with Theorem 4.3.3 such that $a_{11} = a_{22} = 0.2 < \frac{1}{n}$, where $n = 2$.

A neural network consisting of 5 inputs $[y_{p_1}, y_{p_2}, u_1, u_2, u_3]$, 2 hidden layers with 10 nodes each, and 3 outputs is used to implement the controller, i.e., $\Omega^3_{5,10,10,3}$. The emulator network belonged to the class $\Omega^3_{5,10,10,2}$. A learning rate of $\eta = 0.2$ and momentum rate of $\alpha = 0.1$ is used for both networks.

As with the simulation study considered in Chapter 3, the ability of the controller to adapt to a changing input is investigated by considering the following reference input

$$\begin{aligned} r_1(k) &= a(k/200)sgn[\sin(\frac{2\pi k}{100})] \\ r_2(k) &= a(k/200)sgn[\sin(\frac{2\pi k}{200})] \\ r_3(k) &= 0.2a(k/200)sgn[\sin(\frac{2\pi k}{200})] \end{aligned} \tag{4.84}$$



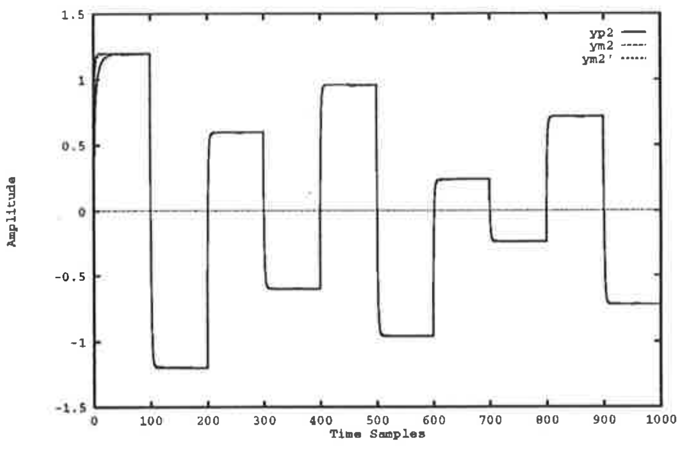


Figure 4.4a: Response of the plant ($y_p = [y_{p_1}, y_{p_2}]^T$), reference model ($y_m = [y_{m_1}, y_{m_2}]^T$) and enhanced reference model ($y'_m = [y'_{m_1}, y'_{m_2}]^T$) for Example 4.4.2

where $a(k/200)$ is a random variable in $\Re[0,1]$ whose value is changed every 200 samples. The various responses for this input are given in Figures 4.4a and 4.4b.

The plant, reference model and enhanced reference model responses shown in Figure 4.4a indicate that the tracking performance of the the enhanced neural control scheme is very good despite the variable nature of the reference input and the complex cross-coupling of variables which exists in the the system. These results compare more than favourably with the performance of the on-line control scheme for the corresponding system in Chapter 3. In particular, the tracking errors have been significantly reduced, such that, after the first few samples, there is no discernible difference between the plant and reference model responses. Furthermore, the small overshoots which occurred in the

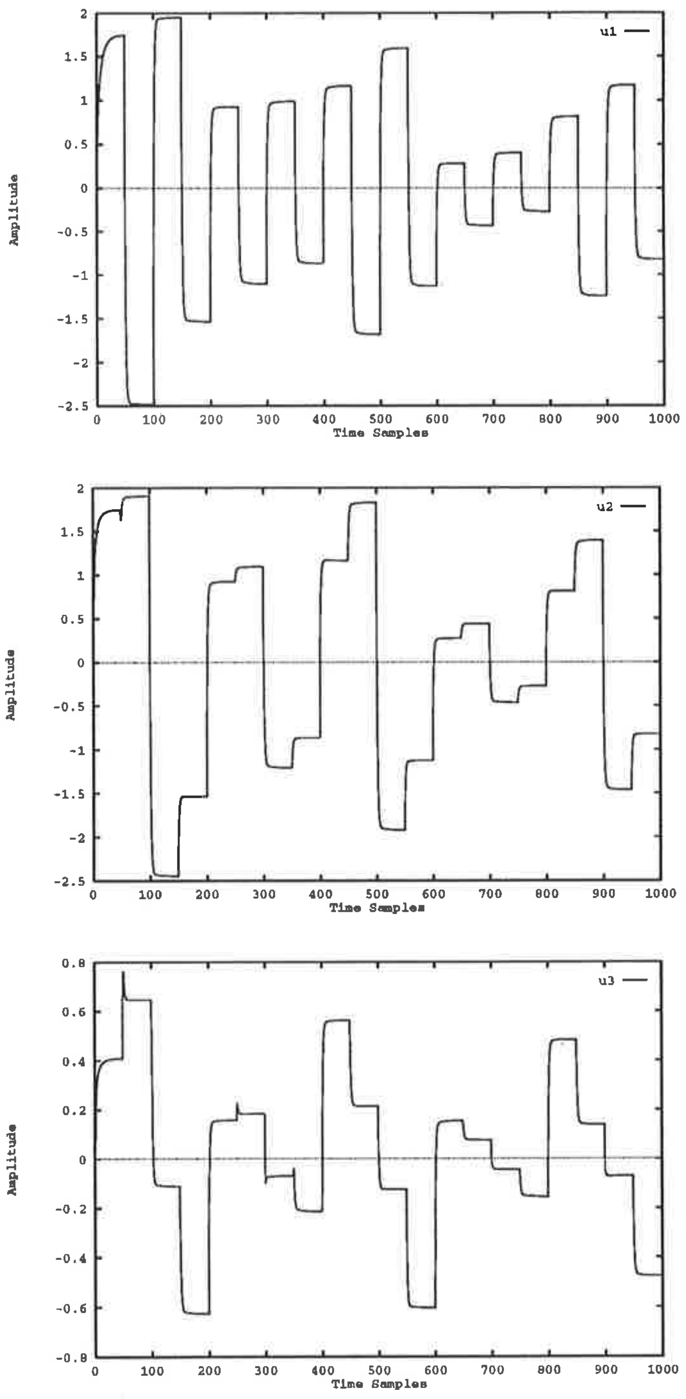


Figure 4.4b: Control input ($u = [u_1, u_2, u_3]^T$) for Example 4.4.2

plant response y_{p_2} due to the more frequently changing cross-coupled response y_{p_1} are eliminated in this approach.

The improvements obtained because of the enhancements made to the control system are further emphasised by the control responses provided in Figure 4.4b. As can be seen in this figure, the control required to achieve the improved tracking performance is far less active than the control response of corresponding example in the previous chapter. This is particularly significant from a practical viewpoint as reduced control activity translates to reduced "wear and tear" on the control mechanism and less energy usage, which in turn means a more economical system.

The excellent performance of the enhanced neural control scheme for this simulation study suggests that the proposed scheme is capable of effectively controlling complex nonlinear multi-input multi-output systems. This augers well for the practical viability of the scheme, as a host of commonly occurring practical systems are multivariable structures.

4.4.3 Nonminimum Phase System

In linear systems theory, a nonminimum phase system is one which has a zero in the right half of the s-plane in a continuous-time environment or outside the unit circle on the z-plane in discrete-time environment. Such systems often pose problems for traditional control schemes as many of these schemes employ an inversion of the plant. For nonminimum phase systems this results in an unstable controller. In nonlinear systems theory, one can no longer represent nonminimum phase systems simply by an appropriate zero. In Chapter 3 a definition is provided for a minimum phase nonlinear system. Based on this definition (Definition 3.2.2), a nonminimum phase nonlinear system can be defined as a system for which the zero dynamic defined by

$$0 = f(0, 0, \ldots, 0; u(k), u(k-1), \ldots, u(k-m+1)) \tag{4.85}$$

where $0 \in \mathbf{R}^n$ and $u \in \mathbf{R}^r$, is unstable, i.e., unbounded inputs lie in the null space of the operator representing the plant.

Nonminimum phase nonlinear systems commonly arise in practice. Examples include the tactical missile and many discretised industrial processes. In fact such systems often arise when a continuous system is discretised [6]. As stated by Slotine and Li [218], if the zero-dynamics is unstable then alternative control strategies to feedback linearisation should be employed. As with traditional control schemes such as feedback linearisation, many neural network based control techniques are unable to cope with nonminimum phase systems. This is primarily because they involve the inversion of the plant being controlled. In the scheme presented here, however, an explicit inversion of the plant is not performed and thus it is capable of dealing with such systems.

EXAMPLE 4.4.3 A second order nonminimum phase system is considered to highlight the effectiveness of the proposed method. The plant equation is given by

$$y_p(k+1) = \tanh(0.7859y_p(k) - 0.3679y_p(k-1) - 0.7267u(k) + 1.3087u(k-1)) \quad (4.86)$$

The nonlinear nature of the system is introduced by the tanh(.) term, which models a saturation effect on the output. Plants such as this one are generally difficult to control because of the nonlinearity in control and the nonminimum phase behaviour.

The nonminimum phase property of this system can be examined by observing the control input $u(k)$ required to keep the output of the system at zero. For the above system, the necessary control input is given by

$$u(k) = \frac{1}{0.7267}(0.7859y_p(k) - 0.3679y_p(k-1) + 1.3087u(k-1)) \quad (4.87)$$

with $u(0) = 0.25$. As shown in Figure 4.5a, the control input required to keep the state at zero is unbounded. Therefore, the system satisfies the definition of a nonminimum phase system given above.

As a counter-example, a linear minimum phase system of the form given below is considered

$$y_p(k+1) = 0.7859y_p(k) - 0.3679y_p(k-1) - 0.7267u(k) + 0.36335u(k-1) \quad (4.88)$$

This system has poles at $z = 0.3930 \pm 0.4621j$ and zeros at $z = 0, 0.5$. As the system has poles and zeros within the unit circle, it is stable and minimum phase. For this system

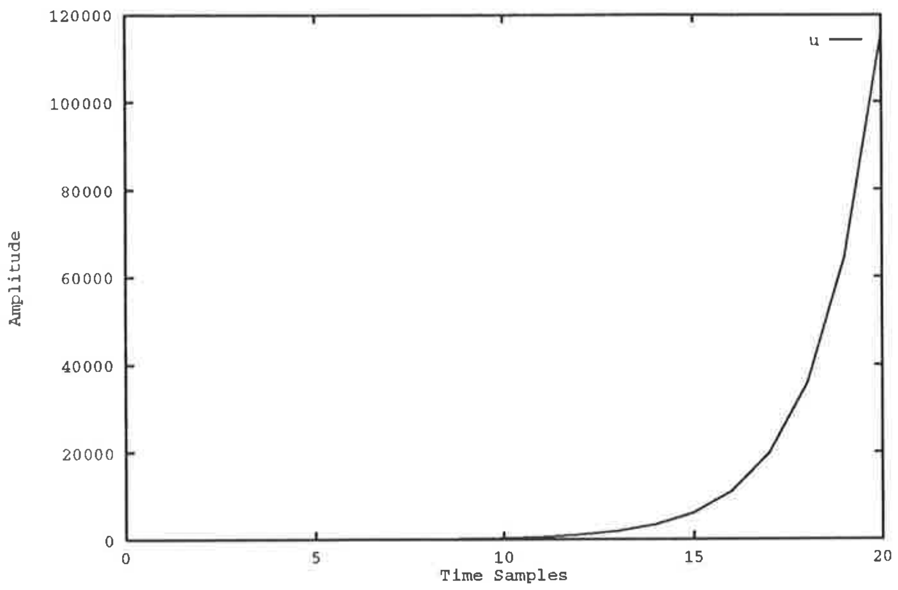


Figure 4.5a: Zero dynamics control input for a nonminimum phase nonlinear system

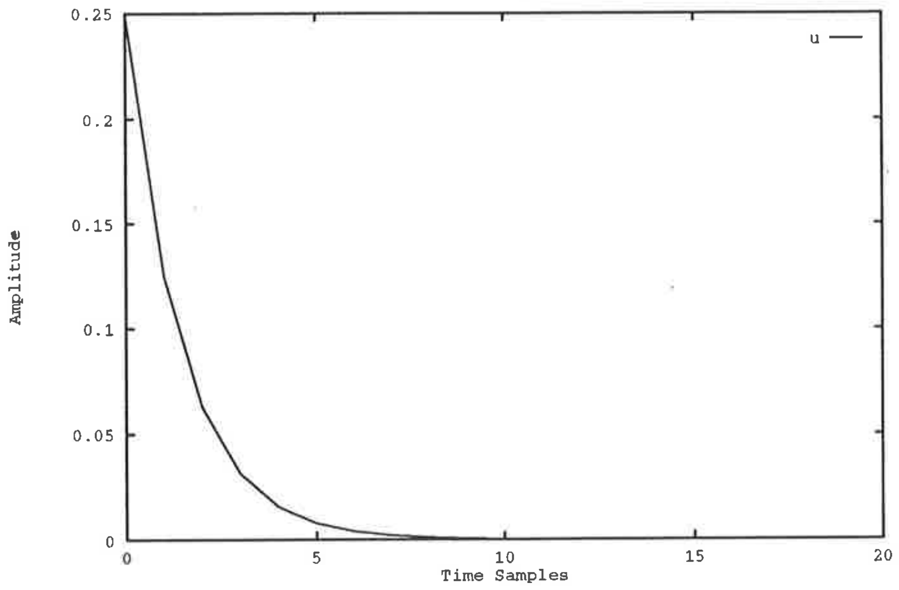


Figure 4.5b: Zero dynamics control input for a minimum phase linear system

the control required to achieve zero dynamics is

$$u(k) = \frac{1}{0.7267}(0.7859y_p(k) - 0.3679y_p(k-1) + 0.36335u(k-1)) \tag{4.89}$$

with $u(0) = 0.25$. This control input is shown in Figure 4.5b. As can be seen, this system is asymptotically stable to the reference point $u = 0$ when the output is kept at zero. This confirms the minimum phase nature of the second system.

The open-loop response of the nonminimum phase nonlinear system as shown in Figure 4.6a undertakes a negative excursion before becoming positive and settling to its steady state value. This is undesirable because it may result in catastrophic failure in a tight control environment, such as navigating a ship through a narrow channel.

The reference model used is a stable second order discrete-time system given by

$$y_m(k+1) = 0.2y_m(k) + 0.2y_m(k-1) + 0.8r(k) \tag{4.90}$$

The enhanced reference model used consisted of (4.90) and

$$y'_m(k+1) = y_m(k+1) - 0.2(y_m(k) - y_p(k)) \tag{4.91}$$

Neural networks consisting of 4 inputs, 2 hidden layers with 20 and 10 nodes and 1 output are used for the controller and emulator. A learning rate of $\eta = 0.2$ is used for both networks. As shown above, a stability constant of $\beta = 0.2$ is used. These parameters provided the best results.

Figure 4.6b shows the controlled response of the plant *without* the enhancements to the model reference scheme being made. The plant output tracks the reference model output, however it still undergoes the negative excursion during its transient period. Figure 4.6c shows the controlled response for the enhanced scheme. As can be seen, excellent tracking is once again achieved, *and* the negative excursion is removed. Furthermore, the response is far quicker and the steady state error is significantly reduced. Hence this represents a significant improvement on many conventional control procedures.

4.4.4 Plant Uncertainty

In this example the robustness of the proposed control scheme to plant uncertainty is investigated. As no mathematical system can exactly model a physical system, it

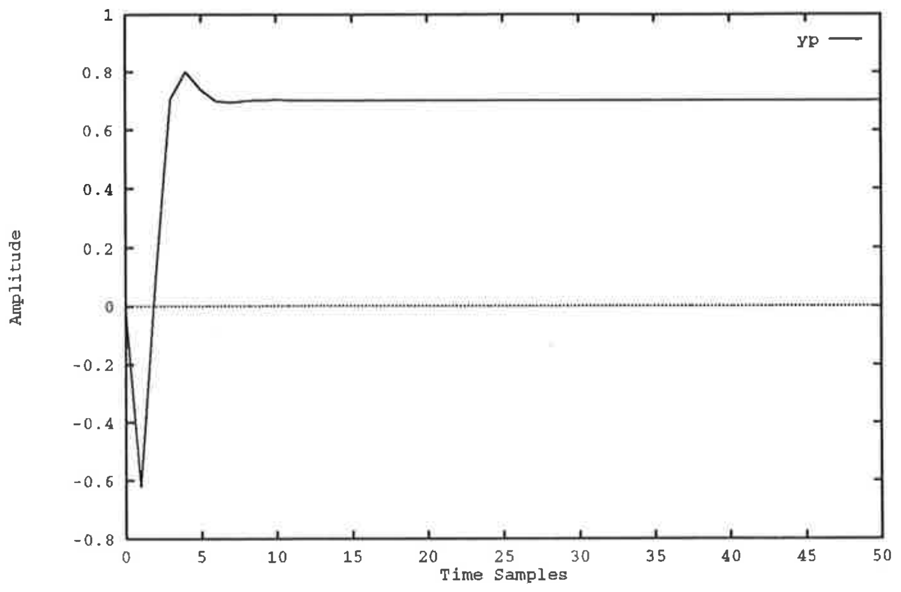


Figure 4.6a: Open-loop step response of the plant (y_p)

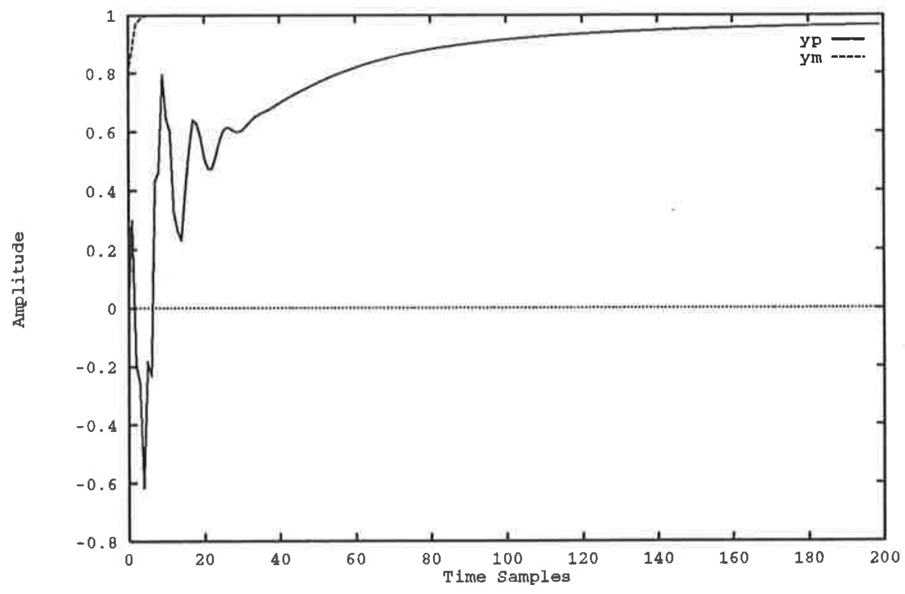


Figure 4.6b: Step response of the plant (y_p) and reference model (y_m) for control *without* the proposed enhancement

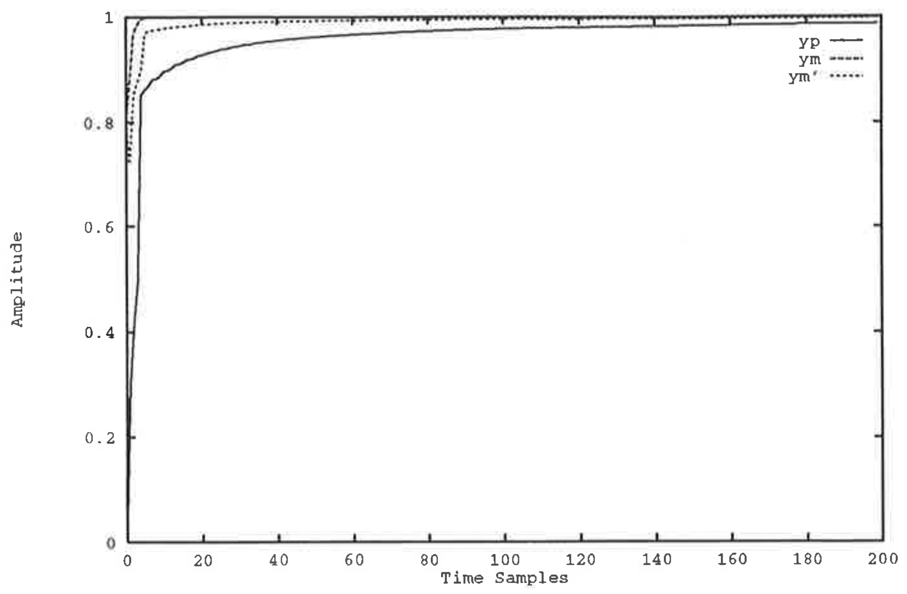


Figure 4.6c: Step response of the plant (y_p), reference model (y_m) and enhanced reference model (y'_m) for the enhanced control scheme

is necessary to be aware of how modelling errors due to the plant uncertainties affect the performance of the control system. Typical sources of uncertainty include unmodelled (high-frequency) dynamics, neglected nonlinearities, over-parameterisation or under-parameterisation, and plant parameter (dynamic) perturbations. In Chapter 2, a technique applicable to nonlinear systems to validate a plant model and help detect over-parameterisation or under-parameterisation in the model is presented. The effects of high frequency dynamics such as additive sensor noise or coloured plant noise are considered in a previous section and in Chapter 3. The latter form of uncertainty, namely plant parameter (dynamic) perturbation is considered in this section.

Dynamic perturbations (or variations in the dynamics of the system) are often due to environmental factors such as temperature, air speed, age or a changing environment. For example in the case of a vehicle, significant dynamic changes result when the car makes the transition from one road surface to another [122]. Another example of a dynamic perturbation is the effect of changing sea-conditions on a ship. As the coefficients of the dynamic ship model are functions of speed [127], this change in environmental conditions results in a variation in the dynamics of the ship system. If the control system performs well for substantial variations in the system dynamics from the design scheme and the stability of the closed-loop system is maintained, then the scheme is said to be robust.

Plant uncertainties of the types described above can be represented in the form of an additive or multiplicative perturbation. For a linear system, this is represented as follows

$$G(z) = G_0(z,\theta) + \Delta G_a(z) \tag{4.92a}$$

$$G(z) = G_0(z,\theta)[1 + \Delta G_m(z)] \tag{4.92b}$$

where ΔG_a is an additive perturbation, ΔG_m is a multiplicative perturbation, G_0 is the nominal plant model and G is the true model of the plant. The perturbations ΔG_a and ΔG_m are known as unstructured uncertainties as they cannot be traced to specific elements of the plant and typically represent unmodelled, neglected or changing dynamics of the system. Structured (or parametric) uncertainty, θ, relates to variations or inaccuracies in the terms actually included in the model of the system, G_0. For example, it may be known that particular parameters in a state-space model vary over a certain range. This represents a structured uncertainty as the variations in the system are known to be due to a particular element in the system. Unstructured uncertainty, ΔG_a or ΔG_m, is usually regarded as more important as disturbances due to unmodelled or variable dynamics in the system arise naturally in practice.

Further details on the relative merits of the additive and multiplicative models and robust control techniques for dealing with them can be found in the collection of papers edited by Dorato [46].

EXAMPLE 4.4.4 Both additive and multiplicative perturbation models are considered to verify the robustness of the neural network based model reference control scheme in the presence of structural variations in the plant. Therefore the two systems considered are given by the following difference equations

$$y_p(k+1) = \frac{y_p^2(k)u(k) + u^3(k)}{1 + y_p^2(k) + y_p(k-1)} + \Delta f_a(k) \tag{4.93a}$$

$$y_p(k+1) = \frac{y_p^2(k)u(k) + u^3(k)}{1 + y_p^2(k) + y_p(k-1)}(1 + \Delta f_m(k)) \tag{4.93b}$$

where the perturbation models used are given by

$$\Delta f_a(k) \quad = \quad \Delta f_m(k) = \begin{cases} 0 & k < 400 \\ \frac{y_p(k)u(k)}{1+y_p(k)} & k \geq 400 \end{cases} \tag{4.94}$$

The reference model considered also consisted of a variable structure. It is given by the following difference equation

$$y_m(k+1) \quad = \quad \begin{cases} 0.2y_m(k) + 0.2y_m(k-1) + r(k) & k < 800 \\ 0.2y_m(k) + 0.5r(k) & k \geq 800 \end{cases} \tag{4.95}$$

In practice a change in the reference model may often occur when the closed-loop response of the system needs to be altered in order to meet the performance requirements. For example, it may be that for the given operating conditions, the response of the closed-loop system needs to be swifter or that the gain of the overall system is insufficient and therefore, the reference model needs to be changed in order to overcome these problems. An example of this situation occurs in the ship steering problem. If the ship is in the open sea, then a slowly responding reference model (say fourth order) may be chosen so that the control activity necessary to achieve the desired response is kept to a minimum, thus reducing the wear-and-tear on the various actuators and rudder mechanism. However, when the ship enters a narrow channel, it often becomes necessary to change the reference model to a faster responding system (say second order) to ensure that the ship is capable of undertaking the relatively quick course changes that may be required in more constricted waters. Therefore, the ability of a control system to handle changes in the desired response is also of major importance.

To demonstrate the variability in the systems, the open-loop step responses for both the additive and multiplicative perturbation cases are provided in Figures 4.7a and 4.7b.

The changes in the plant response for both the additive and multiplicative models are clearly visible at $k = 400$ in Figures 4.7a and 4.7b, respectively. For the additive perturbation model, the mean of the perturbed system is non-zero, whereas the mean of the multiplicative system remains at zero. Therefore, the additive system has a DC bias which may pose a problem for the control system. A further point worth noting about the additive perturbation system is the presence of the large spikes at $k = 500, 700, 900, \ldots$.

