Advanced Neural Network Controllers and Classifiers Based on Sliding Mode Training Algorithms

A thesis submitted by

Van Minh Tri Nguyen

in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

from

The University of Technology, Sydney

2006

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

Production Note:
Signature removed prior to publication.

ACKNOWLEDGMENTS

I would sincerely like to thank the following people who have been of help in the undertaking of my doctoral research.

First and foremost, my supervisor, Professor Hung Tan Nguyen, who has been a great and good teacher in my life. Professor Hung not only gave me invaluable guidance in my research, but also supported me in the best possible way for my spirit. My knowledge and experience over the past three years have been obtained, to a great degree, from Professor Hung, to whom I am very grateful.

Next, my co-supervisor, Associate Professor Quang Phuc Ha, who helped me a great deal and taught me with his expert knowledge of sliding mode control. I am grateful for his challenging feedback, which has improved my research career.

Thank you to Pat Skinner, for her editorial support in the writing of my draft. Her attention to detail and suggestions taught me how to write carefully and academically. I can say that she has made an integral contribution to the completion of my thesis.

Thanks also to my good colleague and friend, Phillip Taylor, for his invaluable help, his expertise in collecting, processing and disseminating the experimental data of the head movement commands for wheelchair control; to Russell Nicholson for his expert guidance in the control laboratory; to all the technical and administration staff in the Faculty of Engineering, for their prompt and helpful support.

I owe sincere thanks to my parents, Nguyen Luan and Luong Thi Buoi, for their encouragement and useful advice during these challenging years. I pursued this work for my parents, who have supported me and expected my success since I was a child. To my siblings, Tinh, Tien, Thuy and Truong, thanks for their good humour and moral support.

Above all, thanks to my dear wife, Quynh Tran, who has always shared all my difficulties and feelings over the last trying years. I am deeply grateful to my wife. To my daughter, Nha Tran, who was born while I was undertaking the research and significantly changed my thought: I dedicate this work to you.

TABLE OF CONTENTS

LIST OF SYMBOLS

N	number of input nodes
K	number of hidden nodes
M	number of output nodes
P	number of patterns in the data set
$\overline{W}_{nk}$	network weight between input node n and hidden node k
W_{km}	network weight between hidden node k and output node m
$\overline{o}_k$	output of hidden node k
o_m	output of output node m
$\mathbf{x}^p$	input vector of pattern p
$\mathbf{d}^p$	target output vector of pattern p
f	activation function
f'	first derivative of the activation function f
E	criterion function of the network error
$\mathbf{w}$	vector of all the network weights
$\frac{\partial E}{\partial \mathbf{w}}$	gradient vector
s	sliding function
$\|\cdot\|$	Euclidean norm for a vector and Frobenius norm for a matrix
$\mathbf{A}$	system matrix
$\mathbf{B}$	input matrix
$\mathbf{C}$	output matrix
$\mathbf{x}$	system state vector
$\mathbf{T}$	transformation matrix
$\mathbf{H}$	parameter vector of the sliding function
$\mathbf{F}, \mathbf{G}$	a pair of controllable canonical matrices
$\mathbf{P}, \mathbf{Q}$	a pair of matrices in the Lyapunov equation
u	control input
r	reference input

Subscripts

ω	speed subsystem
x	tension subsystem

Greek Letters

α	momentum coefficient
η	learning rate
μ	a positive scalar
ε	robust learning rate

LIST OF ABBREVIATIONS

ABP	Adaptive BackPropagation
Adaline	Adaptive linear element
BP	BackPropagation
CFSMBP	Chattering-Free Sliding-Mode BackPropagation
CG	Conjugate Gradient
DBD	Delta-Bar-Delta
DWM	Deterministic Weight Modification
EABPM	Extended Adaptive BackPropagation with Momentum
FNN	Feedforward Neural Network
GA	Genetic Algorithm
GN	Gauss–Newton
GOTA	Globally Optimal Training Algorithm
IAMSS	Iterated Adaptive Memory Stochastic Search
IEEE	the Institute of Electrical and Electronics Engineers
LM	Levenberg–Marquardt
MGFPROP	Magnified Gradient Function Propagation
MLP	MultiLayer Perceptron
NN	Neural Network
QN	Quasi-Newton
QuickProp	Quick Propagation
RMBP	Reach Mode BackPropagation
RPROP	Resilient Propagation
SARPROP	Simulated Annealing Resilient Propagation
SISO	Single-Input Single-Output
SMC	Sliding Mode Control
SuperSAB	Super Self-Adapting Backpropagation
SVC	Static VAR Compensator
TRUST	Terminal Repeller Unconstrained Subenergy Tunnelling
VSS	Variable Structure System

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis presents the research undertaken to develop some novel learning algorithms based on the sliding-mode control techniques for the neural network classifiers and controllers. Although the feedforward neural network with the backpropagation learning algorithm is the most widely used approach for classification and control applications, the slow convergence rate, the local minima problem, the difficulties in system identification and the lack of robustness are the issues existing for these neural network-based systems. The combination of the sliding-mode control techniques and the backpropagation algorithm, as described in this thesis, leads to three novel learning algorithms, which offer effective solutions for these problems.

The first learning algorithm, derived from the integration between the chattering-free sliding-mode control technique and the backpropagation algorithm, can obtain fast and global convergence with less computation. Experiment results relating to the head-movement neural classifier for wheelchair control show that the proposed approach considerably improved the convergence speed, global convergence capability and even the generalisation performance of the neural network classifier, in comparison with various popular learning algorithms.

The second learning algorithm, also derived from the integration between the chattering-free sliding-mode control technique and the backpropagation algorithm, can guarantee the stability and robustness of the neural control system with parameter uncertainties. Based on this stable neural controller, a neural control design methodology is developed for a class of uncertain nonlinear systems with transportation lag, wherein a new training procedure is proposed to avoid the difficult choice of the training inputs always associated with the conventional neural network identifier. The implementation results with a real-time Static VAR Compensator system indicate the effectiveness of the proposed method.

The third on-line learning algorithm, developed from the reaching law method combined with the backpropagation algorithm, offers a robust adaptation approach for the neural control systems with parameter uncertainties and disturbances. The neural control approach is further developed to design a novel decentralised neural controller

for a class of uncertain large-scale systems with bounds of interconnections and disturbances. The stability and robustness of the neural control system are guaranteed based on the Lyapunov synthesis. Real-time implementation results for a Coupled Electric Drives CE8 system show the effectiveness and feasibility of the proposed approach.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Nowadays, soft computing is being applied more and more frequently in commerce and industry (Bonissone et al., 1999). One important component of soft computing is the artificial neural network (NN), and the most widely used neural network architecture is the multilayer feedforward (Haykin, 1995). For training the multilayer feedforward neural network (FNN), the backpropagation (BP) algorithm based on the gradient descent method is the most popular approach. The FNNs with the BP learning algorithm have been used successfully in a variety of industrial applications such as pattern recognition, classification, optimisation, modelling, identification, and control (Meireles et al., 2003). However, there are many problems associated with the BP algorithm. Its major drawbacks are the uncertainty of finding a global minimum of the error criterion function and the excessively long time for training the FNNs. In addition, the BP algorithm is sensitive to perturbations in the noise environments. Therefore, the robustness of the learning algorithm has recently become a major issue.

In the last 20 years, a significant number of different learning algorithms has been developed to improve the BP performances. Some attempts focusing on the dynamic change of learning rate, the addition of a momentum term, the selection of a better energy function, or the modifications of the slope of the activation function have been undertaken. These approaches, however, did not result in significant improvement in the convergence rate and the global convergence capability. A number of second-order gradient methods, such as the Levenberg-Marquardt, quasi-Newton, conjugate gradient and Quickprop algorithms, can remarkably improve the convergence speed, but local minima problems still exist in these methods.

To overcome the problem of local minima, stochastic global optimisation approaches have been applied to the neural networks' training algorithm. The random optimisation method, the genetic algorithm, and the simulated annealing optimisation technique are

some popular directions of the stochastic methods. However, the stochastic approaches sometimes require a very long time to reach a global optimal solution. Other global optimisation methods are the deterministic approaches that can effectively find the global minima. Nevertheless, the computational complexity for these learning algorithms with a global convergence is very high.

One of the powerful tools for tackling the above problems is the combination of the variable structure system (VSS) strategy and the BP learning algorithm (Kaynak et al., 2001). Through the use of the VSS principle, the convergence and robustness properties of the gradient-based learning algorithm are significantly improved. Several pioneering studies using VSS theory in the training of NN weights are reported in the literature. Based on a linear "sliding surface" in Slotine and Sastry (1983), Parlos et al. (1994) proposed his adaptive backpropagation (ABP) algorithm. In the function approximation applications, the ABP algorithm outperforms the BP and QuickProp approaches (Fahlman, 1988) in terms of speed and chance of convergence. Parma et al. (1999) introduced a sliding mode backpropagation algorithm that can speed up the BP algorithm in the function approximation problems. However, these schemes have not been implemented in the pattern recognition and classification applications. Giordano et al. (2004) developed an adaptive learning approach based on the sliding mode control strategy, which guarantees the finite time reachable condition of a zero network error. Nevertheless, this approach was only applied in a neural identifier.

The nonlinear mapping and learning properties of FNNs provide an attractive capability for applying the networks to control systems. After the first IEEE (the Institute of Electrical and Electronics Engineers) conference on neural networks in 1987, a remarkable increasing amount of conference and journal papers was published relating to the neural control systems. These papers showed the significant capability of NNs for dealing with the system nonlinearity, and a number of new NN control structures were also proposed. However, some disadvantages exist in these neural control systems. When the networks are trained to identify nonlinear dynamic systems and their inverses, it is very difficult to choose the appropriate training inputs covering the operating range of the controlled systems. In addition, most of the neural control systems using the on-line learning capability of FNNs still lack rigorous stability guarantees.

Recently, several excellent NN control approaches have been proposed based on Lyapunov's stability theory. One main advantage of these schemes is that the adaptive laws for the multilayer FNNs are derived based on the Lyapunov synthesis and therefore guarantee the stability of the controlled systems without the requirement for off-line training. Some adaptive laws proposed by Jagannathan (2001) and Hayakawa et al. (2005) can actually be classified into the group of the sliding-mode-based learning algorithms. However, these Lyapunov-like methods still encounter difficulties in real-time applications.

In this thesis, a number of novel NN learning algorithms based on the sliding mode techniques are developed. A combination of the chattering-free sliding-mode control scheme and BP produces a fast and globally converged algorithm, which can be applied in the binary problems. Similarly, an on-line learning algorithm is proposed and applied in the neural controller of a class of uncertain single-input single-output (SISO) systems. A novel training scheme is developed for this neural control system so that the difficult choice of the training input signal is eliminated. Another proposed on-line update rule for the FNNs is based on the reaching law method of the VSS theory. A decentralised neural controller using this learning rule can guarantee the stability of a class of large-scale systems with interconnections and disturbances. All the proposed learning algorithms are successfully implemented in real-time systems.

1.2 Research Objectives

The primary objective of the research described in this thesis was to develop the learning algorithms of FNNs based on the VSS theory in order to overcome some problems associated with the neural network classifiers and controllers using the backpropagation learning algorithm. Specifically, the proposed objectives were:

(i) The development of a system with:

1) a novel training algorithm based on the sliding mode technique which assure fast and global convergence of the FNNs, together with

2) a head-movement classifier using FNN and the proposed learning algorithm.

(ii) The development of a system with:

1) a neural controller structure for a class of uncertain SISO systems, together with

2) a novel training algorithm based on the sliding mode technique which guarantees the system stability, and

3) a new training framework of the neural network controller which avoids the difficult choice of the training inputs always associated with the conventional training procedure for system modelling.

(iii) The development of a system with:

1) a novel decentralised controller structure using neural networks for a class of uncertain interconnected nonlinear systems, and

2) a sliding-mode-based learning algorithm which guarantees the stability of the controlled large-scale systems.

The thesis describes the outcome of the research undertaken to fulfil these objectives. Additionally, some minor researches closely associated with the objectives have been undertaken. They include:

. A review of the literature on neural networks from a historical perspective, the BP modifications, and the neural control systems

. Critical comparisons of the new methods with other conventional methods.

. Application details of the proposed neural controllers in some real-time systems including the Static VAR Compensator system and the Coupled Electric Drives system.

. Application details of the neural network head-movement classifier for the wheelchair control system.

1.3 Organisation of Thesis

Chapter 1 briefly introduces artificial neural networks concepts and their problems. Some problems in the NN-based systems can be solved by using VSS theory in the NNs' learning algorithms. This is the objective of the research.

Chapter 2 contains the literature review section. The historical perspective of artificial neural networks is first introduced. Some developments in the backpropagation algorithm for improving convergence rate and global convergence are then reviewed. Furthermore, an overview of the neural control systems based on nonlinear mapping and adaptive learning properties of FNNs is presented. This section also reveals some limitations of the NN control and classification systems and further formulates the research aims.

Chapter 3 introduces some background about the NN learning algorithms and the VSS theory. The BP algorithm and its modifications are classified in six branches, including dynamic learning rate methods, second-order gradient methods, activation function variations, BP with momentum, criterion function variations, and sliding-mode-based approaches. Some discussions on the advantages and disadvantages of these algorithms are presented at the end of this chapter.

A number of novel training algorithms based on the sliding-mode control techniques are proposed in Chapter 4. Two algorithms, named CFSMBP and RMBP, are developed for the neural controller, while the other algorithm, named EABPM, is proposed for the neural classifier. All algorithms are globally converged in the sense of the Lyapunov stability method. Some simulation results with the XOR problem and control problem for an uncertain continuous linear system can illustrate the methods' effectiveness. The proposed EABPM learning algorithm is further developed and tested in a head-movement neural classifier for wheelchair control.

Chapter 5 develops the neural controller design methodology for a class of uncertain SISO system. A sliding function is first defined and is used as the network training error. The proposed CFSMBP learning algorithm for the neural controller is proven to guarantee the system stability. A training procedure for the neural controller is then

developed. Finally, the neural controller is applied in a real-time Static VAR Compensator system.

In Chapter 6, a decentralised neural controller structure is designed for a class of large-scale systems with bounds of interconnections and disturbances. Each neural network controller using a proposed RMBP learning algorithm can stabilise each subsystem of the large-scale systems. Some real-time experiments in a Coupled Electric Drive system are implemented to validate the designed method.

Chapter 7 concludes the thesis with a summary and some possible future research directions.

CHAPTER 2

LITERATURE REVIEW

2.1 Historical Perspective

The modern concept of artificial neural networks (NNs) began when McCulloch and Pitts (1943) introduced the first mathematical model of a biological neuron. Then Donald Hebb (1949) proposed one of the first learning rules for the neurons. In 1958, Rosenblatt established a first learning network, named Perceptron, which could be trained for pattern recognition (Rosenblatt, 1958). Two years later, Widrow and Hoff (1960) proposed an adaptive linear element, called Adaline. Based on the least mean square learning rule, Adaline was successfully applied for cancelling the echo phenomenon in telephone lines. This is considered to be the first industrial application of NNs. However, Minsky and Papert (1969) mathematically proved that the Perceptron could only solve a limited class of problems and predicted a dead end of the neural network field.

Influenced by Minsky and Papert, during the 1970s, only a few pioneering works on NNs were undertaken. Kohonen (1972) and Anderson (1972) independently proposed the mathematical model for associative memory trained by the Hebb rule. In 1974, Werbos originally developed the backpropagation algorithm for feedforward neural networks, and demonstrated their ability to estimate a social communication model (Werbos, 1974). Albus (1975) introduced the "Cerebellar Model Articulation Controller" networks based on his view of human memory models. Grossberg (1976) investigated self-organising networks derived from the human visual systems. In the early 1980s, Hopfield (1982) introduced the first model of recurrent neural networks, which could be implemented by integrated circuit hardware, and Paker (1982) independently discussed the BP algorithm for training feedforward neural networks (FNNs). The BP algorithm was then reinvented and made popular by Rumelhart and McClelland (1986).

After Rumelhart and McClelland answered the criticism of Minsky and Papert, a dramatic increase of interest in NNs occurred. The Institute of Electrical and Electronics Engineers (IEEE) organised the first conference on neural networks in 1987, and the International Neural Networks Society (INNS) was concurrently formed. The first journal of INNS, *Neural Networks*, appeared in early 1988. Another IEEE Neural Networks Society was then formed with one of its publications, *IEEE Transaction on Neural Networks*, commencing in 1990. In the late 1980s to early 1990s, many developments in NNs occurred. Kosko (1987) developed an adaptive "Bi-directional Associative Memory" using the Hebb learning law. Broomhead and Lowe (1988) first introduced "radial basic function networks" under the method of potential function. Other novel NN structures such as the "Boltzmann Machine" (Hinton & Sejnowski, 1986), the "B-splines networks" (Moody, 1989), "functional-link networks" (Chen & Billings, 1992) and "wavelet networks" (Zhang & Benveniste, 1992) were also proposed.

The 1988 DARPA Neural Network study (1988) listed a number of NN applications in commercial applications and motivated the use of neural networks in real life. Since then, neural networks have been applied in many fields such as aerospace, automotive, banking, defence, electronics, entertainment, financial, insurance, manufacturing, medical, robotics, speech, securities, telecommunications, transportation and others. Currently, the multilayer perceptrons (MLPs) are the most used structure in commercial and industrial applications. As shown in the study by Haykin (1995), the percentages of network utilisation are 81.2% for MLPs, 5.4% for Hopfield, 8.3% for Kohonen and 5.1% for the others.

For training FNNs, a group of MLPs, BP is currently the most popular algorithm among others. The idea of increasing the learning robustness of BP in noise environment was first considered in the late 1980s. Hanson and Burr (1988) introduced the Minkowshi-r BP algorithm, and showed that the effect of noise in the target domain is dramatically reduced by using small power values. In White (1989), the linear output error is replaced by a nonlinear suppressor function so that the proposed learning law is insensitive to small perturbations in the underlying probability distribution of the training set. All the above researchers eventually used a signum function of the network

error in their learning algorithms, which are similar to the sliding-mode control schemes.

The variable structure systems (VSSs), with the sliding-mode control (SMC) scheme first proposed in the 1950s (Emelyanov, 1959; Itkis, 1976; Utkin, 1977), are very famous for their robust property. Other contributions to VSS theory have been made during the last decade. In Gao and Hung (1993), the reaching law approach, a general method for designing variable structure controllers, was first presented. The chattering-free sliding-mode control approach (Ertugrul et al., 1995) was developed by combining VSSs and Lyapunov designs. Another integral sliding mode control technique, which guaranteed the robustness of the motion in the whole state space, was further proposed by Utkin and Shi (1996). The use of VSS theory in the learning process was first reported in Zak and Sira-Ramirez (1990), whereby a switching weight adaptation strategy was proposed to impose a discrete-time asymptotically stable linear learning error dynamics. In Parlos et al. (1994), an adaptive BP algorithm based on the choice of a linear sliding surface was proposed and accelerated the convergence rate of BP. In Sira-Ramirez and Morles (1995), a dynamic SMC approach was also proposed for robust adaptive learning in analogue adaptive linear elements (Adalines). Giordano et al. (2004) recently developed the robust learning approach of Sira-Ramirez and Morles for the FNNs, which was successfully applied in a real-time neural network identifier.

Figure 2.1 presents an overview of NN history and VSS techniques, focusing on some milestones in the development of NNs and their learning algorithms.

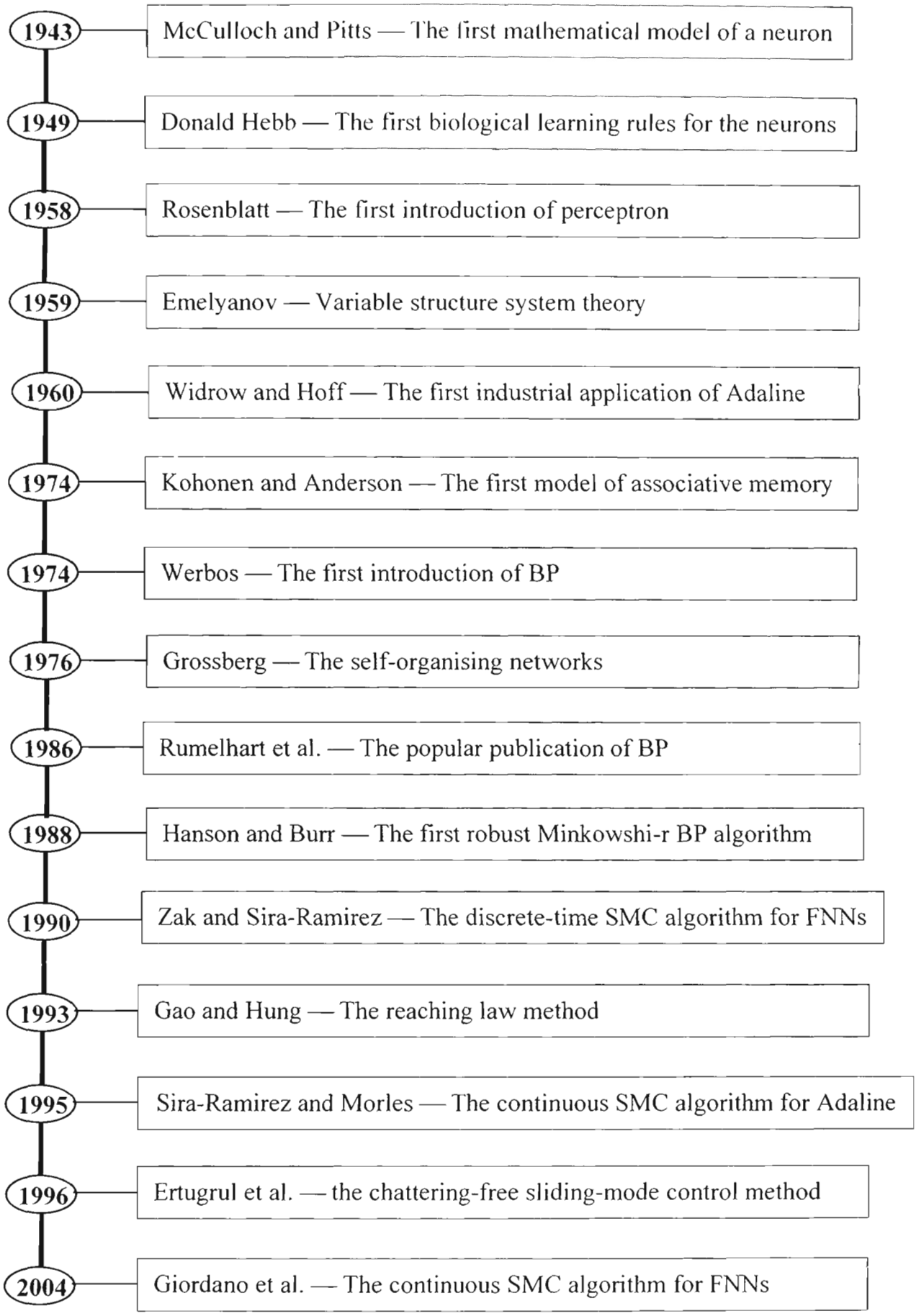


Figure 2.1: Milestones in the development of neural networks.

2.2 Backpropagation Algorithm Developments in Classification

Without doubt, the most popular NN structure is FNN, and the BP is the most widely used learning algorithm in training FNNs. FNNs have successfully been applied in some real problems such as pattern recognition and classification. Despite the general success of BP in training FNNs, there are some major drawbacks that need to be overcome. Firstly, BP will converge to a local minimum of the error function, and fail to find a global optimal solution. The second shortcoming of the BP algorithm is its slow convergence rate. Therefore, the FNN with BP algorithm is sometimes not accepted in real-time systems. Moreover, the limitations of the training algorithms can also influence the network's generalisation ability. However, this problem is being addressed by other researchers in my group.

Recently, many researchers have focused on improving the performance of BP. The following review is divided into two major topics:

1. BP modifications for improving the convergence rate.

2. BP modifications for global convergence.

2.2.1 BP modifications for improving the convergence rate

The BP algorithm based on the steepest-descent technique performs poorly in terms of convergence speed in high dimension and to a fixed step length (Fletcher, 1987). A large learning rate, that is, faster convergence rate, is allowed in some flat regions of the error surface, while in the high slope regions it causes oscillation and divergence of the algorithm. A simple approach to speed up BP is through the addition of a momentum to the weight update rule (Plaut et al., 1986). Thus for a flat region, the momentum leads to increasing the learning rate by a factor $\frac{1}{1-\alpha}$, where $0 < \alpha < 1$ is the momentum coefficient (Tollenaere, 1990). On the other hand, in a region of high fluctuation, the momentum effect vanishes. However, this method did not significantly improve the convergence rate of BP.

Dynamic learning rate methods were also proposed to speed up the algorithm. Vogl et al. (1988) and Battiti (1989) pioneered in adjustments to the learning rate. In their "bold driver" techniques, the value of the square error E is monitored after a weight update. The learning rate is grown exponentially if the value of E decreases. On the other hand, if the value of E increases, the last weight update is rejected and the learning rate is decreased exponentially. Weir (1991) proposed a method for a self-determination adaptive learning rate. This method uses the sign of the energy-weight gradient for computing the optimal learning rate. These methods are actually faster than the BP with momentum, but the improvement in convergence speed is still limited.

Because the characteristic of the error surface depends on every weight dimension, some other researchers applied an individual dynamic learning rate for each weight in the network. Jacobs (1988) introduced the Delta-Bar-Delta (DBD) rule which utilises the sign change of an exponential averaged gradient. Each learning rate is linearly increased if the parameter change has been in the same direction between two iterations. Otherwise, if the direction of the parameter change alternates, the learning rate is exponentially decreased. The SuperSAB (Tollenaere, 1990) and RPROP (Riedmiller & Braun, 1993) methods are also based on the idea of an independent learning rate adaptation. The basic change is to increase the learning rate exponentially instead of linearly, as with the DBD method. The experimental studies by these authors have shown that the individual adaptive learning rate algorithms considerably speed up the learning process. However, these methods require the selection of many added parameters, which affects the convergence speed. Moreover, variable learning rate BP algorithms often become trapped in local minima.

The above variable learning rate BP algorithms attempt to change the learning rate with regard to widely differing eigenvalues of the Hessian matrix. By directly using the Hessian matrix, Newton methods can significantly improve the convergence speed of BP. This algorithm converges quickly if the search region is quadratic or nearly so (Bishop, 1995; Cichocki & Unbehauen, 1993). However, the Newton method requires $P \times W^2$ steps, where W is the number of weights in the network and P is the number of patterns in the data set. Also, the Hessian matrix must be inverted, which requires W^3 steps. With this high computational expense, the Newton method is therefore not commonly used.

To overcome this problem, several methods have been proposed to approximate the Hessian matrix. Becker and Le Cun (1989) proposed the Gauss–Newton method whereby the off-line elements of the Hessian matrix are neglected. Facing another problem of the Newton method when the Hessian matrix is not positive at every point in the error surface, the Gauss–Newton approach may converge towards a maximum or a saddle-point rather than a minimum. By adding a positive-definite symmetric matrix to the Gauss–Newton approximation of the Hessian matrix, the Levenberg–Marquardt (LM) method (Hagan & Menhaj, 1994) can overcome this difficulty. The study results of Hagan et al. (1995) showed that the LM algorithm is one of the fastest algorithms for training FNNs of moderate size. However, because it neglects off-diagonal Hessian terms, it is not able to rotate the search direction as in the exact Newton's method. Alternative approaches (Battiti & Masulli, 1990), known as quasi-Newton (QN) methods, compute an approximation of the inverse Hessian instead of calculating the Hessian directly and then evaluating its inverse. The problem arising from Hessian matrices which are not positive definite are also solved in the QN methods (Barnard, 1992). These methods, however, require large memory storage, for example, a size of $W \times W$ for the QN method and a size of $M \times P \times W$ for the LM method, where M is the number of output nodes. For networks with more than a few thousand weights, this could lead to prohibitive memory requirements.

Another linear searching approach to speed up the learning process is represented by the conjugate gradient (CG) methods (Hert et al., 1991; Moller, 1993). They can be regarded as a form of BP with momentum, in which the learning rate and momentum coefficient are determined automatically at each iteration without explicit knowledge of the Hessian matrix. The conjugate gradient techniques require only W storage, which is better than the QN and LM algorithms. In a similar manner, the QuickProp algorithm (Fahlman, 1988) is a variation of BP with a dynamic momentum that is viewed as an approximation of Newton's rule. Results from applications of the QN, CG and QuickProp methods (Barnard, 1992; Fahlman, 1988; Makram-Ebeid et al., 1989) show that these approaches are the fastest algorithms for training FNNs. The main disadvantage of these second-order gradient methods is the local convergence problem.

Another reason for the slow convergence is the premature saturation of the derivative of the activation function, sometimes referred to as the "flat-spot" problem (Lee et al.,

1993). Some modifications had been suggested to eliminate the flat-spot problem in order to accelerate the convergence speed. Fahlman (1988) suggested altering the derivative of the activation function directly by adding a constant 0.1 to the derivative term. Vitela and Reifman (1997) further set the value of the derivative to a constant value when the activation slope fell in predefined saturation regions. In the MGFPROP method, Ng et al. (2004) magnified the activation derivative by using a power factor $\frac{1}{s}$, where $s \geq 1$. Therefore, the derivative of the activation function is scaled up to speed up the convergence rate. Baum and Wilczek (1988) and Ooyen and Nienhuis (1992) chose a better energy criterion so that the flat-spot problem has been eliminated for the output units. However, the derivative of the activation function still appears for the hidden units. Therefore, the flat-spot problem is only partially solved by employing the entropy criterion. With these modifications, the convergence speed can be improved, but local minima problems still exist.

Parlos et al. (1994) claimed that the slow convergence of BP is due to a small error gradient in the vicinity of a local minimum. Then they introduced an adaptive backpropagation (ABP) algorithm, in which the learning rate is a specific function of the error and of the error gradient in order to accelerate the network convergence. However, Parlos et al. only successfully applied ABP in function approximation. They reported that ABP was found to be as fast as the BP in the binary problems, such as the parity problem or the encoder problem. This leads to fewer studies on ABP in the neural network literature.

2.2.2 BP modifications for global convergence

Another drawback of BP is the local convergence problem. Often trapped in local minima, the BP algorithm usually results in poor performance. There are a number of global optimisation algorithms which can overcome the problem.

The stochastic global optimisation method is one of the attractive solutions. Replacing BP with random optimisation methods (Matyas, 1965; Solis & Wets, 1981), Baba (1989) showed that the new algorithms often successfully found the NN's weights which gave a global minimum of the error function. Another stochastic minimisation algorithm, called IAMSS (iterated adaptive memory stochastic search), characterised by

the use of an adaptive memory in Brunelli (1994), was superior to BP in binary problems. The genetic algorithm (GA), a global optimisation method, has also been introduced to train the neural networks (Whitley et al., 1990). GA often assures the networks' success in searching and converging to the global minimum. Nevertheless, GA can suffer from excessively slow convergence before proving an accurate solution. SARPROP, proposed by Treadgold and Gedeon (1998), is a combination of RPROP (Riedmiller & Braun, 1993) with the global search technique of simulated annealing (SA) (Szu, 1987) in order to maintain quick convergence to the global minimum. The SARPROP remarkably improves both the convergence rate and the global convergence capability of the BP algorithm. However, these stochastic methods theoretically converge to a global optimal solution in a limited space. For example, when the input space is expanded, the random optimisation methods and IAMSS algorithm fail to converge to the global minimum, and the evolutionary-gradient hybrid method (Salomon, 1998) and SARPROP exhibit the same deficiencies as the steepest-descent methods in some problems. In addition, there are no guarantees that the stochastic approaches would reach the global minimum in finite time.

Several deterministic methods which are more efficient than BP have been proposed in literature. Cetin et al. (1993) replaced the batch BP by a global-descent rule. This methodology is based on a global optimisation scheme, acronymed TRUST (terminal repeller unconstrained subenergy tunnelling). Although the global descent approach always guarantees to find the global minima for the function of one variable, the same result cannot be obtained for multivariate functions. Tang and Koehler (1994) used a branch-and-bound based on Lipschitz optimisation methods and developed globally optimal training algorithms (GOTA). However, the effectiveness of GOTA is only improved by using dynamically computed local Lipschitz constants over subsets of the weight space. Ng et al. (2004) used deterministic weight modification (DWM) which reduces the system error by changing the weights of an FNN in a deterministic way. The integration of DWM and MGFPROP can outperform other modified BP algorithms for most of the adaptive problems in terms of the convergence rate and global convergence capability. Nevertheless, the high computational complexity is the major shortcomings for these global convergence learning algorithms.

2.3 Neural Control Developments

From the control theory viewpoint, there is no general controller design methodology for the great diversity of nonlinear systems. In recent years, the NN-based nonlinear control technique has attracted the interest of an increasing number of researchers. The most attractive properties of neural networks in the control field are their nonlinear mapping and learning capabilities. The NNs' ability to represent nonlinear mappings, that is, to model nonlinear systems, is the feature that is most exploited in the synthesis of nonlinear controllers. Moreover, the training process of NNs to minimise an energy criterion function also opens an on-line adaptation region for nonlinear control theory. The following overview of the neural control field is divided into two areas:

1. Neural controllers based on nonlinear mapping.

2. Neural controllers based on adaptive learning.

2.3.1 Neural controllers based on nonlinear mapping

Following this direction, the nonlinear functional mapping properties of NNs are central to their use in control. A number of results have been published showing that an FNN can approximate any nonlinear function with an arbitrarily desired degree of accuracy (Cybenko, 1989; Funahashi, 1989; Hornik et al., 1989). To be specific, these papers proved that a continuous function can be arbitrarily well approximated by an FNN with only a single hidden layer, where each unit in the hidden layer has a continuous sigmoidal nonlinearity. Another group of MLPs, the radial basic function NNs, have also been proven to have the best approximation property (Broomhead & Lowe, 1988; Poggio & Girosi, 1990). However, the number of nodes needed in radial basic function NNs increases with the dimension of the input space (Weigand et al., 1990), which leads to the necessary consideration of the cost of using this approach.

Training an NN using input-output data from real-time nonlinear plants can yield nonlinear dynamical systems and their inverses. The procedure of training an NN to represent the forward dynamics of a system is introduced in Narendra and Parthasarathy (1990). The NN model is often placed in parallel with the system and the error between

the system and network output is used as the network training error. Some useful guidelines for choosing the training input signal may be found in Hagan et al. (2002). Psaltis et al. (1988) also introduced a generalised inverse learning scheme to obtain the inverse model of dynamical systems. In this approach, the inverse model network precedes the system and receives input from the system output. A synthetic training signal is introduced to the plant to obtain a corresponding output, and the network is trained to reproduce its output as the plant output. Another approach to inverse modelling is known as specialised inverse learning (Psaltis et al., 1988). This learning structure contains a neural inverse model preceding the plant and a trained forward model of the system placed in parallel with the plant. The error signal for the training algorithm is the difference between the training input and the forward model output. However, the training input signal must be chosen over a wide range of system inputs and the actual operational inputs are very difficult to define a priori (Hagan et al., 2002; Jordan & Rumelhart, 1992). This is the main drawback of the above training procedures for forward and inverse modelling of dynamical systems.

Neural network models of dynamics systems and their inverses have been subsequently utilised for control. In the literature on NN architectures for control, a large number of control structures have been proposed and used. Direct inverse control, neural internal model control, neural predictive control and neural feedback linearisation control are some popular architectures, among others.

The neural predictive control scheme optimises the plant response over a specified time horizon (Hagan et al., 2002; Soloway & Haley, 1996). This architecture requires a NN plant model, a NN controller, a performance function to evaluate system responses, and an optimisation procedure to select the best control input. The optimisation procedure, however, can be computationally expensive.

An internal model control consists of a NN controller, a NN forward plant model and a robust filter (Hunt & Sbarbaro, 1991; Varshney & Panigrahi, 2005). The difference between the system and the model outputs is used as feedback input to the robustness filter, which then feeds into the NN controller. The NN forward plant model and the NN controller, that is, the inverse plant model, can be trained off-line using data collected from plant operations. As mentioned above, the data set for training the neural model

may not cover a sufficiently large operational range, thus causing poor or even unstable control performance.

A neural feedback linearisation structure uses two trained FNNs to approximate the unknown nonlinear plant (Jin et al., 1993). After cancelling the nonlinear part by feedbacking the NN outputs, a linear state feedback controller can be implemented. There are several variations on the neural feedback linearisation controller (Breemen & Veelenturf, 1996; Hagan et al., 2002). Nevertheless, system performance depends on the accuracy of NN approximation, and is often unsatisfied.

Direct inverse control utilises an inverse system model that cascades with the controlled system. Thus the network acts directly as the controller (Miller et al., 1990) without a carefully trained NN plant model. The significant problem of this scheme is the lack of robustness which can be attributed primarily to the absence of feedback. Ichikawa and Sawa (1992) developed a direct feedback neural controller and a genetic training algorithm for optimisation of the NN parameters in terms of various evaluations. Although the controlled system, in simulations, can face noise, parameter changes and nonlinearities, the approach is difficult to implement in real-time systems. Recently, direct feedback inverse control scheme has been utilised in Nayeri et al. (2004) and Daosud et al. (2005), but the difficulty in choosing the training input signal over the operational range of system inputs is the main disadvantage of this approach.

2.3.2 Neural controllers based on adaptive learning

Most of the NN control approaches in literature have utilised the on-line learning capability of NNs, that is, the adaptive learning approach. Since the real-time system often includes time-varying parameters and unpredicted disturbances, the adaptive neural control can enhance the robustness properties of the controlled systems.

An early "backpropagation-through-time" technique was proposed by Nguyen and Widrow (1990), in which the plant dynamics can be learned off-line by a neural network identifier, and the output error is back-propagated through the plant identifier to obtain an equivalent error for training the neural controller. Recent developments of this approach are presented in Da (2000) and Da and Song (2003), in which a sliding

mode regulator of system error is passed back through the neural identifier for tuning the neural controller parameters. These studies, however, lack rigorous stability guarantees.

The model reference indirect adaptive control using NNs has also been presented by Narendra and Parthasarathy (1990). Here the desired performance of the closed-loop system is specified through a stable reference model. The error between the plant output and the reference model output is used for training the network acting as the controller. When the NN controller treats the plant as a part of the NN output layer, it faces the problem of unknown plant. Thus a forward model network has to be used to identify the input-output behaviour of the plant. The identification model is then used for computing the partial derivatives of the system output with regard to the controller parameters, also known as the plant Jacobian matrix. In a more synthetic framework, Ng (1997) also calculated the plant Jacobian from the neural model and showed the stability of the on-line learning neural controlled system. However, in the stability proof (Ng, 1997), the change in the Lyapunov function lacks the partial derivative of the Lyapunov function with regard to the NN's input states. Tanaka (1996) used the linear difference inclusion approach to analyse the stability of these system configurations, but the results were derived only for a few specific examples.