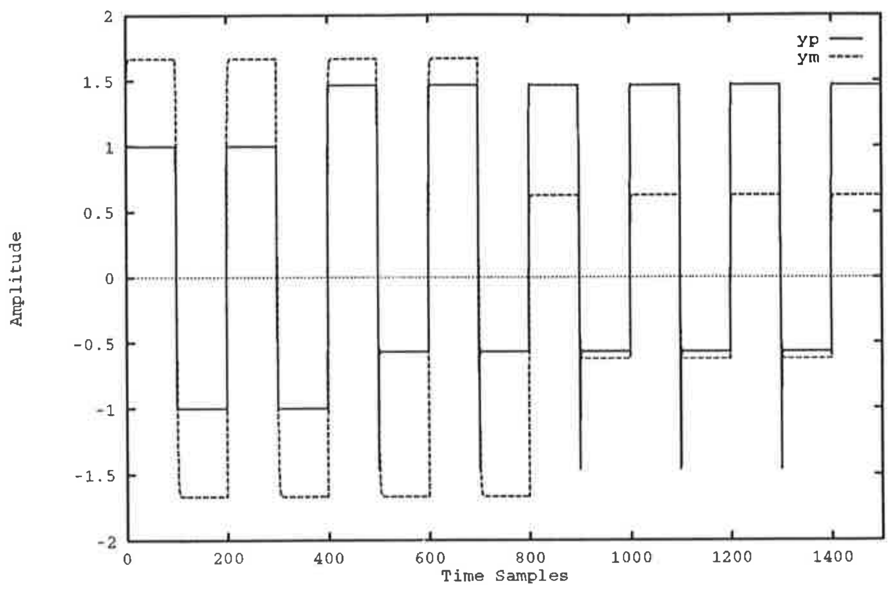


Figure 4.7a: Open-loop response of the plant (y_p) and reference model (y_m) for an additive perturbation model

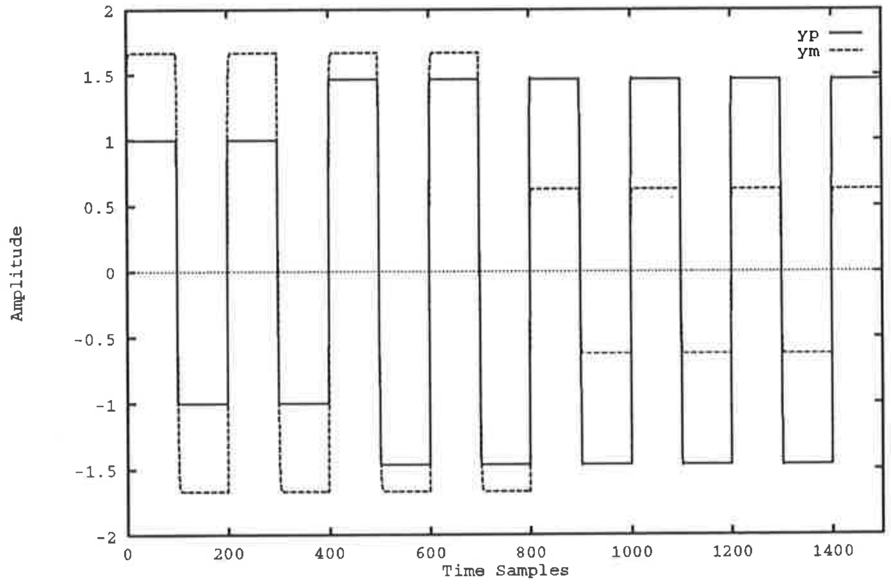


Figure 4.7b: Open-loop response of the plant (y_p) and reference model (y_m) for a multiplicative perturbation model

These spikes are due to the fact that at the instant the control input changes from a positive value to a negative value, the terms $\Delta f_a(k) = \Delta f_m(k)$ and $\frac{y_p^2(k)u(k)+u^3(k)}{1+y_p^2(k)+y_p(k-1)}$ are negative as $y_p(k) > 0$ and $u(k) < 0$ for these values of k. Therefore, the plant response for the additive perturbation system is a larger negative value than the response without the perturbation. However, at the next time sample, the term $\Delta f_a(k)$ becomes positive as $y_p(k) < 0$ and $u(k) < 0$ and thus the plant response is still negative but of a smaller magnitude. This results in the spikes seen in Figure 4.7a. The spikes are not visible for the multiplicative perturbation case because $\mid \Delta f_m(k) \mid < 1$ and therefore the term $1 + \Delta f_m(k)$ is always positive. Therefore, the response of the multiplicative perturbed system is simply a scaled version of the unperturbed system. The change in the reference model is evident at $k = 800$. Although with the time scale chosen, it is difficult to detect the change in response time of the reference model, the change in gain of the system is clearly visible.

The enhanced reference model chosen consisted of (4.95) and

$$y'_m(k+1) = y_m(k+1) - 0.1(y_m(k) - y_p(k)) \tag{4.96}$$

Neural networks belonging to the class $\Omega^3_{3,20,10,1}$ are used for both the controller and emulator networks. A learning rate of $\eta = 0.2$ and a momentum rate of $\alpha = 0.1$ are also used. The plant and reference model responses are provided for both the additive perturbation model and multiplicative perturbation model in Figures 4.8a and 4.8b, respectively. These responses are for the neural control scheme *without* the enhancements proposed in this chapter. The corresponding responses for the enhanced neural control scheme with the enhanced reference model are provided in Figures 4.9a and 4.9b.

As the perturbations do not enter the system until $k = 400$, the performance of the neural controller without the proposed enhancement for both cases is identical over this period. The plant response initially has a significant overshoot. At the instant when the perturbations enter the system, there is a large overshoot for both the additive and multiplicative case, as shown in Figures 4.8a and 4.8b, respectively. In the case of the multiplicative perturbation, the tracking error is reduced significantly. The performance of the additive perturbation system is not as good, with relatively large tracking errors present at the instant when the reference input is changed.

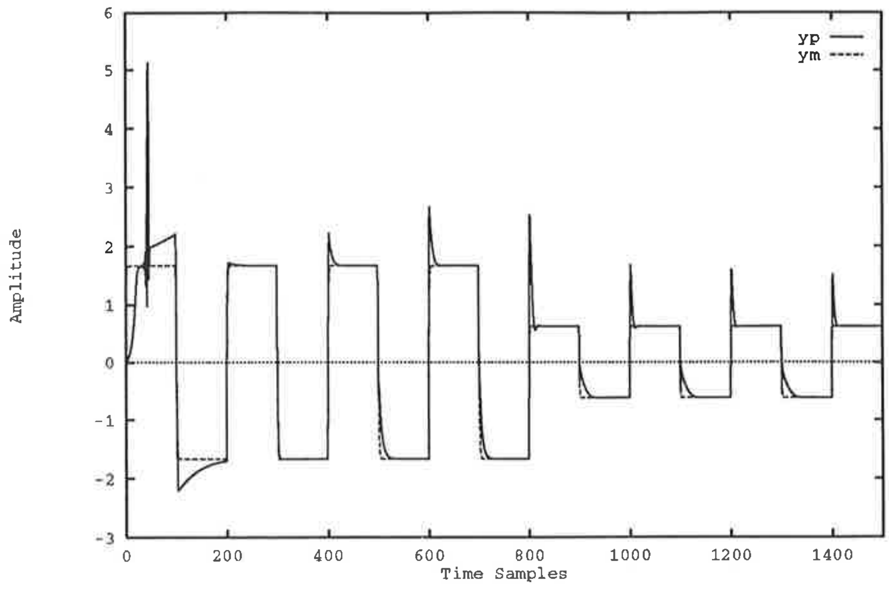


Figure 4.8a: Response of the plant (y_p) and reference model (y_m) for an additive perturbation model *without* the proposed enhancement

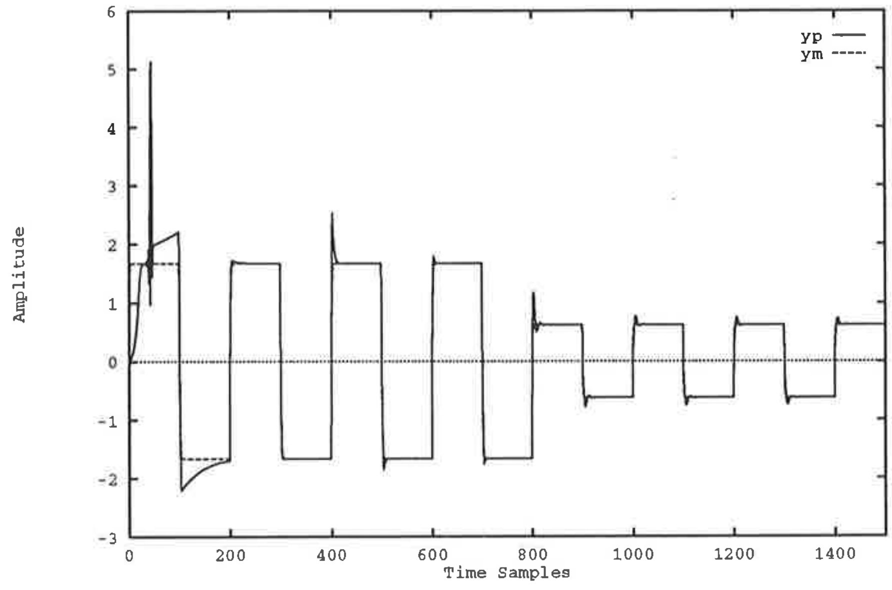


Figure 4.8b: Response of the plant (y_p) and reference model (y_m) for a multiplicative perturbation model *without* the proposed enhancement

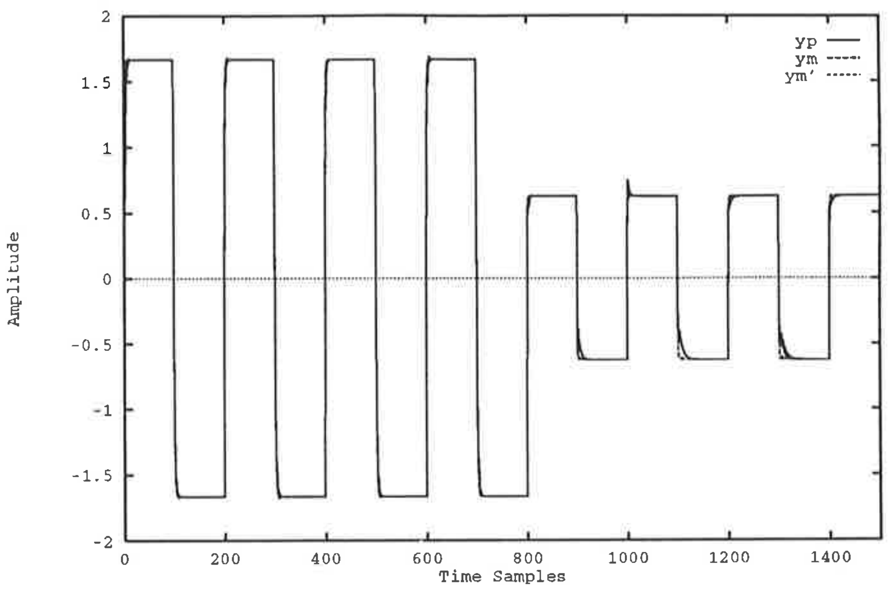


Figure 4.9a: Response of the plant (y_p), reference model (y_m) and enhanced reference model (y'_m) for an additive perturbation model *with* the proposed enhancement

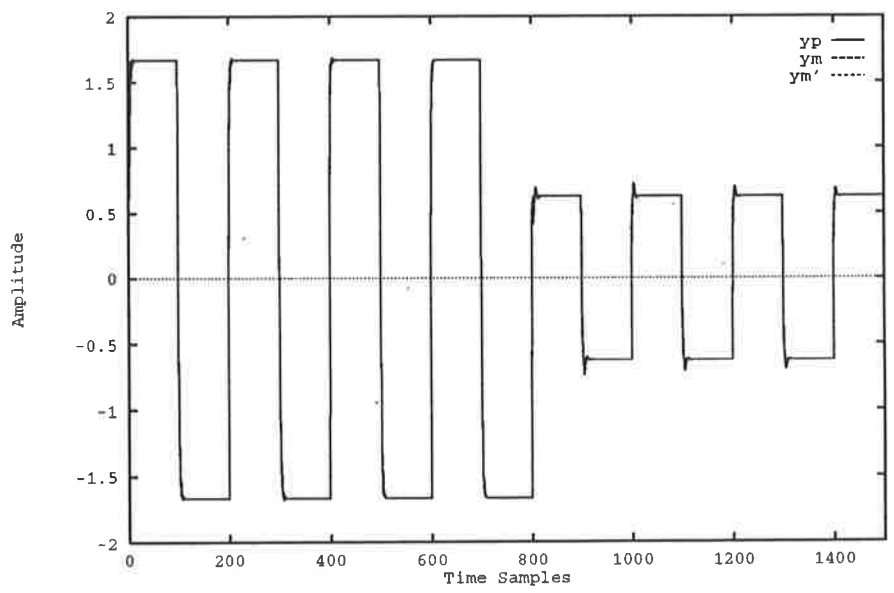


Figure 4.9b: Response of the plant (y_p), reference model (y_m) and enhanced reference model (y'_m) for a multiplicative perturbation model *with* the proposed enhancement

The performance of the control scheme for the multiplicative perturbation case is also very good when the reference model is changed at $k = 800$. Whilst the steady state response of the original neural control system for the additive model (Figure 4.8a) is very good once the reference model is changed ($k \geq 800$), it still displays significant overshoots at the instant of the step changes. The relatively inferior performance of the control system when additive perturbation is present can most likely be attributed to the presence of a DC bias in the plant response.

Figure 4.9a shows the responses for the enhanced neural control scheme with an additive perturbation present. The corresponding responses for the multiplicative perturbation case are provided in Figure 4.9b. Both of these figures highlight the improvement the enhancement makes to the performance of the controller. In particular, the simulation results indicate that the outputs of the unknown plant perfectly track the outputs of the reference model (and enhanced reference model) despite the fact that the structure of the plant is varied during the control process. Furthermore, the control scheme is capable of effectively dealing with the change in reference model. In particular, the relatively large tracking errors present in the original control scheme responses, particularly for the additive perturbation case, are reduced significantly. Finally, the very poor tracking performance in the initial stages of operation ($k = 0, \ldots, 200$) is also significantly improved. Hence, not only does this simulation example demonstrate the effectiveness of the enhanced neural control scheme in controlling an unperturbed nonlinear system, it also demonstrates that the control system has good robustness for time-varying unknown plants and variable reference models.

4.4.5 Marginally Stable Nonlinear Systems

In this section the issue of controlling a nonlinear system which is inherently open-loop marginally stable is considered. As mentioned in Chapter 3, such systems are not uncommon in practice. Classic examples are the ship steering problem [127], the homing missiles guidance problem [78] and the position control of a DC motor problem [10]. In the ship steering problem the open-loop dynamics are such that a small change in rudder angle will result in the ship being locked into a circular path. When the homing missile is in flight, any force at angle to its trajectory will cause it to topple out of balance. For the case of the position control problem, if a fixed input voltage is applied to the DC motor in an open-loop environment, the position of any point on the rotor will continuously change. These three examples highlight the fact that this issue has important practical ramifications.

Many of the neural network based control procedures developed so far are off-line procedures [169, 171]. In these procedures, it is necessary to perform an open-loop identification of the plant. However, this can only be achieved if the system is strictly stable. Therefore, a large class of systems cannot be controlled via these methods. In the method proposed in this chapter, the identification of the plant and the control are done simultaneously and in a closed-loop environment. Therefore, the closed-loop control system can compensate for any tendency towards instability exhibited by the plant. Hence the restriction of open-loop stability on the plant, as in previous work, is removed. Furthermore, it will be shown that the enhancements proposed to the neural adaptive control procedure results in a greatly improved response for the marginally stable system successfully controlled in the previous chapter.

As explained earlier, for the purpose of this simulation study, a marginally stable nonlinear system can be considered to be a stable system cascaded by an integrator. This is expressed as follows

$$\begin{aligned} y'_p(k+1) &= \left(\frac{1}{1-q^{-1}}\right) y_p(k+1) && (4.97a) \\ &= y'_p(k) + y_p(k+1) && (4.97b) \end{aligned}$$

where y'_p is the output of the augmented plant which is nonlinear and marginally stable, y_p is the output of the stable system and q is the shift operator.

The plant considered belongs to a class of systems which are nonlinear in output and control and their respective nonlinearities are *not* separable. Systems belonging to this class are usually the most difficult to control as the control input, $u(k)$, is usually heavily embedded in the nonlinear function.

EXAMPLE 4.4.5 The marginally stable plant expressed in the form of equation (4.97b) is given by

$$y_p'(k+1) = y_p'(k) + \frac{y_p(k)y_p(k-1)u(k) + u^3(k) + 0.5y_p(k-1)}{1 + y_p^2(k) + y_p^2(k-1)} \tag{4.98}$$

The reference model considered is a stable second order discrete-time system given by

$$y_m(k+1) = 0.2y_m(k) + 0.2y_m(k-1) + r(k) \tag{4.99}$$

The enhanced reference model used is formed from (4.99) and (4.11) with the stability constant $\beta = 0.1$.

The performance of the controller is investigated by observing the step responses of the plant and reference model. The adaptive nature of the controller is also investigated by changing the reference input. Therefore the following reference input is used to demonstrate the effectiveness of the controller.

$$r(k) = \begin{cases} s(k-25) & k < 600 \\ \sin(\frac{2\pi k}{100}) & k \geq 600 \end{cases} \tag{4.100}$$

where $s(k-25)$ is a step at $k = 25$.

The plant and reference model responses for this input are given in Figure 4.10a. The control input for the plant response is given in Figure 4.10b. As the plant is second order in output and first order in control, neural networks consisting of 3 inputs, 2 hidden layers with 20 and 10 nodes, respectively, and 1 output are used for the controller and emulator. A learning rate of $\eta = 0.05$ is employed.

Figure 4.10a shows the transient response for the neural controller. This figure highlights the excellent performance of the controller. In fact apart from the first few samples, there is very little difference between the desired reference model output and the plant output. The enhanced reference model output also rapidly converges to the reference model output. The performance of the system does not deteriorate when the reference input is changed, thus demonstrating the ability of the controller to adapt to a change in input. Comparing these results with the responses obtained in Chapter 3 (Figure 3.22a) shows that significant improvement is obtained by employing the enhancement proposed in this chapter. Not only is the tracking error reduced, but the oscillations and overshoot present in the transient response of Chapter 3 are also removed. The trade-off in achieving these

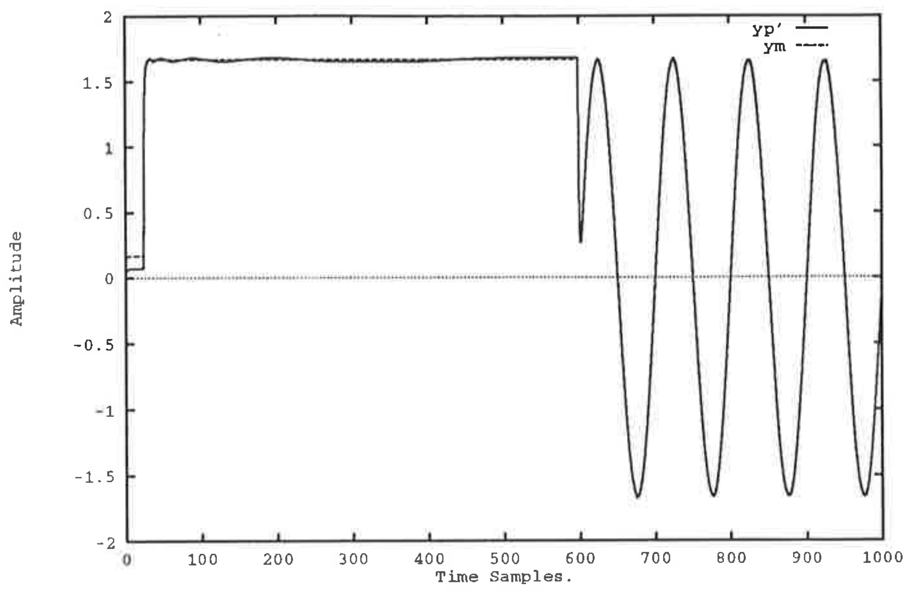


Figure 4.10a: Response of the plant (y_p') and reference model (y_m) for Example 4.4.5 with $r(k) = s(k-25)$ for $k < 600$ and $\sin(\frac{2\pi k}{100})$ for $k \geq 600$

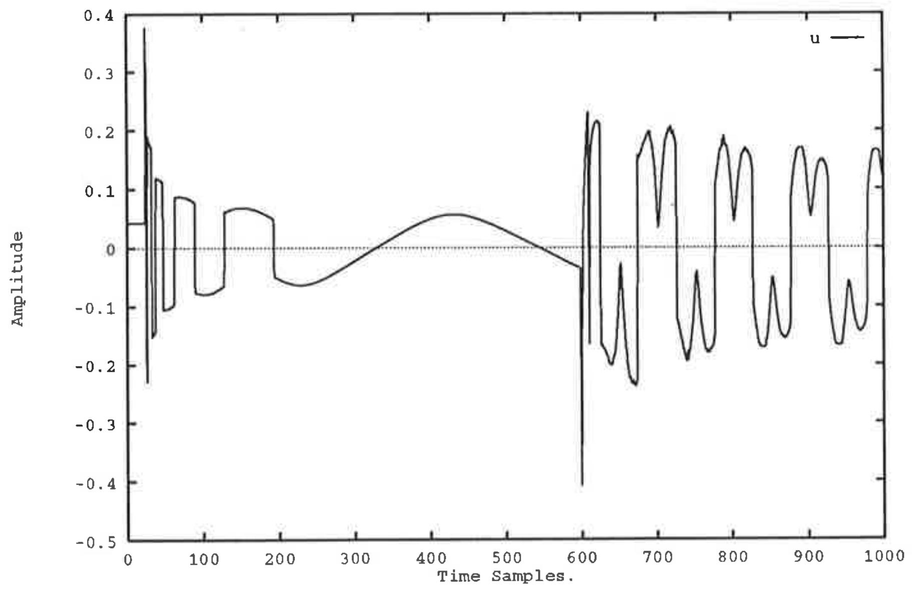


Figure 4.10b: Control input $u(k)$ for Example 4.4.5 with $r(k) = s(k-25)$ for $k < 600$ and $\sin(\frac{2\pi k}{100})$ for $k \geq 600$

tracking results is that the control effort is very active and more so than the control response of the corresponding example in Chapter 3 (Figure 3.22b). As explained earlier high control activity often poses problems for the control mechanism. However, this result is not unexpected as the output of a marginally stable system is being made to track the output of a linear stable system. Therefore, the increased activity is most likely due to the fact that the plant response tracks the linear reference model response far more closely, which for this type of system, requires a great deal more control effort.

4.5 Conclusions

This chapter represents the main theoretical part of this thesis. Two of the the most important concepts of any control system are addressed, namely stability of the neural adaptive controller and convergence of the output error.

The chapter introduces the concept of an enhanced reference model. The origin of this concept lies in the the field of optimal control and in particular an approach known as the optimal decision strategy (ODS). This strategy has been applied to a number of practical problems including robotics, terrain tracking and ship steering. An integral part of this approach is the enhanced reference model, which combines the reference model with an enhanced output function, also known as the desired velocity function. This function is chosen *a priori* by the control system designer. It is shown that the form of the enhanced reference model is intimately related to the control approach adopted. A form previously suggested for a model reference adaptive control scheme is adopted in this thesis.

An enhancement to the neural network based model reference adaptive control scheme presented in Chapter 3 is then suggested. This primarily involves the incorporation of the enhanced reference model into the neural adaptive control scheme. The enhancement gives rise to a number of theorems for which proofs are also furnished. The three main theorems are summarised below.

- Theorems 4.3.1 and 4.3.2 state that in order for the tracking error for the model reference system to converge to zero, certain *sufficient* limit conditions must be satisfied.

- Theorem 4.3.4 states that there exists a Lyapunov function such that the closed-loop system is stable on a neighbourhood of zero. This theorem also gives rise to a further *sufficient* limit condition.

These theorems and their associated proofs address the principle theoretical aims of this chapter and represent two of the major findings of this work. The remainder of the chapter deals with the practical implementation of the enhanced neural control scheme and demonstrating the effectiveness of the scheme via simulation studies.

The neural adaptive control scheme proposed in this chapter is derived for a general multivariable system, such that the only restriction on the form of the enhanced reference model is that the matrix A is stable. However, for the sake of simplicity, in all of the multi-input multi-output systems considered, a further restriction is placed on the matrix A, namely that it is diagonal. This restriction results in Theorem 4.3.3 which states a condition for the existence of an upper bound on the elements of the diagonal matrix A in order for Theorem 4.3.1 to hold.

As with the approach of Chapter 3, the neural adaptive control scheme proposed in this chapter uses an approximation of the controller error to adjust the weights of the controller neural network. Theorem 4.3.5 defines a form for the approximate controller error. Theorem 4.3.6 then states how this error can be obtained. This result is a modification of the controller error result provided in Chapter 3 and originally proposed by Jordan [95].

The effectiveness of the enhanced neural adaptive scheme is then demonstrated through a number of simulation examples. Firstly, the performance of the scheme on the single-input single-output system and multi-input multi-output system considered in the on-line scheme of Chapter 3 is investigated. Comparison with the results provided in Chapter 3 showed that significant improvement is achieved, not only in the accuracy of the tracking responses, but also in terms of reduced control activity.

A practical treatment of the robustness of the scheme to various disturbances is also undertaken. In particular, the scheme is found to be robust to (*i*) variable load disturbances, which arise because of a change in the load conditions on the system, (*ii*) sensor noise, which arises because of errors or noise induced by the measuring devises and (*iii*) dynamic plant noise, which can be produced by a host of sources, including noise due to internal components, unmodelled high frequency dynamics or environmental factors such as wind or waves.

Several commonly occurring non-ideal systems are also considered to highlight the practical viability of the scheme. Firstly, a nonminimum phase nonlinear system is considered. Such a system can be defined as having an unstable zero dynamics. It is shown that the proposed scheme is capable of dealing with such systems mainly because it does not involve the explicit inversion of the plant dynamics. It is also shown that this approach removes the problematic negative excursion present in the (positive) step response of nonminimum phase systems.

The robustness of the scheme to variations in the dynamics of the system is also investigated. Such variations are common in practical systems and can represent changes in the environmental or operating conditions of the system. It is shown that these variations can be represented by two types of perturbation models, namely additive perturbations or multiplicative perturbations. The scheme is shown to be extremely effective in dealing with both forms of perturbations. The effect of a change in the reference model and therefore a change in the desired closed-loop response of the system is also investigated. The scheme is also found to be robust to these changes.

Finally, the ability of the proposed scheme to control a marginally stable nonlinear system is investigated. This system is simulated by cascading an integrator on to the output of a stable nonlinear system. The performance of the scheme is found to be very good particularly in comparison with the results for the corresponding example in Chapter 3.

In summary, this chapter provides the theoretical and practical justifications for the neural adaptive control scheme proposed in this thesis. Results for the stability of the control scheme and convergence of the tracking error are provided to establish the practical viability of the approach. However, the results are based on *sufficient* limit conditions. The derivation of a *necessary and sufficient* condition, which represents the natural extension of this work, remains a very difficult task. Despite the increasing volume of work being conducted into stability and convergence results for neural adaptive control schemes, general results have proven extremely elusive.

Chapter 5

Application Example – Anti-Skid Brake System

5.1 Introduction and Overview

Demonstrating the effectiveness of a new control strategy, such as the neural adaptive control scheme proposed in this thesis, via a simulation study is a rewarding exercise. Not only does it help to clarify and reinforce the theoretical concepts raised by the proposed scheme, it also provides an indication of its practical feasibility and limitations. From an engineering viewpoint, a great deal of knowledge can be gained from simulation studies. Further knowledge about the practical feasibility of a proposed control scheme can be gained from its application to "real world" problems. This has provided the motivation for a great deal of research in the application of neural control methodologies to a wide range of practical areas such as robotics, power systems, biomedical and biotechnical engineering, process control, engine control and so-on. A good collection of some recent application based neural control papers can be found in [64]. However, because of the restrictive costs of practical test equipment and the lack of availability or accessibility of the "real world" system, most of the existing works on neural control applications have been based on computer simulations of the actual system. It is perhaps for this reason that so many neural control methodologies are tested on the inverted pendulum or ball and beam apparatus. The relatively low cost and availability of such systems make them a good test-bed for a new approach. However, the primary limitation with the inverted pendulum and similar apparatus is that they are essentially laboratory test-beds and they do not really represent the significantly more complex and interesting "real world" systems. A similar problem was encountered in the adaptive control area during its formative years. However, since then, many theoretical developments have been made and tested on industrial control problems such as aircraft control, ship autopilots, process control, robotics and power systems [64]. Whilst the ideal situation would be to test a proposed control scheme on an actual physical system, a realistic and relatively detailed simulation of the system can often prove to be equally beneficial. Certainly, it provides an additional knowledge on the "real world" potential of the control approach compared with simulation studies based on arbitrary systems.

In this chapter, the application of the model reference neural adaptive control scheme proposed in this thesis to an anti-skid brake system (ABS) is presented. The objective of an ABS is to maximise tyre traction by preventing the wheels from locking during braking whilst maintaining adequate vehicle stability and steerability. Basically an anti-skid brake system comprises sensors to detect imminent wheel-locking together with a system to modulate the applied brake pressure so that the braked wheel maintains a tangential speed slightly less than the linear speed of the vehicle. This enables the tyres to retain most of their lateral force capability which in turn helps to keep the vehicle steerable and the overall system stable. Furthermore, the stopping distance of a vehicle equipped with ABS is, in most cases, reduced in comparison with a stop achieved by wheel-locking. Wheel lock is particularly undesirable for a number of reasons. Under normal driving conditions, a driver judiciously applies the brake to ensure wheel lock does not occur. However, under an emergency situation, the natural instinct is to apply maximum brake effort, thus resulting in either (*i*) the front wheels locking up or (*ii*) the rear wheels locking up or (*iii*) all four wheels locking up. If situation (*i*) occurs then the driver will be unable to steer the vehicle[1]. If the rear wheels lock, but the front wheels are not locked then the vehicle generally loses directional stability. Situation (*iii*) will cause the car to skid in a straight line [177]. To prevent the above situation from occurring,

[1]This assumes a front wheel steering vehicle.

an anti-skid brake system is employed.