Instead of extracting the controller error from plant output error, the reinforcement learning control approach (Barto, 1990) attempts to determine target controller outputs that would lead to an increase in the measure of plant performance. The adaptive critic controller (Sutton & Barto, 1998) consists of two NNs. The first action network operates as an inverse controller and the second critic network predicts the future performance of the system. Reinforcement learning, an approximation of dynamic programming, is used to optimise future performance. However, the lack of a global stability proof is the major problem of these methods.

With the advantage of no NN plant model, direct adaptive neural control schemes have been developed to ease the unknown plant Jacobian. In Psaltis et al. (1988), a dynamics is treated as an unmodifiable layer of the NN, and the error is backpropagated through this dynamics. In the method proposed by Chen and Pao (1989), the system output error is transformed into the network output error using an inverse transfer matrix of the

system dynamics. Another approach (Kawato et al., 1987) used a fixed feedback gain stage to generate a transformed error signal for updating the NNs. Saerens and Soquet (1991) and Zhang et al. (1995) used the sign of the plant Jacobian in an attempt to eliminate the demand of a preceding learning stage. Venugopal et al. (1995) put a gain layer NN between the NN controller and the system. The Jacobian effects are therefore controlled by updating the gain layer weights. Recently, direct adaptive neural control schemes have been discussed in a new framework of sliding mode control approach (Efe et al., 2003; Nguyen et al., 2004b; Tsai et al., 2004). In Efe et al. (2003), control error is extracted from a prescribed reaching mode equation to avoid the plant Jacobian requirement. Nguyen et al. (2004b) used the sliding function mentioned in Tsai et al. (2004) for training the direct neural controller without involving the Jacobian matrix. However, the stability and error convergence have not been fully proven for these NN-based control systems.

Sanner and Slotine (1992) explained the instability of the above neural control systems as being because of the gradient descent on-line training methods. Thus a number of NN control approaches based on Lyapunov's stability theory have been proposed (Hayakawa et al., 2005; He, 2002; Hovakimyan et al., 2002; Jagannathan, 2001). In these schemes, the adaptive laws for the multilayer NNs are derived based on the Lyapunov method and therefore guarantee the stability of the controlled systems without the requirement for off-line training. Using the Lyapunov approach, Jagannathan (2001) proposed a novel weight updates of a multilayer NN controller so that the tracking error and the NN weight estimates are uniformly ultimately bounded. However, the proposed update rule is too complicated to applied in a real-time system. Hovakimyan et al. (2002) developed an adaptive output feedback control methodology for nonlinear systems using single-hidden-layer FNNs. A gradient-type learning algorithm for on-line updating the NN weights is proposed so that the tracking error and the weight estimate error are ultimately bounded. But such adaptive control schemes are still difficult to implement in real-time systems.

In He (2002), a feedback control algorithm is proposed that utilises a set of NNs to compensate for the effect of the system's nonlinearities, and a proposed learning algorithm for on-line updating the weight parameters of NNs can guarantee the Lyapunov stability of the control system. This learning algorithm is actually similar to

the robust backpropagation training algorithm with a dead zone, introduced in (Song et al. (1999). The selection of a small dead zone parameter will provide a small bound of the tracking error. However, the dead zone parameter can not be set to zero, because this may cause divergence of the NN in the present of disturbance (Song, 1998). Hayakawa et al. (2005) developed a state feedback neural adaptive control scheme for nonlinear uncertain nonnegative and compartmental systems. Using the Lyapunov-like methods, an update rule of the NN weights was proposed and guarantees ultimate boundedness of the error signal. In this on-line update algorithm, the sign of the output error is actually used, thus this algorithm can be classified into the group of the sliding-mode-based learning algorithms.

Other researchers basically developed the sliding-mode control scheme in the training of NNs (Efe & Kaynak, 2000; Giordano et al., 2004). In Efe and Kaynak (2000), an appropriate combination of BP and VSS theory leads to a novel update law which can benefit from the robustness property of the VSS approach and the error minimisation property of the BP algorithm. But the heuristic choice of many learning parameters is the obvious drawback of this algorithm. Motivated from the research of Ramirez and Morles (1995), Giordano et al. (2004) developed an adaptive learning approach based on sliding-mode control strategy which guarantees the finite time reachability of zero network error condition. Nevertheless, this approach is only applied in a neural identification scheme of a real-time industrial manipulator.

2.4 Discussion and conclusion

In this chapter, the NN history and the SMC techniques are introduced. The overview shows that the FNN is the most popular NN architecture and BP is the most used algorithm for training FNNs. However, some major drawbacks associated with BP such as local and slow convergence problems require further developments. A number of approaches, such as the BP with momentum, the dynamic learning rate methods, the use of entropy criterion or the activation function modifications, have been undertaken, but the convergence rate and global convergence capability are not significantly improved. Some second-order gradient methods, including the Levenberg-Marquardt, quasi-Newton, conjugate gradient and QuickProp algorithms, can lead to high convergence

speed. However, large memory requirements and local minima convergence are major limitations of these approaches.

Both stochastic and deterministic global optimisation methods are developed to overcome the local minima convergence problem. The random optimisation method, IAMSS, GA, and SARPROP often assure a convergence to the global minimum, but these stochastic approaches sometimes require a long training duration and a limit of input space. Other deterministic methods, named TRUST, GOTA or DWM, can effectively find the global minima. Nevertheless, the computational complexity for these learning algorithms with a global convergence is very high.

From the chattering-free sliding-mode control technique, an extended ABP algorithm which can maintain fast convergence of the error function is proposed in this thesis. In addition, a momentum term which stabilises the learning process is combined with the extended ABP to form a novel algorithm, named EABPM. The EABPM algorithm, also viewed as a deterministic method, is developed from the variable structure system theory. Theoretically, the error function in the EABPM method exponentially converges to zero in the sense of Lyapunov stability. EABPM therefore produces significant improvement of the convergence speed while ensuring global convergence of the training method. Experimental results in the head-movement classifier also show that the EABPM always quickly converges the system error to a small acceptable level.

For the control purpose, the nonlinear mapping and learning capability of NNs open bright prospects for nonlinear control developments, and a large number of NN control structures have been proposed in the literature. Direct inverse control, neural internal model control, neural predictive control, neural feedback linearization control are some neural control structures based on nonlinear mapping, which are briefly mentioned in this chapter. The difficulty of obtaining the training data set which covers a sufficiently large operational range is the main drawback of these schemes. Moreover, most neural control systems based on on-line learning strategies such as backpropagation-through-time, model reference neural adaptive control, reinforcement learning control, and direct adaptive control, still lack stability guarantees.

The research described in this thesis is focused on a type of control system architecture in which the FNN serves as a direct feedback nonlinear controller. A sliding function using as the network training error is defined to omit the need of an NN identifier. An on-line sliding-mode-based learning algorithm, named CFSMBP, is proposed to speed up the incremental BP approach. The NN controllers using the CFSMBP and BP learning algorithms are theoretically proven to stabilise the controlled system. A training procedure is then developed to ensure the NN weights converge to optimal values and to avoid the difficult choice of sufficient training signals associated with the existing training procedure. Therefore, when applying the trained NN controller in the system, the output performance can satisfy the desired requirements in the presence of parameter changes and nonlinearities. Real-time experiments for the Static VAR Compensator System show that a real-time neural controller has been designed and implemented successfully.

The research further developed a decentralised neural control scheme for the large-scale systems with interconnections and disturbances. The whole system has a parallel structure so that each subsystem uses a multilayer NN which works as the direct adaptive neural controller. A novel sliding function is defined for training the direct neural controller without involving the Jacobian matrix. An on-line NN learning algorithm based on the reaching law method is proposed so that the closed-loop controlled system is asymptotically stable. The scheme is also applied successfully in the real-time Coupled Electric Drive systems.

CHAPTER 3

NEURAL NETWORK LEARNING ALGORITHMS

Learning algorithms are the heart of artificial NNs. When the feedforward NNs are utilised in the control and classification systems, the integration between variable structure system (VSS) theory and the learning process can solve some problems of these NN-based systems. Therefore, this chapter presents some background about sliding mode techniques and NN learning algorithms. Until now, there have been hundreds of publications proposing various learning algorithms of FNNs. It is not possible to report all the learning algorithms published in the thesis, however, in this chapter, a few popular learning algorithms for the multilayer FNNs have been considered. These algorithms are well cited in the research literature and are related to the new learning algorithms proposed in the next chapter.

3.1 The Backpropagation Algorithm

The backpropagation algorithm (Paker, 1982; Rumelhart & McClelland, 1986; Werbos, 1974) is the most well known and widely used among other learning algorithms described in the literature. In this algorithm, an error function is defined as the mean-square difference between the desired output and the actual output of the FNN. The BP algorithm is based on steepest-descent techniques extended to each of the layers in the network by the chain rule. There are two types of backpropagation algorithm, the *batch* learning algorithm and the *incremental* learning algorithm. In the batch learning method, the weights are updated after all patterns are presented, while weight updating in the incremental learning is performed at every iteration after the presentation of an input pattern. The batch learning method is more robust, since the training step averages over all the training patterns. On the other hand, the incremental learning approach appeals to some on-line adaptation applications.

Consider the basic structure of a feed-forward network with a single hidden layer, as shown in Figure 3.1. The neural network consists of N input nodes, K hidden nodes, and

M output nodes. Sigmoid functions are used as the activation functions for both the hidden and output layer.

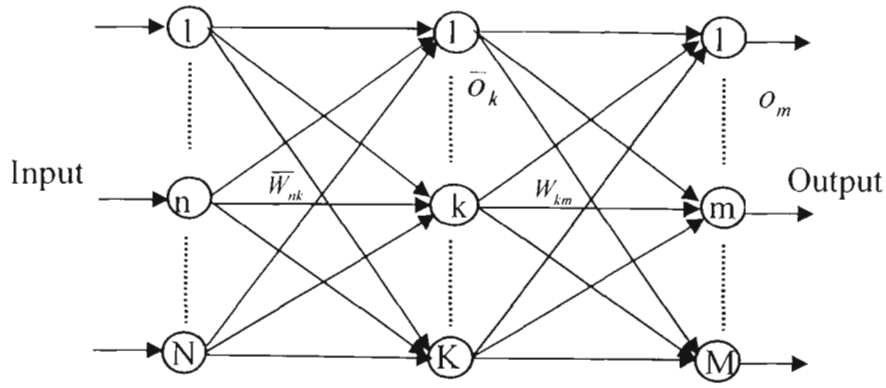


Figure 3.1: Structure of a feedforward neural network with one hidden layer.

Let $\bar{W}_{nk}$ be the network weight of the connection between the input node n and the hidden node k, W_{km} the network weight of the connection between the hidden node k and the output node m, $\bar{o}_k$ and o_m the output of the hidden node k and the output of the output node m, respectively. For each training pattern p, let $\mathbf{x}^p$ be the input values, and $\mathbf{d}^p$ the target output values.

The standard BP algorithm based on batch learning is shown in the following:

1. (Initialisation) Initialise all weights to small random values $W_{km}(0)$ and $\bar{W}_{nk}(0)$. Choose a small positive learning rate η, maximum number of iterations $t_{\max}$ and a very small maximum tolerable error $E_{\max}$. Set the initial iteration $t = 0$.

2. (Forward propagation) Select training pairs $\{\mathbf{x}^p, \mathbf{d}^p | p = 1,..., P\}$ from the training set, propagate the inputs forward through the network using the equation:

$$o_m{}^p = f_o\left(net_m\right) = f_o\left(\sum_{k=1}^{K} W_{km}\left(t\right)\bar{o}_k{}^p\right)$$

$$\bar{o}_k{}^p = f_h\left(net_k\right) = f_h\left(\sum_{n=1}^{N} \bar{W}_{nk}\left(t\right)x_n{}^p\right)$$

where $f_o\left(.\right)$ is the activation function for output node, $f_h\left(.\right)$ is the activation function for hidden node.

The criterion function of the error is calculated as

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[d_m{}^p - o_m{}^p\right]^2 \tag{3.1}$$

If E is not greater than $E_{\max}$, or the number of iterations exceeds a value $t_{\max}$, then the algorithm is stopped. Otherwise, go to step 3.

3. (Backward propagation) Compute the changes of the weights for the next iteration

$$\Delta W_{km}\left(t\right) = -\eta\frac{\partial E}{\partial W_{km}\left(t\right)} = \eta\sum_{p=1}^{P}\delta_m{}^p\bar{o}_k{}^p$$

$$\Delta \bar{W}_{nk}\left(t\right) = -\eta\frac{\partial E}{\partial \bar{W}_{nk}\left(t\right)} = \eta\sum_{p=1}^{P}\bar{\delta}_k{}^p x_n{}^p$$

where

$$\delta_m{}^p = \left(d_m{}^p - o_m{}^p\right)f_o'\left(net_{om}\right)$$

$$\bar{\delta}_k{}^p = f_h'\left(net_{hk}\right)\sum_{m=1}^{M}\delta_m{}^p W_{km}\left(t\right)$$

Update the weights

$$W_{km}(t+1) = W_{km}(t) + \Delta W_{km}(t)$$
$$\bar{W}_{nk}(t+1) = \bar{W}_{nk}(t) + \Delta \bar{W}_{nk}(t)$$

Set $t = t+1$ and go to step 2.

In the incremental learning method, the criterion function in (3.1) is replaced by

$$E = E^p = \frac{1}{2}\sum_{m=1}^{M}\left[d_m{}^p - o_m{}^p\right]^2 \tag{3.2}$$

For simplification, define a weight vector of all the NN's weights as

$$\mathbf{w} = \left[\bar{W}_{11},...,\bar{W}_{nk},...,\bar{W}_{NK},W_{11},...,W_{km},...,W_{KM}\right]^T \tag{3.3}$$

The BP algorithm can now be written in the form

$$\Delta \mathbf{w}(t) = -\eta \frac{\partial E}{\partial \mathbf{w}(t)} \tag{3.4}$$

where the gradient term $\frac{\partial E}{\partial \mathbf{w}(t)}$ is sometimes expressed as $\mathbf{g}(t)$ or $\nabla E(\mathbf{w}(t))$, and its components $\Delta w_i(t) = -\eta \frac{\partial E}{\partial w_i(t)}$; $i = N \times K + K \times M$.

The BP algorithm with a simple computation has been successfully applied in a variety of areas (Meireles et al., 2003). However, BP suffers from the following disadvantages. Firstly, BP can converge to a local minimum of the error function (Bianchini et al., 1994), and fail in finding a global optimal solution. Secondly, the convergence speed of BP is too slow, which is often unacceptable in real-time systems. Its slowness is due to the use of the steepest-descent technique with a fixed chosen step length (Fletcher, 1987). Another reason for the slow convergence is the premature saturation of the derivative of the activation function, sometimes referred as the "flat-spot" problem (Lee

et al., 1993). Another less-considered major problem is the robustness, that is, how well the FNN trained by BP will perform in the presence of noise. The concept of robustness is a prime requirement when the NNs are used for control systems with external disturbances and parameter uncertainties.

There are numerous modifications to BP with the goal of increased speed of convergence, avoidance of local minima, and/or improvement in the NN's robustness properties. A classification tree is shown in Figure 3.2 to help our discussion. As can be seen from the figure, the methods proposed to improve the original BP performance can be divided into groups: (1) Dynamic learning rate methods, (2) Second-order gradient methods, (3) Activation function variations, (4) BP with momentum, (5) Criterion function variations, and (6) Sliding-mode-based learning algorithms.

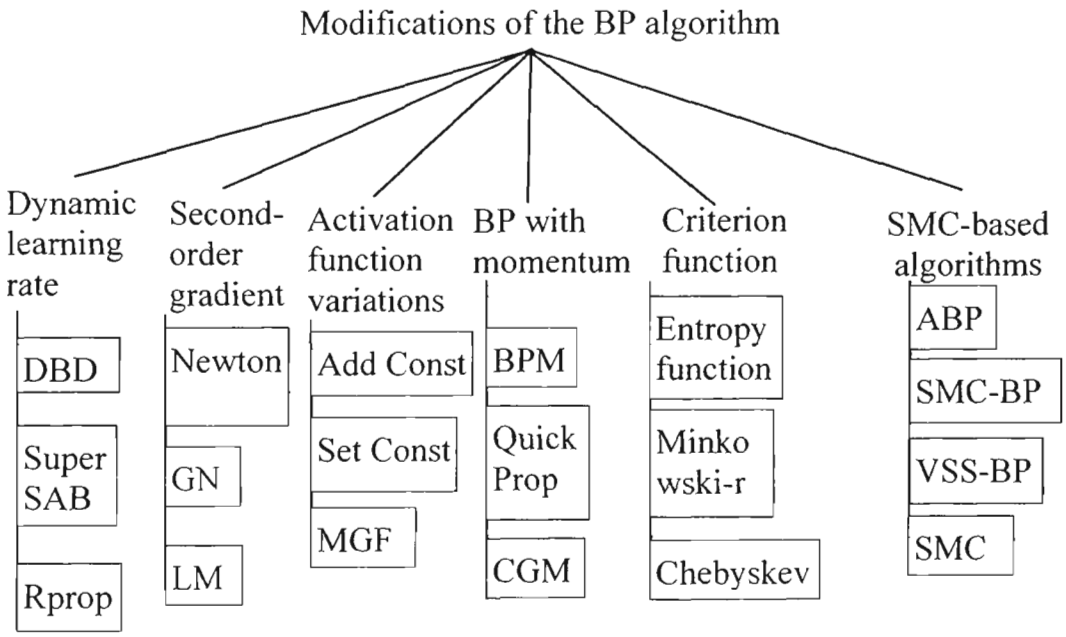


Figure 3.2: Tree classification of different modified backpropagation algorithms

3.2 Variable Learning Rate Backpropagation Algorithms

The convergence speed of BP is directly affected by the learning rate η. If η is small, the search path will closely approximate the gradient path, but convergence will be very slow, due to the large number of update steps needed to reach a local minimum. On the other hand, if η is large, convergence initially will be very fast, but the algorithm will eventually oscillate and thus not reach a minimum. The adaptive learning rate hence can speed up the learning process. Some of these techniques will be discussed below.

3.2.1 Delta-bar-delta (DBD) (Jacobs, 1988)

The Delta-bar-delta learning rules use the adaptive rate to speed up the convergence. If the parameter change has been in the same direction for several iterations, the algorithm increases the learning rate by adding a constant. If the direction of the parameter change alternates, then the learning rate is decreased by multiplying a constant smaller than 1. This is prompted by the idea that if the weight changes are oscillating, the minimum is between the oscillations and a smaller step size might find that minimum. The step size may be increased again once the error has stopped oscillating. The DBD algorithm is based on the batch learning method and can be described as follows:

1. (Initialisation) Initialise all weights to small random values $w_i(0)$. Choose small positive learning rates $\eta_i(0)$, maximum number of iterations $t_{\max}$ and very small maximum tolerable error $E_{\max}$. Set the increase factor η^+, the decrease factor η^-, the initial iteration $t=1$, $\overline{w}_i(0)=0$, filter coefficient β.

2. (Forward propagation) Select training pairs $\{\mathbf{x}^p, \mathbf{d}^p \mid p=1,...,P\}$ from the training set, propagate the inputs forward through the network to attain outputs

$$o_m{}^p;\ p=1,...,P;\ m=1,...,M.$$

The criterion function of the error is calculated as

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[d_m{}^p - o_m{}^p\right]^2$$

If E is not greater than $E_{\max}$, or the number of iterations exceeds a value $t_{\max}$, then the algorithm is stopped. Otherwise, go to step 3.

3. (Backward propagation) Compute the derivative of the criterion function with regard to weights as

$$\frac{\partial E}{\partial w_i(t)}$$

Change the learning rate by the rule

$$\bar{w}_i(t) = (1-\beta)\frac{\partial E}{\partial w_i(t)} + \beta \bar{w}_i(t-1)$$

$$\text{if } \left(\frac{\partial E}{\partial w_i(t)} \times \bar{w}_i(t-1) > 0\right) \text{ then } \eta_i(t) = \eta_i(t-1) + \eta^+$$

$$\text{else if } \left(\frac{\partial E}{\partial w_i(t)} \times \bar{w}_i(t-1) < 0\right) \text{ then } \eta_i(t) = \eta_i(t-1) \times \eta^-$$

$$\text{else } \eta_i(t) = \eta_i(t-1)$$

Update the weights and the gradient and learning rates

$$w_i(t) = w_i(t-1) - \eta_i \frac{\partial E}{\partial w_i(t)}$$

$$\eta_i(t-1) = \eta_i(t); \quad \frac{\partial E}{\partial w_i(t-1)} = \frac{\partial E}{\partial w_i(t)}$$

Set $t = t + 1$ and go to step 2.

Some values are recommended for $\eta^+ = 0.095$ and $\eta^- = 0.9$, $\beta = 0.7$. When η^+, η^- are set to zero, the learning rates assume a constant value as in the standard backpropagation algorithm.

3.2.2 RPROP (Riedmiller & Braun, 1993)

The basic principle of RPROP is to eliminate the saturated influence of the magnitude of the gradient, thus only the sign of the gradient is used to find the proper update direction. RPROP uses independent update step size η_j for every connection. Furthermore, these step sizes are adapted with regard to the sign of the actual and the last derivative. The step sizes are bounded by upper and lower limits in order to avoid oscillation and arithmetic underflow of the floating point values. Finally, local backtracking is applied to those connections where sign changes of the gradient are detected. The RPROP algorithm can be described as follows:

1. (Initialisation) Initialise all weights to small random values $w_i(0)$. Choose small positive learning rates $\eta_i(0)$, maximum number of iterations $t_{\max}$ and very small maximum tolerable error $E_{\max}$. Set the increase factor η^+, the decrease factor η^-, the initial iteration $t = 1$, $\Delta w_i(0) = 0$.

2. (Forward propagation) Select training pairs $\{\mathbf{x}^p, \mathbf{d}^p | p = 1,..., P\}$ from the training set, propagate the inputs forward through the network to attain outputs

$$o_m{}^p;\ p = 1,..., P;\ m = 1,..., M.$$

The criterion function of error is calculated as

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[d_m{}^p - o_m{}^p\right]^2$$

If E is not greater than $E_{\max}$, or the number of iterations exceeds a value $t_{\max}$, then the algorithm is stopped. Otherwise, go to step 3.

3. (Backward propagation) Compute the derivative of the criterion function with regard to weights as

$$\frac{\partial E}{\partial w_i(t)}$$

Change the learning rate by the rule

if $\left(\frac{\partial E}{\partial w_i(t)} \times \frac{\partial E}{\partial w_i(t-1)} > 0\right)$ then

begin $\eta_i(t) = \eta_i(t-1) \times \eta^+$

if $\eta_i(t) > \eta_{\max}$ then $\eta_i(t) = \eta_{\max}$

$$\Delta w_i(t) = -\eta_i(t) \times \operatorname{sgn}\left(\frac{\partial E}{\partial w_i(t)}\right)$$

$$w_i(t) = w_i(t-1) + \Delta w_i(t)$$

end

else if $\left(\frac{\partial E}{\partial w_i(t)} \times \frac{\partial E}{\partial w_i(t-1)} < 0\right)$ then

begin $\eta_i(t) = \eta_i(t-1) \times \eta^-$

$$\frac{\partial E}{\partial w_i(t)} = 0$$

if $\eta_i(t) < \eta_{\min}$ then $\eta_i(t) = \eta_{\min}$

$$w_i(t) = w_i(t-1) - \Delta w_i(t-1)$$

end

else

begin $\eta_i(t) = \eta_i(t-1)$

$$\Delta w_i(t) = -\eta_i(t) \times \operatorname{sgn}\left(\frac{\partial E}{\partial w_i(t)}\right)$$

$$w_i(t) = w_i(t-1) + \Delta w_i(t)$$

end

Update the gradient, weight changes and learning rates

$$\eta_i(t-1) = \eta_i(t); \quad \frac{\partial E}{\partial w_i(t-1)} = \frac{\partial E}{\partial w_i(t)}; \quad \Delta w_i(t) = \Delta w_i(t-1)$$

Set $t = t+1$ and go to step 2.

The recommended values of the parameters are $\eta^+ = 1.2$, $\eta^- = 0.5$, $\eta_{\max} = 50.0$, $\eta_{\min} = 0.000001$.

3.3 Second-order Gradient Methods

The variable learning rate BP algorithms actually try to change the learning rate with regard to widely differing eigenvalues of the Hessian matrix. By directly using the Hessian matrix, Newton methods can significantly improve the convergence speed of BP. The Hessian matrix $\mathbf{H}_w$ is the matrix of second derivatives of E with regard to the weights $\mathbf{W}$ as

$$\mathbf{H}_w(t) = \frac{\partial^2 E}{\partial \mathbf{w}(t)^2}, \tag{3.5}$$

with its components $H_{w_{ij}}(t) = \frac{\partial^2 E}{\partial w_i(t)\partial w_j(t)}$

3.3.1 Newton method

The Newton method is based on a quadratic model $\tilde{E}(\mathbf{w})$ of the criterion $E(\mathbf{w})$ and uses only the three terms in the Taylor series expansion of E about the weight vector $\mathbf{w}(i)$:

$$\tilde{E}(\mathbf{w}(t)+\Delta\mathbf{w}(t)) = E(\mathbf{w}(t)) + \nabla E(\mathbf{w}(t))\Delta\mathbf{w}(t) + \tfrac{1}{2}\Delta\mathbf{w}(t)^T \nabla^2 E(\mathbf{w}(t))\Delta\mathbf{w}(t)$$

This quadratic function is minimised by solving the equation $\nabla\tilde{E}(\mathbf{w}(t)+\Delta\mathbf{w}(t)) = 0$, which leads to the Newton's method

$$\Delta\mathbf{w}(t) = -\eta\left[\mathbf{H}_w(t)\right]^{-1}\frac{\partial E}{\partial \mathbf{w}(t)} \tag{3.6}$$

This algorithm converges quickly if the search region is quadratic or nearly so (Cichocki & Unbehauen, 1993). However, the Newton method is not commonly used because computing the Hessian matrix is computationally expensive. To overcome the problem, several methods have being proposed to approximate the Hessian matrix.

3.3.2 Gauss–Newton method (Becker & Le Cun, 1989)

Becker and Le Cun proposed the Gauss–Newton method whereby the off-line elements of $\mathbf{H}_w$ are neglected, thus arriving at the approximation

$$\Delta w_i(t) = -\eta \left[\frac{\partial^2 E}{\partial w_i^2(t)} \right]^{-1} \frac{\partial E}{\partial w_i(t)} \tag{3.7}$$

However, in the regions of very small curvature, such as plateaus, $\Delta w_i(t)$ in Equation (3.7) is dramatically increased. Furthermore, the Hessian matrix may not be positive definite at every point in the error surface.

3.3.3 Levenberg–Marquardt (LM) method (Hagan & Menhaj, 1994)

The LM method overcomes these difficulties by adding a positive scalar μ to the Gauss–Newton approximation of the Hessian:

$$\Delta w_i(i) = -\eta \left[\frac{\partial^2 E}{\partial w_i^2(t)} + \mu \right]^{-1} \frac{\partial E}{\partial w_i(t)} \tag{3.8}$$

μ could also change adaptively by

$$\mu = \begin{cases} \mu\beta & E(t+1) \geq E(t) \\ \dfrac{\mu}{\beta} & E(t+1) < E(t) \end{cases}$$

where β is a value defined by the user. The LM algorithm can be described as:

1. (Initialisation) Initialise all weights to small random values $w_i(0)$. Choose a small positive coefficient μ, maximum number of iterations $t_{\max}$ and very small maximum tolerable error $E_{\max}$. Set parameter ϑ, the initial iteration $t = 1$.

2. (Forward propagation) Select training pairs $\{\mathbf{x}^p, \mathbf{d}^p | p = 1,...,P\}$ from the training set, propagate the inputs forward through the network to attain outputs and error vector

$$o_m{}^p; \quad \mathbf{e} = \left[d_m{}^p - o_m{}^p \right]; \quad p = 1,...,P; \quad m = 1,...,M$$

The criterion function over all patterns is calculated as

$$E(t-1) = \frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} \left[d_m{}^p - o_m{}^p \right]^2$$

If E is not greater than $E_{\max}$, or the number of iterations exceeds a value $t_{\max}$, then the algorithm is stopped. Otherwise, go to step 3.

3. (Backward propagation) Compute the derivative of output error $o_m{}^p$ with regard to weights to attain the Jacobian matrix

$$\mathbf{J} = \left[\frac{\partial o_m{}^p}{\partial w_i(t)} \right]; \quad i = N \times K + K \times M; \quad p = 1,...,P; \quad m = 1,...,M$$

Update the weights

$$\Delta w_i(t) = -\left[\mathbf{J}^T \mathbf{J} + \mu \mathbf{I} \right]^{-1} \mathbf{J}^T \mathbf{e} \tag{3.9}$$

$$w_i(t) = w_i(t-1) + \Delta w_i(t)$$

4. Recompute the criterion function with the new weights $w_i(t)$.

$$E(t)=\frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[d_m{}^p-o_m{}^p\right]^2$$

If $\left(E(t)<E(t-1)\right)$ then $\mu=\mu/\vartheta$, set $w_i(t-1)=w_i(t)$, $t=t+1$ and go to step 2.

Else $\mu=\mu\times\vartheta$: $w_i(t)=w_i(t-1)$ and go to step 3.

The recommended values of the parameters are $\vartheta=5$, $\mu=0.01$.

3.4 Activation Function Variations

During training, if a neuron in an FNN receives a weights signal *net*, a sum of all input signals presented to this neuron, with a large magnitude, this neuron outputs a value close to one of the saturation levels of its activation function, as in Figure 3.3. If the corresponding target value is substantially different from that of the saturated neuron, one can say that the neuron has entered a flat spot. When this happens, the size of the weight update, due to the derivative of activation function approaching zero, will be very small, and it will take an excessively long time to leave the flat spot.

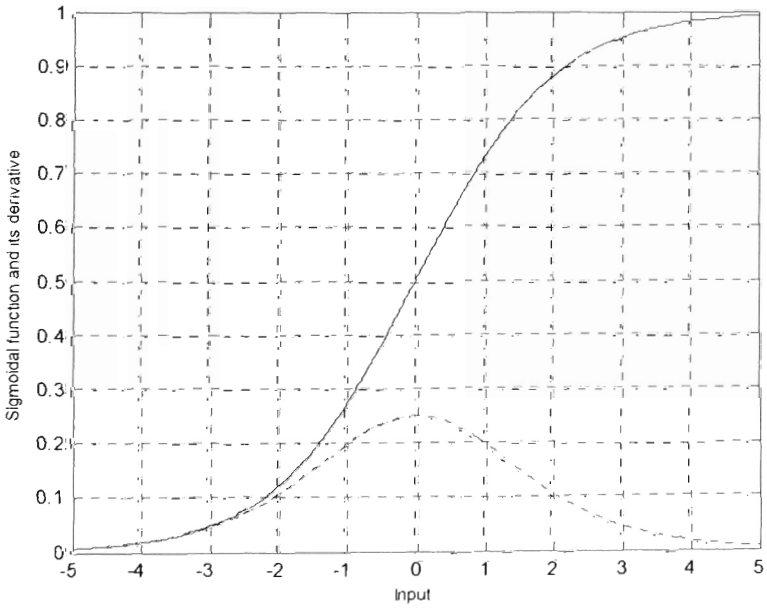


Figure 3.3: Sigmoid function (solid line) and its derivative (dotted line).

Fahlman (1988) suggested altering the derivative of the activation function directly by adding a constant 0.1 to the derivative term. Thus the derivative will be scaled up and will not be too small.

$$f'_{new}(net) = f'(net) + 0.1, \tag{3.10}$$

where $f'(net)$ can be $f'_o(net_m)$ for the output node or $f'_h(net_k)$ for the hidden node.

Vitela and Reifman (1997) set the value of the derivative to a constant value when the activation level falls in predefined saturation regions.

$$f'(net) = 0.09 \text{ as } f(net) \le 0.1 \text{ or } f(net) \ge 0.9 \tag{3.11}$$

Ng et al. (2004) magnified the activation derivative by using a power factor $\frac{1}{S}$, where $s \ge 1$. Therefore, the derivative of the activation function is scaled up to speed up the convergence rate.

$$f'_S(net, s) = \left[f'(net)\right]^{\frac{1}{S}} \tag{3.12}$$

These methods actually tried to solve the premature saturation problem accompanying the derivative of activation function. Therefore, the NNs applying these approaches can speed up the convergence rate, but the local minima problem still exists.

3.5 Backpropagation with Momentum Methods

One efficient and commonly used method that allows a larger learning rate without divergent oscillations occurring is the addition of a momentum to the BP algorithm.

3.5.1 Backpropagation with momentum (BPM) (Plaut et al., 1986)

This learning algorithm introduces a momentum term in the weight changes as

$$\Delta w_i(t) = -\eta \frac{\partial E}{\partial w_i(t)} + \alpha \Delta w_i(t-1) \tag{3.13}$$

where the momentum factor $0 < \alpha < 1$, $\Delta w_i(t) = w_i(t) - w_i(t-1)$.

This algorithm can be described in the following steps:

1. (Initialisation) Initialise all weights to small random values $w_i(0)$. Choose a small positive learning rate η, maximum number of iterations $t_{\max}$ and very small maximum tolerable error $E_{\max}$. Set the initial iteration $t = 1$, $\Delta w_i(0) = 0$.

2. (Forward propagation) Select training pairs $\{\mathbf{x}^p, \mathbf{d}^p | p = 1,..., P\}$ from the training set, propagate the inputs forward through the network to attain outputs

$$o_m{}^p;\ p = 1,..., P;\ m = 1,..., M.$$

The criterion function of the error is calculated as

$$E = \frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} \left[d_m{}^p - o_m{}^p \right]^2$$

If E is not greater than $E_{\max}$, or the number of iterations exceeds a value $t_{\max}$, then the algorithm is stopped. Otherwise, go to step 3.

3. (Backward propagation) Compute the derivative of the criterion function with regard to weights as

$$\frac{\partial E}{\partial w_i(t)}$$

Change the weights as $\Delta w_i(t) = -\eta \frac{\partial E}{\partial w_i(t)} + \alpha \Delta w_i(t-1)$

$$w_i(t) = w_i(t-1) + \Delta w_i(t)$$

Update the weight changes $\Delta w_i(t-1) = \Delta w_i(t)$

Set $t = t+1$ and go to step 2.

3.5.2 Quickprop (Fahlman, 1988)

Fahlman (1988) proposed a variation of BPM, called Quickprop, that employs a dynamic momentum by

$$\alpha(t) = \frac{\frac{\partial E}{\partial w_i(t)}}{\frac{\partial E}{\partial w_i(t-1)} - \frac{\partial E}{\partial w_i(t)}} \tag{3.14}$$

With this adaptive coefficient $\alpha(t)$, if the current slope is persistently smaller than the previous one but has the same sign, then $\alpha(t)$ is positive, and the weight change will accelerate. If the current slope is in the opposite direction from the previous one, meaning the weights are crossing over a minimum, then $\alpha(t)$ is negative, and the weight change starts to decelerate. To prevent the weights from growing too large, a small weight decay can be used.

The Quickprop algorithm can be described in the following steps:

1. (Initialisation) Initialise all weights to small random values $w_i(0)$. Choose a small positive learning rate η, maximum number of iterations $t_{\max}$ and very small maximum tolerable error $E_{\max}$. Set parameter μ, the initial iteration $t = 1$, gradient $\frac{\partial E}{\partial w_i(0)} = 0$, $\Delta w_i(0) = \Delta w_i(1) = 0$.

2. (Forward propagation) Select training pairs $\{\mathbf{x}^p, \mathbf{d}^p \mid p = 1,...,P\}$ from the training set, propagate the inputs forward through the network to attain outputs

$$o_m{}^p;\ p = 1,...,P;\ m = 1,...,M.$$

The criterion function of the error is calculated as

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[d_m{}^p - o_m{}^p\right]^2$$

If E is not greater than $E_{\max}$, or the number of iterations exceeds a value $t_{\max}$, then the algorithm is stopped. Otherwise, go to step 3.

3. (Backward propagation) Compute the derivative of the criterion function with regard to weights as

$$\frac{\partial E}{\partial w_i(t)}$$

4. Changes of the weights are as follows:

If $(\Delta w_i(t) > 0)$ then

begin

if $\left(\frac{\partial E}{\partial w_i(t)} > 0\right)$ then $\Delta w_i(t+1) = -\eta \frac{\partial E}{\partial w_i(t)}$

if $\left(\frac{\partial E}{\partial w_i(t)} > \frac{\mu}{1-\mu}\frac{\partial E}{\partial w_i(t-1)}\right)$ then $\Delta w_i(t+1) = \Delta w_i(t) + \mu \times \Delta w_i(t-1)$

else $\Delta w_i(t+1) = \Delta w_i(t) + \frac{\frac{\partial E}{\partial w_i(t)} \times \Delta w_i(t-1)}{\frac{\partial E}{\partial w_i(t-1)} - \frac{\partial E}{\partial w_i(t)}}$

end

Else if $\left(\Delta w_i(t) < 0\right)$ then

begin

if $\left(\frac{\partial E}{\partial w_i(t)} < 0\right)$ then $\Delta w_i(t+1) = -\eta \frac{\partial E}{\partial w_i(t)}$

if $\left(\frac{\partial E}{\partial w_i(t)} < \frac{\mu}{1-\mu}\frac{\partial E}{\partial w_i(t-1)}\right)$ then $\Delta w_i(t+1) = \Delta w_i(t) + \mu \times \Delta w_i(t-1)$

else $\Delta w_i(t+1) = \Delta w_i(t) + \frac{\frac{\partial E}{\partial w_i(t)} \times \Delta w_i(t-1)}{\frac{\partial E}{\partial w_i(t-1)} - \frac{\partial E}{\partial w_i(t)}}$

end

Else $\Delta w_i(t+1) = -\eta \frac{\partial E}{\partial w_i(t)}$

Update the weights, changes of weights and gradient

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$\frac{\partial E}{\partial w_i(t-1)} = \frac{\partial E}{\partial w_i(t)}$$

$$\Delta w_i(t) = \Delta w_i(t-1);\ \Delta w_i(t+1) = \Delta w_i(t)$$

Set $t = t+1$, $\frac{\partial E}{\partial w_i(0)} = 0$ and go to step 2.

The recommended value of the parameter μ is 1.5.