A great deal of research and development effort has been conducted into anti-skid brake systems because it is a recognised fact that they can produce significant improvements in the safety performance of commercial and passenger vehicles. In a recent study of accidents involving heavy goods vehicles and cars in Great Britain [195], it is concluded that the use of ABS is likely to have affected (prevented or reduced in severity) 10% of accidents involving heavy good vehicles alone and 9% of accidents involving heavy good vehicle and cars. This translates to about 120 lives that could have been saved through the use of ABS. Studies conducted in the then West Germany have verified a reduction in the number and seriousness of accidents involving ABS-equipped vehicles compared with conventional vehicles [58]. Research is still being undertaken to develop even more safe and reliable anti-skid brake systems and in conjunction with the recent developments in traction control (also known as anti-wheel spin regulation (ASR)), the ultimate aim is to further improve the active safety of vehicles and thus the general safety on roads [58, 183, 212].

In the last few years, there has been a recognition of the suitability of applying neural networks to the modelling and control of vehicle systems. In particular, the ability of neural networks to adapt to highly nonlinear and time-varying dynamical systems has provided the motivation for much of the work in this area. Recently, Ohno *et al.* [175] have developed an automatic braking system for automobiles using neural networks. The aim of the proposed system is to make the vehicle decelerate smoothly and come to a stop at a specified position behind the preceding stopped vehicle. To achieve this, the vehicle speed is controlled via the neural network controller by varying the brake solenoid current according to the distance between the two vehicles. The controller is shown to be able to adapt to changes in the weight of the vehicle and the gradient of the road. Majors *et al.* [133] have developed a neural network methodology to control the air-to-fuel ratio of automotive fuel injection systems using the cerebellar model articulation controller (CMAC) architecture. The resultant fuel-injection controller regulates the A/F ratio to a value chosen to result in driveability and good emission performance. It is also shown to be very effective in learning the nonlinearities of the engine system and in dealing with the time-delays inherent in the engine sensors. A neural network based engine control system is also considered by Morita [159]. In this approach, a backpropagation neural network is used to control combustion parameters of an automobile gasoline engine. In particular, the neural controller is shown to be effective in controlling the spark ignition rate and fuel injection volume as well as being able to compensate for conditions such as the warming-up of the engine and variations due to environmental factors. The application of neural networks to active suspension systems has been addressed by Hampo and Marko [65] and Smith *et al.* [219]. In [65], a number of neural control approaches are applied to an active suspension system to maintain a car at a level attitude. The fault tolerance of the approaches to actuator failures and sensor failures is also investigated. In [219], an artificial intelligence approach to the control of a semi-active suspension system is presented. The approach uses a heuristic search strategy to derive an optimal control for the suspension system, which is then learnt by an artificial neural network to provide an on-line nonlinear optimal control law. Neural networks are applied to the lateral control of autonomous vehicles by Kornhauser [115]. A machine vision system is proposed in which the driver's view of the oncoming highway environment is captured. Backpropagation and adaptive resonance neural networks are investigated for processing the resulting images and generating acceptable steering commands for the vehicle. However, despite the recent interest in applying neural network methodologies to vehicle systems, there does not seem to be any reported work on the application of neural control to the ABS problem.

The anti-skid brake system is a challenging control problem because the vehicle–brake dynamics are highly nonlinear with uncertain time-varying parameters. These parameter variations are due to factors such as changes in the brake pad coefficient of friction, changes in the gradient of a road, and variations in the friction characteristics of the tyre/road contact. The latter variations are perhaps the most significant and are produced by differences in the methods of road surface construction, contaminants on the road, changes in tyre parameters such as tread wear and variations in the speed of the vehicle [67, 122, 230]. Furthermore, detailed models of the vehicle–brake system or accurate predictions of the tyre/road surface conditions are currently difficult and expensive to obtain for practical anti-skid brake systems [230]. Therefore, not only does an ABS have to be able to adapt to changing environmental conditions (e.g. varying road conditions), it must also be robust to dynamic uncertainties and variations. Currently most commercial anti-skid brake systems are based on a look-up table approach. These look-up tables are calibrated through iterative laboratory experiments and engineering field tests. Therefore, these systems are not adaptive and issues such as robustness are not addressed. More recently some theoretical studies of anti-skid brake systems have begun to appear in the literature [53, 177, 229, 230, 253]. In particular, conventional control approaches such as sliding control, describing function methods and optimal control have been used. In many of these approaches, the effects of the above parameter variations have been considered. Even more recently, an adaptive anti-skid brake system based on fuzzy logic control is presented by Layne *et al.* [122]. It is shown in the previous chapter that the proposed neural control scheme possesses the properties described above. In particular, the adaptive nature of the approach will help to ensure that the performance of the proposed neural network based ABS is still satisfactory even when adverse road conditions are encountered.

The chapter is organised as follows. A mathematical model based on the single-wheel rotational dynamics and the linear vehicle dynamics is presented in Section 5.2. The interaction between these dynamics and the effect of different road surfaces on the model is also considered. A discretised version of the vehicle–brake system dynamics based on an Euler discretisation process is also presented. An overview of existing ABS approaches is presented in Section 5.3. This will provide a good background for some of the difficulties associated with this problem. The motivation for the work presented in this chapter is the belief that neural networks are extremely suitable for the anti-skid brake system and the lack of any previously reported work on this application. Therefore, in Section 5.4, the unique approach of applying the neural control scheme proposed in this thesis to the ABS problem is considered. The effectiveness of the approach, particularly to changing road conditions is then demonstrated via simulation results. Section 5.5 then provides some concluding remarks.

5.2 Derivation of the System Dynamics

The simplified model of the vehicle–brake dynamics presented by Layne *et al.* in [122] is employed in this study. The mathematical model consists of single wheel rotational dynamics, linearised vehicle dynamics and the interactions between them. The actuator dynamics, wind resistance effects and all the dynamics associated with the suspension system and steering mechanism are not considered. A list of the relevant variables and parameters and, where applicable, their typical values is provided in Table 5.1.

parameter	physical meaning	typical value[a]
B_w	viscous friction of the wheel	4 N s
B_v	viscous friction of the vehicle	6 N s
J_w	rotational inertia of the wheel	1.13 N m s^2
M_v	vehicle mass	4x342 kg
R_w	wheel radius of free rolling tyre	0.33 m
g	gravitational acceleration constant	9.81 m s^{-2}
n_w	number of wheels	4
θ	angle of incline of the road	rad
N_v	normal force at the tyre	N
V_v	vehicle linear speed	m s^{-1}
ω_v	angular speed of a free spinning wheel	rad s^{-1}
ω_w	wheel angular speed	rad s^{-1}
T_b	braking torque at the wheel	N m
T_t	torque generated due to the slip between the wheel and the road	N m
F_θ	force applied to the car due to the gradient of the road	N
F_t	tyre friction force	N
μ	coefficient of braking force	–
λ	wheel slip	–

[a]These values are obtained from Fling and Fenton [53] and are for a 1969 Plymouth sedan

Table 5.1: vehicle–brake system parameters and variables

The differential equation describing the vehicle dynamics is obtained from Newton's law by summing the total forces which are applied to the vehicle during braking, i.e.,

$$\dot{V}_v(t) = -\frac{1}{M_v}[n_w F_t(t) + B_v V_v(t) + F_\theta(t)] \tag{5.1}$$

where $F_\theta(t) = M_v g \sin\theta(t)$ is the force applied to the car as a result of a vertical gradient in the road and $F_t(t) = \mu(\lambda)N_v(\theta(t))$ is the tyre friction force which is a function of the tyre normal force $N_v(\theta(t)) = \frac{M_v}{n_w} g \cos\theta(t)$. The equation for the normal force assumes that the weight of the car is distributed evenly amongst all the wheels.

The dynamics of the wheel can be determined from Newton's law, i.e., the angular acceleration of the wheel ($\dot{\omega}_w$) is the sum of the rotational torques which are applied to the wheel divided by the moment of inertia of the wheel. This results in the following differential equation

$$\dot{\omega}_w(t) = \frac{1}{J_w}[-T_b(t) - B_w \omega_w(t) + T_t(t)] \tag{5.2}$$

where $T_t(t) = R_w F_t(t)$ is the torque generated due to the slip between the wheel and the road surface. A block diagram of this system is shown in Figure 5.1.

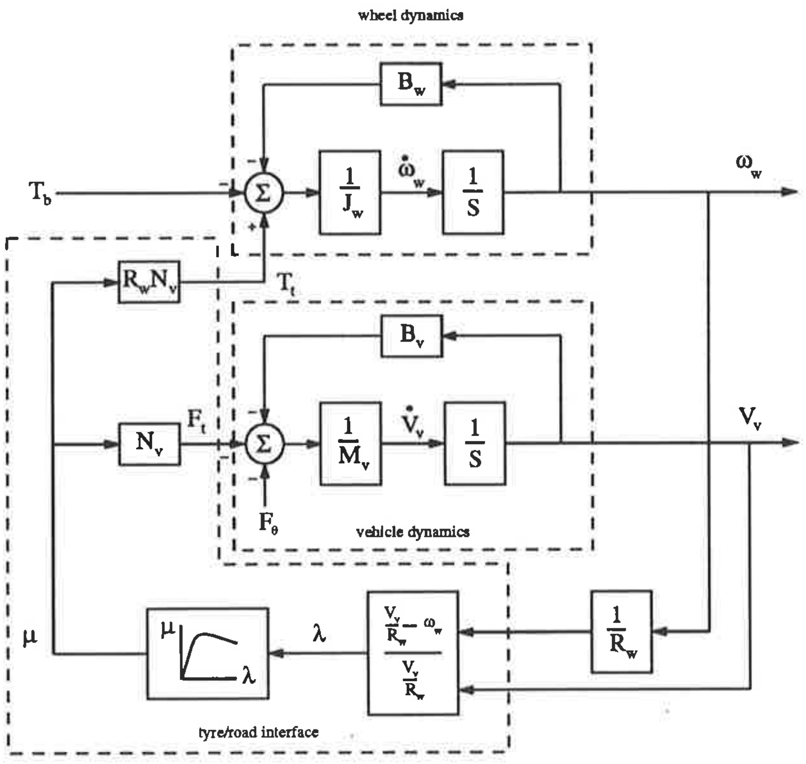


Figure 5.1: Block diagram of the vehicle/wheel/road dynamics

The most significant variable in the vehicle–brake system is the wheel slip, λ. For a braking operation, its value is the difference between the vehicle speed, or more accurately, the angular speed of the free spinning wheel, $\omega_v = \frac{V_v}{R_w}$, and the wheel speed, ω_w, normalised by the vehicle speed, i.e.,

$$\lambda(t) = \frac{\frac{V_v(t)}{R_w} - \omega_w(t)}{\frac{V_v(t)}{R_w}}$$

$$= \frac{\omega_v(t) - \omega_w(t)}{\omega_v(t)} \tag{5.3}$$

Wheel slip arises because of the fact that when a braking force is applied to a tyre, a frictional force is developed at the contact spot between the tyre and the ground. At the same time, the tyre tread within and just in front of the contact area is subject to tension. As a result, the tread elements are stretched before entering the contact region, and thus the distance travelled by the tyre per revolution when subject to this braking torque is greater than the free rolling case [238, 251].

Wheel slip is usually chosen as the controlled variable in anti-skid braking systems because of its influence on the tyre friction force F_t, and its corresponding torque, T_t. This influence arises due to the relationship between the wheel slip and the adhesion coefficient (or braking friction coefficient), μ. The adhesion coefficient is the ratio between the frictional force F_t and the normal force N_v, and for a given level of wheel slip, it has a unique value for each road surface. The nonlinear relationship between the adhesion coefficient and the wheel slip is represented by a $\mu - \lambda$ characteristic. These characteristics are strongly influenced by both the material and method of construction of the road as well as tyre parameters such as the tread wear. Factors such as the vehicle speed and vertical load also have an influence on this characteristic. It is the $\mu - \lambda$ relationship which is the major source of nonlinearity in the system. The adhesion coefficient–wheel slip characteristics for four common road surfaces are given in Figure 5.2.

Analytical expressions which accurately describe the relationship between the adhesion coefficient and wheel slip are not readily available. Therefore, most ABS engineers in the automobile industry obtain these characteristics from field experiments using specialised test equipment. For the study presented here, the above characteristics are therefore based on experimentally measured $\mu - \lambda$ data provided by Harned *et al.* in [67].

From equation (5.3) and Figure 5.2, it is apparent that $\lambda = 0$ corresponds to a free rolling wheel ($\omega_w = \omega_v$ and $\mu(\lambda) = 0$) while $\lambda = 1$ represents a wheel that is locked ($\omega_w = 0$). It is the latter case that an ABS is designed to prevent from occurring. At the same time, an anti-skid brake system attempts to maximise the tyre traction by maximising the coefficient of braking friction, μ. As is expected, the braking friction coefficient is largest for dry asphalt, slightly reduced for wet asphalt, greatly reduced for loose gravel

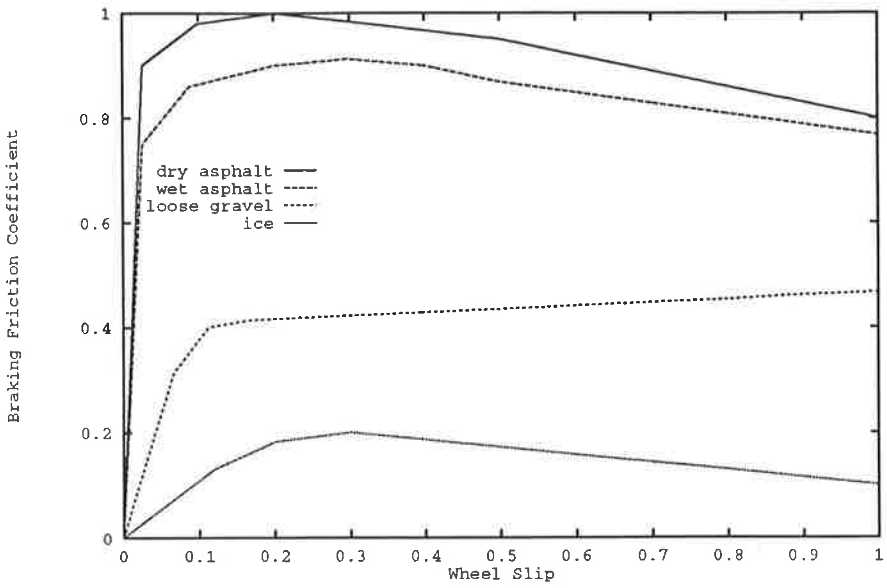


Figure 5.2: Adhesion coefficient vs wheel slip for various road surfaces

and most significantly reduced for ice. Apart from the loose gravel surface, each of the $\mu - \lambda$ characteristics consists of a positive slope region and a negative slope region. When the system is operating in the negative slope region, the vehicle is unstable and sensitive to disturbances, and the cornering properties of the tyres may easily result in a loss of directional control or lateral stability [230]. The positive slope region of the $\mu - \lambda$ characteristic is the stable region of operation for the vehicle–brake system. Therefore, it is desirable to operate in this region of the curve. For the unique case of loose gravel, maximum braking friction occurs when the wheels are locked, i.e., $\lambda = 1$ and $\omega_w = 0$. This is because when a car is skidding on loose gravel, the gravel bunches up in front of the wheel, resulting in an increase in the frictional effect. However, when the wheels are rolling, the gravel gets pushed aside and thus the adhesion effect is less. As a result, the $\mu - \lambda$ curve for loose gravel consists only of positive slope regions and therefore the adhesion coefficient increases with slip to its maximum value at $\lambda = 1$.

Based on the $\mu - \lambda$ characteristics, the objective of an anti-skid brake system would be to regulate the wheel slip so as to maximise the adhesion coefficient for a given road surface. However to ensure that the system does not move into the unstable region of operation, a slightly more conservative value of desired slip is chosen by many ABS engineers. Typically the vehicle–brake system is made to operate within its positive slope $\mu - \lambda$ region and it is regulated to the slip ($\lambda_{\mu_{\max}}$) which produces maximum braking friction only when a maximum braking force is required. For the four surfaces considered in this study, a conservative value of slip of $\lambda = 0.15$ is used.

As the particular road surface has a significant and unique effect on the braking characteristic, a controller which can operate under all road conditions and is robust to transitions in road surfaces is desired. A neural network based controller offers much promise in this area. Furthermore, detailed models of the vehicle–brake system or accurate estimation of the tyre/road surface conditions are currently difficult and expensive to obtain. Therefore, the ability of the neural network control scheme proposed in this thesis to deal with model inaccuracies and uncertainty is advantageous for an anti-skid braking system.

A difficulty in implementing an anti-brake system is that the slip is difficult to measure directly. As shown above, the slip is related to the angular speed of the wheel and the linear speed of the vehicle. Many of the commercial anti-skid brake systems utilise wheel-based sensors to measure the angular speed and/or angular acceleration of the wheel. Typically such a sensor consists of a wheel driven toothed disc and an inductive pick up which produces a continuous electrical signal proportional to the angular speed of the wheel [177]. However, the difficulty in determining a value of slip arises because measuring the linear speed and/or acceleration of the vehicle during braking is a problematic task. In particular, during a braking operation, the presence of wheel slip means that the linear speed of the vehicle is no longer directly related to the angular wheel speed via the equation $V_v = R_w\omega_w$. Unlike an anti-spin device, where the vehicle speed can be obtained from sensors measuring the angular speed of the nondriven wheels, wheel sensors cannot be used to accurately obtain the vehicle speed in ABS mode as braking is carried out on all four wheels. In many commercial anti-skid brake systems the vehicle speed is extrapolated from an initial speed and the integration of an acceleration signal [241] or the relationship $V_v = R_w\omega_w$ is used and the resultant errors are accounted for in some way [243].

As far as test systems are concerned, an extra free rolling wheel can be used to measure the vehicle speed [67]. However, this is of course not practical for a commercial system. In the approaches proposed by Tan *et al.* [230, 229] it is assumed that vehicle speed measurements are available. However, it is also pointed out that sensors to provide accurate vehicle speed measurements are generally expensive and so the ABS schemes proposed in [230, 229] are not readily implementable. Recently Doppler radar ground speed sensors have been developed and implemented in experimental ABS and 4WD/Off-Road ASR systems[2] [132, 161, 241]. These sensors measure the vehicle speed according to the Doppler radar principle. Typically such sensors consist of a transmitter/receiver which sends an extremely-high-frequency (EHF) signal in the direction of the road surface. The road acts as a reflector, and as a result of the Doppler effect, a frequency shift of the reflected signal occurs. The vehicle speed can then be found from the Doppler frequency shift. These sensors have already been fitted to some agricultural vehicles such as tractors and are relatively inexpensive [132].

In [122], Layne *et al.* assume that vehicle acceleration measurements are available via an accelerometer[3]. However, it is also stated that the use of an accelerometer introduces problems such as noise and integration error and that their effects on the ABS need to be investigated thoroughly. Recently, a scheme which addresses the noise issue in accelerometer measurements is proposed by Watanabe *et al.* [243]. A Kalman filtering approach for accurately estimating the vehicle speed is presented. Accelerometer measurements are used to complement the measurements obtained from the wheel based sensors to provide a more accurate vehicle speed. An extended Kalman filter is used to reduce the high frequency noise associated with the accelerometer measurements. An accelerometer located near the centre of mass is also utilised to measure the vehicle deceleration by Fling and Fenton in [53]. However it is shown that an accelerometer alone is not suitable for estimating the vehicle deceleration because of its internal offset error. It is also explained that wheel tachometers should not be used during emergency braking operations as they have the tendency to develop limit-cycle behaviour. An estimation circuit is devised which combines the accelerometer and wheel tachometer measurements to cancel

[2]Four wheel drive (4WD) systems suffer from the same problem as anti-skid brake systems in that in both cases there are no free-rolling wheels from which the vehicle speed can be obtained.

[3]Accelerometers typically consist of a pendulum or resonant fork arranged such that any deflection in the position of the pendulum or changes in the resonant frequency of the fork is proportional to the acceleration of the vehicle. Further details on accelerometers can be found in [208] and [220].

out the effect of the DC offset in the accelerometer and the limit-cycle oscillations in the wheel tachometer output to produce acceleration and velocity estimates. An estimation scheme to determine the vehicle speed is also adopted by Satoh and Shiraishi [209]. In the scheme presented, the estimation is performed by using a memory which stores the peak value of wheel speed. This value is reduced by a specified rate whenever the actual wheel speed decreases sharply. The output of the memory is assumed to approximate the vehicle speed.

Hence, in this study, it is assumed that some form of estimation scheme could be implemented such that a measurement of the vehicle speed is available. This assumption allows a dynamic system equation for the wheel slip with the vehicle and wheel speed as state variables to be derived as follows:

From (5.3) the wheel slip can be written as

$$\lambda(t) = 1 - \frac{\omega_w(t)}{\omega_v(t)} \tag{5.4}$$

Therefore,

$$\begin{aligned} \dot{\lambda}(t) &= \frac{\omega_w(t)\dot{\omega}_v(t) - \dot{\omega}_w(t)\omega_v(t)}{\omega_v^2(t)} \\ &= \frac{\frac{\omega_w(t)\dot{\omega}_v(t)}{\omega_v(t)} - \dot{\omega}_w(t)}{\omega_v(t)} \\ \dot{\lambda}(t) &= \frac{(1-\lambda(t))\dot{\omega}_v(t) - \dot{\omega}_w(t)}{\omega_v(t)} \end{aligned} \tag{5.5}$$

For the sake of simplicity, the vehicle system equation (5.1) is rewritten as follows:

$$\begin{aligned} \dot{\omega}_v(t) &= \frac{\dot{V}_v(t)}{R_w} = \frac{1}{R_w}[-\frac{1}{M_v}(B_v V_v(t) + n_w F_t(t) + F_\theta(t))] \\ &= -\frac{B_v R_w}{M_v R_w}\omega_v(t) - \frac{n_w N_v(\theta(t))}{M_v R_w}\mu(\lambda) - \frac{F_\theta(t)}{M_v R_w} \\ &= -a_2\omega_v(t) - a_1\mu(\lambda) - a_0 \end{aligned} \tag{5.6}$$

where $a_2 = \frac{B_v}{M_v}$, $a_1 = \frac{n_w N_v(\theta(t))}{M_v R_w} = \frac{g n_w \cos\theta(t)}{n_w R_w}$, $a_0 = \frac{F_\theta(t)}{M_v R_w} = \frac{g \sin\theta(t)}{R_w}$, and the wheel dynamics, (5.2), can be rewritten as:

$$\begin{aligned} \dot{\omega}_w(t) &= -\frac{B_w}{J_w}\omega_w(t) - \frac{1}{J_w}T_b(t) + \frac{1}{J_w}T_t(t) \\ &= -b_2\omega_w(t) - b_1 T_b(t) + b_0\mu(\lambda(t)) \end{aligned} \tag{5.7}$$

where $b_2 = \frac{B_w}{J_w}$, $b_1 = \frac{1}{J_w}$, $b_0 = \frac{R_w N_v(\theta(t))}{J_w} = \frac{M_v g R_w \cos\theta(t)}{n_w J_w}$.

As the neural controller and subsequent theory proposed in this thesis are based on discrete-time systems, the above differential equations are discretised using an Euler discretisation process. This results in the following difference equations:

$$\lambda((k+1)h) = (2-\frac{\omega_v((k+1)h)}{\omega_v(kh)})\lambda(kh)-\frac{\omega_w((k+1)h)-\omega_w(kh)}{\omega_v(kh)}+\frac{\omega_v((k+1)h)}{\omega_v(kh)}-1 \quad (5.8)$$

where

$$\begin{aligned}
\omega_v((k+1)h) &= (1-\frac{B_v}{M_v}h)\omega_v(kh) - \frac{n_w}{M_v R_w}hF_t(kh) - \frac{F_\theta}{M_v R_w}h \\
&= (1-a_2 h)\omega_v(kh) - a_1 h\mu(\lambda(kh)) - a_0 h \qquad (5.9) \\
\omega_w((k+1)h) &= (1-\frac{B_w}{J_w}h)\omega_w(kh) - \frac{1}{J_w}hT_b(kh) + \frac{1}{J_w}hT_t(kh) \\
&= (1-b_2 h)\omega_w(kh) - b_1 hT_b(kh) + b_0 h\mu(\lambda(kh)) \qquad (5.10)
\end{aligned}$$

where h is the sampling period, the control is the braking torque, T_b, the measurable states of the system are $\omega_v((k+1)h)$ and $\omega_w((k+1)h)$, and the overall system response is the slip, $\lambda((k+1)h)$. It is this quantity which will be controlled by the neural network based anti-skid brake system.

5.3 Existing Approaches

In recent years a number of manufacturers such as Bosch, Honda, Toyota, Borg-Warner, etc., have developed and produced anti-skid brake systems for commercial vehicles. For reasons of commercial sensitivity, intricate details of such systems are not readily available in the published literature, although the papers by Leiber and Czinczel [124], and more recently, Sigl and Czinczel [212] provide a good general overview of the development of ABS and ASR systems at Robert Bosch GmbH. From these and other publications, it is known that anti-skid brake systems generally consist of sensors, a control unit and brake pressure modulators (usually solenoid valves). Typically the sensors monitor the angular speed of the wheel and then produce a continuous electrical signal proportional to this value. The sensor signal is then transmitted to the control unit. Based on the signal it receives from the sensor, the control unit calculates an estimate of the value of the slip. Using an experimentally derived look-up table, it then determines the necessary action to be implemented for the current value of slip. Usually the control action would be either to apply brake pressure, hold the current brake pressure or release the existing brake pressure. The brake pressure modulator receives the command signal from the control unit through an electronic switch and it then adjusts the wheel cylinder brake pressure accordingly. The design, tuning and calibration of commercial anti-skid brake systems are based on experimental trials rather than using analytical results. Therefore commercial anti-skid brake systems are not adaptive and issues such as robustness are not addressed. As a result, the performance of such systems may degrade if harsh or changing road conditions are encountered.

A number of analytical approaches to the design of anti-skid brake systems have recently been reported. In [53], Fling and Fenton employ a describing function approach. This involves using a feedback compensator to provide anti-skid behaviour and determining the design parameters for the nonlinear compensator scheme by a describing function analysis. The aim of the compensation is to ensure that when the slip exceeds the value for which the adhesion is at its maximum, wheel lock-up is prevented and the operation is directed towards a steady state condition. In this condition a stable small amplitude limit cycle is established around the peak of the $\mu - \lambda$ curve. The scheme is successfully tested on a 1969 Plymouth test vehicle.

In the paper by Zellner [255], a frequency response approach to the design of a feedback control for an ABS is adopted. Firstly, this entails deriving a mathematical model for the vehicle–brake system. A mathematical model of the brake pressure modulator is then derived. The frequency response of the overall system is then analysed from which an appropriate feedback control structure is derived. The frequency response criteria for the controller/modulator/vehicle dynamics are that it should resemble an integrator in the 2–6 Hz region and have sufficient open-loop phase margin to allow a cross over frequency in this range. These requirements are chosen so that a stable limit cycle with a frequency of at least 2 Hz could be achieved. The desired frequency range also represents the required bandwidth for the closed-loop system. As practical anti-lock modulators are found to look like integrators in the 2–6 Hz region, it is concluded that the desired controller should be a pure gain system with constant phase lag over this frequency range. Several feedback controllers, each with a different feedback variable (wheel speed, slip, wheel angular acceleration, and wheel angular jerk), are simulated and tested. The results show that a wheel angular acceleration feedback controller meets the desired requirements most successfully.