3.5.3 Scaled conjugate gradient method

In scaled conjugate gradient methods, a new search direction in two successive steps of optimal steepest descent is a compromise between the current gradient ∇E and the previous search direction

$$\mathbf{d}(t) = -\mathbf{g}(t) + \alpha \mathbf{d}(t-1)$$

with $\mathbf{g}(t) = \frac{\partial E}{\partial \mathbf{w}(t)}$, $\mathbf{d}(0) = -\mathbf{g}(0)$.

From the relation $\Delta \mathbf{w}(t) = \eta(t)\mathbf{d}(t)$, the weight vector update rule becomes

$$\Delta \mathbf{w}(t) = -\eta(t)\frac{\partial E}{\partial \mathbf{w}(t)} + \alpha(t)\Delta \mathbf{w}(t-1), \tag{3.15}$$

In practice, α, which plays the role of an adaptive momentum, is chosen according to the Hestenes-Stiefel rule (Moller, 1993):

$$\alpha(t) = \frac{\mathbf{g}(t)^T\left[\mathbf{g}(t) - \mathbf{g}(t-1)\right]}{\mathbf{d}(t-1)^T\left[\mathbf{g}(t) - \mathbf{g}(t-1)\right]},$$

or the Polack-Ribiere rule (Hert et al., 1991):

$$\alpha(t)=\frac{\mathbf{g}(t)^{T}\left[\mathbf{g}(t)-\mathbf{g}(t-1)\right]}{\mathbf{g}(t-1)^{T}\,\mathbf{g}(t-1)},$$

or the Fletcher-Reeves rule:

$$\alpha(t)=\frac{\mathbf{g}(t)^{T}\,\mathbf{g}(t)}{\mathbf{g}(t-1)^{T}\,\mathbf{g}(t-1)}$$

The optimal step size $\eta(t)$ is chosen to minimise the energy function along the search direction

$$\eta(t)=\min_{\eta} E\left(\mathbf{w}(t)+\eta(t)\mathbf{d}(t)\right)$$

These algorithms can be described as follows:

1. (Initialisation) Initialise all weights to small random values $w_i(0)$. Choose a small positive learning rate η, maximum number of iterations $t_{\max}$ and very small maximum tolerable error $E_{\max}$. Set the initial iteration $t=1$, $\mathbf{g}(0)=\dfrac{\partial E}{\partial \mathbf{w}(0)}=0$, $\mathbf{d}(0)=-\mathbf{g}(0)$, $\Delta w_i(0)=0$.

2. Estimate the minimisation step

$$\eta(t)=-\frac{\mathbf{d}(t)^{T}\,\mathbf{g}(t)}{\mathbf{d}(t)^{T}\,\mathbf{H}(t)\mathbf{d}(t)+\lambda(t)\left\|\mathbf{d}(t)\right\|^{2}}$$

Where $\lambda(t)$ is a fudge factor to make the denominator positive and $\mathbf{H}(t)$ is an approximation of the Hessian matrix.

Update the weights $\Delta \mathbf{w}(t) = \eta(t)\mathbf{d}(t)$

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \Delta \mathbf{w}(t)$$

3. (Forward propagation) Select training pairs $\{\mathbf{x}^p, \mathbf{d}^p | p = 1,..., P\}$ from the training set, propagate the inputs forward through the network to attain outputs

$$o_m{}^p;\ p = 1,..., P;\ m = 1,..., M.$$

The criterion function of the error is calculated as

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[d_m{}^p - o_m{}^p\right]^2$$

If E is not greater than $E_{\max}$, or the number of iterations exceeds a value $t_{\max}$, then the algorithm is stopped. Otherwise, go to step 3.

4. (Backward propagation) Compute the derivative of the criterion function with regard to weights as

$$\mathbf{g}(t) = \frac{\partial E}{\partial \mathbf{w}(t)}$$

Calculate $\alpha(t)$ as

$$\alpha(t) = \begin{cases} \dfrac{\mathbf{g}(t)^T\left[\mathbf{g}(t) - \mathbf{g}(t-1)\right]}{\mathbf{d}(t-1)^T\left[\mathbf{g}(t) - \mathbf{g}(t-1)\right]} & \text{(Hestenes-Stiefel)} \\ \dfrac{\mathbf{g}(t)^T\left[\mathbf{g}(t) - \mathbf{g}(t-1)\right]}{\mathbf{g}(t-1)^T\,\mathbf{g}(t-1)} & \text{(Polak-Ribiere)} \\ \dfrac{\mathbf{g}(t)^T\,\mathbf{g}(t)}{\mathbf{g}(t-1)^T\,\mathbf{g}(t-1)} & \text{(Fletcher-Reeves)} \end{cases}$$

5. Find the conjugate direction $\mathbf{d}(t)$ as

$$\mathbf{d}(t) = -\mathbf{g}(t) + \alpha(t)\mathbf{d}(t-1)$$

Set $t = t+1$ and go to step 2.

3.6 Criterion Function Variations

In order to increase the speed of convergence, to avoid local minima convergence and to improve the robust property, other criterion/error functions are used, from which new versions of the BP rule can be derived.

3.6.1 The instantaneous entropy criterion

The instantaneous entropy criterion was proposed by Baum and Wilczek (1988), wherein

$$E(\mathbf{w}) = \tfrac{1}{2}\sum_{m=1}^{M}\left[(1+d_m)\ln\left(\frac{1+d_m}{1+o_m}\right)+(1-d_m)\ln\left(\frac{1-d_m}{1-o_m}\right)\right] \tag{3.16}$$

With hyperbolic tangent activations at both hidden and output layers, employing the *incremental* BP learning, the weight changes are obtained as

$$\begin{aligned}\Delta W_{km}(t+1) &= \eta\beta(d_m - o_m)\bar{o}_k \\ \Delta \bar{W}_{nk}(t+1) &= \eta\beta^2(d_m - o_m)(1-\bar{o}_k^{\,2})x_n\end{aligned} \tag{3.17}$$

From this equation, the derivative of the activation function has been eliminated, and the output units do not have a flat-spot problem. However, the derivative of the activation function still appears for the hidden units. Therefore, the flat-spot problem is only partially solved by employing the entropy criterion.

3.6.2 The Minkowski-*r* criterion function

The Minkowski-*r* criterion function was proposed by Hanson and Burr (1988) as:

$$E(\mathbf{w}) = \frac{1}{r}\sum_{m=1}^{M}\left|d_m - o_m\right|^r \tag{3.18}$$

which leads to the weight update rule as

$$\begin{aligned}\Delta W_{km}(t+1) &= \eta\,\mathrm{sgn}(d_m - o_m)\left|d_m - o_m\right|^{r-1} f_o'(net_m)\,\bar{o}_k \\ \Delta W_{nk}(t+1) &= \eta\left[\sum_{m=1}^{M}\mathrm{sgn}(d_m - o_m)\left|d_m - o_m\right|^{r-1} W_{km}(t)\, f_o'(net_m)\right] f_h'(net_k)\, x_n\end{aligned} \tag{3.19}$$

Hanson and Burr (1988) showed that decreasing the power values r can significantly improve convergence and may dramatically reduce noise in the target domain.

The idea of increasing the learning robustness of BP in noisy environments can be placed in a more general statistical framework (White, 1989). Robustness of learning refers to insensitivity to small perturbations in the underlying probability distribution $p(\mathbf{x})$ of the training set. These statistical techniques motivate the replacement of the linear error $e_m = d_m - o_m$ by a nonlinear error suppressor function $f_e(e_m)$ that is compatible with the underlying probability density function $p(\mathbf{x})$. For example,

$$f_e(e_m) = \mathrm{sgn}(d_m - o_m)\left|d_m - o_m\right|^{r-1} \tag{3.20}$$

with $1 \le r \le 2$.

This error suppressor leads to the weight update rule in Equation (3.19). This update rule actually can be classified as the sliding-mode-based learning algorithm.

3.6.3 The Chebyskev norm

The quadratic cost function (3.2) can be generalised to have the norm-based form (Burrascano, 1991) as

$$E = \frac{1}{p}\sum_{m}(d_m - o_m)^p \text{ where } 1 \le p \le \infty \tag{3.21}$$

The L_∞ norm, also called the Chebyshev norm, of the cost function is as

$$E^{\infty} = \sup_{m}\left(\left|d_m - o_m\right|\right) \tag{3.22}$$

Where $\sup(.)$ denotes a function selecting the largest component of the error vector, while the other error components are negligible. The weight update rule is obtained as

$$\begin{aligned}
\Delta W_{km}(t+1) &= \eta \operatorname{sgn}(d_m - o_m) f_o'(net_m)\bar{o}_k \\
\Delta \bar{W}_{nk}(t+1) &= \eta\left[\sum_{m=1}^{M} \operatorname{sgn}(d_m - o_m) W_{km}(t) f_o'(net_m)\right] f_h'(net_k) x_n
\end{aligned} \tag{3.23}$$

The algorithm (3.23) is a specific case of the update rule (3.19) with $r = 1$.

3.7 Sliding-mode-based Learning Algorithms

From the robustness point of view, VSS theory offers high-performance solutions to the problem of NNs' parameter tuning. VSS theory was first proposed by Emelyanov (1967), and was later developed by Itkis (1976) and Utkin (1977). In general, the sliding mode control (SMC) system can be separated into the reaching mode and sliding mode, as shown in Figure 3.4 (for a second-order system). In the reaching mode, a system trajectory starting from anywhere on the phase plane moves toward a desired dynamics, called the sliding surface, and reaches the surface in finite time. Remaining on the sliding surface, or in the sliding mode, the trajectory asymptotically tends to the origin of the phase plane. In the sliding mode, the system is robust against external disturbances and system uncertainties.

Numerous contributions to VSS theory have been made during the last decade. The integral sliding mode control (Utkin & Shi, 1996), the reaching law method (Gao & Hung, 1993) and the chattering-free sliding-mode approach (Ertugrul et al., 1995) are some examples of these new contributions. In those approaches mentioned, the Lynapunov stability theorem is satisfied in different ways so that the system state trajectory is driven toward and remains on the sliding surface.

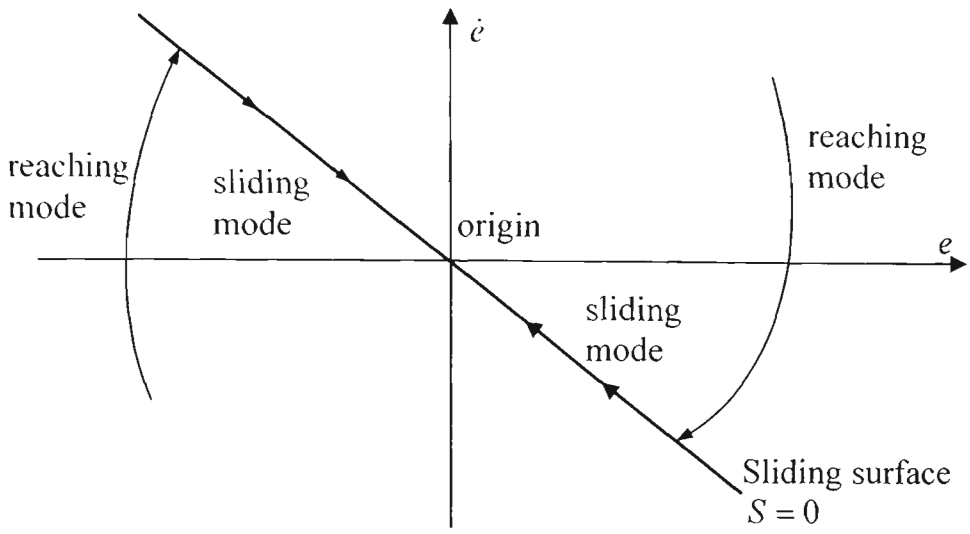


Figure 3.4: State trajectories of a variable structure system.

Several studies utilising VSS theory in the training algorithm of NNs are reported in the literature. Zak and Sira-Ramirez (1990) developed a switching weight adaptation strategy and showed to impose a discrete-time asymptotically stable linear learning error dynamics. However, the authors just developed the robust algorithm for the FNNs with discontinuous activation functions. The pioneering study of Sira-Ramirez and Morles (1995) primarily discussed the use of the sliding-mode control strategy in learning algorithm of Adalines. A dynamical adaptive learning scheme based on sliding-mode control ideas is proposed, and it represents a simple, but robust, mechanism for guaranteeing the finite time reachability of zero error condition. This approach is also highly insensitive to bounded external perturbation inputs and output measurement noise. However, this robust algorithm is only developed for the Adalines. In the following section, recent studies accounting for the sliding-mode-based training performance of FNNs are briefly considered.

3.7.1 Adaptive BP (ABP)

The adaptive BP algorithm was proposed by Jervis and Fitzgerald (1993) and Parlos et al. (1994). The algorithm updates the weights in the direction of steepest descent, but with a learning rate that is a specific function of the error and of the error gradient norm:

$$\Delta \mathbf{w}(t) = \rho(E) \frac{\mathbf{g}(t)}{\|\mathbf{g}(t)\|^2} \tag{3.24}$$

where $$\rho(E) = \begin{cases} \eta \\ \eta E \\ \eta \tanh\left(\frac{E}{E_0}\right) \end{cases},$$

η and E_0 are positive constants, representing the algorithm step size and the error normalisation factor, respectively.

The adaptive BP algorithm with $\rho(E) = \eta E$ is described as follows:

1. (Initialisation) Initialise all weights to small random values $w_i(0)$. Choose a small positive learning rate η, maximum number of iterations $t_{\max}$ and very small maximum tolerable error $E_{\max}$. Set the initial iteration $t = 1$, $\Delta w_i(0) = 0$.

2. (Forward propagation) Select training pairs $\{\mathbf{x}^p, \mathbf{d}^p | p = 1,..., P\}$ from the training set, propagate the inputs forward through the network to attain outputs

$$o_m{}^p;\ p = 1,..., P;\ m = 1,..., M.$$

The criterion function of the error is calculated as

$$E = \frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} \left[d_m{}^p - o_m{}^p \right]^2$$

If E is not greater than $E_{\max}$, or the number of iterations exceeds a value $t_{\max}$, then the algorithm is stopped. Otherwise, go to step 3.

3. (Backward propagation) Compute the derivative of the criterion function with regard to weights as

$$\frac{\partial E}{\partial w_i(t)}$$

Calculate the denominator $\Psi = \left[\frac{\partial E}{\partial \mathbf{w}(t)}\right]^T \frac{\partial E}{\partial \mathbf{w}(t)}$

Changes of weights are as $\Delta w_i(t) = -\frac{\eta E}{\Psi}\frac{\partial E}{\partial w_i(t)}$

$$w_i(t) = w_i(t-1) + \Delta w_i(t)$$

Set $t = t+1$ and go to step 2.

3.7.2 Sliding-mode BP algorithm (SMC-BP)

A new learning algorithm based on the SMC strategy was proposed by Parma et al. (1999). For an FNN as shown in Figure 3.2, the sliding surfaces are first defined as:

For the output layer,

$$s_m(t) = e_{2m}(t) + Ce_{2m}(t), \quad C > 0$$
$$e_{1m}(t) = (d_m - o_m) f_o'(net_m)$$
$$e_{2m}(t) = e_{1m}(t) - e_{1m}(t-1)$$

For the hidden layer, $s_{kH} = e_{2kH} + C_H e_{1kH}, \quad C_H > 0,$

with
$$e_{1kH} = f_h'(net_k)\sum_{m=1}^{M} e_{1m} W_{km}(t)$$
$$e_{2kH}(t) = e_{1kH}(t) - e_{1kH}(t-1)$$

Basing on the SMC principle, the weight update rule is obtained as

$$\Delta W_{km}(t) = \alpha \left| e_{1m}(t) \right| \operatorname{sgn}\left(s_m(t)\right) \bar{o}_k(t)$$
$$\Delta \bar{W}_{nk}(t) = \beta \left| e_{1kl}(t) \right| \operatorname{sgn}\left(s_{kl}(t)\right) x_n(t) \tag{3.25}$$

From the discrete-time sliding mode condition, some bounds on $\alpha > 0, \beta > 0$ are proposed such that the network trained by the proposed algorithm is global convergence. The SMC-BP algorithm resembles the incremental BP one, except for two subtle changes:

1. The absolute value of the error is used instead of the actual error.

2. The sign of the sliding surfaces is added so that the learning problem is reduced to a standard SMC problem. Convergence of the algorithm is therefore guaranteed.

The weight update rule (3.25) is actually similar to the rule in Equation (3.19) with $r = 2$.

3.7.3 Variable structure system technique for BP algorithm (VSS-BP)

In Efe et al. (2000), a sliding function from the change of weights is defined as

$$s(t) = \Delta \mathbf{w}(t)$$

From the reaching law approach (Gao & Hung, 1993), the reaching condition is obtained as

$$\Delta s = -Q_T \tanh\left(\frac{s}{\phi}\right) - K_T s = A(\mathbf{w}),$$

where Q_T, K_T are the gains and ϕ is the width of the boundary layer.

Then a VSS-based algorithm was proposed to force $\Delta \mathbf{w}$ to zero

$$\Delta \mathbf{w}_{VSS}(t) = \beta \min\left\{ \left| \frac{\Delta \mathbf{w}(t-1)}{\mathbf{g}(t)} \right|, \left| \frac{A(\mathbf{w})}{\mathbf{g}(t)} \right| \right\} \mathbf{g}(t) + A(\mathbf{w}), \quad 0 < \beta < 1$$

By combining the BP algorithm and VSS-based algorithm, a fast and robust algorithm is obtained as

$$\Delta \mathbf{w}(t) = -\eta_1 \mathbf{g}(t) + \eta_2 \min\left\{\left|\frac{\Delta \mathbf{w}(t-1)}{\mathbf{g}(t)}\right|, \left|-\frac{A(\mathbf{w})}{\mathbf{g}(t)}\right|\right\}\mathbf{g}(t) + \eta_3 A(\mathbf{w}) \qquad (3.26)$$

One obvious drawback of the VSS-BP algorithm is that it now contains five heuristic parameters: $\eta_1, \eta_2, \eta_3, Q_T, K_T$.

3.7.4 On-line learning in FNN based on the sliding mode concept (SMC)

Giordano et al. (2004) developed an adaptive learning scheme for an FNN with time-varying input vector and an output scalar, as shown in Figure 3.5.

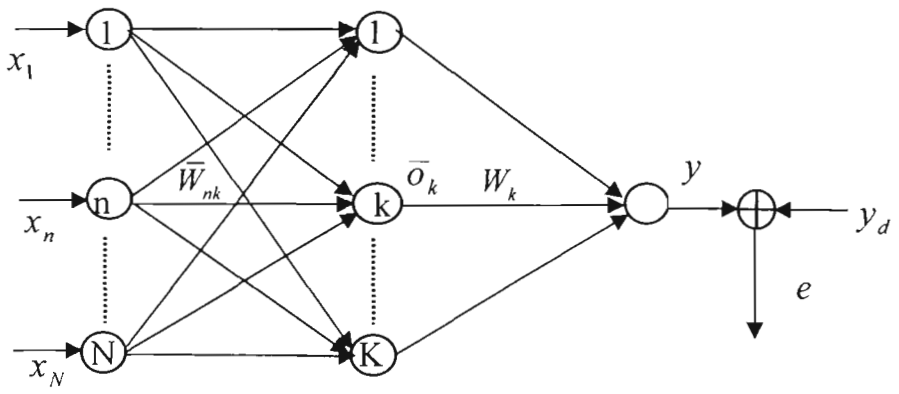


Figure 3.5: Structure of a feedforward neural network with one output neuron.