According to Yeh *et al.* [253], the inherent nonlinearity of the ABS dynamics means that only part of the picture can be seen via the frequency domain approaches of Fling and Fenton [53] and Zellner [255]. Additionally, the feedback control approach presented in [255] is very much dependent on the the operating conditions and is therefore not adaptive to changes in road surface or other parameter variations. In the approach presented in [253], a piecewise linear model of the tyre force is implemented to simplify the nonlinearity of the ABS dynamics. A number of assumptions about the system dynamics are made to obtain a simplified second order dynamic system with the braking torque and slip as state variables. The dynamics of the ABS are then depicted as state trajectories in the $T_b - \lambda$ phase plane. The conjugate boundary method is used to produce boundaries for the state trajectories known as the conjugate prediction boundary and the reselection boundary. These are used to determine the stability of a particular limit cycle and the global tendency of the slip dynamics. This analysis is then used to determine the appropriate prediction condition and reselection condition[4] for the ABS control law.

A discrete-time vehicle traction control is presented in the paper by Tan and Tomizuka [230]. Vehicle traction control is composed of both anti-skid braking and anti-spin acceleration. Its aim is to maximise tyre traction by preventing wheel lock-up during braking and wheel spin during acceleration. The basic idea behind the approach presented in [230] is to group the time-varying uncertainties of the vehicle–brake system together, discretise the continuous-time system, locally linearise the resultant discrete-time system and then derive a linear feedback scheme based on the principles of sliding mode control. The scheme is successfully tested for the ABS operation on a test cell and an actual vehicle. In the paper by Tan and Chen an optimal control approach to traction control is provided for the continuous-time vehicle–brake system. The optimal solution results in a "bang-bang" control to achieve regulation of the wheel slip at its corresponding

[4]The prediction condition determines when the brake pressure should be released to prevent lock-up while the reselection condition determines when the brake pressure should be applied again once the chance of lock-up is averted.

peak adhesion coefficient value in the $\mu - \lambda$ curve ($\lambda_{\mu_{\max}}$). The optimal control solution is shown to be equivalent to the zeroth order sliding mode control. Using this result, sufficient conditions are then developed via Lyapunov stability theory to design variable structure control laws for the vehicle traction control system.

5.4 Neural Network Based Anti-skid Brake System

5.4.1 System Description

The primary objective of the neural network based anti-skid brake system (NN-ABS) presented in this chapter is to regulate the wheel slip to the prespecified value of 0.15. From the adhesion coefficient versus wheel slip characteristic presented in Figure 5.2, it can be seen that this value corresponds to a level of wheel slip slightly less than the value for which the adhesion coefficient is maximum on most surfaces. Generally a slightly conservative value of slip, such as the one chosen, helps to ensure stable operation. In the proposed NN-ABS it is assumed that knowledge of the particular type of road surface is not available. Therefore, the target value of slip of 0.15 is used regardless of the road surface. As explained earlier, this will not produce the optimum braking performance on loose gravel where a minimum stopping distance is achieved when the wheels lock up ($\lambda = 1$). For the desired slip to be chosen based upon the road surface encountered, an estimation scheme which identifies the road surface conditions would be needed. In [230], it is suggested that this can be achieved by using a weighted least-squares estimation scheme in conjunction with a wheel slip traction control algorithm. However, for the purpose of this study a constant desired slip is used.

A block diagram of the neural network based anti-skid brake system is shown in Figure 5.3.

The inputs to the neural network adaptive controller are the wheel slip (λ), vehicle speed (ω_v), wheel speed (ω_w) and reference input (λ_r). The output from the NN-ABS controller is the braking torque (T_b). It is assumed that if the system is implemented on an actual vehicle, wheel sensors would be available to provide measurement of the angular wheel

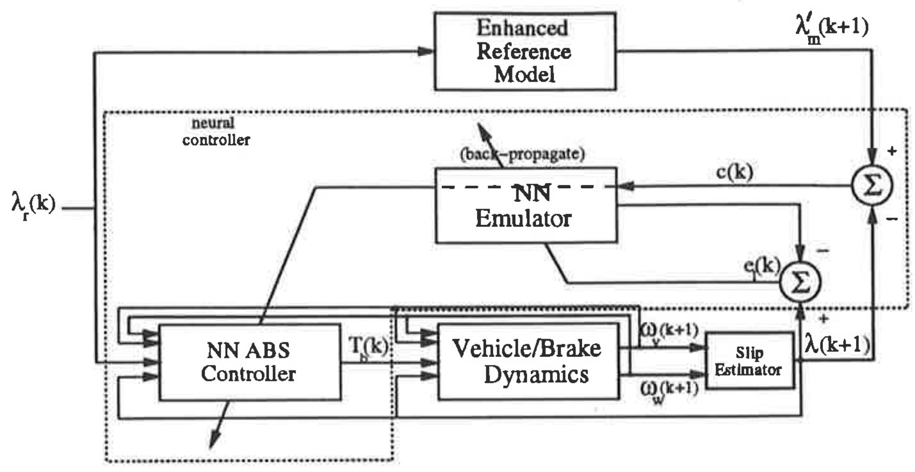


Figure 5.3: Block diagram of the neural network anti-skid brake system

speed (ω_w) and either an estimation scheme or a sensor such as the Doppler ground radar would provide a value of linear vehicle speed (V_v) from which the equivalent angular speed ($\omega_v = \frac{V_v}{R_w}$) could be obtained. Therefore, in the NN-ABS scheme, it is assumed that the measurable system parameters are the equivalent angular vehicle speed and the angular wheel speed. The value of slip is estimated via equation (5.8). The value of braking friction coefficient (μ) used in the vehicle–brake model, equations (5.9) and (5.10), is calculated from the wheel slip using the $\mu - \lambda$ characteristic provided in Figure 5.2. The wheel slip is assumed to be the overall output response. Therefore the neural network emulator models the entire system dynamics, i.e., the vehicle–brake dynamics and the wheel slip dynamics. However, as described in earlier chapters, this is only necessary so that an estimate of the plant Jacobian $\frac{\partial \lambda}{\partial T_b}$ can be obtained. Therefore, neural network based anti-skid brake system can be described by the following equations:

The overall system output (wheel slip) is governed by the nonlinear difference equation

$$\lambda((k+1)h) = f(\omega_v(kh), \omega_w(kh), T_b(kh), \lambda(kh)) \tag{5.11}$$

where $f(.)$ is an arbitrary nonlinear function representing the relationships defined by equations (5.8), (5.9) and (5.10).

The control strategy is to find a suitable braking torque

$$T_b(kh) = g(\omega_v(kh), \omega_w(kh), \lambda(kh), \lambda_r(kh); W_c) \tag{5.12}$$

where g is a neural network parameterised by the set of weights W_c, such that for a stable enhanced reference model governed by

$$\lambda_m((k+1)h) = f_m(\lambda_m(kh), \lambda_r(kh)) \tag{5.13}$$

and

$$\lambda'_m((k+1)h) = \lambda_m((k+1)h) - \beta(\lambda_m(kh) - \lambda(kh)) \tag{5.14}$$

the tracking error satisfies the following relationship

$$|e_T((k+1)h)|_1 = |\lambda_m((k+1)h) - \lambda((k+1)h)|_1 \leq \varepsilon_T \tag{5.15}$$

where $\varepsilon_T \geq 0$ is a predefined tolerance, $f_m(.)$ is a stable linear function, $\beta < 1$ is a stability constant, g is a neural network parameterised by the weights W_c, k is the time index, λ_m is the desired level of slip, λ'_m is the enhanced slip and $|.|_1$ is the vector norm.

As shown in the previous chapter, this can be achieved by minimising the cost function given by

$$E = \frac{1}{2}\sum_{k=0}^{N-1}(\lambda'_m((k+1)h) - \lambda((k+1)h))^2 \tag{5.16}$$

which gives rise to the following controller weight update equation

$$W_c((k+1)h) = W_c((k+1)h) + \eta e_c(kh)\frac{\partial T_b(kh)}{\partial W_c(kh)} \tag{5.17}$$

where the controller error is given by

$$e_c(kh) = (1-\beta)(\lambda'_m((k+1)h) - \lambda((k+1)h))\frac{\partial\hat{\lambda}((k+1)h)}{\partial T_b(kh)} \tag{5.18}$$

and $\frac{\partial\hat{\lambda}((k+1)h)}{\partial T_b(kh)}$ is an approximate of the plant Jacobian obtained by backpropagating through the neural network emulator which is defined by the following equation

$$\hat{\lambda}((k+1)h) = \hat{f}(\omega_v(kh), \omega_w(kh), T_b(kh), \lambda(kh); W_I) \tag{5.19}$$

where $\hat{f}$ is a neural network parameterised by the weights W_I.

From a practical viewpoint, the control unit for the above scheme would implement the neural network controller, neural network emulator, enhanced reference model and slip estimation scheme. It would receive the speed measurements from the sensors (ω_w, ω_v) and calculate the corresponding value of slip and desired value of slip using equations

(5.8) and (5.13) and (5.14), respectively. The control unit would then adjust the weights of the adaptive controller based on the values of the inputs, tracking error and the approximate Jacobian using equations (5.17) and (5.18) to produce the required braking torque T_b. From an implementation perspective, the NN-ABS control unit would also need to account for the brake actuator dynamics. Typically a brake actuator can be modelled as a nonlinear relationship between the braking torque and the solenoid current used to drive the brake actuator. The nonlinear relationship is a hysteresis characteristic between the solenoid current and the brake pressure. The brake pressure is proportional to the braking torque. The following relationship can be derived from Ohno *et al.* [175]:

$$T_b = K\mu_{bp}P(I_s) \tag{5.20}$$

where I_s is the solenoid current, P is the brake cylinder pressure which is a nonlinear (hysteresis) function of the solenoid current, μ_{bp} is the brake pad friction coefficient and K is a constant. Tan and Tomizuka [230] approximate the brake actuator by a slow first order system with a gain whose value depends on the states of the solenoid valves and the brake cylinder pressure. Therefore, the control unit in a practical NN-ABS would either transform the braking torque output to an appropriate value of solenoid current using a brake actuator model such as those described above or the NN controller would be trained such that its output is the appropriate level of solenoid current. This value would then be sent to the brake actuator to achieve the desired braking action. The primary difference between these two approaches is that in the latter case, the backpropagated error from the neural network emulator can be used to modify the controller weights directly, whereas in the first case, the error would need to be transformed by the actuator model as the output of the controller, for this case, is actually the braking torque. In the simulation study performed here, the above actuator models are not employed because of the unavailability of information on the typical values of the constants in the actuator models, in particular brake pad friction coefficient μ_{bp} and the constant K. However, the results published in [89] and [175] demonstrate that neural network control schemes are capable of effectively dealing with systems consisting of hysteresis nonlinearities. Therefore, incorporating the actuator dynamics into the system should not pose any problems for the NN-ABS scheme.

5.4.2 Simulation Results

In the simulation study, neural networks consisting of 4 inputs, 2 hidden layers with 25 and 20 nodes, respectively, and 1 output are used. A learning rate of $\eta = 0.02$ is used for both neural networks. On average, these values provided the best results for all of the scenarios considered. Learning is done completely on-line, so that once the ABS is activated, the neural networks adapts to the system requirements. The reference model used is given by the following difference equation

$$\lambda_m((k+1)h) = (1-20h)\lambda_m(kh) + 20h\lambda_r(kh) \tag{5.21}$$

As can be seen, the reference model is a linear system with unity gain, so that the desired level of wheel slip is represented by the reference input $\lambda_r = 0.15$. The reference model is chosen to have a quick response time, thus ensuring that the reference slip value rapidly reached the desired slip. The enhanced reference output is chosen to be

$$\lambda'_m((k+1)h) = \lambda_m((k+1)h) - 0.1(\lambda_m(kh) - \lambda(kh)) \tag{5.22}$$

Fast sampling with h = 1ms is chosen here because generally in an ABS the changes in the system take place very rapidly, i.e., in the order of one second. Assuming that the NN-ABS on a current generation floating-point DSP chip such as the TMS320C40, this sampling period is realistic. This can be verified by an approximate operation count for the neural network based anti-skid brake system described above. By far the most computationally intensive part of the system is the modification of the network weights. This is achieved by the backpropagation algorithm which consists of three parts: a feedforward stage, an error feedback stage and a weight modification stage. For the neural network structure described above, the NN-ABS would require approximately 9200 operations (addition, multiplication, division and assignment). Assuming a conservative figure of 2 cycles per instruction, where the instruction cycle time for the TMS320C40 is 40 ns, the proposed NN-ABS would result in a computation time of approximately 0.74 ms. Furthermore, this analysis does not take into account an improvement in the efficiency of the algorithm which could be obtained by exploiting the parallel nature of the neural networks.

Effects of Different Road Surfaces

The first simulation scenario considered is the performance of the neural network based anti-skid brake system for dry asphalt, wet asphalt, loose gravel and icy road surfaces. The vehicle is travelling on a level surface ($\theta = 0$) and transitions between the road surfaces are not considered. The ABS is initiated when the vehicle speed is 60 km/h and the wheel slip is 0.5. This is a typical value of wheel slip at which ABS is initiated, because the wheels are tending towards locking-up and, on most surfaces, this value corresponds to the unstable (negative slope) region on the $\mu - \lambda$ characteristic. The resultant wheel and vehicle speed, wheel slip and braking torque responses are provided for the above four surfaces in Figures 5.4a, 5.4b, 5.4c and 5.4d, respectively.

The dry asphalt, wet asphalt and loose gravel cases all display similar results. For these three road surfaces, there is virtually no discernible difference between the actual value of slip and the desired slip response. Furthermore, the braking torque is initially quite small, thus allowing the angular wheel speed to increase to the level required to produce a slip of 0.15. This corresponds to the 0–0.2 second portion of the response. It can also be observed that during this period, the responses for the loose gravel case are far "smoother" than the dry asphalt and wet asphalt cases. This can be attributed to the fact that in the $\lambda = 0.15 - 0.5$ region of the $\mu - \lambda$ curves (Figure 5.2), the asphalt characteristics consist of a number of piecewise linear regions of different slopes, whereas the loose gravel characteristic consists of only one value of slope. Therefore, the progression of the slip from its initial value of 0.5 to the desired level of 0.15 is "smoother" for loose gravel than for the other surfaces. As a result, the speed responses and torque response are also "smoother" for loose gravel. Therefore this phenomenon is a result of the approximation of the $\mu - \lambda$ characteristic used rather than the NN-ABS scheme.

Once the wheel speed reaches the value for which the wheel slip is 0.15, the braking torque increases linearly which results in a linear deceleration of the vehicle. The reason for the linearly increasing braking torque can be explained by considering equations (5.9) and (5.10). For a constant value of slip and a level road ($\theta = 0$), the frictional force F_t and its corresponding torque T_t are constant and the force F_θ which is applied to the car as a result of an incline in the road is zero. By taking this into account and after

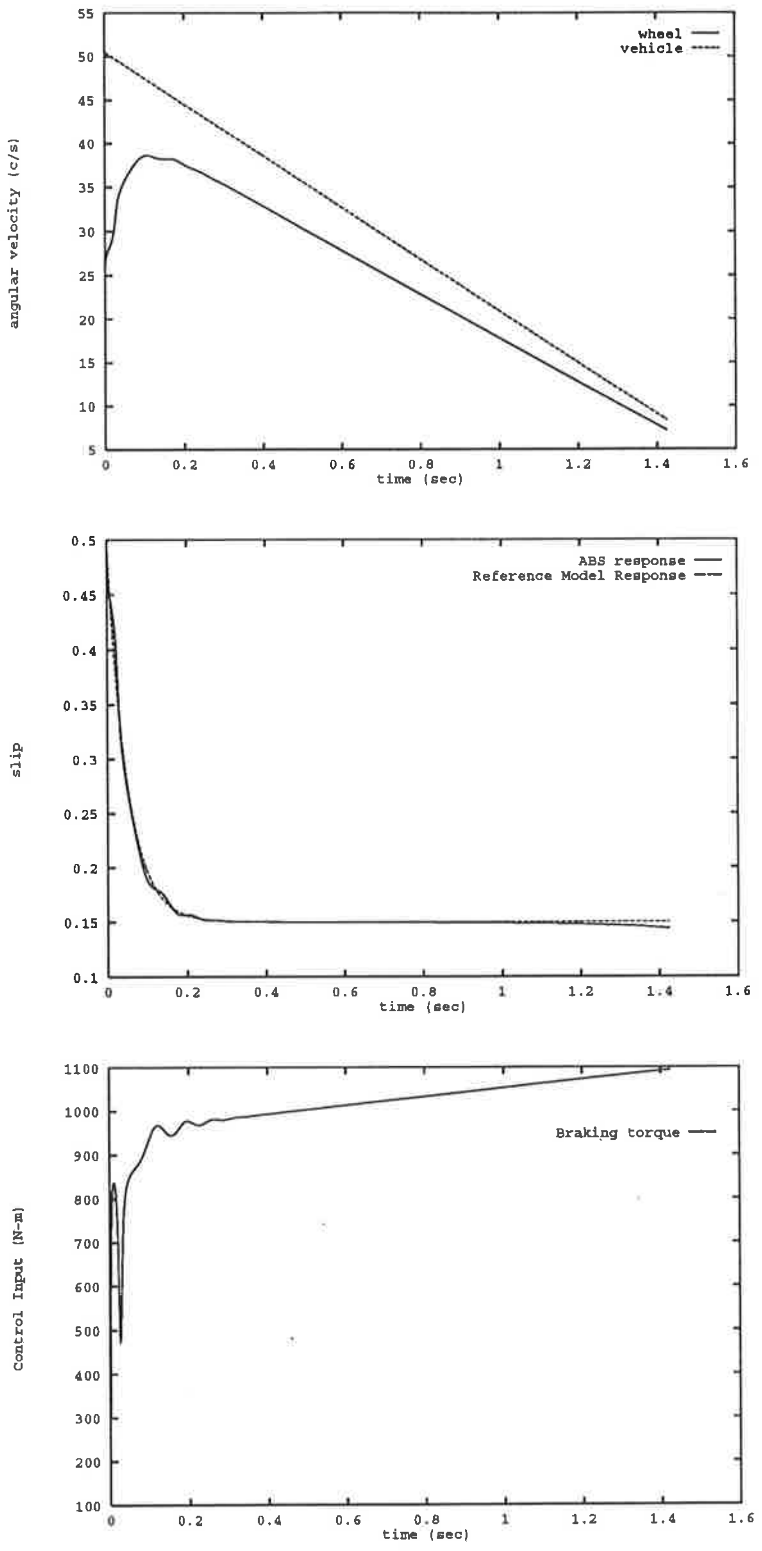
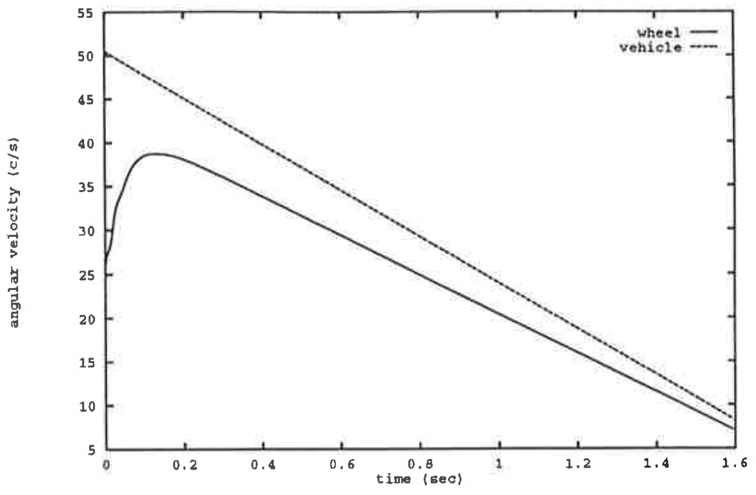
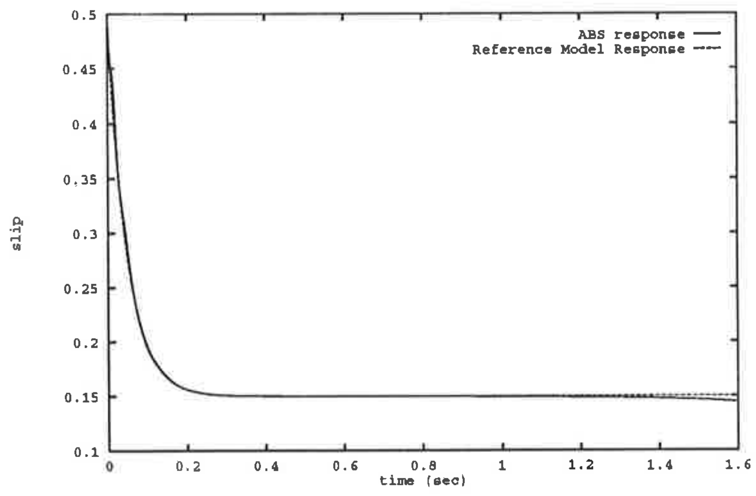


Figure 5.4a: Angular velocity, wheel slip and braking torque responses for a dry asphalt road surface





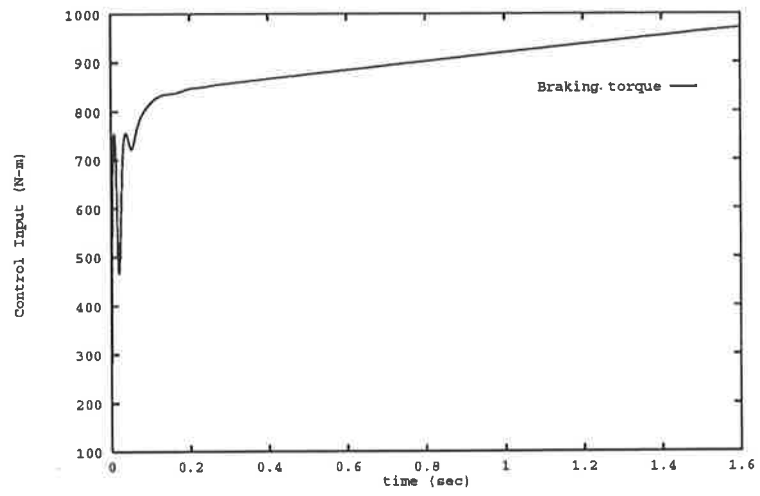


Figure 5.4b: Angular velocity, wheel slip and braking torque responses for a wet asphalt road surface

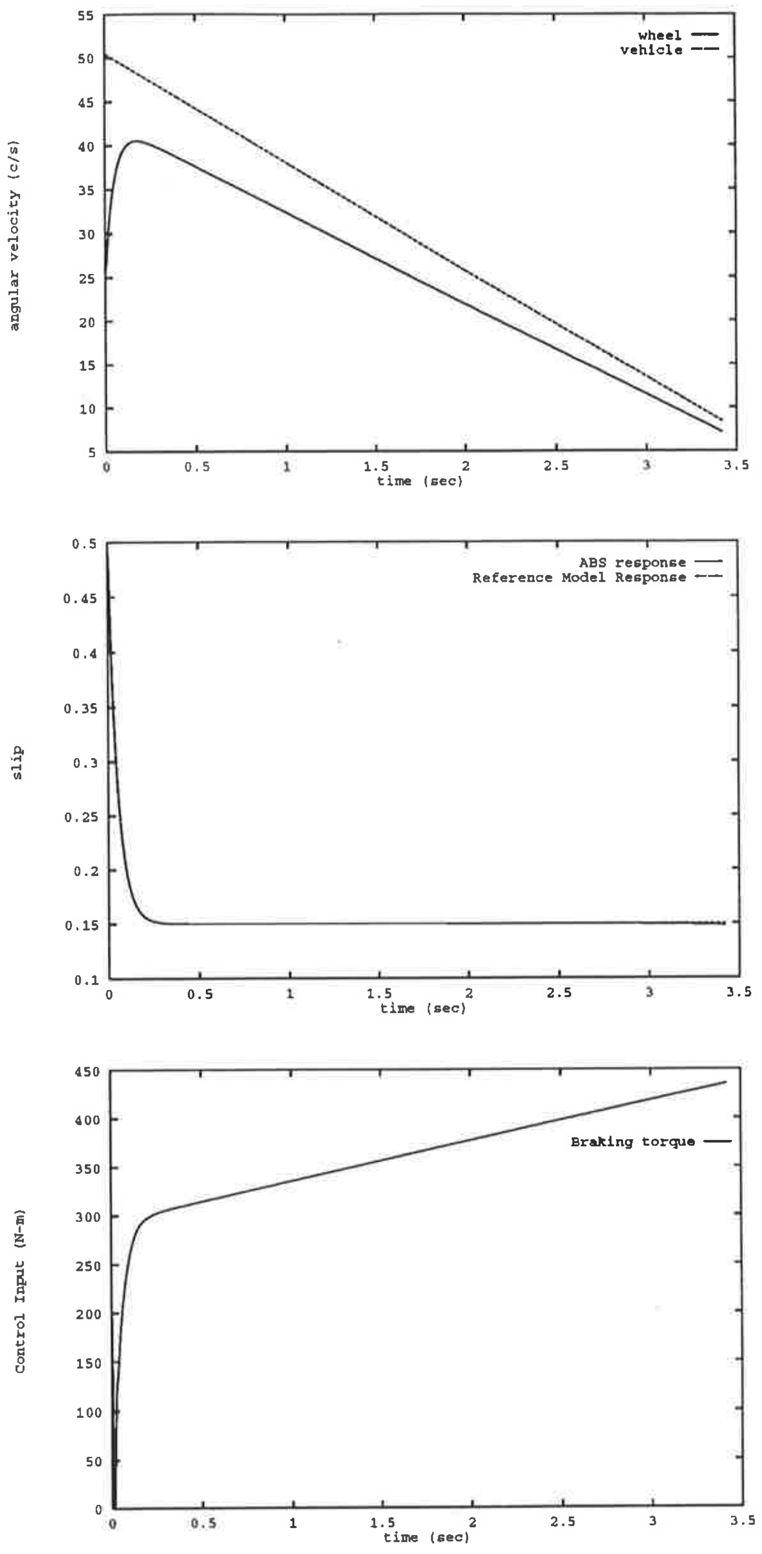


Figure 5.4c: Angular velocity, wheel slip and braking torque responses for a loose gravel road surface