Giordano et al. assumed that the weight matrices, input vector derivate and derivative of the desired output are bounded

$$\|\mathbf{W}\| \le B_w \quad , \quad \|\bar{\mathbf{W}}\| \le B_{\bar{w}}$$
$$\|\dot{\mathbf{x}}(t)\| \le B_{\dot{x}}, \quad |\dot{y}_d(t)| \le B_{\dot{y}}$$

And the adaptive law for the weight vectors is attained as

$$\dot{W}_k(t) = -\left(\frac{\bar{o}_k}{\bar{\mathbf{o}}^T\bar{\mathbf{o}}}\right)\alpha\,\mathrm{sgn}\left(e(t)\right)$$
$$\dot{\bar{W}}_{nk}(t) = -\left(\frac{W_k x_n}{\mathbf{x}^T\mathbf{x}}\right)\alpha\,\mathrm{sgn}\left(e(t)\right) \tag{3.27}$$

where $\mathbf{x}$ is the vector of the time-varying inputs, $\bar{\mathbf{o}}$ is the vector of the output signals of the neurons in the hidden layer, $e(t) = y(t) - y_d(t)$ is the output error, the slope of activation function $f_h'(.)$ is assumed not to be larger than B_A, and the learning rate is chosen as:

$$\alpha > KB_A B_W B_x B_{\bar{W}} + B_y .$$

This algorithm assures the output error converges to zero in a finite time t_h, which may be estimated as

$$t_h \le \frac{|e(0)|}{\alpha - KB_A B_W B_x B_{\bar{W}} - B_{\dot{y}}} \tag{3.28}$$

3.8 Discussion and conclusion

In the *batch* learning procedure, after all training data are presented, E is calculated and the NN's weights are updated. Thus it can be said that the error criterion E is a function of all weight variables. The change of E in a very small incremental time can be approximated as $\Delta E \approx \left[\frac{\partial E}{\partial \mathbf{w}}\right]^T \Delta\mathbf{w}$.

The BP algorithm yields the dynamics $\Delta E \approx -\eta\left[\frac{\partial E}{\partial \mathbf{w}}\right]^T \frac{\partial E}{\partial \mathbf{w}}$. In the vicinity of a local minimum, $\frac{\partial E}{\partial \mathbf{w}}$ is very small, which results in a slow convergence rate of the BP approach. Moreover, the training process stops when $\Delta E = 0$ in the local minima, that is, $\frac{\partial E}{\partial \mathbf{w}} = 0$, and the NN becomes trapped in the local minima.

The Newton algorithm leads to the equation $\Delta E \approx -\eta \left[\frac{\partial E}{\partial \mathbf{w}}\right]^T \mathbf{H}(t)^{-1} \frac{\partial E}{\partial \mathbf{w}}$. Because the eigenvalues $\{\lambda_1,....,\lambda_n\}$ of $\mathbf{H}(t)$ are not always positive in the error surface, the training procedures may be diverged. In the Levenberg-Marquardt algorithm, this problem is overcome by using a matrix $\mathbf{H}(t)+\mu\mathbf{I}$, and the positive scalar μ is chosen such that all eigenvalues $\{\lambda_1+\mu,...,\lambda_n+\mu\}$ are positive. However, these second-order gradient methods still face the problem of getting trapped in the local minima since the training process always stops at the local minima, that is, $\frac{\partial E}{\partial \mathbf{w}} = 0$.

The inclusion of momentum can generally improve the convergence rate of the BP algorithm, as proven in Bishop (1995). But the BPM algorithm remains inefficient. Another heuristic approach, the QuickProp algorithm, utilises a dynamic momentum factor, thus leading to a significant improvement in convergence speed, in comparison to the BPM algorithm. Nevertheless, if the local quadratic fit is a parabola with a maximum, the QuickProp algorithm leads to an uphill step. Although the scaled conjugate gradient method can be the fastest algorithm for training FNNs, its convergence to local minima is the main drawback of this BP variation.

In the adaptive BP algorithm, the change in error can be obtained as $\Delta E \approx -\eta E$. Therefore, the training algorithm will converge to the global minimum, that is, $E = 0$, with a convergence rate η. This BP modification can show the capability of utilising VSS theory in BP to improve the convergence rate and avoid the local minima problem. However, Parlos et al. (1994) only successfully applied ABP in function approximation. They reported that ABP was found to be as fast as the BP in the parity problem or the encoder problem. This has led to fewer studies on ABP in the neural network literature.

Although the on-line learning algorithm proposed by Giordano et al. (2004) represents a simple, but robust, mechanism for guaranteeing the finite time reachable condition of the zero error function, it is only successfully applied in a real-time neural identifier. Moreover, in the case where the value of $\bar{\mathbf{o}}'\bar{\mathbf{o}}$ in (3.27) reaches zero, the change of weights becomes infinite. This results in the divergence of the SMC algorithm.

In conclusion, a number of popular NN learning algorithms are introduced in this chapter. These BP variations are distributed into some categories, including the dynamic learning rate approaches, the second-order gradient methods, activation function and criterion function variations, backpropagation with momentum, and especially the combination of VSS theory and BP. Major advantages and disavantages of these learning algorithms are then discussed. Motivated from the above algorithms, three novel learning algorithms are developed and presented in the next chapter.

CHAPTER 4

ADVANCED NEURAL NETWORK TRAINING ALGORITHMS BASED ON SLIDING-MODE CONTROL TECHNIQUES

This chapter describes the integration between the variable structure system (VSS) theory and the backpropagation (BP) learning algorithm to obtain some robust algorithms. Using the chattering-free sliding-mode control technique and the reaching law approach, three different learning algorithms are proposed. One batch leaning algorithm is developed for classification problems, and results in a fast and global convergence of the system. Two on-line learning schemes are developed for control applications, and the neural controllers mathematically guarantee the stabilities of the systems with parameter uncertainty and external disturbances. Their effectiveness is illustrated through some simulation results, and is further tested in a neural head-movement classifier for wheelchair control. Finally, discussion on the results is presented in the end of the chapter.

4.1 Introduction

In the VSS theory (Hung et al., 1993), the reaching law method is the most general scheme among the others. For the continuous system, the derivative of a predefined sliding function $s(t)$ in the reaching law method (Gao & Hung, 1993) is described as

$$\dot{s}(t) = -\varepsilon \operatorname{sgn}(s(t)) - qs(t), \quad \varepsilon > 0, q > 0. \tag{4.1}$$

The reaching time for the system (4.1) to move from an initial position $s(0)$ to the sliding surface $s = 0$ is finite, and is given by $T_r = \frac{1}{q} \ln \frac{q|s(0)| + \varepsilon}{\varepsilon}$.

If $\varepsilon = 0$, the reaching rule is similar to the chattering-free SMC strategy (Ertugrul et al., 1995; Nguyen, 1998):

$$\dot{s}(t) = -qs(t), \quad q > 0, \tag{4.2}$$

and the sliding variable $s(t)$ in the rule (4.2) exponentially converges to zero with a convergence rate q.

If $q = 0$, the reaching rule becomes the standard SMC strategy (Utkin, 1977):

$$\dot{s}(t) = -\varepsilon \operatorname{sgn}(s(t)), \quad \varepsilon > 0, \tag{4.3}$$

and the law (4.3) forces the sliding variable $s(t)$ to zero in a finite time $T_s = \frac{|s(0)|}{\varepsilon}$. If ε is too small, the reaching time will be too long. On the other hand, a ε too large will cause severe chattering. Apparently, by adding the proportional term $-qs$, the zero convergence of the sliding function in the reaching law method is faster than those of the other schemes.

The discrete version of the chattering-free SMC strategy becomes

$$\Delta s = s(t+1) - s(t) = -q_T s(t), \tag{4.4}$$

and the sliding function will exponentially converge to zero if $0 < q_T = q.\Delta t < 1$.

When the VSS theory is applied in the NN training process, the criterion function of the network error is often defined from the sliding variable as

$$E = \frac{1}{2}s^2 \tag{4.5}$$

When the change of E is satisfied by the law (4.1), (4.2), or (4.4), the error function will asymptotically converge to zero. Therefore, using the reaching law approach and the chattering-free SMC method in the BP algorithm can yield a number of novel training algorithms, which are presented in the next section. Figure 4.1 shows the tree classification of these algorithms, wherein the acronym of EABPM refers to the

extension of the ABP algorithm with momentum, the acronym of CFSMBP denotes the chattering-free sliding-mode control technique combined with the backpropagation algorithm, and the acronym of RMBP denotes the reaching law method combined with the backpropagation algorithm.

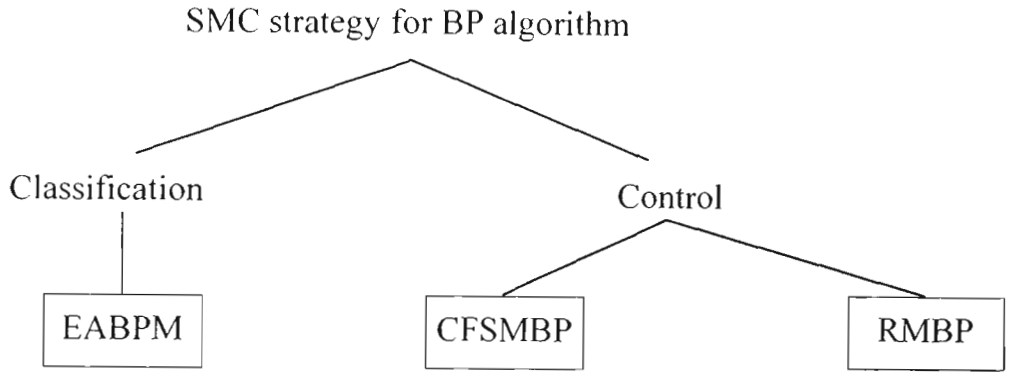


Figure 4.1: Tree classification of the proposed learning algorithms.

4.2 Sliding-mode-based Neural Networks for Classification Applications

4.2.1 Feedforward neural network structure for classification applications

Consider the structure of a feedforward network with a single hidden layer, as shown in Figure 4.2. The neural network consists of N+1 input nodes, K+1 hidden nodes, and M output nodes. Sigmoid functions, $f(v)=\frac{1}{1+e^{-v}}$, are used as the activation functions for both the hidden and output layer. This NN structure is often utilised in the classification applications (Haykin, 1995).

Let $\overline{W}_{nk}$ be the network weight for the input node n and the hidden node k, $\tilde{\overline{\mathbf{W}}}$ the augmented weight matrix between the input layer and the hidden layer, W_{km} the network weight for the hidden node k and the output node m, $\tilde{\mathbf{W}}$ the augmented weight matrix between the hidden layer and the output layer, $\overline{o}_k$ and o_m the outputs of the hidden node k and the output node m, respectively, $\tilde{\overline{\mathbf{o}}}$ the augmented output vector of

the hidden layer. For each training pattern p, let $\mathbf{x}^p$ be the input pattern p, $\tilde{\mathbf{x}}^p$ the vector of augmented inputs, and $\mathbf{d}^p$ the target output values.

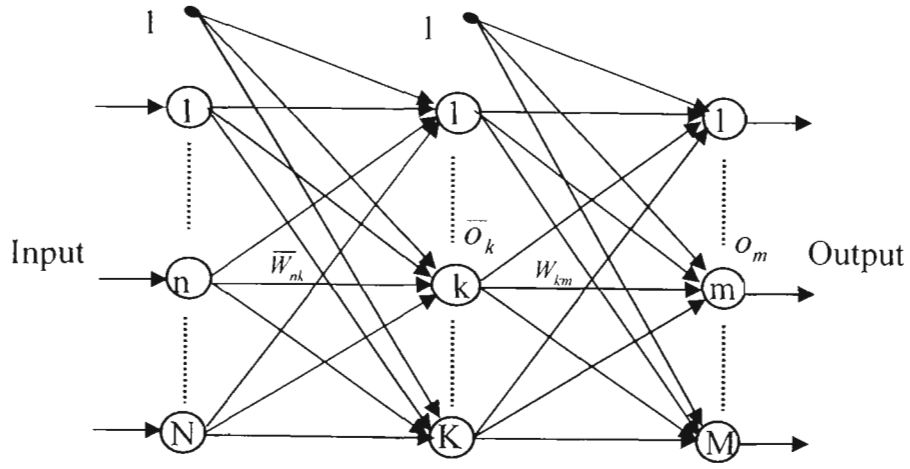


Figure 4.2: Structure of a feedforward network with one hidden layer.

After presenting all training pairs $\{\mathbf{x}^p, \mathbf{d}^p | p = 1,..., P\}$ to the network, the error criterion function is calculated as

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[d_m{}^p - o_m{}^p\right]^2 \tag{4.6}$$

4.2.2 Sliding-mode-based learning algorithm for the FNN

The *batch* BP algorithm is the gradient descent method which changes the network weights to minimise the error function E. However, the BP algorithm is slow in convergence, and often becomes trapped in a local minimum. As discussed in Chapter 3, in the vicinity of a local minimum, $\frac{\partial E}{\partial \mathbf{w}}$ is very small, which results in a slow convergence rate of the BP approach. Moreover, the changes of weights stop when $\Delta E = 0$ in the local minima, that is, $\frac{\partial E}{\partial \mathbf{w}} = 0$, and the NN becomes trapped in the local minima.

In the adaptive BP (ABP) algorithm (Parlos et al., 1994), the change in error can be obtained as $\Delta E = -\eta E$. From the chattering-free SMC strategy, the training process will converge to the global minimum, that is, $E = 0$, with a convergence rate η. However,

as reported in Parlos et al. (1994), the ABP algorithm is not an effective approach for solving the binary problems, such as the parity problem or the encoder problem.

An extended learning algorithm of ABP, as in Equation (3.24) with $\rho(E)=\eta E$, for the FNN, as described in Figure 4.2, is derived from as:

$$\begin{aligned}\Delta W_{km}(t) &= \frac{\eta}{\left(\sum_{p=1}^{P}\delta_m{}^p\bar{o}_k{}^p\right)^2+\left(\sum_{p=1}^{P}\bar{\delta}_k{}^p x_n{}^p\right)^2}\sum_{p=1}^{P}\delta_m{}^p\bar{o}_k{}^p \\ \Delta \bar{W}_{nk}(t) &= \frac{\eta}{\left(\sum_{p=1}^{P}\delta_m{}^p\bar{o}_k{}^p\right)^2+\left(\sum_{p=1}^{P}\bar{\delta}_k{}^p x_n{}^p\right)^2}\sum_{p=1}^{P}\bar{\delta}_k{}^p x_n{}^p\end{aligned} \tag{4.7}$$

where

$$\begin{aligned}\delta_m{}^p &= \left(d_m{}^p - o_m{}^p\right) f_o'\left(net_{om}\right) \\ \bar{\delta}_k{}^p &= f_h'\left(net_{hk}\right)\sum_{m=1}^{M}\delta_m{}^p W_{km}(t).\end{aligned}$$

There are two problems associated with this algorithm. Firstly, when an arbitrary weight of the FNN reaches a local minimum, that is, $\frac{\partial E}{\partial w_i}=0$, the change of this weight obtained zero. Therefore, it is possible for the NN to be trapped in a local minimum. In addition, the denominator in the right side of (4.7) is proportional to the slopes of the activation functions $f_o\left(net_{om}\right)$ and $f_h\left(net_{hk}\right)$. Therefore, if the slope values reach zero, the changes of the network weights become infinite. These problems are the reasons why the ABP algorithm cannot applied in the binary problems.

To tackle the disadvantages of ABP, a momentum term is added to establish a novel algorithm:

$$\Delta W_{km}(t)=\frac{\eta}{\left(\sum_{p=1}^{P}\delta_m{}^p\bar{o}_k{}^p\right)^2+\left(\sum_{p=1}^{P}\bar{\delta}_k{}^p x_n{}^p\right)^2}\sum_{p=1}^{P}\delta_m{}^p\bar{o}_k{}^p+\alpha\Delta W_{km}(t-1)$$
$$\Delta \bar{W}_{nk}(t)=\frac{\eta}{\left(\sum_{p=1}^{P}\delta_m{}^p\bar{o}_k{}^p\right)^2+\left(\sum_{p=1}^{P}\bar{\delta}_k{}^p x_n{}^p\right)^2}\sum_{p=1}^{P}\bar{\delta}_k{}^p x_n{}^p+\alpha\Delta \bar{W}_{nk}(t-1) \tag{4.8}$$

By adding the momentum term, the zero gradient value in a local minimum will not affect the changes of the network weights.

Moreover, to avoid the zero reachability of the slope of the activation function, a modification consisting of adding a constant 0.01 to the values of $f_o'(net_{om})$ and $f_h'(net_{hk})$ is approached.

The meaning of adding the momentum term to ABP can also be explained by the chattering-free SMC theory. Define a sliding function as

$$s=\Delta\mathbf{W}(t-1). \tag{4.9}$$

From the chattering-free sliding-mode-based condition (4.4), the sliding condition is satisfied as

$$\Delta s=-q_T s \tag{4.10}$$

where q_T is the positive constant, $0<q_T<1$.

Substituting Equation (4.9) into Equation (4.10) yields to

$$\begin{aligned}&\Delta\mathbf{w}(t)-\Delta\mathbf{w}(t-1)=-q_T\Delta\mathbf{w}(t-1)\\&\Delta\mathbf{w}(t)=\alpha\Delta\mathbf{w}(t-1);\quad 0<\alpha=(1-q_T)<1\end{aligned} \tag{4.11}$$

The law introduced in (4.11) will assure $\lim_{t\to\infty}\Delta\mathbf{W}(t)=0$, and the convergence rate is decided by the appropriate choice of α.

A combination of the laws formulated in (4.7) and (4.11) will meet the objectives of both the parametric stabilisation and the cost convergence. Therefore, the learning algorithm (4.8) can obtain fast and global convergence.

The algorithm (4.8) is named as EABPM, which refers to the Extension of the ABP algorithm with Momentum. Other versions of the BP with momemtum schemes, the QuickProp and scaled conjugate gradient methods cannot apply in EABPM. The QuickProp algorithm indeed uses a crude approximation of the Hessian. Therefore, this algorithm will be diverged when error points are near a maximum of the error surface. For the scaled conjugate gradient method, the adaptive learning rate was determined by performing a line minimisation along the search direction.

The proposed algorithm for training the FNN in classification problems can be described as follows:

1. (Initialisation) Initialise all weights to small random values $W_{km}(0)$ and $\bar{W}_{nk}(0)$. Choose a small positive learning rate η, momentum coefficient α, maximum number of iterations $t_{\max}$ and a very small maximum tolerable error $E_{\max}$. Set the iteration number $t = 0$ and $\Delta W_{km}(0) = \Delta \bar{W}_{nk}(0) = 0$.

2. (Forward propagation) Select training pairs $\{\mathbf{x}^p, \mathbf{d}^p | p = 1,..., P\}$ from the training set, propagate the inputs forward through the network using the equation:

$$o_m{}^p = f_o(net_m) = f_o\left(\sum_{k=1}^{K} W_{km}(t)\bar{o}_k{}^p\right)$$

$$\bar{o}_k{}^p = f_h(net_k) = f_h\left(\sum_{n=1}^{N} \bar{W}_{nk}(t)x_n{}^p\right)$$

where $f_o(.)$ is the activation function for output node, $f_h(.)$ is the activation function for hidden node.