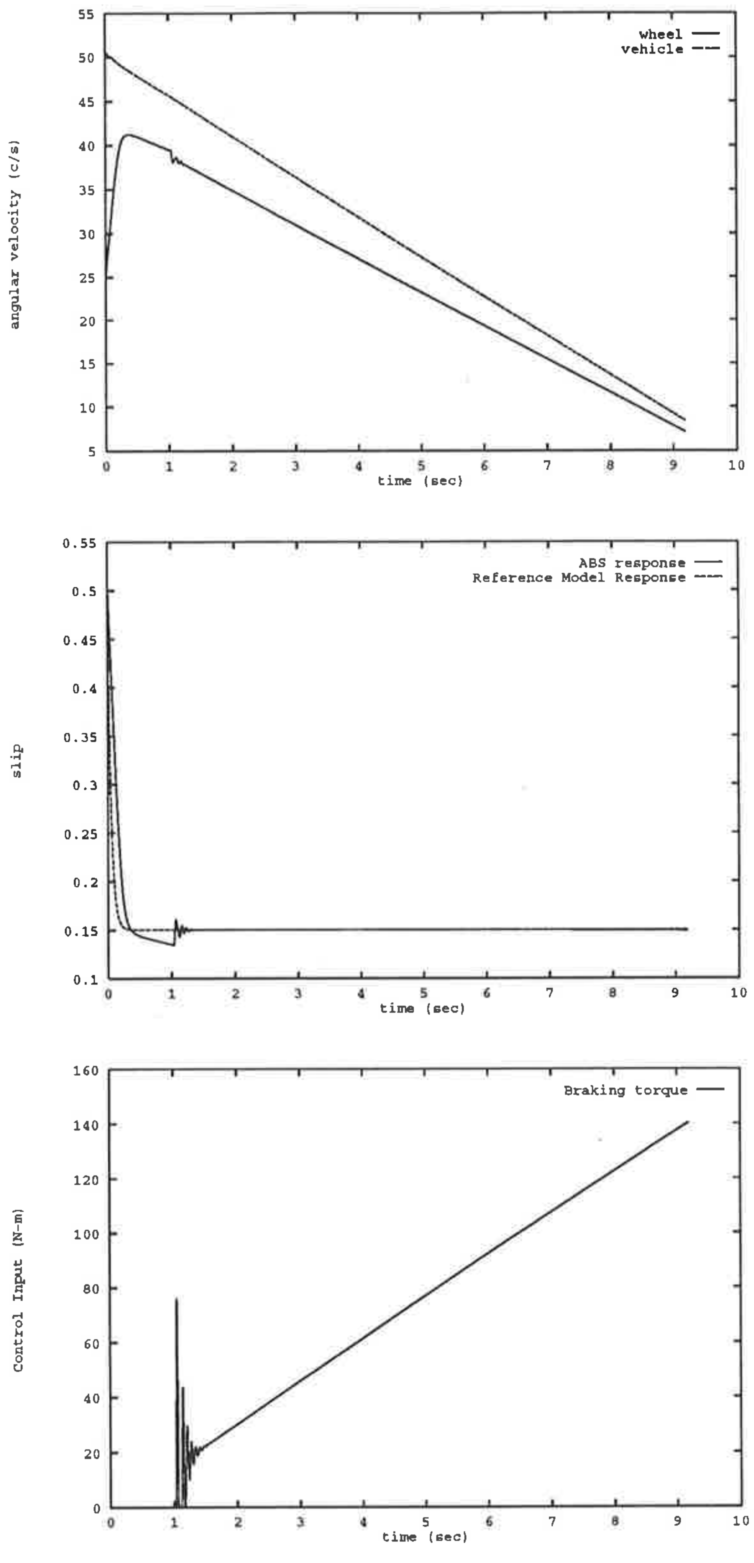


Figure 5.4d: Angular velocity, wheel slip and braking torque responses for an icy road surface

introducing the values for the system parameters, the wheel and vehicle speed dynamic equations can be written as:

$$\begin{aligned}
\omega_v((k+1)h) &= (1-\frac{B_v}{M_v}h)\omega_v(kh) - \frac{n_w}{M_v R_w}hF_t(kh) - \frac{F_\theta}{M_v R_w}h \\
&= (1-0.0044h)\omega_v(kh) - 29.73h\mu(\lambda) \\
\omega_v(10^{-3}(k+1)) &\approx \omega_v(10^{-3}k) - 0.2973\mu(\lambda) \qquad (5.23)
\end{aligned}$$

and

$$\begin{aligned}
\omega_w((k+1)h) &= (1-\frac{B_w}{J_w}h)\omega_w(kh) - \frac{1}{J_w}hT_b(kh) + \frac{1}{J_w}hT_t(kh) \\
&= (1-3.54h)\omega_w(kh) - 0.885hT_b(kh) + 979.785h\mu(\lambda) \\
\omega_w(10^{-3}(k+1)) &\approx \omega_w(10^{-3}k) - 8.85\times 10^{-3}T_b(10^{-3}k) + 0.980\mu(\lambda) \qquad (5.24)
\end{aligned}$$

It can be seen from the equation (5.23) that most of the vehicle deceleration is due to the term related to the tyre friction force ($29.73h\mu(\lambda)$). Furthermore, the vehicle speed term related to the viscous friction of the vehicle, namely $(1-0.0044h)\omega_v(kh)$, is approximately $\omega_v(kh)$. Therefore, if the slip is regulated to a fixed value, for example $\lambda = 0.15$, the vehicle will decelerate linearly. However, to ensure that the slip remains at this fixed level, the wheel must also decelerate linearly by an appropriate amount. It can be seen from the wheel speed equation (5.24) that the term related to the torque produced by wheel slip ($979.785h\mu(\lambda)$) is a constant which works against the other terms. Furthermore, the wheel speed term related to the viscous friction of the wheel, namely $(1-3.54h)\omega_w(kh)$, is approximately $\omega_w(kh)$ for typical values of h. Therefore, the wheel speed is effectively changed by an amount solely related to the braking torque and the constant wheel slip torque term. Hence, the braking torque must increase linearly by a rate which not only counteracts the constant effect of the frictional torque, but also results in a linear wheel speed deceleration which maintains the desired slip.

The responses for an icy surface are provided in Figure 5.4d. The NN-ABS behaviour for an icy road surface is significantly different from its behaviour for the other road surfaces. Although the slip response eventually reaches the value 0.15, and the system output exhibits excellent tracking from then on, its behaviour for $t \leq 0.15$ s is unique to an icy road surface. During this period no control action is undertaken as the system is already acting in an unstable region and even a small braking torque is likely to cause the wheels to lock-up. Once the slip drops below the desired level and thus the system is operating in a stable region, the controller generates the necessary braking torque so that the system slip response tracks the desired slip response. As the frictional effects are much smaller on ice than on other surfaces, the braking torque that can be applied to maintain the slip at 0.15 is smaller. Therefore, the vehicle takes much longer to slow down.

An observation which can be made about all of the responses is that the system does not exhibit the limit cycle behaviour for which many other anti-skid brake systems are designed. As indicated in [122], the likely effects of using a system which is not designed to exhibit this behaviour is unknown and would require further investigation.

The slight deviation of the slip responses away from the desired response in the latter stages of the ABS operation can be attributed to the fact that at low speeds the estimated value of slip becomes less accurate and very sensitive [122]. As a consequence, the slip becomes difficult to control and therefore the simulations are stopped when the vehicle had slowed to 10 km/h. As explained in [53], anti-skid brake systems are often disabled below a threshold speed because of the chance of unwanted braking action being initiated when the noise (and limit cycle) amplitude is large relative to the vehicle speed. Therefore, disabling the ABS at low speeds is a common practice in commercial systems.

The above responses show that performance of the neural network based anti-skid brake system is very good for the surfaces considered. An additional indication of the effectiveness of the approach can be obtained by comparing the stopping distance for the NN-ABS with the stopping distance obtained from a locked-wheel stop. The stopping distance of a vehicle is defined as the reaction distance plus the braking distance, where the reaction distance is the distance travelled by a vehicle while a driver recognises the need to use the brakes and actually starts to physically apply the brakes, and the braking distance is the distance travelled by the vehicle once the brakes have been applied until the vehicle stops. For an alert driver, the reaction distance is on average 12.5 m (3/4 second) at 60 km/h and it increases to 21 m at 100 km/h. The reaction distance is independent of the type of braking action employed and therefore in the following analysis only the braking distances are considered.

The braking distances for the NN-ABS and wheel-lock stop for a vehicle initially travelling at 60 km/h and 100 km/h are provided in Tables 5.2 and 5.3, respectively. These two initial speeds are chosen as they correspond to the two most common speed limits in Australia.

road surface	ABS	lock-up	Δ_{lock} %
dry asphalt	14.24	17.60	-19.1
wet asphalt	15.94	18.33	-13.0
loose gravel	34.20	30.06	+11.2
ice	91.83	134.67	-31.8

Table 5.2: Braking distances (metres) for the neural network anti-skid brake system versus a locked-wheel stop for an initial vehicle speed of 60 km/h

road surface	ABS	lock-up	Δ_{lock} %
dry asphalt	39.44	48.74	-19.1
wet asphalt	44.20	50.76	-13.0
loose gravel	94.49	83.06	+13.8
ice	226.60	357.53	-36.6

Table 5.3: Braking distances (metres) for the neural network anti-skid brake system versus a locked-wheel stop for an initial vehicle speed of 100 km/h

As can be seen from these results the braking distances are reduced appreciably when the anti-skid brake system is employed in all of the cases except for the loose gravel road surface. This is reflected in the values for the percentage decrease in stopping distance for the NN-ABS in comparison with the results for a locked-wheel stop (Δ_{lock}). Both the loose gravel road surface and the icy road surface are somewhat unique and these results are analysed in detail below. Notice, however, that the percentage decrease for the wet asphalt surface is significantly smaller than the dry asphalt. This can be attributed to the fact that for a slip of $\lambda = 0.15$, the adhesion coefficient for a wet asphalt road surface ($\mu = 0.882$) is similar to its adhesion coefficient for a slip of $\lambda = 1.0$ ($\mu = 0.768$). The corresponding values for a dry asphalt road surface are $\mu = 0.99$ and $\mu = 0.8$. Therefore, the magnitude of the frictional force F_t does not change as significantly on a wet asphalt surface as it does on a dry asphalt road surface when the slip is changed from $\lambda = 0.15$ to $\lambda = 1.0$. Consequently, the corresponding stopping distances also reflect this result.

As explained earlier, a minimum stopping distance is achieved for a loose gravel surface when a locked-wheel stop is employed. Therefore the results obtained for this case are as expected. However, in the simplified model considered in this study, it is assumed that the frictional effects of the road surface are the same on all of the wheels. The situation on loose gravel is more complex than this. If the braking system is such that front wheel braking occurs, then the above assumption is valid. The alternative scenario is that the braking system results in rear wheel braking when the front wheels are locked. This occurs with rear wheel ABS. In this situation, the gravel bunches up in front of the front wheels and thus the rear wheels, which track the path of the front wheels, operate on a vastly different road surface [67]. Therefore the design of a commercial anti-skid brake system for gravel surfaces would need to account for these two situations.

Another expected result is that the braking distance for an icy surface is significantly greater than the other road surfaces. However, even for this worst case scenario, the braking distance for the NN-ABS is reduced appreciably in comparison with the results for a locked-wheel stop. In fact the magnitude of Δ_{lock} is greatest for an icy road surface. This can be attributed to the small frictional force and torque components of the system dynamics for icy road surfaces. Therefore the relative effect of the increase in braking friction coefficient achieved by the ABS ($\mu(\lambda = 1.0) \rightarrow \mu(\lambda = 0.15)$) is more significant.

An observation which can be made by comparing the results for the initial vehicle speed of $V_v(0) = 60$ km/h and the initial vehicle speed of $V_v(0) = 100$ km/h is that the braking distance approximately varies with $V_v^2(0)$. As explained earlier, the deceleration of the vehicle is virtually constant and therefore the equation of motion $V_v^2(t) = V_v^2(0) + 2as$ can be approximately applied.

As a means of comparison, the braking distance results of the fuzzy model reference learning control (FMRLC) ABS approach presented by Layne *et al.* [122] for dry asphalt, wet asphalt and icy road surfaces and the loose gravel braking distance results of Satoh and Shiraishi [209] are considered. Due to the differences in the $\mu - \lambda$ characteristics and/or the system parameters of the approaches, the absolute values of braking distances cannot be directly compared. However, they are still provided to give an indication of the order of the braking distances involved. Instead, the percentage change in braking distance of the the ABS approaches with respect to a locked-wheel stop (Δ_{lock}) for the particular system dynamics is used to compare the approaches. For comparison with the results of [122], a desired slip of $\lambda = 0.2$ is chosen and an initial vehicle speed of 56 mph is considered. The braking distance results are provided in Table 5.4.

	FMRLC			NN		
road surface	ABS	lock-up	Δ_{lock} %	ABS	lock-up	Δ_{lock} %
dry asphalt	32.72	38.42	-14.8	31.32	39.11	-19.7
wet asphalt	35.30	39.86	-11.5	34.84	40.78	-14.6
ice	151.07	247.26	-38.9	167.44	290.37	-42.3

Table 5.4: Comparison of the braking distances (metres) for the fuzzy logic ABS approach of Layne *et al.* and the proposed neural network anti-skid brake

The results demonstrate that the NN-ABS approach produces a larger decrease in braking distances than the fuzzy logic approach presented in [122] for the three road surfaces considered. The differences in the braking distances between the two approaches for a locked-wheel stop can be attributed to differences in the $\mu - \lambda$ characteristics. This effect is more pronounced for an icy road surface because the relative influence braking friction has on the system is greater for icy surfaces.

The results provided in [209] are for a vehicle equipped with an anti-skid brake system operating on a gravel road with a desired slip of $\lambda = 0.1$ and an initial speed of 75 km/h. The braking distance results for the two ABS schemes considered in [209](H-ALB and I-ALB) and the NN-ABS approach are provided in Table 5.5.

	ABS	lock-up	Δ_{lock} %
I-ALB	50.3	37.3	+34.9
H-ALB	43.6	37.3	+16.9
NN-ABS	53.3	46.9	+13.8

Table 5.5: Comparison of the braking distances (metres) for the ABS approaches of Satoh and Shiraishi and the proposed neural network anti-skid brake for a gravel road surface

Whilst the absolute braking distances for a wheel-lock stop and an ABS stop are greater using the neural network approach than corresponding values obtained using the approaches presented in [209], the values for the percentage change in braking distance indicate that the NN-ABS produces favourable results in comparison to the other approaches. As mentioned earlier, a locked-wheel stop produces the optimum braking performance on a gravel road surface. Therefore an ABS approach which regulates the slip to a value other than $\lambda = 1.0$ will produce a greater braking distance than the locked-wheel stop. This is the case for the approaches considered in the above analysis. The results of [209] are obtained from tests conducted on an actual vehicle travelling on a gravel road. However, the simulation results for the NN-ABS approach are obtained from a simplified model of the vehicle system dynamics and an approximation of the adhesion characteristic of the gravel road surface. Therefore the differences in absolute values in stopping distances between the approaches presented in [209] and the NN-ABS approach proposed here can most likely be attributed to differences in the adhesion coefficient levels and the approximations made in the system model. However, the simulation results still indicate that the NN-ABS approach offers much promise for the braking of vehicles on gravel surfaces.

Effects of a Road Gradient

The next simulation test investigates the effects of an inclination in the road surface on the performance of the NN-ABS. Road gradients of $\theta = 15°$ and $\theta = -15°$ are considered. These values correspond to very steep roads. The braking distance for these two gradients on the four road surfaces previously considered are given in Tables 5.6 and 5.7. An initial vehicle speed of 60 km/h is considered.

road surface	ABS	lock-up	Δ_{lock} %	$\Delta_{\theta=0°}$
dry asphalt	11.61	13.66	-15.0	-18.5
wet asphalt	12.67	14.09	-10.0	-20.5
loose gravel	21.47	19.83	+8.3	-37.2
ice	34.64	39.33	-11.9	-62.3

Table 5.6: Braking distances (metres) for the neural network anti-skid brake system versus a locked-wheel stop for an initial vehicle speed of 60 km/h and a road inclination of $\theta = +15°$

Firstly considering the case of a positive road gradient, the results show that the braking distance has been reduced considerably in comparison with the corresponding results for a level road (Table 5.2). Intuitively, this result is as expected. This is because when a positive gradient is present, the force $F_\theta(kh) = M_v g sin(\theta(kh))$ acts in the opposite direction to the motion of the car, thus helping to slow the vehicle down. This is despite

road surface	ABS	lock-up	Δ_{lock} %	$\Delta_{\theta=0^\circ}$
dry asphalt	20.19	27.34	-26.2	+41.7
wet asphalt	23.66	29.08	-18.6	+48.4
loose gravel	100.62	72.31	+39.1	+194.2
ice	unable to stop the car			

Table 5.7: Braking distances (metres) for the neural network anti-skid brake system versus a locked-wheel stop for an initial vehicle speed of 60 km/h and a road inclination of $\theta = -15^\circ$

the fact that the tyre friction force $F_t(kh) = \mu(\lambda(kh))N_v(\theta(kh))$ is reduced slightly for a gradient of $\theta = 15^\circ$ (for $\theta = 0^\circ$: $N_v = 3355$ N, $F_\theta = 0$ N whereas for $\theta = 15^\circ$: $N_v = 3241$ N, $F_\theta = 3473$ N).

By examining the equations of motion provided earlier, it can be seen that the force due to the gradient F_θ is constant for a given road incline and is independent of the adhesion properties of the road surface. For an adhesive road surface such as asphalt, the frictional forces are of a similar magnitude to the force F_θ, whereas for less adhesive surfaces such as gravel and, in particular, ice, the frictional forces are comparatively much smaller. Therefore the force produced by the gradient in the road has a more significant effect on vehicle braking for gravel and icy road surfaces. This is reflected in the results provided in Table 5.6, where the percentage decrease in braking distance of the NN-ABS for a road with a gradient of $\theta = 15^\circ$ compared with the corresponding results for a level surface ($\Delta_{theta=0^\circ}$) increases as the frictional effects of the road surface decreases.

For the case of $\theta = -15^\circ$, the values of $\Delta_{\theta=0^\circ}$ indicate that the braking distances are increased significantly compared with the corresponding results for a level surface. The reason for the increased braking distances is that for a negative incline the force F_θ acts in the direction of the motion of the vehicle. Therefore, the force partially counteracts the frictional forces working to slow the vehicle down, thus resulting in a longer stopping distance. For the case of an icy road surface, the force F_θ is sufficiently large compared with the frictional forces, so that the ABS is unable to stop the vehicle. This can be confirmed by considering the equation of motion for the vehicle:

$$\omega_v((k+1)h) = (1 - \frac{B_v}{M_v}h)\omega_v(kh) - \frac{n_w}{M_v R_w}hF_t(kh) - \frac{F_\theta}{M_v R_w}h \qquad (5.25)$$

For $\theta = -15°$, the tyre friction force is $F_t = \frac{M_v}{n_w} gcos(\theta)\mu(\lambda) = 3240.7\mu(\lambda)$ N and the force due to the incline is $F_\theta = M_v gsin(\theta) = 3473.4$ N. Therefore the vehicle equation of motion becomes

$$\omega_v(10^{-3}(k+1)) \approx \omega_v(10^{-3}k) - 0.0287\mu(\lambda) + 0.007694 \tag{5.26}$$

where the sampling period is $h = 1$ ms. For an initial slip of $\lambda = 0.5$, the adhesion coefficient for an icy surface is $\mu = 0.172$, which results in the following equation of motion

$$\omega_v(10^{-3}(k+1)) \approx \omega_v(10^{-3}k) + 0.00277 \tag{5.27}$$

Therefore the vehicle speed increases linearly. In fact, it can be shown that if the road gradient is steeper than $\theta > \arctan(\mu)$ then the vehicle is unable to be stopped. Therefore for an icy surface and an initial adhesion coefficient of $\mu = 0.172$ ($\lambda = 0.5$), the critical road incline is $\theta = -9.75°$.

Variation in Road Surface

It was earlier stated that one of the principle benefits for using neural networks in an anti-skid brake system is their ability to adapt to changes in the environmental conditions without a significant degradation in performance. Therefore in the final set of simulations the ability of the neural network based anti-skid brake system to deal with transitions in the road conditions is investigated. In particular, four road surface transitions which commonly occur under Australian conditions are investigated.

The first simulation study involves the situation where the brakes are applied on a dry asphalt road and during the braking the vehicle moves on to a gravel road surface. The second case involves the situation where the vehicle is initially on wet asphalt and it then moves onto a wet gravel road surface. To take into account the effects of water on the braking friction coefficient of a dry loose gravel road surface, the following equation is used:

$$\mu_{\text{wet gravel}}(\lambda) = \frac{\mu_{\text{wet asphalt}}(\lambda)}{\mu_{\text{dry asphalt}}(\lambda)} \mu_{\text{dry gravel}}(\lambda) \tag{5.28}$$

This equation assumes that the frictional effects of water over all road surfaces is constant for a given level of wheel slip. Harned *et al.* [67] provide a $\mu - \lambda$ characteristic for river bottom gravel. This sort of gravel appears to consist of smooth stones which are well set in the road (river bottom) surface. Therefore, this road surface exhibits significantly different adhesion behaviour from the expected behaviour of a water-covered loose gravel road surface. In particular, the frictional effects of river bottom gravel do not increase with wheel slip as is the case for dry loose gravel. Intuitively, one would not expect that a film of water on a gravel surface would prevent the gravel from bunching up in front of the tyre. Hence, river bottom gravel is not considered in the simulation study and instead the approximation described above is assumed to be more appropriate.

The third simulation scenario involves the transition from wet asphalt to an icy road surface and the final scenario is the reverse of this situation. The responses for the vehicle and wheel speed, wheel slip and braking torque for these four scenarios are provided in Figures 5.5a, 5.5b, 5.5c and 5.5d, respectively. In all of the results, the initial vehicle speed is 60 km/h, the initial level of slip is $\lambda = 0.5$ and level road surfaces are considered.

The results for the first three road transition simulations demonstrate that the NN-ABS approach is capable of dealing with the relatively drastic changes in road conditions encountered in these scenarios with minimal effect on its ability to produce the desired slip response. In fact apart from a short oscillation (< 0.1 s) which occurs when the transition takes place, the slip closely tracks the desired response. In particular, the NN-ABS performs very well for the most drastic road transition (wet asphalt-ice) as well as the most common of these transitions in Australia (dry asphalt - loose gravel). The braking torque results for the first three simulations are comprised of two distinct regions. The first region consists of a braking torque of a relatively large magnitude, while after the transition occurs, the braking torque is much smaller. This can be explained by the fact that in these road transition scenarios, the vehicle moves from a surface with a relatively large coefficient of braking friction to a surface which does not offer much frictional resistance. Therefore, initially a large braking torque is allowed without lock-up occurring, but once the transition takes place, the braking torque must fall in order to maintain the desired level of slip on the reduced friction surfaces. This is further emphasised by the fact that both the vehicle and wheel deceleration reduces once the transition occurs, i.e., N_v and T_t reduce in magnitude once the transition occurs. Therefore, the the torque T_b required to maintain the same slip is smaller and consequently the vehicle deceleration

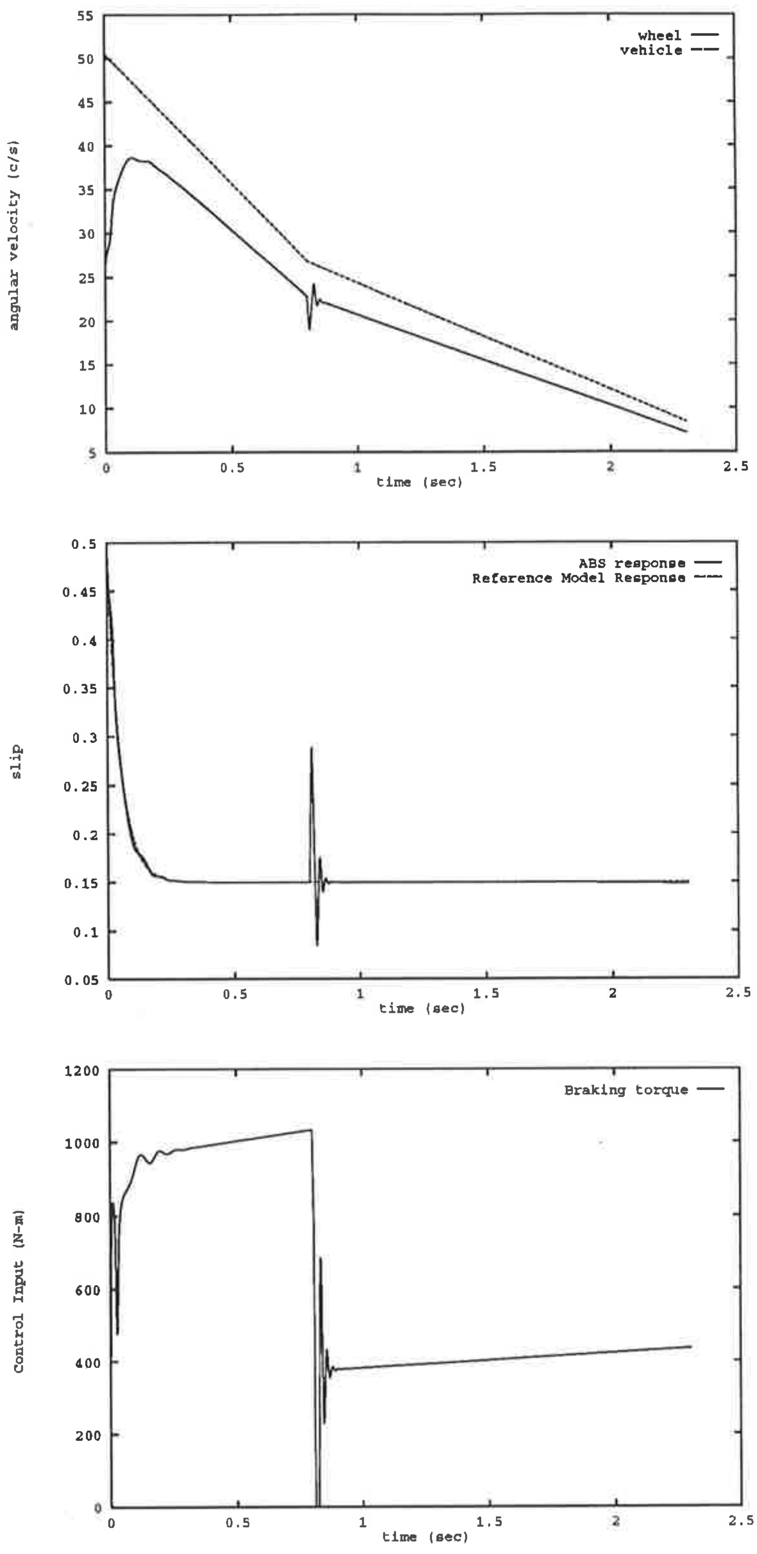


Figure 5.5a: Angular velocity, wheel slip and braking torque responses for a dry asphalt - loose gravel transition

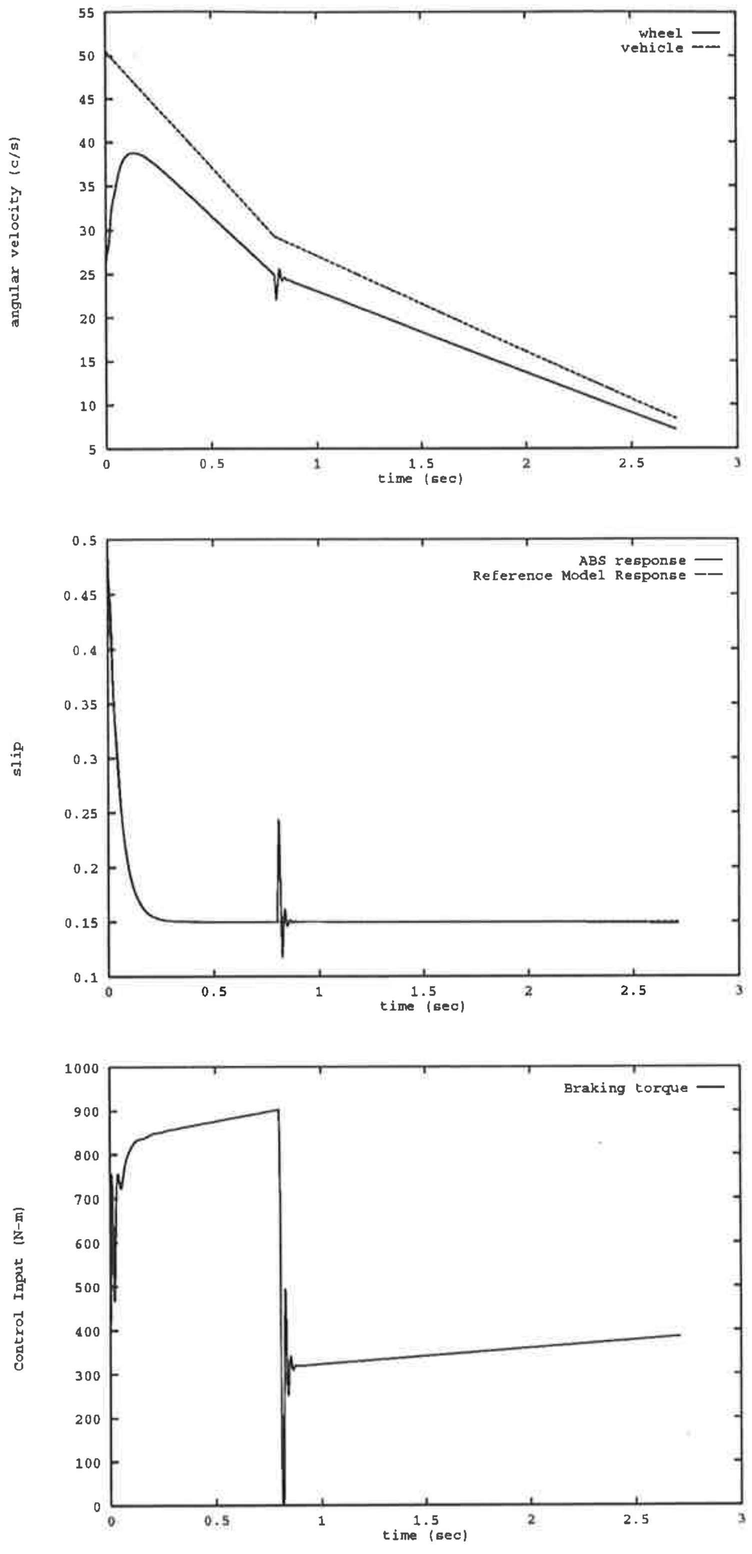


Figure 5.5b: Angular velocity, wheel slip and braking torque responses for a wet asphalt - wet gravel transition

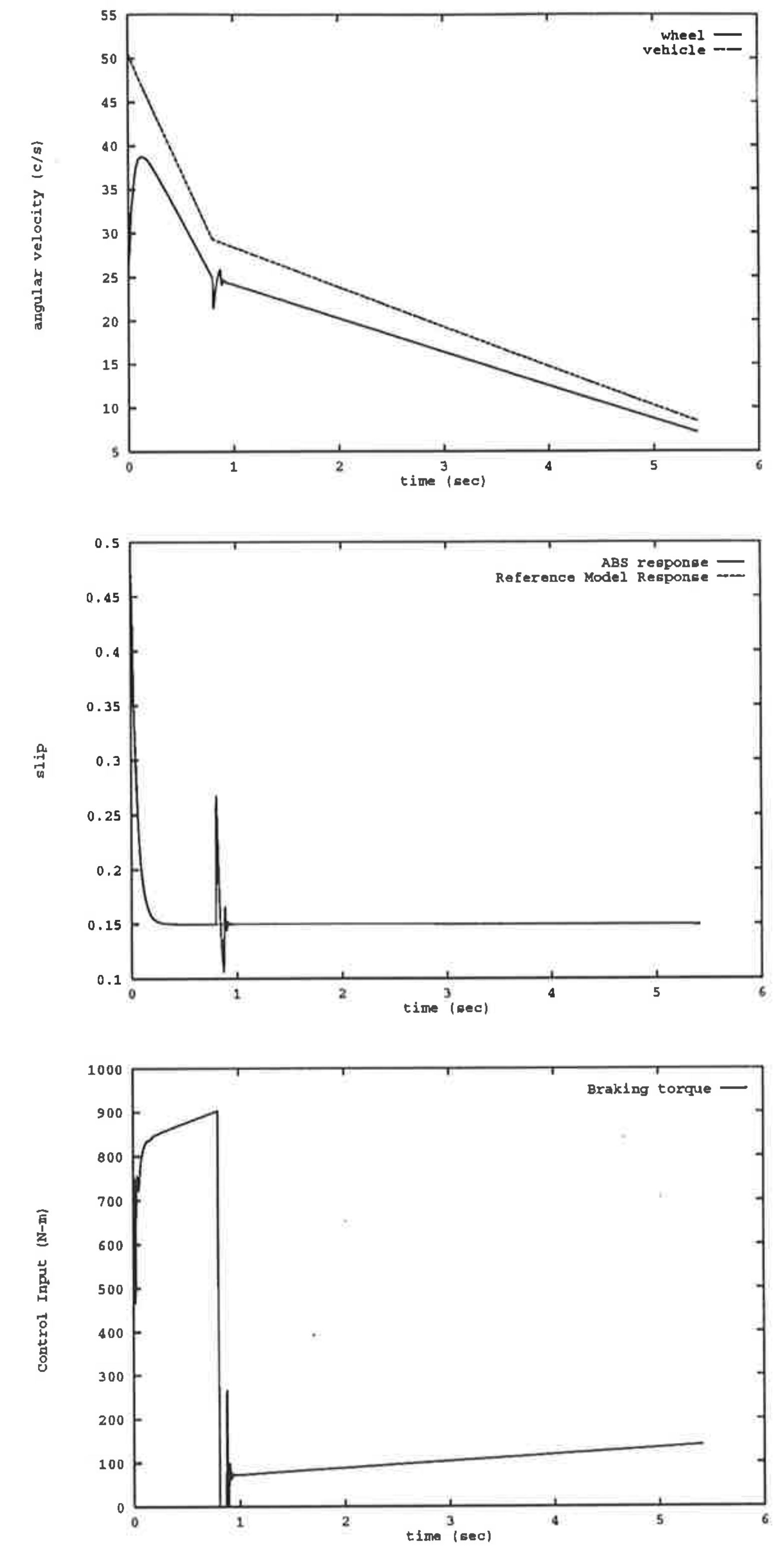


Figure 5.5c: Angular velocity, wheel slip and braking torque responses for a wet asphalt - ice transition

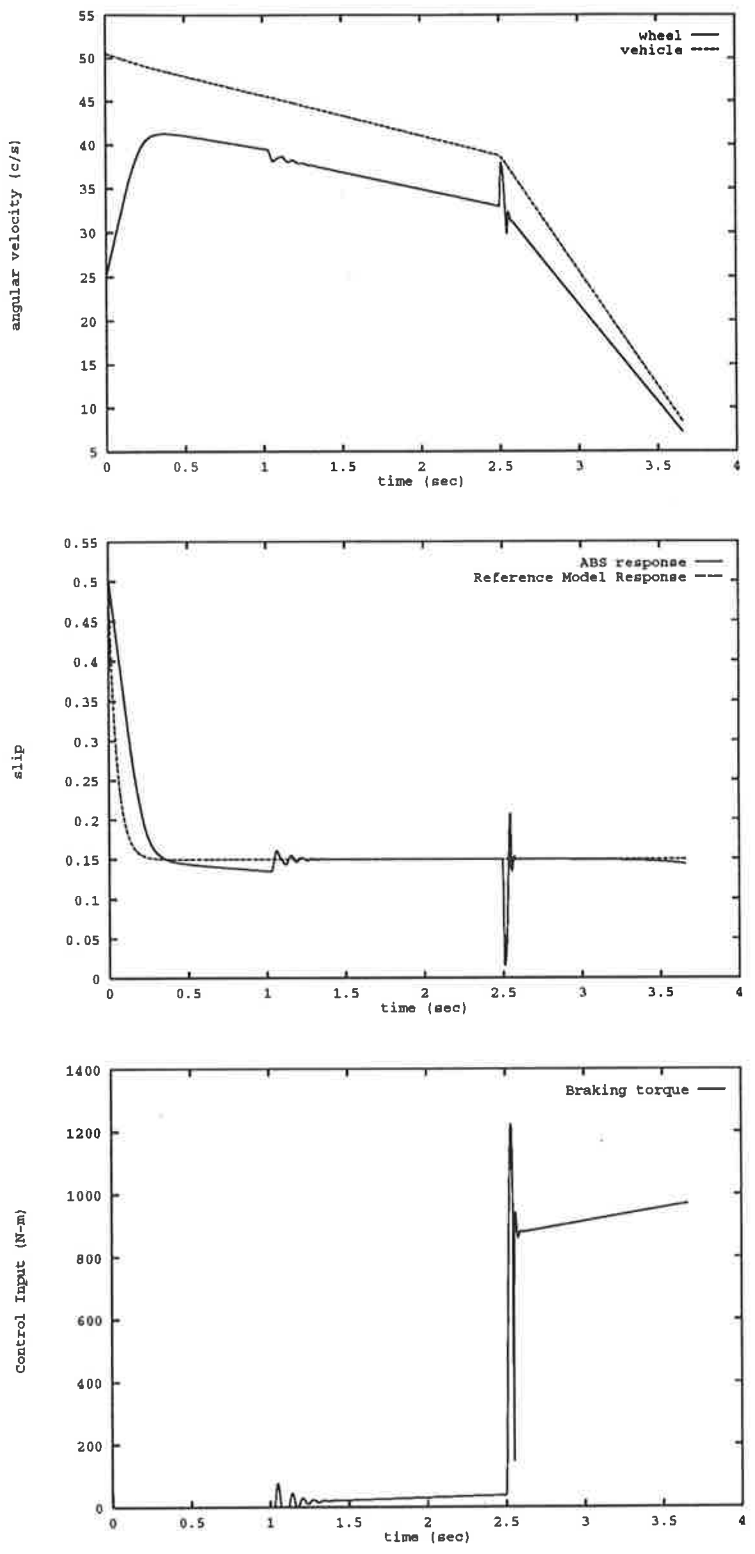


Figure 5.5d: Angular velocity, wheel slip and braking torque responses for a ice - wet asphalt transition

decreases in magnitude. A point worth noting about the results for the wet asphalt-ice transition is that, unlike the situation where only an icy road surface is present (Figure 5.4d), the braking torque is not set to zero for a substantial period of time once an icy surface is encountered. This is because when the transition to an icy surface occurs, the slip is already at a sufficiently low level so that a braking torque can be applied without fear of the wheels locking-up immediately. However as explained above, the braking torque is of a lesser magnitude than its value for the wet asphalt surface.

In the final scenario considered, the brakes are initially applied on an icy road surface, which provides little frictional resistance to the vehicle, before it moves onto the more adhesive wet asphalt road. The results provided in Figure 5.5d demonstrate that, as with the case of an icy surface alone, the NN-ABS initially does not provide any braking torque. Once the slip drops to a level which would not result in an immediate wheel-lock, braking torque of a small magnitude is applied to achieve and maintain a wheel slip of $\lambda = 0.15$. After the road surface transition occurs, the braking torque increases greatly, as do the frictional force F_t and torque T_t which results in a sharp increase in magnitude of the vehicle and wheel deceleration. Consequently the vehicle is rapidly brought to a stop.

As with the results for single road surfaces, the NN-ABS approach does not display the limit cycle behaviour for which other ABS are designed. However, the slip response still deviates slightly from the desired response in the latter stages of braking. As indicated earlier, this is as a result of the slip estimation scheme.

road transitions	ABS	lock-up	Δ_{lock} %
dry asphalt - loose gravel	19.90	22.39	-11.1
wet asphalt - wet gravel	23.49	23.57	-3.4
wet asphalt - ice	41.92	66.35	-36.8
ice - wet asphalt	46.28	51.42	-10.0

Table 5.8: Braking distances (metres) for the neural network anti-skid brake system versus a locked-wheel stop for transitions in the road surface with an initial vehicle speed of 60 km/h

Finally, the braking distances for the NN-ABS scheme and a locked-wheel stop for the road surface transitions considered above are provided in Table 5.8. For all of the road surface transitions considered, the neural network anti-skid brake system produces a significant reduction in braking distance compared with a wheel-lock stop. Hence, these results further emphasise the effectiveness of the proposed approach in dealing with drastic, but commonly occurring, environmental changes whilst still achieving the desired slip performance.

5.5 Conclusions

In this chapter, the proposed neural network adaptive control scheme is applied to the practical problem of an automotive anti-skid brake system. The principle objective of the proposed NN-ABS is to maximise the tyre traction during a braking operation by regulating the vehicle–brake dynamics to a suitable level of slip for any given road surface.

The anti-skid brake system is chosen to demonstrate the practical feasibility of the neural adaptive control scheme as it is a challenging control problem due to the highly nonlinear vehicle–brake dynamics. Furthermore, the dynamics vary significantly with changes in the environmental conditions, in particular the type of road surface. Therefore an anti-skid brake system must be robust to dynamic uncertainties and variations. In the previous chapter, the ability of the proposed neural adaptive control scheme to deal with such disturbances is demonstrated for arbitrary nonlinear systems. Hence, the anti-skid brake system considered in this chapter provides an opportunity to verify the effectiveness of the neural control scheme for a practical problem.

The simulation study undertaken involves firstly deriving a mathematical model for the vehicle–brake system. The dynamic model is derived from Newton's laws by considering the rotational torques applied to the wheel and the total forces applied to the vehicle during braking. The dynamics are shown to be strongly influenced by the frictional effects of the road surface present. These effects are incorporated into the model by way of a nonlinear relationship known as the $\mu - \lambda$ characteristic which relates the braking friction coefficient (μ) to the wheel slip (λ) for any given surface. The $\mu - \lambda$ characteristics are shown to be unique for a particular road surface. The NN-ABS regulates the vehicle–brake system to a level of wheel slip which produces approximately maximum braking friction for most road surfaces. The vehicle dynamics are then discretised so that they could be incorporated into the neural network adaptive control framework provided in the previous chapter.

Several existing anti-skid brake system approaches are studied in detail. One of the main results from this analysis is that many of the current ABS schemes are based on iterative laboratory/field tests and are therefore not adaptive to dynamic variations. Several of the problems associated with the commercial implementation of such schemes are also highlighted and discussed. In particular, the problems associated with obtaining an accurate vehicle speed measurement during braking are explained. Several of the methods used to overcome this problem are also presented.

The neural network based anti-skid brake system is then presented. In this approach, the controller neural network is designed to produce the braking torque required to ensure that the vehicle–brake system produced a wheel slip which tracked the desired slip response characterised by a reference model. A second neural network is used to produce the system Jacobian which is in turn used to modify the controller weights appropriately. The scheme is shown to be effective in regulating the system to the desired slip for four different road surfaces. It is also shown to result in a significant reduction in the stopping distance of the vehicle compared with a locked-wheel stop for all of the road surfaces except loose gravel. However, it is demonstrated that for any brake system, the minimum stopping distance on loose gravel is achieved by locking the wheels. Comparison with other ABS approaches further demonstrates the promising performance of the NN-ABS scheme. The scheme is also demonstrated to be capable of effectively dealing with gradients in the road, except for the case of a very steep negative gradient on an icy road surface. However, this scenario is shown to be impossible to control for any anti-skid brake system. Finally, the ability of the NN-ABS approach to deal with transitions in road surfaces is also demonstrated. Such transitions result in a change of the frictional forces applied to the vehicle and thus produce variations in the dynamics of the vehicle–brake system. The adaptive nature of the NN-ABS means that such changes are able to be effectively dealt with. Hence the the simulation study undertaken shows that the neural network based anti-skid brake system is able to produce effective braking responses for the vehicle even when it is subject to severe environmental conditions.

The obvious extension of the research reported in this chapter would be to test the NN-ABS approach on either a specifically equipped brake test cell or a test vehicle. This would involve issues such as the modelling of actuator dynamics, as well as the dynamics associated with the suspension and steering mechanisms, implementation of the algorithm on a microprocessor, the effects of sensor noise and other disturbances, the choice of sampling rates, etc., being considered. Furthermore, a means of accurately estimating the vehicle speed would also need to be implemented. Several suggestions on how to achieve this are provided in the chapter.

A modification to the NN-ABS scheme would be required to deal with the common practical situation of a split-μ surface. This is because in the simulation study conducted in this chapter, it is assumed that the frictional effects of the road surface are the same for all four wheels. However, this is not the case when the vehicle is travelling on a split-μ surface. In this situation some of the wheels are on a road surface with a low adhesion coefficient, while the remaining wheels are on a road surface with a higher adhesion coefficient. Such surfaces result in a yawing moment which makes the vehicle more difficult to control. Furthermore, with a split-μ surface, the wheels on the less adhesive surface would lock-up before the other wheels. This would require the anti-skid brake system to either independently control each wheel or control a set of wheels (e.g. a separate ABS for the left wheels and right wheels). However, as the NN-ABS system considered in this chapter provides an equal braking response to all four wheels, some modification of the scheme would be required.

Finally, it is shown by the results for a loose gravel road surface that a wheel slip which produces a minimum stopping distance for one surface does not necessarily produce the best results for another surface. Therefore, another practical issue which would need to be considered would be the development of a scheme which could determine the type of road surface present and then evaluate a value of slip to produce maximum braking friction for that surface. This would help to produce an optimum ABS response for all road surfaces. However, the results provided in this chapter indicate that the neural network based anti-skid brake system offers much promise and also demonstrates the practical feasibility of the neural network adaptive control scheme proposed in this thesis.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

The motivation for the research described in this thesis is the potential for neural network based adaptive control schemes to provide a unified approach to the control of a general class of nonlinear system. As explained in the review of existing linear and nonlinear control methodologies provided in Chapter 1, most of the conventional control approaches are restricted by the type of system that they can handle. Neural networks have demonstrated an ability to approximate continuous nonlinear functions to any degree of accuracy. Furthermore, neural networks are inherently adaptive and have an ability to learn. These factors make them ideal for the control of general nonlinear systems in which there are uncertainties regarding the system dynamics and its environment. However, as explained in this thesis and in other literature, the analytical study of neural adaptive control schemes is still a difficult problem and important control system concepts such as stability and convergence remain difficult to prove for all situations. Despite the significant progress made towards these issues in this thesis, the development of theoretical results for neural network control architectures remains an open and vibrant area of research.

In most of the neural network based control procedures presented in the literature, the neural network is generally used to model some part or all of the system dynamics.

This application exploits the principle advantage of neural networks, i.e., their ability to perform functional approximation roles. Several neural network architectures have been proposed for modelling dynamic systems, including modified versions of static networks such as the multilayer perceptron, and dynamic networks such as recurrent neural networks. The neural adaptive control approach proposed in this thesis also uses neural networks to emulate the nonlinear system dynamics as well as to synthesise the control. Therefore, in Chapter 2 a review of some of the most commonly used neural networks for modelling dynamic systems is provided. In particular, attention is paid to the time delay neural network. This network is used in the proposed control scheme, mainly because of its simplicity and its success in applications such as speech processing and the related time-series prediction, as well as in identification and control.

A commonly occurring problem in control systems is the incorrect parameterisation of the system model. In the context of time delay neural networks, this relates to an incorrect choice of the number of delayed values of control and output terms. Whilst in the proposed neural adaptive control scheme it is assumed that correct *a priori* knowledge of the required model parameterisation is available, it is shown in Chapter 2 that techniques exist which can provide information on the validity of a particular model. Therefore, even this knowledge is not crucial to the success of the scheme.

As has been mentioned repeatedly in this thesis, the area of neural control (or neuro-control or connectionist-control) has been extremely active over recent years. As with the field of adaptive control during its formative years, the advances in neural control schemes have been in many directions. By far the most popular approach amongst the control community is neural adaptive control, where neural networks are combined with a traditional control framework. Several neural adaptive control approaches are examined in Chapter 2, not only to provide a survey of this area, but also to highlight some of the advantages and limitations of the existing approaches. Furthermore, this provides the opportunity to compare and contrast the proposed approach with some existing schemes.

A basic neural network based model reference adaptive control scheme is considered in Chapter 3. Model reference adaptive control is chosen as a framework for the approach for a number of reasons. Firstly, it is a control scheme which is very amenable to being implemented in a neural network based scheme. This is because it generates the desired closed-loop response for the system which can then be used by a supervised learning scheme. Secondly, it enables the desired closed-loop dynamics of the system to be incorporated into a structured and easily analysable form. This, in turn, provides the system designer with a means of easily changing the desired dynamics to meet different performance specifications. The approach presented in Chapter 3 combines two existing neural control approaches, namely forward modelling and model reference neural adaptive control. The approach is demonstrated to require very little information about the system in order to provide effective control. It is shown to provide a unified approach to controlling a general class of discrete-time nonlinear system, including those previously found by others to be the least analytically tractable and thus difficult to control. In particular, the approach can deal with nonlinear systems which are non-affine in control and where the control is heavily embedded within the nonlinearities of the system dynamics. This is in contrast to existing geometric control schemes for nonlinear system and many other neural adaptive control schemes. The approach utilises two neural networks; one to emulate the plant dynamics and the other to synthesise the control. It is further shown that the neural network which emulates the plant is only required to provide an approximation of the plant Jacobian, which is then used to update the weights of the controller neural network. Several alternative methods of obtaining an approximation of the Jacobian are discussed, but each one is shown to be deficient in some way.

The scheme is shown to be able to be implemented in both an off-line form and an on-line (truly adaptive) form. The issues associated with these two forms of the scheme are discussed. Simulation examples are used to highlight the effectiveness of the approach for a wide range of nonlinear systems, including a multi-input multi-output plant and a marginally stable plant. The effective performance of the approach in the presence of load disturbances, sensor noise and dynamic plant noise is also demonstrated. However, as stated in Chapter 3, the approach has two major shortcomings. Firstly, the scheme does not include any theoretical guarantees for the convergence of the tracking error. This represents a fundamental deficiency with the basic approach, because for model reference adaptive control schemes the principle aim is to ensure that the tracking error converges to a neighbourhood of zero. A second critical property of model reference control schemes, and for that fact, any control scheme, is the stability of the overall system. This issue is also not addressed in the scheme presented in Chapter 3. Therefore, despite the success of the scheme in simulation studies, it is concluded that these issues would need to be addressed in order for the proposed neural network based model reference adaptive control scheme to be practically feasible.

The main theoretical results of this thesis are provided in Chapter 4. Most importantly, theorems which address the convergence of the tracking error and the stability of the closed-loop system are provided. The proofs for these theorems are also presented in this chapter. In order to derive these theorems, it is first necessary to make an enhancement of the existing model reference neural adaptive control scheme. This enhancement primarily involves the introduction of an enhanced reference model which provides a desired response for the controlled plant. The enhanced reference model has a form which allows sufficient limit conditions to be derived which guarantees the convergence of the tracking error between the output of the original reference model and the output of the controlled plant. An iterative search routine is also incorporated into the scheme to generate a control which meets the limit condition. The proposed enhancement has its origins in the fields of optimal control, in particular an approach known as the optimal decision strategy, and on-line optimisation schemes. The enhancement also gives rise to results which guarantee the stability of the closed-loop system. A modified weight update equation is also derived so that the controller neural network weights are updated appropriately.

The effectiveness of the new neural adaptive control scheme is demonstrated through simulation studies. Comparison of the results with the scheme presented in Chapter 3 demonstrates that significant improvements in the tracking performance are achieved with the new scheme for stable, minimum phase nonlinear systems in a noise-free environment. However, in practice such an ideal environment is rarely encountered. Firstly, all practical systems are subject to disturbances of some form. Typically these can include noise induced in the measuring devices, noise resulting from internal components, noise due to unmodelled high frequency dynamics, changes in the load conditions on the system, or environmental disturbances such as turbulence, waves, gusts of wind, etc. Furthermore, many practical systems are also subject to time-varying parameters or dynamics which may arise because of factors such as aging components, component failure or changing environmental conditions. The robustness of a control scheme to such disturbances and variations is an important factor in determining its practical feasibility. Therefore, simulation studies are conducted to demonstrate the ability of the proposed scheme to effectively deal with disturbances of various forms (DC load disturbance, white sensor noise and coloured plant noise) as well as variations in the plant dynamics. Secondly, marginally stable systems and nonminimum phase systems commonly occur in practice. Both of these types of systems cause problems for many existing control approaches. Some classic examples of marginally stable nonlinear systems are ships and homing missiles. These systems are unable to be controlled via approaches which require the open-loop identification of the system dynamics. Some common examples of nonminimum phase nonlinear systems are the tactical missile and many discretised industrial processes. These systems are unable to be controlled via methods which employ an inversion of the plant dynamics, such as feedback linearisation. Thus the ability of any control scheme to effectively deal with these types of non-ideal, but common, systems augers well for its practical feasibility. Therefore, simulation studies are presented in Chapter 4 to demonstrate the effectiveness of the proposed neural adaptive control scheme in controlling a marginally stable nonlinear system and a nonminimum phase nonlinear system.

Finally in Chapter 5 a simulation study based on the "real world" problem of an automobile anti-skid brake system is undertaken to further demonstrate the practical feasibility of the proposed approach. The anti-skid brake system is a challenging control problem because of the highly nonlinear vehicle–brake dynamics which vary significantly with changes in the environmental conditions, particularly the road surface conditions. Therefore an anti-skid brake system must be able to adapt to dynamic variations. Most of the existing anti-skid brake systems are based on experimentally derived look-up table approaches and are therefore not adaptive to the variations which commonly occur in vehicle systems. The results presented in the previous chapters of this thesis suggest that proposed neural network control is ideal for this problem. The aim of the neural network based anti-skid brake system proposed in Chapter 5 is to regulate the wheel slip to a level which provides near-maximum tyre traction on most road surfaces. However, as the measurable system parameters in most anti-skid brake systems are the wheel speed and vehicle speed (or acceleration) an estimation scheme is typically employed to obtain a value of slip. Therefore, in the proposed scheme the neural network emulator models both the vehicle–brake dynamics and the slip estimation scheme. This is necessary so that the Jacobian of the controlled variable (slip) with respect to the control variable (braking torque) can be obtained. The Jacobian is then used to modify the weights of the neural network controller so that the braking torque necessary to achieve the desired slip is provided. The neural network based anti-skid brake system is demonstrated to provide effective braking performance on four different road surfaces and for the case where a transition in the road surface occurs. Although the system is not implemented in practice, a number issues related to the practical implementation of the scheme are also discussed. These include modelling of actuator dynamics, schemes to accurately measure the vehicle speed and the computational requirements of the proposed scheme.

In summary, this thesis describes the development of a neural network based model reference adaptive control scheme for discrete-time non-affine nonlinear systems. The original contribution of the research is outlined in the body of the thesis. However, the broader aims of the work that have been achieved are:

- A detailed and objective review of nonlinear control systems with a view to understanding the merits and limitations of the existing approaches.
- A thorough review of the issues associated with the use of artificial neural networks for the modelling and control of nonlinear systems.
- The development of a neural network based model reference adaptive control scheme for which several conventional control system properties and concepts, such as stability of the closed-loop system, convergence of the tracking error, and robustness to disturbances, and dynamic variations are addressed.
- The investigation, via extensive simulation studies, of the effectiveness of the proposed approach on systems subject to common practical problems such as marginally stable or nonminimum phase behaviour and disturbances of various forms.

- The application of the approach to a practical problem such as the anti-skid brake system.

The work pertaining to the application of the proposed approach to marginally stable systems is published in reference [140]. The application of the approach to nonminimum phase nonlinear systems is published in reference [141]. The research pertaining to the convergence and stability results and the practical treatment of the robustness of the system to disturbances and plant variations is reported in reference [143]. The work on the application of the method to an anti-skid brake system is reported in references [142, 145]. Finally, research related to stability and convergence issues of the proposed neural adaptive control scheme and its application to defence systems is reported in [144].

6.2 Recommendations for Future Work

The research presented in this thesis and in the above publications has demonstrated that neural network based model reference adaptive control is potentially an extremely viable and effective practical approach to the control of nonlinear systems. However, this conclusion is based on theoretical analyses and simulation studies only. In view of the development of a practical neural adaptive controller, there are some theoretical and practical areas in which further research is suggested.

1. The results for the stability of the overall system and convergence of the tracking error which are provided in Chapter 4 are based on *sufficient* limit conditions. Therefore, a natural extension of this work is the derivation of *necessary and sufficient* conditions for the convergence of the tracking error. However, because of the nonlinear nature of the neural networks used, this is an extremely challenging task. A possible line of investigation would be to incorporate global stability and convergence considerations into the development of a learning algorithm for the neural network structure chosen. Research along these lines has been undertaken in [186, 234, 235] for feedback/input-output linearisation approaches for continuous-time affine (in control) nonlinear systems.