The criterion function of the error is calculated as

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[d_m{}^p - o_m{}^p\right]^2 \tag{4.12}$$

If E is not greater than $E_{\max}$, or the number of iterations exceeds a value $t_{\max}$, then the algorithm is stopped. Otherwise, go to step 3.

3. (Backward propagation) Compute the changes of the weights for the next iteration:

$$\Delta W_{km}(t) = \frac{\eta}{\left(\sum_{p=1}^{P}\delta_m{}^p\bar{o}_k{}^p\right)^2 + \left(\sum_{p=1}^{P}\bar{\delta}_k{}^p x_n{}^p\right)^2}\sum_{p=1}^{P}\delta_m{}^p\bar{o}_k{}^p + \alpha\Delta W_{km}(t-1)$$

$$\Delta \bar{W}_{nk}(t) = \frac{\eta}{\left(\sum_{p=1}^{P}\delta_m{}^p\bar{o}_k{}^p\right)^2 + \left(\sum_{p=1}^{P}\bar{\delta}_k{}^p x_n{}^p\right)^2}\sum_{p=1}^{P}\bar{\delta}_k{}^p x_n{}^p + \alpha\Delta \bar{W}_{nk}(t-1)$$

where

$$\delta_m{}^p = \left(d_m{}^p - o_m{}^p\right)\left(f_o'(net_{om}) + 0.01\right)$$

$$\bar{\delta}_k{}^p = \left(f_h'(net_{hk}) + 0.01\right)\sum_{m=1}^{M}\delta_m{}^p W_{km}(t)$$

Update the weights

$$W_{km}(t+1) = W_{km}(t) + \Delta W_{km}(t)$$

$$\bar{W}_{nk}(t+1) = \bar{W}_{nk}(t) + \Delta \bar{W}_{nk}(t)$$

Set $t = t+1$ and go to step 2.

4.2.3 Classification application using the sliding-mode-based neural networks

The XOR problem, a type of the binary problems, is the most popular benchmark at present. Thus it would be worthwhile to solve the XOR problem using the EABPM algorithm, as described in Section 4.2.2.

An FNN structure with $N=2,\ K=2,\ M=1$ is chosen for this task. The mean square error is defined as

$$E=\frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[d_m{}^p-o_m{}^p\right]^2$$

The maximum tolerable error $E_{\mathrm{m}}=0.002$ and the maximum epochs $l_m=3\times10^4$ are set. The initial weights are drawn at random between -0.1 to 0.1. In the first experiments, a first set of weights is randomly generated and kept constant while the learning parameters are tuned to obtain the fastest possible convergence. Apparently, the optimal learning parameters, which correspond with the fastest convergence of the NN, are obtained by a trial and error approach. Thus these parameters can reach different values for various applications. Secondly, each experiment is performed 50 times for 50 different sets of initial weights on various learning algorithms, including the BPM, ABP, and proposed EABPM algorithms. The simulation results are shown in Table 4.1, where the optimal learning parameters were obtained from the tunning process, % means the percentage of global convergence, which counts the percentage of trials (over 50 trials) that successfully converge within $l_m=3\times10^4$ epochs, R means the average number of epochs to converge over the converged trials.

Table 4.1: Performance comparison for the XOR problem.

Leaning algorithm	Optimal learning parameters	R	%
BPM	$\eta^*=0.95,\ \alpha^*=0.89$	2312.4	100
ABP	$\eta^*=0.01$	816	52
EABPM	$\eta^*=0.015,\ \alpha^*=0.06$	699.98	100

Discussion:

As seen from the results exhibited in Table 4.1, our proposed algorithm, EABPM, outperforms the BP, ABP approaches in terms of global convergence capability. Among the algorithms that have 100% of global converged trials, the proposed algorithm is really faster than BPM. Although ABP is faster than BPM, its global convergence capability is very poor. These results demonstrate that the proposed EABPM algorithm can guarantee fast and global convergence for training the FNN in the classification problems.

4.2.4 Sliding-mode-based neural network head-movement classifier for wheelchair control

Introduction

Powered wheelchairs are commonly the means of transport of people with lower-extremity disability. Conventional electrical wheelchairs are not always appropriate for mobility disabilities relating to quadriplegia. To overcome this problem associated with joystick control, various wheelchair-guidance interfaces, such as a sip and puff (Schimeisser & Seamone, 1979), chin controller (Schimeisser & Seamone, 1979), eyewink controller (Crisman et al., 1991), voice (Mazo et al., 1995; Rockland & Reisman, 1998) and head movement (Joseph & Nguyen, 1998; Nguyen et al., 2004a), have been designed.

Head movement is a natural form of pointing a certain direction, and can be used to replace the joystick while still allowing for similar control. A number of researchers have developed head-movement wheelchair control for severely disabled people. Coyle (1995) developed an ultrasonic non-contact head controller integrated with a small vocabulary voice recognition system for the wheelchairs. In Tew (1988), a photo quadrant sensor, which is made up of four photodiodes fabricated on one substrate, is used to determine the head movement.

An alternative telemetric head movement device using a tilt sensor and wireless technology was recently designed for the control of powered wheelchairs (Joseph & Nguyen, 1998). This system had many benefits for the disabled wheelchair users, such as comfortable and easy use, and adaptability to each individual user via neural networks (Taylor & Nguyen, 2003). In 2004, this head movement controller was successfully extended using embedded LINUX and neural networks (Nguyen et al., 2004a).

Powered wheelchair control using head movement commands

The whole powered wheelchair control system is described in Figure 4.3.

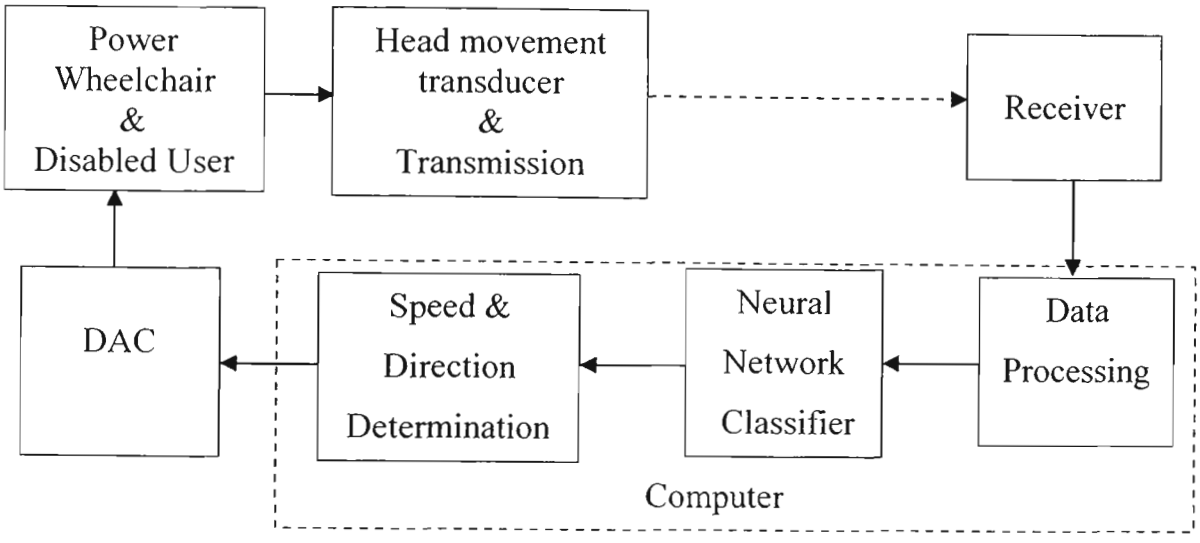


Figure 4.3: Diagram of the powered wheelchair control system using head movement commands.

When a user tips or nods her/his head for the desired direction, the output signals of a head movement transducer inside the user's cap are transmitted via the telemetry system. In the receiver, the signal is received and is then normalised to a value range of zero-to-one for the neural network classifier. The neural network's weights are trained to store vital information about the patterns.

In the real-time system, the trained neural network can classify the input signal into intended commands corresponding to the user's head movements. Afterwards, a simple set of heuristics is utilised to determine the corresponding speed and direction of the wheelchair. These speed and direction outputs are in turn converted to the 0 to 5V

output via a digital to analogue converter (DAC). Therefore, the powered wheelchair is controlled easily by the user's head movement.

A number of disabled people voluntarily participated in the test with approval from the UTS Human Research Ethics Committee and informed consent from the volunteers. They had high-level spinal cord injuries with C5 lesion (having difficulty in using their arms freely) and C4 lesion (cannot move their arms at all). Four head nods in the forward, backward, left and right directions and a sideways shake of the head for stopping the wheelchair are recorded. The head movement is sampled at a rate of 20Hz upon a request from a computer, and the movement transducer signals with regard to two coordinates x and y are measured, wherein the coordinate axis x is horizontal and straight out in front of the user, and the coordinate axis y is horizontal and perpendicular to the axis x. A sliding window of 20 samples from each coordinate produces each pattern with 40 inputs. Figures 4.4 and 4.5 show the posture of some sample data collected from users with C5 and C4 lesion, respectively.

As seen in Figures 4.4 (a) and 4.4 (b), when a user with C5 lesion inclines or moves back his/her head, the transducer signals with regard to coordinate x have pulse postures, and those with regard to coordinate y are rather flat. When the user's head is tilted left or right, the signals with regard to coordinate x are virtually flat, and those of the coordinate axis y are fluctuated, as in Figures 4.4 (c) and 4.4 (d). Two axis signals in Figure 4.4 (e) are changed as the user shakes the head. In addition, the data signals in Figure 4.5 are different with those in Figure 4.4 because the disabled with C4 lesion have difficulty in moving the head. Nevertheless, basing on the nonlinear classification capability of the neural networks, the head-movement neural classifier can effectively work with different disabled people.

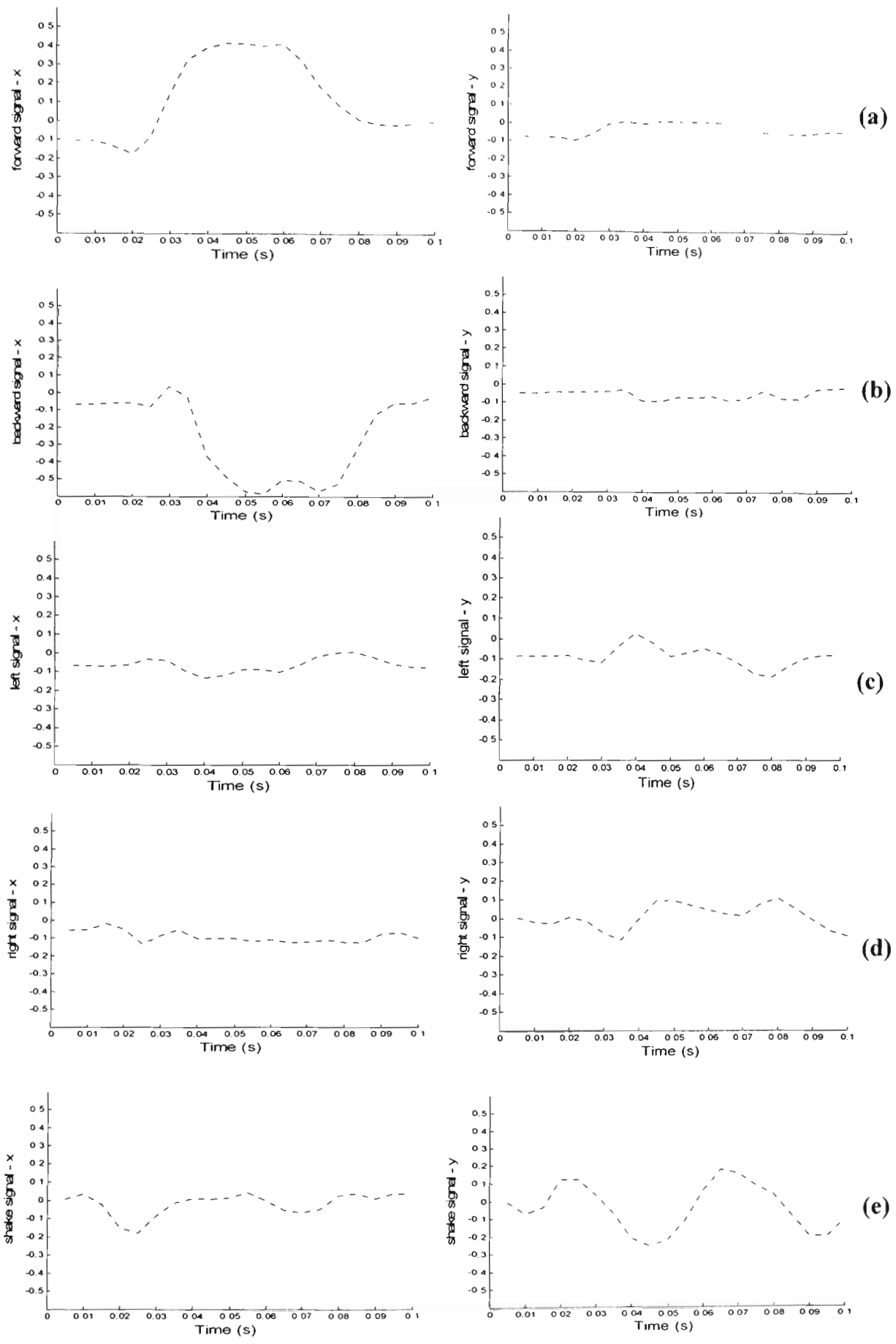


Figure 4.4: The posture of two axis data collected from a C5 user with the (a) forward, (b) backward, (c) left, (d) right, (e) stop commands.

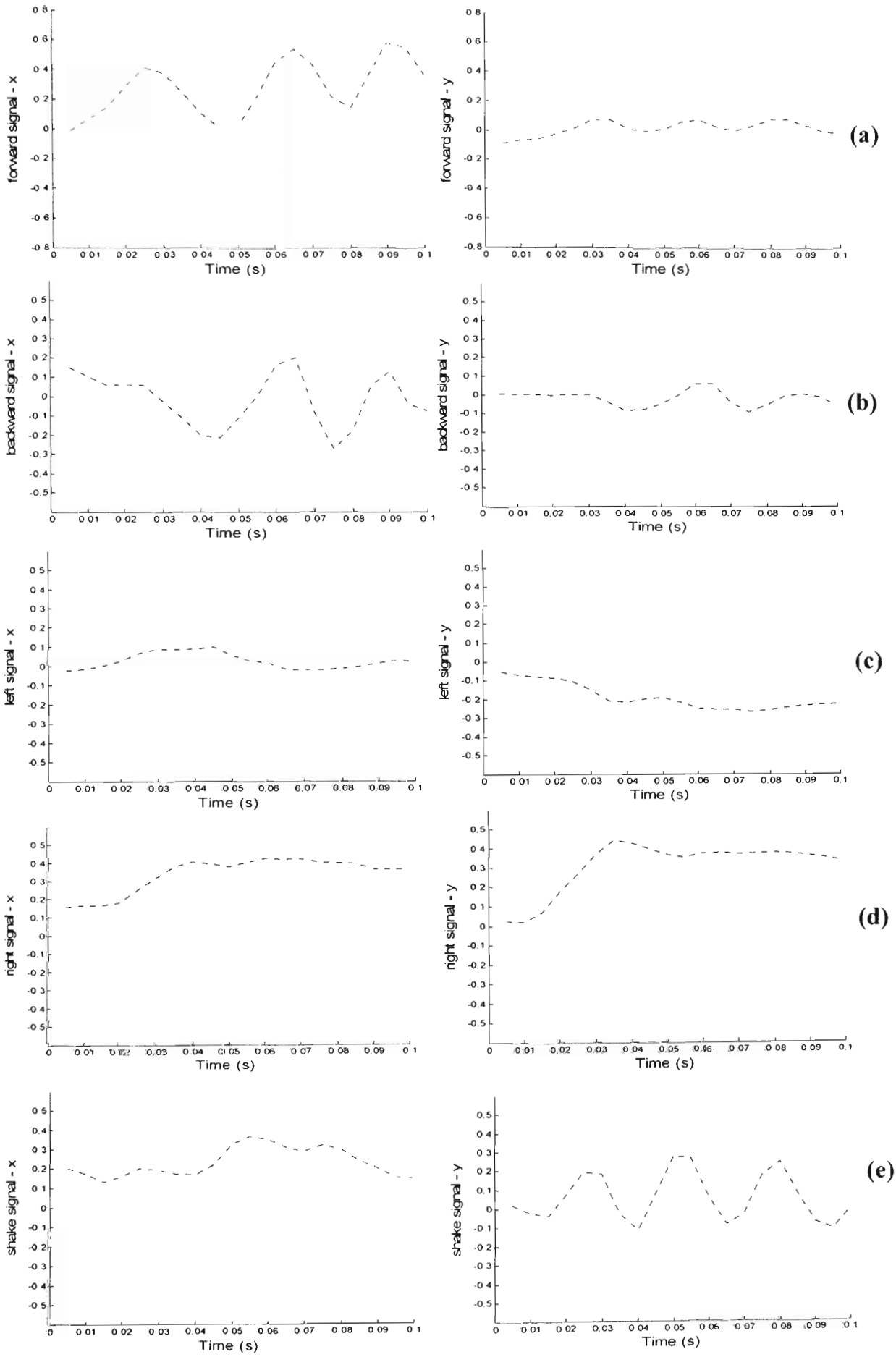


Figure 4.5: The posture of two axis data collected from a C4 user with the (a) forward, (b) backward, (c) left, (d) right, (e) stop commands.

Sliding-mode-based neural network classifier for head movement commands

The feedforward neural network with one hidden layer is used for the head-movement classifier. The input layer consists of 41 nodes, with the first 20 being the input from the coordinate x, the next 20 being the input from the coordinate y and an augmentation input of 1. The output layer consists of five nodes corresponding to each of five commands: forward, backward, left, right and stop. The target value of each output is either 1.0 or 0.0. By a trial and error approach, the hidden node number of 9 is chosen to obtain better generalisation property. A bias of 1 is also augmented to the hidden layer.

The EABPM learning algorithm presented in Section 4.2.2 is used for training the feedforward neural network. Moreover, a 0.05–0.95 threshold with margin criterion is chosen in the training process. Any output over 0.95 is set to a one and anything under 0.05 is set to a zero before back propagation occurs.

For improving generalisation of the network, a non-convergent method described in Finnoff et al. (1993) can be utilised to avoid "overfitting". The whole data set is divided into three parts: training set, validation set and test set. The training set is used for training the NN's weights. The validation set is utilised to stop the training process when the validation error starts to increase. The test set is used to estimate the expected performance (generalisation) of the trained network on new data.

The proposed training procedure is shown in the following steps:

1. (Initialisation) Initialise all weights to small random values $W_{km}(0)$ and $\bar{W}_{nk}(0)$. Choose a small positive learning rate η, maximum epoch value l_m, large value E_{vmin} and very small maximum tolerable error E_m. Set number of epochs $l = 0$, $count = 0$.