2. It is well known that for nonlinear control systems certain conditions must be satisfied by the system in order to ensure the existence of a solution to the control problem [90, 173, 218]. These conditions are often quite involved and hard to verify. Therefore, several assumptions are made without full justification to ensure that a solution to the problem exists without the need to verify the existence conditions and to assist in the development of the new theoretical results. The most critical of these assumptions are those regarding the controllability and observability of the system. As pointed out in [166], there is no simple way of checking the validity of these assumptions in the nonlinear control context given the present state of nonlinear control theory. Theoretical justifications for assumptions regarding the controllability and stabilisation of neural network control systems are addressed in [126] for state-space models. However, as stated in [126], this problem becomes far more complex when only input-output data is available and the state of the system is not accessible, as is the case in the work presented in this thesis. Therefore, further research can be conducted into the extremely challenging problem of developing theoretical results to justify the assumptions made in the scheme and then gradually relaxing these assumptions. The results achieved in linear adaptive control theory and for feedback linearisable systems may provide some guidance for this issue.

3. In the work presented in this thesis only discrete-time systems are considered. This is primarily because the backpropagation algorithm has traditionally been provided in a discrete format. Recently, there has been some work on the application of neural network techniques to the adaptive control of continuous-time nonlinear systems [31, 186, 204]. There are a number of motivations for this work. Firstly, the discretisation of physical continuous-time nonlinear systems often results in highly complex discrete-time models [173] and may even introduce nonminimum phase behaviour [6]. Secondly, the full potential of neural control schemes will only be realised if they are implemented in hardware. Currently, the dominant trend is towards analog hardware realisations of neural network architectures [147], although recently a hybrid analog-digital approach has been tested [44]. Finally, continuous-time learning rules have recently been developed. Therefore, a further

area of research can be the continuous-time realisation of the proposed neural network based model reference adaptive control. This would primarily entail a re-working of the theoretical results in a continuous-time framework.

4. The neural control scheme proposed in this thesis is shown, via simulation studies, to be robust to variations in the dynamics of the system. However, theoretical results to guarantee the robustness of neural adaptive control schemes still remain elusive. As with stability and convergence results, much insight into robust approaches to neural adaptive control systems can be gained from existing results in the "conventional" control literature. One approach which has shown promise is the use of a sliding control scheme in conjunction with a neural control scheme [204]. In this paper, a sliding mode controller takes over from the neural controller whenever the state of the system moves outside the region on which the neural network approximation is valid. The work presented in [204] is for continuous-time systems. However, the design of sliding controllers for discrete-time systems is a more difficult task than for the continuous-time case [215]. Therefore, the development of discrete-time sliding control strategies for the proposed scheme would be a fruitful and challenging area of research. The scheme proposed in this thesis is also shown to provide effective control in the presence of various forms of disturbances. However, apart from the bias compensation scheme, explicit techniques to minimise the effect of the disturbances on the system are not considered. The topic of disturbance rejection in neural network based identification and control of nonlinear systems is discussed in [160]. In the approach, the identifier and controller structures are modified to compensate for the effect of disturbances on the plant output. The disturbances are assumed to be additive and are generated by an unforced dynamical system. Specific structures of nonlinear systems are considered. Some simplifying (local linearisation) assumptions are provided to deal with Model IV type nonlinear systems. As with the neural adaptive control scheme presented in [169], an explicit control law is used. This control law is based on the structure of the nonlinear system under consideration about which some partial knowledge is assumed. Therefore, further work can be conducted on extending the work provided in [160] to more general nonlinear systems and disturbance models (including

multiplicative noise) and applying the results to the proposed neural adaptive control scheme. Because of the generalised nature of the proposed scheme, this would be a challenging task.

5. In order for any new and novel control scheme to gain acceptance amongst industrial engineers, it must be tested and proven on actual "real world" systems. In this thesis, a simulation study of the application of the neural adaptive control scheme to anti-skid brake system is performed. Further work on the application of the proposed scheme to an actual anti-skid brake system or any other practical system is recommended. This would require a number of issues to be investigated. In particular, the principle bottleneck in any neural control system is the speed of learning in the neural networks. Slow learning can limit the usefulness of the neural networks in control problems under a real-time environment. This problem is further exacerbated by the iterative search routine conducted in the proposed scheme. Therefore, methods of improving the speed of learning would need to be investigated. A brief analysis of the computational requirements of the proposed neural network based anti-skid brake system indicates the approach is feasible if the system is implemented on a specialised DSP chip. More extensive analysis would be required to determine the feasibility of implementing the scheme on existing hardware systems without suffering from the memory bandwidth bottle-neck problem. In terms of the practical realisation of the scheme, hardware based neural network systems which take advantage of the inherent parallelism found in neural network architectures offer the most promise for the future. Another potential problem with the hardware implementation of the scheme is the need for large amounts of memory to store the weights of the (generally) large networks required for most control applications. Therefore, the memory requirements of the scheme would also need to be considered carefully. These factors point towards the need for customised VLSI hardware implementation of neural control schemes. This would be an extremely challenging and potentially fruitful area of research [44].

In summary, the application of neural networks to the adaptive control of nonlinear (and linear) systems is still a very open area of research, in both the theoretical and application domains. There are still a number theoretical issues which need to be addressed or developed further. Significant insight into methods for addressing these issues can be gained from the many theoretical results established in adaptive control theory in the past two decades. The full potential of neural control schemes will only be realised through hardware implementation. In particular, customised VLSI implementations offer the most promise. This is also a burgeoning and potentially fruitful area of endeavour. It is the author's belief (and hope) that the area of neural adaptive control will continue to be a profitable and stimulating area of research and eventually the widespread industrial implementation of neural control schemes will come to fruition.

Appendix A

Backpropagation Learning Algorithm

The backpropagation learning algorithm was originally proposed by Werbos in 1974 [244] and then rediscovered by Parker (1985) [179] and Rumelhart et al. (1986) [200] It arguably remains the most commonly used supervised learning algorithm. The backpropagation algorithm is essentially an extension of the gradient descent algorithm to the multilayer perceptron.

The backpropagation algorithm essentially provides a means of adapting the weights W_{ij} in a MLP to learn the training pair $\{x_j, d_i\}$ in a minimum least squares sense.

Consider a network with M layers, $m = 1, \ldots, M$
y_i^m is the output of neuron i in layer m,
y_i^0 is the ith input $= x_i$,
W_{ij}^m is the weight for the connection from the jth node in layer $m-1$ to the ith in layer m,
d_i is the desired response of the ith output node and
N is the number of nodes in the output layer.

The cost function that is minimised is given by

$$E = \frac{1}{2}\sum_{i=1}^{N}(d_i - y_i^M)^2$$

$$= \frac{1}{2}\sum_{i=1}^{N}(d_i - g(\sum_j W_{ij}^M g(\sum_k W_{jk}^{M-1} \dots g(\sum_q W_{pq}^1 x_q))))^2 \tag{A.1}$$

The net input to node i in layer m is given by

$$h_i^m = \sum_j W_{ij}^m y_j^{m-1} \tag{A.2}$$

This results in an output

$$y_i^m = g(h_i^m) = g(\sum_j W_{ij}^m y_j^{m-1}) \tag{A.3}$$

where $g(.)$ is a sigmoidal activation function.

The gradient descent algorithm for adjusting the weights results in the following weight update equation.

$$W_{ij}^m(k+1) = W_{ij}^m(k) + \Delta W_{ij}^m(k) \tag{A.4}$$

where

$$\Delta W_{ij}^m(k) = -\eta \frac{\partial E}{\partial W_{ij}^m(k)} \tag{A.5}$$

The change in error criterion with respect to the weights is given by

$$\begin{aligned} \frac{\partial E}{\partial W_{ij}^m} &= \frac{\partial E}{\partial y_i^m}\frac{dy_i^m}{dh_i^m}\frac{\partial h_i^m}{\partial W_{ij}^m} \\ &= \frac{\partial E}{\partial y_i^m} g'(h_i^m) y_j^{m-1} \end{aligned} \tag{A.6}$$

where the iteration index k is neglected for simplicity sake and g' denotes the derivative of g. For the output layer, $\frac{\partial E}{\partial y_i^m} = y_i - d_i$ Therefore the effective error for the output layer is defined as

$$\begin{aligned} \delta_i^M &= g'(h_i^M)[d_i - y_i^M] \\ &= g'(\sum_j W_{ij}^M y_j^{M-1})[d_i - y_i^M] \end{aligned} \tag{A.7}$$

For the $M-1$ layer connections W_{jk}^{M-1}, the chain rule must be employed to obtain the term $\frac{\partial E}{\partial y_i^{M-1}}$ i.e.

$$\frac{\partial E}{\partial W_{jk}^{M-1}} = \frac{\partial E}{\partial y_j^{M-1}}\frac{\partial y_j^{M-1}}{\partial W_{jk}^{M-1}}$$

$$\begin{aligned}
&= \sum_i (d_i - y_i^M) g'(h_i^M) W_{ij}^M g'(h_j^{M-1}) y_k^{M-2} \\
&= \sum_i \delta_i^M W_{ij}^M g'(h_j^{M-1}) y_k^{M-2} \\
&= \delta_j^{M-1} y_k^{M-2} \qquad \text{(A.8)}
\end{aligned}$$

where $\delta_j^{M-1} = g'(h_j^{M-1}) \sum_i \delta_i^M W_{ij}^M$. Therefore the effective error from layer M is propagated backwards to layer $M-1$ to obtain its effective error which is used to update the weights in that layer.

Bibliography

[1] R. J. Adams and S. S. Banda. "Robust Flight Control Design Using Dynamic Inversion and Structured Singular Value Synthesis". *IEEE Transactions on Control Systems Technology*, 1(2):80–92, June 1993.

[2] M. Agarwal and D. E. Seborg. "Self-Tuning Controllers for Nonlinear Systems". *Automatica*, 23(2):209–214, 1987.

[3] D. J. Amit. *Modeling Brain Function: The World of Attractor Neural Networks.* Cambridge University Press, 1989.

[4] K. J. Åström. "Theory and Applications of Adaptive Control - A Survey". *Automatica*, 19:471–486, September 1983.

[5] K. J. Åström. "Adaptive Feedback Control". *Proceedings of the IEEE*, 75(2):185–217, September 1987.

[6] K. J. Åström, P. Hagander, and J. Sternby. "Zeros of Sampled Systems". *Automatica*, 20:31–38, January 1984.

[7] K. J. Åström and B. Wittenmark. "On Self Tuning Regulators". *Automatica*, 9:185–199, March 1973.

[8] K. J. Åström and B. Wittenmark. "Self-Tuning Controllers Based on Pole-Zero Placement". *Proc. IEE Pt D: Control Theory and Applications*, 127:120–130, May 1980.

[9] K. J. Åström and B. Wittenmark. *Adaptive Control.* Addison-Wesley, 1989.

[10] K. J. Åström and B. Wittenmark. *Computer Controlled Systems : Theory and Design*. Prentice-Hall, 1990.

[11] M. R. Azimi-Sadjadi and R-J Liou. "Fast Learning Process of Multilayer Neural Networks Using Recursive Least Squares Method". *IEEE Transactions on Signal Processing*, 40(2):446–450, February 1992.

[12] A. Balestrino, G. De Maria, and A. S. Zinober. "Nonlinear Adaptive Model-Following Control". *Automatica*, 20(5):559–568, 1984.

[13] R. D. Barnard. "An Optimal-Aim Control Strategy for Nonlinear Regulation Systems". *IEEE Transactions on Automatic Control*, 20(2):200–208, April 1975.

[14] R. D. Barnard. "Comments on "The Control of Robot Manipulators with Bounded Input"". *IEEE Transactions on Automatic Control*, 34(9):1022–1024, September 1989.

[15] A. G. Barto, R. S. Sutton, and C. W. Anderson. "Neurolike adaptive elements that can solve difficult learning control problems". *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(5):834–846, 1983.

[16] E. B. Baum and D. Haussler. "What size net gives valid generalization?". *Neural Computation*, 1:151–160, 1989.

[17] W. J. Bigley and V. J. Rizzo. "Wideband Linear Quadratic Control of a Gyro-Stabilised Electro-Optical Sight System". *IEEE Control Systems Magazine*, 7(4):20–24, August 1987.

[18] S. A. Billings, B. Jamaluddin, and S. Chen. "Properties of Neural Networks with Applications to Modelling Nonlinear Dynamical Systems". *International Journal of Control*, 55(1):193–224, 1992.

[19] S. A. Billings and W. S. F. Voon. "Structure Detection and Model Validity Tests in the Identification of Nonlinear Models". *IEEE Proceedings Pt D: Control Theory and Applications*, 130(4):193–199, July 1983.

[20] S. A. Billings and W. S. F. Voon. "Correlation Based Model Validity Tests for Nonlinear Models". *International Journal of Control*, 44(1):235–244, 1986.

[21] E. K. Blum and L. K. Li. "Approximation Theory and Feedforward Networks". *Neural Networks*, 4:511–515, 1991.

[22] M. Bodson, J. N. Chiasson, R. T Novotnak, and R. B. Rekowski. "High-Performance Nonlinear Feedback Control of a Permanent Magnet Stepper Motor". *IEEE Transactions on Control Systems Technology*, 1(1):5–14, March 1993.

[23] U. Borison. "Self-Tuning Regulators for a Class of Multivariable Systems". *Automatica*, 15:209–215, March 1979.

[24] U. Borison and R. Syding. "Self-Tuning Control of an Ore Crusher". *Automatica*, 12:1–7, January 1976.

[25] J. A. Bossi and M. A. Langehough. "Multivariable Autopilots for a Bank-to-Turn Missile". In *American Control Conference*, pages 567–572, 1988.

[26] D. S. Broomhead and D. Lowe. "Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks". Technical Report Memorandum 4148, Royal Signals and Radar Establishment, Great Malvern, Worcs. WR14 3PS, U.K., 1988.

[27] G. A. Carpenter and S. Grossberg. "A Massively Parallel Architecture for a Self-organising Neural Pattern Recognition Machine". *Computer Vision, Graphics and Image Processing*, 37:54–115, 1987.

[28] F-C. Chen. "Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control". *IEEE Control Systems Magazine*, 10(3):44–48, April 1990.

[29] F-C. Chen and H. K. Khalil. "Adaptive Control of Nonlinear Systems Using Neural Networks - A Deadzone Approach". *American Control Conference*, pages 667–672, 1991.

[30] F-C. Chen and H. K. Khalil. "Adaptive Control of Nonlinear Systems Using Neural Networks". *International Journal of Control*, 55(6):1299–1317, 1992.

[31] F-C. Chen and C. C. Liu. "Adaptive Control of Nonlinear Continuous-Time Systems Using Neural Networks - General Relative Degree and MIMO cases". *International Journal of Control*, 58(2):317–335, 1993.

[32] S. Chen, C.F. Cowan, S. A. Billings, and P.M. Grant. "Parallel Recursive Prediction Error Algorithm for Training Layered Neural Networks". *International Journal of Control*, 51(6):1215–1228, 1990.

[33] S. Chen, C.F. Cowan, and P.M. Grant. "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks". *IEEE Transactions on Neural Networks*, 2:302–309, March 1991.

[34] D.L. Chester. "Why two hidden layers are better than one". In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, pages 265–268, 1990.

[35] J. Chiasson. "Dynamic Feedback Linearization of the Induction Motor". *IEEE Transactions on Automatic Control*, 38(10):1589–1594, October 1993.

[36] J. Chiasson and M. Bodson. "Nonlinear Control of a Shunt DC Motor". *IEEE Transactions on Automatic Control*, 38(11):1662–1666, November 1993.

[37] A. Chotai, P. C. Young, and M. A. Behzadi. "Self-Adaptive Design of a Nonlinear Temperature Control System". *IEE Proceedings Pt. D: Control Theory and Applications*, 138(1):41–49, January 1991.

[38] D. W. Clarke and P. J. Gawthrop. "Self-Tuning Controller". *Proceedings of the IEE*, 122:929–934, September 1975.

[39] D. W. Clarke and P. J. Gawthrop. "Self-Tuning Control". *Proceedings of the IEE*, 126:633–640, June 1979.

[40] G. Cybenko. "Approximation by Superpositions of Sigmoidal Functions". *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[41] B. D'Andrea and J. Levine. "C.A.D. for Nonlinear Systems Decoupling, Perturbation Rejection and Feedback Linearization with Applications to the Dynamic Control of a Robot Arm". In M. Fleiss and M. Hazewinkel, editors, *Algebraic and Geometric Methods in Nonlinear Control Theory*, pages 545–572. Reidel, 1987.

[42] W. Dayawansa, W. M. Boothby, and D. L. Elliot. "Global State and Feedback Equivalence of Nonlinear Systems". *System and Control Letters*, 6:229–234, 1985.

[43] P.B. Deshpande and R.H. Ash. *Computer Process Control.* Instrument Society of America, 1981.

[44] M. R. DeYong, R. L. Findley, and C. Fields. "The Design, Fabrication and Test of a New VLSI Hybrid Analog-Digital Neural Processing Element". *IEEE Transactions on Neural Networks*, 3(3):363–374, May 1992.

[45] D. Dochain and G. Bastin. "Adaptive Identification and Control Algorithms for Nonlinear Bacterial Growth Systems". *Automatica*, 20:621–634, September 1984.

[46] P. Dorato, editor. *Robust Control.* IEEE Press, 1987.

[47] P. Dorato, L. Fortuna, and G. Muscato. *Robust Control for Unstructured Perturbations: An Introduction.* Springer-Verlag, 1992.

[48] P. Dorato and R. K. Yedavalli, editors. *Recent Advances in Robust Control.* IEEE Press, 1990.

[49] G. A. Dumont and P. R. Belanger. "Self-Tuning Control of a Titanium Dioxide Kiln". *IEEE Transactions on Automatic Control*, 23(4):532–538, August 1978.

[50] B. Egardt. "Unification of Some Discrete-Time Adaptive Control Schemes". *IEEE Transactions on Automatic Control*, 25(4):693–697, August 1980.

[51] G. Feng. "Stable Identification of Nonlinear Dynamic Systems Using RBF Nets". In *Proceedings of the 12th IFAC World Congress*, volume 9, pages 269–272, 1993.

[52] M. Fliess and M. Hazewinkel, editors. *Algebraic and Geometric Methods in Nonlinear Control Theory.* D. Reidel, 1986.

[53] R. T. Fling and R. E. Fenton. "A Describing-Function Approach to Antiskid Design". *IEEE Transactions on Vehicular Technology*, VT-30(3):134–144, August 1981.

[54] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems.* Addison-Wesley, 1994. 3rd Edition.

[55] K. Funahashi. "On the Approximate Realization of Continuous Mappings by Neural Networks". *Neural Networks*, 2:183–192, 1989.

[56] L. Gao, L. Chen, Y. Fan, and H. Ma. "A Nonlinear Control System Design for Power Systems". *Automatica*, 28(5):975–979, 1992.

[57] C. J. Goh and J. L. Noakes. "Neural Networks and Identification of Systems with Unobserved States". *ASME Journal of Dynamic Systems*, 115:196–203, 1993.

[58] E. Göhring. "Commercial Vehicle Electronics - An Improvement in Safety, Economy and Environmental Compatibility". *Proceedings of the Institution of Mechanical Engineers: Traction Control and Anti-wheel-spin Systems*, C360/88:1–13, 1988.

[59] G. C. Goodwin and D. Q. Mayne. "A Parameter Estimation Perspective of Continuous-Time Model Reference Adaptive Control". *Automatica*, 23(1):57–70, January 1987.

[60] G. C. Goodwin, P. J. Ramadge, and P. E. Caines. "Discrete-Time Multivariable Adaptive Control". *IEEE Transactions on Automatic Control*, 25(3):449–456, June 1980.

[61] J. W. Grizzle and P. V. Kokotović. "Feedback Linearization of Sampled-Data Systems". *IEEE Transactions on Automatic Control*, 33(9):857–859, September 1988.

[62] A. Guez and J. Selinsky. "A Trainable Neuromorphic Controller". *J. Robotic Systems*, 5(4):24–32, 1988.

[63] M. M. Gupta, editor. *Adaptive Methods for Control System Design.* IEEE Press, 1986.

[64] M. M. Gupta and D. H. Rao, editors. *Neuro-Control Systems: Theory and Applications.* IEEE Press, 1994.

[65] R. J. Hampo and K. A. Marko. "Investigation of the Application of Neural Networks to Fault Tolerant Control of an Active Suspension System". In *1992 American Control Conference*, volume 1, pages 11–15, Chicago, Illinois, June 1992.

[66] C. C. Hang and P. C. Parks. "Comparative Studies of Model Reference Adaptive Control Systems". *IEEE Transactions on Automatic Control*, 18:419–428, October 1973.

[67] J. L. Harned, L. E. Johnston, and G. Scharpf. "Measurement of Tire Brake Force Characteristics as Related to Wheel Slip (Antilock) Control System Design". SAE Paper 690214, 1969.

[68] E.J. Hartman, J.D. Keeler, and J.M. Kowalski. "Layered Neural Networks with Gaussian Hidden Units as Universal Approximations". *Neural Computation*, 2:210–215, 1990.

[69] J. Hauser, S. Sastry, and G. Meyer. "Nonlinear Control Design for Slightly Nonminimum Phase Systems: Application to V/STOL Aircraft". *Automatica*, 28(4):665–679, 1992.

[70] R. Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, 1990.

[71] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

[72] J. J. Hopfield. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982. In "Neurocomputing", J. A. Anderson and E. Rosenfeld editors.

[73] J. J. Hopfield. "Neurons with graded responses have collective computational properties like those of two-state neurons". *Proceedings of the National Academy of Sciences*, 81:3088–3092, 1984. In "Neurocomputing", J. A. Anderson and E. Rosenfeld editors.

[74] K. Hornik, M. Stinchcombe, and H. White. "Multilayered Feedforward Networks are Universal Approximators". *Neural Networks*, 2:359–366, 1989.

[75] D. A Hoskins, J. N. Hwang, and J. Vagners. "Iterative Inversion of Neural Networks and its Applications to Adaptive Control". *IEEE Transactions on Neural Networks*, 3(2):292–301, March 1992.

[76] S.C. Huang and Y.F. Huang. "Bounds on the number of hidden neurons in multilayer perceptrons". *IEEE Transactions in Neural Networks*, 2(1):47–55, January 1991.

[77] W. Y. Huang and R. P. Lippmann. "Neural Net and Traditional Classifiers". In D. Z. Anderson, editor, *Neural Information Processing Systems*, pages 387–396. American Institute of Physics, 1988.

[78] D.G. Hull, J.L. Speyer, and C.Y. Tseng. "Maximum Information Guidance for Homing Missiles". *Journal of Guidance*, 8:pp. 494–497, July-August 1985.

[79] K.J. Hunt and D. Sbarbaro. "Neural Networks for Nonlinear Internal Model Control". *IEE Proceedings, Pt. D : Control Theory and Applications*, 138(5):431–438, Sept. 1991.

[80] K.J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop. "Neural Networks for Control Systems - A Survey". *Automatica*, 28(6):1083–1112, 1992.

[81] L. R. Hunt, R. Su, and G. Meyer. "Global Transformations of Nonlinear Systems". *IEEE Transactions on Automatic Control*, AC-28(1):24–31, January 1983.

[82] D. Hush, B. Horne, and J. M . Salas. "Error Surfaces for Multi-Layer Perceptrons". *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):1153–1161, September/October 1992.

[83] D. R. Hush and B. G. Horne. "Progress in Supervised Neural Networks : What's New Since Lippmann?". *IEEE Signal Processing Magazine*, 10(1):8–39, January 1993.

[84] IEEE Control Systems Society. "Special Section on Neural Networks". *IEEE Control Systems Magazine*, 8(2):3–31, April 1988.

[85] IEEE Control Systems Society. "Special Section on Neural Networks for Control Systems". *IEEE Control Systems Magazine*, 9(3):25–59, April 1989.

[86] IEEE Control Systems Society. "Special Issue on Neural Networks in Control Systems". *IEEE Control Systems Magazine*, 10(3), April 1990.

[87] IEEE Control Systems Society. "Special Issue on Neural Networks in Control Systems". *IEEE Control Systems Magazine*, 12(2), April 1992.

[88] T. Ionescu and R. V. Monopoli. "Discrete Model Reference Adaptive Control with an Augmented Error Signal". *Automatica*, 13:507–517, September 1977.

[89] T. Ishikawa, J. Tsuji, H. Ohmori, and A. Sano. "Novel Configuration of Nonlinear Adaptive Control Incorporating Neural Network". In *Proceedings of the 12th IFAC World Congress*, volume 9, pages 483–488, 1993.

[90] A. Isidori. *Nonlinear Control Systems : An Introduction.* Springer Verlag, 1989.

[91] R. A. Jacobs. "Increased Rates of Convergence Through Learning Rate Adaptation". *Neural Networks*, 1(4):295–308, 1988.

[92] B. Jakubczyk. "Feedback Linearization of Discrete-Time Systems". *Systems and Control Letters*, 9:411–416, 1987.

[93] R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee, P. S. Lewis, and S. Qian. "Function Approximation and Time Series Prediction with Neural Networks". Technical Report LA-UR 90-21, Los Alamos National Laboratory, 1990.

[94] R. D. Jones, Y. C. Lee, S. Qian, C. W. Barnes, K. R. Bisset, G. M. Bruce, G. W. Flake, K. Lee, L. A. Lee, W. C. Mead, M. K. O'Rourke, I. J. Poli, and L. E. Thode. "Nonlinear Adaptive Networks : A Little Theory, a Few Applications". Technical Report LA-UR 91-273, Los Alamos National Laboratory, 1991.

[95] M. I. Jordan. "Supervised learning and systems with excess degrees of freedom". Technical Report COINS 88-27, Computer and Information Sciences, University of Massachusetts, Amherst, MA, May 1988.

[96] M. I. Jordan and R. A. Jacobs. "Learning to Control an Unstable System with Forward Modeling". In D. Touretzky, editor, *Advances In Neural Information Processing Systems 2*, pages 324–331. Morgan Kaufmann, 1990.

[97] C. G. Källström. "Control of Yaw and Roll by a Rudder/Fin-Stabilization System". *6th Ship Control Systems Symposium*, 3:1–9, 1981.

[98] C. G. Källström, K. J. Åström, N. E. Thorell, J. Eriksson, and L. Sten. "Adaptive Autopilots for Tankers". *Automatica*, 15:241–254, May 1979.

[99] B. L. Kalman and S. C. Kwasny. "Why Tanh: Choosing a Sigmoidal Function". In *International Joint Conference on Neural Networks*, pages IV: 578–581, Baltimore, 1992.

[100] R. E. Kalman. "Design of a Self-Optimizing Control System". *Transactions of the ASME*, 80:468–478, February 1958.

[101] I. Kanellakopoulos, P. V. Kokotivić, and A. S. Morse. "Adaptive Output-Feedback Control of Systems with Output Nonlinearities". *IEEE Transactions on Automatic Control*, 37(11):1666–1682, November 1992.

[102] I. Kanellakopoulos, P. V. Kokotović, and R. Marino. "Robustness of Adaptive Control Schemes Under EMC". *IFAC Symposium on Nonlinear Control System Design*, pages 192–197, 1989.

[103] I. Kanellakopoulos, P. V. Kokotović, and R. Marino. "An Extended Direct Scheme for Robust Adaptive Nonlinear Control". *Automatica*, 27(2):247–255, 1991.

[104] I. Kanellakopoulos, P. V. Kokotović, and R. H. Middleton. "Indirect Adaptive Output-Feedback Control of a Class of Nonlinear Systems". *Proceedings of the 28th Conference on Decision and Control*, December 1990.

[105] I. Kanellakopoulos, P. V. Kokotović, and R. H. Middleton. "Observer-based Adaptive Control of Nonlinear Systems Under Matching Conditions". *Proc. 1990 Automatic Control Conference*, pages 549–555, 1990.

[106] I. Kanellakopoulos, P. V. Kokotović, and A. S. Morse. "Systematic Design of Adaptive Controllers for Feedback Linearizable Systems". *IEEE Transactions on Automatic Control*, 36(11):1241–1253, November 1991.

[107] L. Keviczky, J. Hetthessy, M. Hilger, and J. Kolostori. "Self-Tuning Adaptive Control of Cement Raw Material Blending". *Automatica*, 14:525–532, November 1978.

[108] H. K. Khalil. *Nonlinear Systems*. Collier-Macmillian, 1992.

[109] T. Kohonen. "Self-organized formation of topologically correct feature maps". *Biological Cybernetics*, 43:59–69, 1982. In "Neurocomputing", J. A. Anderson and E. Rosenfeld editors.

[110] P. V. Kokotović. "Recent Trends in Feedback Design: An Overview". *Automatica*, 21(3):225–236, 1985.

[111] P. V. Kokotović, editor. *Foundations of Adaptive Control*. Lecture Notes in Control and Information Sciences. Springer-Verlag, 1991.

[112] P. V. Kokotović, I. Kanellakopoulos, and A.S. Morse. "Adaptive Feedback Linearization of Nonlinear Systems". In P.V. Kokotović, editor, *Foundations of Adaptive Control*, pages 311–346. Springer-Verlag, 1991.

[113] P. V. Kokotović, H. K. Khalil, and J. O'Reilly, editors. *Singular Perturbation Methods in Control: Analysis and Design*. IEEE Press, 1986.

[114] K.-M Koo and J.-H. Kim. "Robust Control of Robot Manipulators with Parametric Uncertainty". *IEEE Transactions on Automatic Control*, 39(6):1230–1233, June 1994.

[115] A. L. Kornhauser. "Neural Network Approaches to Lateral Control of Autonomous Highway Vehicles". SAE Paper 912871, 1991.

[116] G. Kreisselmeier and K. S. Narendra. "Stable Model Reference Adaptive Control in the Presence of Bounded Disturbances". *IEEE Transactions on Automatic Control*, 27(6):593–611, December 1982.

[117] I. D. Landau and R. Lozano. "Unification of Discrete-Time Explicit Model Reference Adaptive Control Designs". *Automatica*, 17:593–611, July 1981.

[118] Y. D. Landau. *Adaptive Control : The Model Reference Approach*. Dekker, 1979.

[119] K. J. Lang, A. H. Waibel, and G. E. Hinton. "A Time-Delay Neural Network Architecture for Isolated Word Recognition". *Neural Networks*, 3(1):23–43, 1990.

[120] A. Lapedes and R. Farber. "Nonlinear Signal Processing Using Neural Networks : Prediction and Signal Modeling". Technical Report LA-UR-87-2662, Los Alamos National Laboratories, Los Alamos, New Mexico, 1987.

[121] K. T. Law, D. J. Hill, and N. R. Godfrey. "Robust Controller Structure for Coordinated Power System Voltage Regulator and Stabilizer Design". *IEEE Transactions on Control Systems Technology*, 2(3):220–232, September 1994.

[122] J. R. Layne, K. M. Passino, and S. Yurkovich. "Fuzzy Learning Control for Antiskid Braking Systems". *IEEE Transactions on Control Systems Technology*, 1(2):122–129, June 1993.

[123] K. A. Lee and H. Yee. "Self Tuning Load Frequency Controller for Interconnected Power Systems Including Effects of Nonlinearities". *Proceedings of the 1990 American Control Conference*, pages 2100–2105, May 1990.

[124] H. Leiber and A. Czinczel. "Four Years of Experience with 4-Wheel Antiskid Brake Systems (ABS)". SAE Paper 830481, 1983.

[125] Jesse Leitner. Personal communication, December 1994. Aerospace Engineer, U. S. Air Force Phillips Lab.

[126] A. U. Levin and K. S. Narendra. "Control of Nonlinear Dynamical Systems Using Neural Networks: Controllability and Stabilization". *IEEE Transactions on Neural Networks*, 4(2):192–206, March 1993.

[127] C. C. Lim. *Autopilot Design for Ship Control.* PhD thesis, University of Technology, Loughborough, 1980.

[128] R. P. Lippman. "An Introduction to Computing with Neural Nets". *IEEE Acoustics, Speech and Signal Processing Magazine*, pages 4–22, April 1987.

[129] R. P. Lippman. "Pattern Classification Using Neural Networks". *IEEE Communications Magazine*, pages 47–64, November 1989.

[130] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification.* MIT Press, 1983.

[131] A. P. Loh and K. F. Fong. "Backpropagation Using Generalised Least Squares". In *Proceedings of the 12th IFAC World Congress*, pages 592–597, 1993.

[132] E. B. MacLaurin and D. A. Crolla. "Wheel-spin Control for On/Off-Road Vehicles". *Proceedings of the Institution of Mechanical Engineers: Traction Control and Anti-wheel-spin Systems*, C363/88:33–41, 1988.

[133] M. Majors, J. Stori, and D-I. Cho. "Neural Network Control of Automotive Fuel-Injection Systems". *IEEE Control Systems Magazine*, 14(3):31–36, June 1994.

[134] I. M. Y. Mareels, B. D. O. Anderson, R. R. Bitmead, M. Bodson, and S. S. Sastry. "Revisiting the MIT Rule for Adaptive Control". *Proceedings of the 2nd IFAC Workshop on Adaptive Systems*, 1986.

[135] R. Marino. "Feedback Linearization Techniques in Robotics and Power Systems". In M. Fleiss and M. Hazewinkel, editors, *Algebraic and Geometric Methods in Nonlinear Control Theory*, pages 523–543. Reidel, 1987.

[136] R. Marino, I. Kanellakopoulos, and P. Kokotović. "Adaptive Tracking for Feedback Linearizable SISO Systems". *Proceedings of the 28th IEEE Conference on Decision and Control*, pages 1002–1007, 1989.

[137] R. Marino and P. Tomei. "Dynamic Output Feedback Linearization and Global Stabilization". *System and Control Letters*, 17:115–121, 1991.

[138] R. Marino and P. Tomei. "Global Adaptive Output-Feedback Control of Nonlinear Systems, Part I: Linear Parameterization". *IEEE Transactions on Automatic Control*, 38(1):17–32, January 1993.

[139] R. Marino and P. Tomei. "Global Adaptive Output-Feedback Control of Nonlinear Systems, Part I: Nonlinear Parameterization". *IEEE Transactions on Automatic Control*, 38(1):33–48, January 1993.

[140] S.K. Mazumdar and C.C. Lim. "Adaptive Controller for Marginally Stable Nonlinear Systems Using Neural Networks". *TENCON '92: IEEE Region 10 International Conference*, pages 535–539, Nov. 1992.

[141] S.K. Mazumdar and C.C. Lim. "A Stable Neural Controller for Nonminimum Phase". *Fourth Australian Conference on Neural Networks*, pages 138–141, Feb. 1993.

[142] S.K. Mazumdar and C.C. Lim. "A Neural Network Based Anti-Skid Brake System". *Submitted*, 1995.

[143] S.K. Mazumdar and C.C. Lim. "An Neural Adaptive Control Scheme for Discrete-Time non-Affine Nonlinear Systems". *Submitted*, 1995.

[144] S.K. Mazumdar and C.C. Lim. "Investigation of Stability and Convergence Issues for an Enhanced Model Reference Neural Adaptive Control Scheme (Invited Paper)". In *Electronics Technology Directions To The Year 2000 - ETD 2000*, 1995.

[145] S.K. Mazumdar and C.C. Lim. "Investigation of the Use Neural Networks for Anti-Skid Brake System Design". *Submitted*, 1995.

[146] W. S. McCullogh and W. Pitts. "A logical calculus of ideas immanent in nervous activity". *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. In "Neurocomputing", J. A. Anderson and E. Rosenfeld editors.

[147] C. Mead. *Analog VLSI and Neural Systems.* Addison-Wesley, 1989.

[148] W. C. Mead, R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, L. A. Lee, and M. K. O'Rourke. "Prediction of Chaotic Time Series Using CNLS Net". Technical Report LA-UR-91-720, Los Alamos National Laboratory, 1991.

[149] A. C. Messer and M. J. Grimble. "Introduction to Robust Ship Track-Keeping Control Design". *Transactions of the Institute of Measurement and Control*, 15(3):104–110, 1993.

[150] D. E. Miller and E. J. Davison. "An Adaptive Controller Which Provides Lyapunov Stability". *IEEE Transactions on Automatic Control*, 34(6):599–609, June 1989.

[151] W. T. Miller III, R. S. Sutton, and P. J. Werbos, editors. *Neural Networks for Control.* MIT Press, Cambridge, Massachusetts, 1991.

[152] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.

[153] S. Monaco. "Nonlinear Systems in Discrete Time". In M. Fleiss and M. Hazewinkel, editors, *Algebraic and Geometric Methods in Nonlinear Control Theory*, pages 411–430. Reidel, 1987.

[154] S. Monaco, D. Normand-Cyrot, and S. Stornelli. "On the Linearizing Feedback in Nonlinear Sampled Data Control Schemes". In *Proceedings of the 25th IEEE Conference on Decision and Control*, volume 3, pages 2056–2060, Athens, Greece, 1986.

[155] S. Monaco and S. Stornelli. "A Nonlinear Feedback Control Law for Attitude Control". In M. Fleiss and M. Hazewinkel, editors, *Algebraic and Geometric Methods in Nonlinear Control Theory*, pages 573–595. Reidel, 1987.

[156] R. V. Monopoli. "Lyapunov's Method for Adaptive Control System Design". *IEEE Transactions on Automatic Control*, 12(3):334–335, June 1967.

[157] R. V. Monopoli. "Model Reference Adaptive Control with Augmented Error Signal". *IEEE Transactions on Automatic Control*, 19(4):474–484, October 1974.

[158] J. Moody and C.J. Darken. "Fast Learning in Networks of Locally-Tuned Processing Units". *Neural Computation*, 1:281–294, 1989.

[159] S. Morita. "Optimization Control for Combustion Parameters of Gasoline Engines using Neural Networks - In the Case of On-Line Control". *JSAE Review*, 14(3):4–9, July 1993.

[160] S. Mukhopadhyay and K. S. Narendra. "Disturbance Rejection in Nonlinear Systems Using Neural Networks". *IEEE Transactions on Neural Networks*, 4(1):53–62, January 1993.

[161] Y. Nakayama, F. Kawahata, and K. Shirai. "Development of Linear Hydraulic ABS". *JSAE Review*, 14(1):36–41, 1993.

[162] K. Nam and A. Arapostathis. "A Model Reference Adaptive Control Scheme for Pure-Feedback Nonlinear Systems". *IEEE Transactions on Automatic Control*, 33(9):803–811, September 1988.

[163] K. S. Narendra. "The Maturing of Adaptive Control". In P. V. Kokotović, editor, *Foundations of Adaptive Control*, Lecture Notes in Control and Information Sciences, pages 3–36. Springer-Verlag, 1991.

[164] K. S. Narendra, I. H. Khalifa, and A. M. Annaswamy. "Error Models for Stable Hybrid Adaptive Systems". *IEEE Transactions on Automatic Control*, 30(4):339–347, April 1985.

[165] K. S. Narendra and L. S. Valavani. "Stable Adaptive Controller Design - Direct Control". *IEEE Transactions on Automatic Control*, 23(4):570–583, August 1978.

[166] K.S. Narendra. "Adaptive Control Using Neural Networks". In R.S. Sutton W. T. Miller III and P.J. Werbos, editors, *Neural Networks for Control*, chapter 5, pages 115–142. MIT Press, 1991.

[167] K.S Narendra and A.M. Annaswamy. *Stable Adaptive Systems.* Prentice-Hall, 1989.

[168] K.S. Narendra, R. Ortega, and P. Dorato, editors. *Advances in Adaptive Control.* IEEE Press, 1991.

[169] K.S. Narendra and K. Parthasarathy. "Identification and Control of Dynamical Systems Using Neural Networks". *IEEE Transactions on Neural Networks*, 1:4–27, March 1990.

[170] D. Nguyen and B. Widrow. "Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights". In *International Joint Conference on Neural Networks*, pages III: 21–26, San Diego, 1990.

[171] D. Nguyen and B. Widrow. "Neural Networks for Self-Learning Control Systems". *IEEE Control Systems Magazine*, 10(3):18–23, April 1990.

[172] R. A. Nichols, R. T. Reichert, and W. J. Rugh. "Gain Scheduling for $mboxH_\infty$ Controllers: A Flight Control Example". *IEEE Transactions on Control Systems Technology*, 1(2):69–79, June 1993.

[173] H. Nijmeijer and A. J. van der Schaft. *Nonlinear Dynamical Control Systems.* Springer-Verlag, 1990.

[174] K. Ogata. *Discrete-Time Control Systems.* Prentice-Hall, 1987.

[175] H. Ohno, T. Suzuki, K. Aoki, A. Takahasi, and G. Sugimoto. "Neural Network Control for Automatic Braking Control System". *Neural Networks*, 7(8):1303–1312, 1994.

[176] R. Ortega, C. Canudas, and S. I. Seleme. "Nonlinear Control of Induction Motors: Torque Tracking with Unknown Load Disturbances". *IEEE Transactions on Automatic Control*, 38(11):1675–1680, November 1993.

[177] H. Ouwerkerk and R. R. Guntur. "Skid Prediction". *Vehicle Systems Dynamics*, 1(2):67–88, November 1972.

[178] J. Park and I.W. Sandberg. "Universal Approximation Using Radial-Basis-Function Networks". *Neural Computation*, 3:246–257, 1991.

[179] D. B. Parker. "Learning Logic". Technical Report TR-47, Centre for Computing Research, Economics and Management, Massachusetts Institute of Technology, Cambridge MA, 1985.

[180] P. C. Parks. "Lyapunov Redesign of Model Reference Adaptive Control Systems". *IEEE Transactions on Automatic Control*, 11:363–367, July 1966.

[181] B. A. Pearlmutter. "Learning State Space Trajectories in Recurrent Neural Networks". *Neural Computation*, 1(2):263–269, 1989.

[182] V. Peterka. "Predictor-based Self-tuning Control". *Automatica*, 20:39–50, January 1984.

[183] E. Petersen and J. Rothen. "ABS with Integrated Drive Slip Control (ASR) for Air-braked Commercial Vehicles - Modular Concept, Functional Principles, Performance". *Proceedings of the Institution of Mechanical Engineers: Traction Control and Anti-wheel-spin Systems*, C374/88:123–134, 1988.

[184] F. J. Pineda. "Generalization of Back-Propagation to Recurrent Neural Networks". *Physical Review Letters*, 59(19):2229–2232, November 1987.

[185] T. Poggio and F. Girosi. "Networks for Approximation and Learning". *Proceedings of the IEEE*, 78(9):1481–1497, September 1990.

[186] M.M. Polycarpou and P.A. Ioannou. "Identification and Control of Nonlinear Systems Using Neural Network Models : Design and Stability Analysis". Technical Report 91-09-01, Department of Electrical Engineering Systems, University of Southern California, Los Angeles CA, Sept 1991.

[187] J. Pomet and L. Praly. "Adaptive Nonlinear Regulation: Equation Error from the Lyapunov Equation". *Proceedings of the 28th IEEE Conference on Decision and Control*, pages 1008–1013, 1989.

[188] D. L. Prager and P. E. Wellstead. "Multivariable Pole-Assignment Self-Tuning Regulators". *Proc. IEE Pt D: Control Theory and Applications*, 128:9–18, January 1981.

[189] D. Psaltis, A. Sideris, and A. A. Yamamura. "A Multi-Layered Neural Network Controller". *IEEE Control Systems Magazine*, 8(2):17–20, April 1988.

[190] V. Rehbock, C. C. Lim, and K. L. Teo. "A Stable Constrained Optimal Model Following Controller for Discrete-Time Nonlinear Systems Affine in the Control". *To Appear in Control-Theory and Advanced Technology*, 1994.

[191] V. Rehbock, K.L. Teo, and L.S. Jennings. "A Linear Programming Approach to On-Line Constrained Optimal Terrain Tracking Systems". *Submitted*, 1992.

[192] V. Rehbock, K.L. Teo, and L.S. Jennings. "An On-Line Tracking Algorithm for a Class of Systems Subject to Noise". *Submitted*, 1993.

[193] R. T. Reichert. "Dynamic Scheduling of Modern-Robust-Control Autopilot Design for Missiles". *IEEE Control Systems Magazine*, 12(5):35–42, October 1992.

[194] A. J. Robinson and F. Fallside. "Static and Dynamic Error Propagation Networks with Application to Speech Coding". In D. Z. Anderson, editor, *Neural Information Processing Systems*, pages 632–641. American Institute of Physics, 1988.

[195] B. J. Robinson and A. R. Duffin. "The Performance and Reliability of Anti-lock Braking Systems". *Proceedings of the Institution of Mechanical Engineers: Braking of Road Vehicles*, C444/051/93:115–126, 1993.

[196] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain". *Psychological Review*, 65:386–408, 1958. In "Neurocomputing", J. A. Anderson and E. Rosenfeld editors.

[197] F. Rosenblatt. *Principles of Neurodynamics.* Spartan, 1962.

[198] D. W. Ruck, S. K. Rogers, M. Kabrisky, P. S. Maybeck, and M. E. Oxley. "Comparative Analysis of Backpropagation and the Extended Kalman Filter for Training Multilayer Perceptrons". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):686–691, June 1992.

[199] W. J. Rugh. "Analytical Framework for Gain Scheduling". *IEEE Control Systems Magazine*, 11(1):79–84, January 1991.

[200] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning Internal Representation By Error Propagation". In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Vol. 1: Foundations*, pages 318–362. MIT Press, 1986.

[201] M. Saerens and A. Soquet. "Neural controller based on back-propagation algorithm". *IEE Proceedings-F : Radar and Signal Processing*, 138(1):55–62, 1991.

[202] R.M. Sanner and J-J. Slotine. "Direct Adaptive Control Using Gaussian Networks". Technical Report NSL-910303, Nonlinear Systems Laboratory, Massachusetts Institute of Technology, Cambridge MA, March 1991.

[203] R.M. Sanner and J-J. Slotine. "Stable Adaptive Control and Recursive Identification Using Radial Gaussian Networks". *Proceedings, IEEE Conference on Decision and Control*, Dec. 1991.

[204] R.M. Sanner and J-J. Slotine. "Gaussian Networks for Direct Adaptive Control". *IEEE Transactions on Neural Networks*, 3(6):837–864, Nov. 1992.

[205] S. S. Sastry and M. Bodson. *Adaptive Control : Stability, Convergence and Robustness.* Prentice-Hall, 1989.

[206] S. S. Sastry and P. V. Kokotović. "Feedback Linearization in the Presence of Uncertainties". *International Journal of Adaptive Control and Signal Processing*, 2:327–346, 1988.

[207] S.S. Sastry and A. Isidori. "Adaptive Control of Linearizable Systems". *IEEE Transactions Automatic Control*, AC-34(11):1123–1131, November 1989.

[208] D. W. Satchell and J. C. Greenwood. "Silicon Microengineering for Accelerometers". *Proceedings of the Institution of Mechanical Engineers*, C46/87:191–193, 1987.

[209] M. Satoh and S. Shiraishi. "Excess Operation of Antilock Brake System on a Rough Road". *Proceedings of the Institution of Mechanical Engineers: Braking of Road Vehicles*, C18/83:125–133, 1983.

[210] J. S. Shamma and M. Athans. "Gain Scheduling: Potential Hazards and Possible Remedies". *IEEE Control Systems Magazine*, 12(3):101–107, June 1992.

[211] C. L. Sheperd and L. Valavani. "Autopilot Design for Bank-to-Turn Missile Using LQG/LQR Methodology". In *American Control Conference*, pages 579–586, 1988.

[212] A. Sigl and A. Czinczel. "ABS/ASR5 - The New ABS/ASR5 System for Passenger Cars". *Proceedings of the Institution of Mechanical Engineers: Braking of Road Vehicles*, C444/058/93:81–87, 1993.

[213] S. Singhal and L. Wu. "Training Feed-Forward Networks with the Extended Kalman Filter". In *International Conference on Acoustics, Speech and Signal Processing*, pages 1187–1190, Glasgow, 1989.

[214] A. S. C. Sinha, S. Kayabar, and H. O. Yurtseven. "State-Space Approach to Nonlinear Model Reference Adaptive Control". *International Journal of Systems Science*, 23(5):833–837, 1992.

[215] H. Sira-Ramirez. "Nonlinear Discrete Variable Structure Systems in Quasi-Sliding Mode". *International Journal of Control*, 54(1):1171–1187, 1992.

[216] H. Sira-Ramirez, M. Zribi, and S. Ahmad. "Adaptive Dynamical Feedback Regulation Strategies for Linearizable Uncertain Systems". *International Journal of Control*, 57(1):121–139, 1993.

[217] J. M. Skowronski. "Adaptive Nonlinear Model Following and Avoidance Under Uncertainty". *IEEE Transactions on Circuits and Systems*, 35(9):1082–1087, September 1988.

[218] J-J. E. Slotine and W. Li. *Applied Nonlinear Control.* Prentice Hall, 1991.

[219] J. C. Smith, K. C. Cheok, and N. Huang. "Optimal Parametric Control of a Semi-Active Suspension System Using Neural Networks". In *1992 American Control Conference*, volume 2, pages 963–967, Chicago, Illinois, June 1992.

[220] T. G. Smithson. "A Review of the Mechanical Design and Development of High Performance Accelerometer". *Proceedings of the Institution of Mechanical Engineers*, C49/87:175–190, 1987.

[221] T. Söderström and P. Stoica. *System Identification.* Prentice Hall, 1989.

[222] E. D. Sontag. "Feedback Stabilization Using Two-Hidden-Layer Net". Technical Report SYCON-90-11, Rutgers Center for Systems and Control, Department of Mathematics, Rutgers University, New Brunswick, NJ 08903, 1990.

[223] E. D. Sontag. "Feedback Stabilization Using Two-Hidden-Layer Nets". *IEEE Transactions on Neural Networks*, 3(6):981–990, November 1992.

[224] M. W. Spong. "On the Robust Control of Robot Manipulators". *IEEE Transactions on Automatic Control*, 37(11):1782–1786, November 1992.

[225] M.W. Spong, J.S. Thorp, and J.M. Kleinwaks. "The Control of Robot Manipulators with Bounded Input". *IEEE Transactions on Automatic Control*, AC-31(6):483–490, 1986.

[226] K. Stokbro, D.K. Umberger, and J.A. Hertz. "Exploiting Neurons with Localised Receptive Fields to Learn Chaos". *Complex System*, 4:603–622, Dec. 1990.

[227] R. Su. "On the Linear Equivalents of Nonlinear Systems". *Systems and Control Letters*, 2(1):48–52, July 1982.

[228] R. Su and L. R. Hunt. "A Canonical Expansion for Nonlinear Systems". *IEEE Transactions on Automatic Control*, 31(7):670–673, July 1986.

[229] H-S. Tan and Y-K. Chin. "Vehicle Antilock Braking and Traction Control: A Theoretical Study". *International Journal of Systems Science*, 23(3):351–365, 1992.

[230] H-S. Tan and M. Tomizuka. "Discrete-Time Controller Design for Robust Vehicle Traction". *IEEE Control Systems Magazine*, 10(2):107–113, April 1990.

[231] D.G. Taylor, P.V. Kokotovic, R. Marino, and I. Kanellakopoulos. "Adaptive Regulation of Nonlinear Systems with Unmodelled Dynamics". *IEEE Transactions Automatic Control*, AC-34(4):405–412, April 1989.

[232] A. Teel, R. Kadiyala, P. Kokotovic, and S. Sastry. "Indirect Techniques for Adaptive Input Output Linearization of Nonlinear Systems". *Proc. 1990 Automatic Control Conference*, pages 79–80, 1990.

[233] M. Tsujii, H. Takeuch, K. Oda, and M. Ohba. "Application of Self-Tuning to Automotive Cruise Control". *Proceedings of the 1990 American Control Conference*, pages 1843–1848, May 1990.

[234] E. Tzirkel-Hancock and F. Fallside. "A Direct Control Method for a Class of Nonlinear Systems Using Neural Networks ". Technical Report CUED/F-INFENG/TR.65, Cambridge University Engineering Department, Cambridge, England, March 1991.

[235] E. Tzirkel-Hancock and F. Fallside. "Stable Control of Nonlinear Systems Using Neural Networks ". Technical Report CUED/F-INFENG/TR.81, Cambridge University Engineering Department, Cambridge, England, July 1991.

[236] I. Vajk, M. Vajta, L. Keviczky, R. Haber, J. Hetthessy, and K. Kovacs. "Adaptive Load-frequency Control of the Hungarian Power System". *Automatica*, 21:129–137, March 1985.

[237] M. Vidyasagar. "New Directions of Research in Nonlinear System Theory". *Proceedings of the IEEE*, 74(8):1060–1091, August 1986.

[238] B. P. Volfson. "Wheel Slip Phenomena". *International Journal of Vehicle Design*, 6(4/5):556–560, 1985.

[239] A. Waibel. "Modular Construction of Time-Delay Neural Networks for Speech Recognition". *Neural Computation*, 1:36–49, 1989.

[240] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. "Phoneme Recognition Using Time-Delay Neural Networks". *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339, March 1989.

[241] H. Wallentowitz, G. Egger, E. Herb, and H. Krusche. "Stability and Traction Control for Four-Wheel Drive Passenger Vehicles". *Proceedings of the Institution of Mechanical Engineers: Traction Control and Anti-wheel-spin Systems*, C365/88:49–60, 1988.

[242] K. Warwick, G. W. Irwin, and K. J. Hunt, editors. *Neural Networks for Control and Systems.* IEE Control Engineering Series. Peter Peregrinus Ltd., London, 1992.

[243] K. Watanabe, K. Kobayashi, and K. C. Cheok. "Absolute Speed Measurement of Automobiles from Noisy Acceleration and Erroneous Wheel Speed Measurements". SAE Paper 920644, 1992.

[244] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[245] P. J. Werbos. "Backpropagation Through Time : What it does and how to do it". *Proc. IEEE*, 78(10):1550–1560, October 1990.

[246] B. Widrow and M. E. Hoff. "Adaptive Switching Circuits". *IRE WESCON Convention Record*, pages 96–104, 1960. In "Neurocomputing", J. A. Anderson and E. Rosenfeld editors.

[247] B. Widrow and M.A. Lehr. "Thirty Years of Adaptive Neural Networks". *Proceedings IEEE*, 78(9):1415–1441, September 1990.

[248] B. Widrow and F. W. Smith. "Pattern-Recognition Control Systems". In *Proceedings of the Symposium on Computer and Information Sciences (COINS)*, pages 288–317, Washington, 1963.

[249] D.E. Williams, B. Friedland, and A.N. Madiwale. "Modern Control Theory for Design of Autopilots for Bank-to-Turn Missiles". *Journal of Guidance, Control and Dynamics*, 10(no. 4):pp. 378–386, 1987.

[250] R. J. Williams and D. Zipser. "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks". *Neural Computation*, 1(2):270–280, 1989.

[251] J. Y. Wong. *Theory of Ground Vehicles.* John Wiley and Sons, 1978.

[252] B. E. Ydstie. "Stability of the Direct Self-Tuning Regulator". In P. V. Kokotović, editor, *Foundations of Adaptive Control.* Springer-Verlag, 1991.

[253] E. C. Yeh, C-Y. Kuo, and P-L. Sun. "Conjugate Boundary Method for Control Law Design of Anti-Skid Brake System". *International Journal of Vehicle Design*, 11(1):40–62, 1990.

[254] R. Żbikowski and P. J. Gawthrop. "A Survey of Neural Networks for Control". In K. Warwick, G. W. Irwin, and K. J. Hunt, editors, *Neural Networks for Control and Systems*, chapter 3, pages 31–50. Peter Peregrinus, 1992.

[255] J. W. Zellner. "An Analytical Approach to Antilock Brake System Design". SAE Paper 840429, 1984.

[256] J. Zhang and S. Lang. "Explicit Self-tuning Control for a Class of Non-linear Systems". *Automatica*, 25(4):593–596, 1989.

[257] M. Zhihong and M. Palaniswami. "Robust Tracking Control for Rigid Robotic Manipulators". *IEEE Transactions on Automatic Control*, 39(1):154–159, January 1994.

[258] J. M. Zurada. *Artificial Neural Systems.* West Publishing Company, 1992.