2. (Forward propagation) Select training pairs $\{\mathbf{x}^p, \mathbf{d}^p | p = 1, ..., P\}$ from the training set and propagate the inputs forward through the network:

$$o_m{}^p = f_o\left(net_m\right) = f_o\left(\sum_{k=1}^{K} W_{km}(t)\,\overline{o}_k{}^p\right)$$

$$\overline{o}_k{}^p = f_h\left(net_k\right) = f_h\left(\sum_{n=1}^{N} \overline{W}_{nk}(t)\,x_n{}^p\right)$$

$$e_m{}^p = d_m{}^p - o_m{}^p$$

$$\text{If } \left(\left(-0.05 > e_m{}^p\right) \text{ or } \left(e_m{}^p < 0.05\right)\right) \text{ then set } e_m{}^p = 0$$

Finally, calculate the mean square error $E = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left[d_m{}^p - o_m{}^p\right]^2$.

Check whenever the total error is acceptable within l_m epochs. If $E \le E_m$ or $l \ge l_m$, then go to step 6. Otherwise, go to step 3.

3. (Backward propagation) Compute the changes of the weights for the next iteration

$$\Delta W_{km}(t) = \frac{\eta}{\left(\sum_{p=1}^{P}\delta_m{}^p\overline{o}_k{}^p\right)^2 + \left(\sum_{p=1}^{P}\overline{\delta}_k{}^p x_n{}^p\right)^2}\sum_{p=1}^{P}\delta_m{}^p\overline{o}_k{}^p + \alpha\Delta W_{km}(t-1)$$

$$\Delta \overline{W}_{nk}(t) = \frac{\eta}{\left(\sum_{p=1}^{P}\delta_m{}^p\overline{o}_k{}^p\right)^2 + \left(\sum_{p=1}^{P}\overline{\delta}_k{}^p x_n{}^p\right)^2}\sum_{p=1}^{P}\overline{\delta}_k{}^p x_n{}^p + \alpha\Delta \overline{W}_{nk}(t-1)$$

where

$$\delta_m{}^p = \left(d_m{}^p - o_m{}^p\right)\left(f_o'\left(net_{om}\right) + 0.01\right)$$

$$\overline{\delta}_k{}^p = \left(f_h'\left(net_{hk}\right) + 0.01\right)\sum_{m=1}^{M}\delta_m{}^p W_{km}(t)$$

Update the weights

$$W_{km}(t+1) = W_{km}(t) + \Delta W_{km}(t)$$
$$\bar{W}_{nk}(t+1) = \bar{W}_{nk}(t) + \Delta \bar{W}_{nk}(t)$$

4. (Validation procedure) Select validation pairs $\{\mathbf{x}_v^p, \mathbf{d}_v^p \mid p = 1,..., P_v\}$ from the validation set and propagate the inputs forward through the network:

$$o_m^p = f_o(net_m) = f_o\left(\sum_{k=1}^{K} W_{km}(t)\bar{o}_k^p\right)$$
$$\bar{o}_k^p = f_h(net_k) = f_h\left(\sum_{n=1}^{N} \bar{W}_{nk}(t)x_{vn}^p\right)$$
$$e_m^p = d_{tm}^p - o_m^p$$

If $\left(\left(-0.05 > e_m^p\right) \text{ or } \left(e_m^p < 0.05\right)\right)$ then set $e_m^p = 0$.

Finally, calculate the mean square error $E_v = \frac{1}{2}\sum_{p=1}^{P_v}\sum_{m=1}^{M}\left[d_{vm}^p - o_m^p\right]^2$.

If $E_v > E_{v\min}$ then

Save the weights at the minimum validation error $\begin{matrix} W_{km\min}(t) = W_{km}(t) \\ \bar{W}_{nk\min}(t) = \bar{W}_{nk}(t) \end{matrix}$,

and set $E_{v\min} = E_v$.

5. Update training parameters, set $k = k+1$, $l = l+1$ and go to step 2.

6. If $l \geq l_m$, terminate the training process and report a false trial to converge. Otherwise, go to step 7.

7. (Test procedure) Set the pattern number $p = 1$, select a pattern from the test set $\{\mathbf{x}_v^p, \mathbf{d}_v^p \mid p = 1,..., P_t\}$ and propagate the inputs forward through the network:

$$o_m{}^p = f_o(net_m) = f_o\left(\sum_{k=1}^{K} W_{km}(t)\,\bar{o}_k{}^p\right)$$

$$\bar{o}_k{}^p = f_h(net_k) = f_h\left(\sum_{n=1}^{N} \bar{W}_{nk}(t)\,x_{tn}{}^p\right)$$

If $\bar{o}_k^p > 0.75$ then set $\bar{o}_k^p = 1$

Otherwise, set $\bar{o}_k^p = 0$.

If all $\bar{o}_k^p = d_{tm}{}^p$ then conclude the pattern is classified well, $count = count + 1$.

Set $p = p + 1$.

If $p <= P_t$, where P_t is the number of patterns in the test set, then go to step 7

Otherwise, terminate the training process,

and report the accuracy $= \dfrac{count}{P_t} \times 100\%$.

4.2.5 Experimental Results of the Sliding-mode-based Neural Network Classifier for Head Movement Commands

For comparison of the performances, a number of learning algorithms in Table 4.2 are tested on the head-movement neural classifier. All the learning algorithms are in the batch mode.

A sensible strategy, as presented in Wessels and Barnard (1992), is used for choosing the magnitudes of the initial weights to avoid the premature saturation problem. In the neural network head-movement classifier, initial weights were drawn at random between -0.47 and 0.47. The maximum epoch value is set to 3×10^4. With the chosen

criterion function, the maximum tolerable error E_m can be set to 0.00125, which means that all output errors $e_m^p; m = 1,...,M; p = 1,...,P$ must approach zero.

Table 4.2: Description of all learning algorithms used in the neural network head-movement classifier.

Learning algorithm	Abbreviation	Algorithm parameters
Backpropagation with momentum in batch mode	BPM	η learning rate α momentum coefficient
Adaptive backpropagation	ABP	η learning rate
RPROP	RPROP	η^+ increase factor η^- decrease factor
Magnified gradient function method	MGFPROP	η learning rate α momentum coefficient s magnification factor
The proposed EABPM algorithm	EABPM	η learning rate α momentum coefficient

In the experiments, the data used for training the neural classifier is randomly divided into three sets. The training set includes 30 patterns of each command class, the validation set contains other 30 patterns of each command class, and other patterns, including 24 patterns of each command class, are used in the test set.

The first experiments are the tuning process to obtain the optimal parameters of each learning algorithm. The first set of weights is randomly generated and kept constant while the learning parameters are tuned to obtain the fastest possible convergence. Because this is a trial and error approach, the optimal learning parameters in Table 4.3 for the head-movement classifier are different from those of the XOR problem, as shown in Table 4.1.

In the next experiments, 50 initial random weight settings are chosen, and each of the algorithms is executed using these sets of weights with the optimal learning parameters obtained from the tuning process.

Table 4.3 shows the experimental results for the 41-10-5 network, where % means the percentage of global convergence, which counts the percentage of trials (over 50 trials) that successfully converge within $l_m = 3\times10^4$ epochs, R means the average number of epochs to converge over the converged trials, and ACR means the average accuracy of the network when it is tested in the test set over the converged trials.

Table 4.3: Performance comparison for the neural network head-movement classifier.

Learning algorithm	Optimal parameters	%	R	ACR (%)
BPM	$\eta^* = 0.28$, $\alpha^* = 0.1$	44	13350	86.13±5.84
ABP	$\eta^* = 0.51$	72	680.14	84.31±6.93
RPROP	$\eta^+ = 1.58$, $\eta^- = 0.74$	68	641.79	78.1±9.49
MGFPROP	$\eta^* = 0.28, \alpha^* = 0.1, s^* = 5$	98	2305.4	81.39±13.5
EABPM	$\eta^* = 0.42, \alpha^* = 0.78$	100	423.84	91.35±5.0

Discussion:

In Table 4.3, the EABMP has 100% global convergence capability while the BPM, ABP and RPROP methods fail to converge in more than 28% of the trials. Additionally, the average epochs needed for the EABMP are approximately 423.84, which is the best result, compared with the other algorithms. Surprisingly, the percentage of correctly

classified patterns in the test set for the EABPM is 91.35±5.0, which is the highest accuracy result, compared with the other algorithms. Thus, our proposed algorithm, EABM, outperforms the BPM, ABP, RPROP and MGFPROP approaches in terms of the convergence rate, global convergence capability and classified accuracy.

4.3 Chattering-free Sliding-mode-based Neural Networks for Control Applications

4.3.1 Feedforward neural network structure for control applications

Consider a class of feedforward NN with one hidden layer and one output node, as shown in Figure 4.6. Cybenko (1989), Hornik et al. (1989) and Funahashi (1989) independently proved that this kind of feedforward NNs can approximate any nonlinear function with an arbitrary desired degree of accuracy. Currently, this feedforward NN has been used in the neural controller design for replacing any unknown nonlinear function (He, 2002; Hovakimyan et al., 2002), or used as a direct feedback neural controller (Daosud et al., 2005; Hayakawa et al., 2005; Nayeri et al., 2004). In a control system, the order of the system state is often represented by n. Therefore, the following definitions will be used for this FNN.

The multilayer neural network, as shown in Figure 4.6, consists of $n+1$ input nodes, $m+1$ hidden nodes, and one output node. The hyperbolic tangent, $f_h(v)=\frac{1-e^{-v}}{1+e^{-v}}$, is used as the activation function for the hidden nodes, and the linear function, $f_o(v)=v$, is used as the activation function for the output node.

$\mathbf{x}=[x_1,...,x_n]^T$ is the vector of the time-varying input signals. A constant input of 1, affecting the bias, is assigned to the vector of augmented inputs $\tilde{\mathbf{x}}=(x_1,...,x_n,1)^T$.

$\bar{\mathbf{W}}=\begin{bmatrix} \bar{W}_{11} & \cdots & \bar{W}_{1n} \\ \vdots & \ddots & \vdots \\ \bar{W}_{m1} & \cdots & \bar{W}_{mn} \end{bmatrix}$ is the matrix of the time-varying connections' weights between the neurons in the input layer and the neurons in the hidden layer.

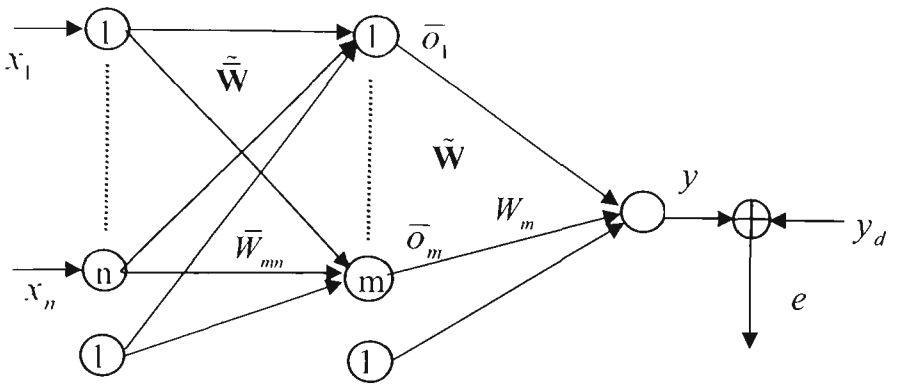


Figure 4.6: Structure of a feedforward neural network with one output neuron.

$\tilde{\bar{\mathbf{W}}} = \begin{bmatrix} \bar{W}_{11} & \cdots & \bar{W}_{1(n+1)} \\ \vdots & \ddots & \vdots \\ \bar{W}_{m1} & \cdots & \bar{W}_{m(n+1)} \end{bmatrix}$ is the matrix of augmented weights by including the bias weights components $\bar{W}_{i(n+1)};\ i = 1,...,m$.

$\bar{\mathbf{o}} = [\bar{o}_1,...,\bar{o}_m]^T$; $\bar{o}_i = f_{hi}(net_{hi})$; $net_{hi} = \sum_{j=1}^{n} \bar{W}_{ij} x_j + \bar{W}_{i(n+1)}$ is the vector of the output signal of the neurons in the hidden layer. A constant input of 1, affecting the bias, is assigned to the vector of augmented outputs $\tilde{\bar{\mathbf{o}}} = [\bar{o}_1,...,\bar{o}_m,1]^T$. $\mathbf{W} = [W_1,...,W_m]$ is the row vector of the connections' weights between the neurons in the hidden layer and the output node, and $\tilde{\mathbf{W}} = [W_1,...,W_m,W_{m+1}]$ is the row vector of augmented weights by including the bias weights component W_{m+1}. The output signal $y(t)$ can be calculated as

$$y(t) = \tilde{\mathbf{W}}\tilde{\bar{\mathbf{o}}} = \sum_{i=1}^{m+1} W_i \bar{o}_i = \sum_{i=1}^{m} W_i f_{hi}\left(\sum_{j=1}^{n} \bar{W}_{ij} x_j + \bar{W}_{i(n+1)} \right) + W_{m+1}. \qquad (4.13)$$

The scalar $y_d(t)$ represents the time-varying desired output of the NN.

4.3.2 Chattering-free sliding-mode-based learning algorithm for the FNN

The zero value of the learning error $e(t)$ is defined as a sliding surface:

$$s(e) = e(t) = y(t) - y_d(t) = 0, \tag{4.14}$$

and the error criterion function is defined as

$$E = \frac{1}{2}e^2. \tag{4.15}$$

The research objective is to propose a novel learning algorithm so that the learning error $e(t)$ will quickly and globally converge to zero. Moreover, the FNN will be used for control purposes, thus the incremental learning approach should be considered for on-line adaptation applications.

Motivated from the research of Giordano et al. (2004), the first learning algorithm for the weight vector $\tilde{\mathbf{W}}$ and the weight matrix $\tilde{\tilde{\mathbf{W}}}$ is proposed as:

$$\begin{aligned} \dot{\tilde{\mathbf{W}}} &= -\frac{\eta}{\left\| \tilde{\tilde{\mathbf{o}}}^T \right\|^2} e \tilde{\tilde{\mathbf{o}}}^T \\ \dot{\tilde{\tilde{\mathbf{W}}}} &= -\frac{\eta}{\left\| \mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T \right\|^2} e \mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T \end{aligned} \tag{4.16}$$

where $\|.\|$ denoted Euclidean norm for a vector and Frobenius norm for a matrix.

In the case when the same input $\mathbf{x}$ is presented to the NN, from (4.13), (4.15) and (4.16), the change of E is obtained as

$$\dot{E} = \sum_{i=1}^{m+1} \frac{\partial E}{\partial W_i} \dot{W}_i + \sum_{i=1}^{m} \sum_{j=1}^{n+1} \frac{\partial E}{\partial \bar{W}_{ij}} \dot{\bar{W}}_{ij} = -\eta E. \tag{4.17}$$

From the chattering-free SMC strategy, the condition (4.17) guarantees that the error $e(t)$ globally converges to zero, and the convergence rate is decided by the appropriate choice of η.

However, when the FNN is applied in the control system, the input vector $\mathbf{x}$ is time-varying. Therefore, the change of E becomes

$$\dot{E} = \sum_{i=1}^{m+1} \frac{\partial E}{\partial W_i} \dot{W}_i + \sum_{i=1}^{m} \sum_{j=1}^{n+1} \frac{\partial E}{\partial \bar{W}_{ij}} \dot{\bar{W}}_{ij} + \sum_{j=1}^{n} \frac{\partial E}{\partial x_n} \dot{x}_n = -\eta E + \sum_{j=1}^{n} \frac{\partial E}{\partial x_n} \dot{x}_n , \qquad (4.18)$$

and the system stability is difficult to obtain. This is the problem of the system stability proof demonstrated by Ng (1997).

For a class of linear systems with unmatched uncertainties, the feedforward NN controller with the learning algorithm (4.16) is proven to guarantee stability and robustness. These results with some simulations were presented in Nguyen (2005).

A problem of the learning algorithm (4.16) is caused by the zero reachability of the derivative of the activation function of the hidden nodes. The small derivative value can lead to a very large change of weights and diverges the algorithm convergence. To overcome this problem, the learning algorithm (4.16) is then developed to another version as follows:

$$\begin{aligned} \dot{\mathbf{W}} &= -\frac{\eta}{\left\|\tilde{\bar{\mathbf{o}}}^T\right\|^2 + \left\|\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\right\|^2} e \tilde{\bar{\mathbf{o}}}^T \\ \dot{\bar{\mathbf{W}}} &= -\frac{\eta}{\left\|\tilde{\bar{\mathbf{o}}}^T\right\|^2 + \left\|\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\right\|^2} e \mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T \end{aligned} \qquad (4.19)$$

where $\mathbf{F}_h' = \begin{bmatrix} f_{h1}' & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f_{hm}' \end{bmatrix}$, f_{hi}', $(i = 1,...,m)$ is the derivative of $\bar{o}_i$ with regard to $\left(\sum_{j=1}^{n} \bar{W}_{ij} x_j + \bar{W}_{i(n+1)} \right)$.

Because the dominator $\left\|\tilde{\bar{\mathbf{o}}}^T\right\|^2 + \left\|\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\right\|^2$ in (4.19) is larger than or equal to one, the problem of the zero slope value is eliminated completely.

In the case when the same input $\mathbf{x}$ is presented to the NN, the learning algorithm (4.19) also leads to the change of E as:

$$\dot{E} = \sum_{i=1}^{m+1} \frac{\partial E}{\partial W_i} \dot{W}_i + \sum_{i=1}^{m} \sum_{j=1}^{n+1} \frac{\partial E}{\partial \bar{W}_{ij}} \dot{\bar{W}}_{ij} = -\eta E .$$

This equation is the sliding condition of the chattering-free sliding mode control approach. Therefore, the learning algorithm (4.19) is a variation of BP derived from the chattering-free sliding mode control theory, and is named as CFSMBP.

In the next section, the feedforward NN with the learning algorithm (4.19) acts as a direct adaptive controller for a class of uncertain single-input single-output systems.

4.3.3 Control applications of the chattering-free sliding-mode-based neural networks

Consider the following continuous-time uncertain system:

$$\begin{aligned} \dot{\mathbf{x}} &= (\mathbf{A} + \Delta\mathbf{A})\mathbf{x} + (\mathbf{B} + \Delta\mathbf{B})u, \qquad \mathbf{x}(0) = \mathbf{x}_0 \\ y &= \mathbf{C}\mathbf{x} \end{aligned} \tag{4.20}$$

where $\mathbf{x} = [x_1 \quad \cdots \quad x_n]^T \in \mathbf{R}^n$ is the system state vector, $u \in \mathbf{R}$ is the control input, $\mathbf{A} \in \mathbf{R}^{n \times n}$ is the system matrix, $\mathbf{B} \in \mathbf{R}^{n \times 1}$ is the input matrix and $\mathbf{C} \in \mathbf{R}^{1 \times n}$ is the output matrix so that $\mathbf{A}, \mathbf{C}$ satisfy the observability condition, and $\mathbf{A}, \mathbf{B}$ are in the controllable canonical form:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & . & . \\ . & 0 & 1 & . \\ . & . & . & . \\ a_1 & . & . & a_n \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0 \\ . \\ 0 \\ b \end{bmatrix}, \tag{4.21}$$

And $\Delta\mathbf{A} \in \mathbf{R}^{n \times n}, \Delta\mathbf{B} \in \mathbf{R}^n$ represent parameter uncertainties in the plant model.

Assumption 4.1: *The uncertain matrices* $\Delta\mathbf{A}, \Delta\mathbf{B}$ *are assumed to be bounded:*

$$\begin{aligned} \|\Delta \mathbf{A}\| &\le \rho_A \\ \|\Delta \mathbf{B}\| &\le \rho_B \end{aligned} \tag{4.22}$$

The control objective is to regulate the state $\mathbf{x}$ in (4.20) to zero, and this is obtained via a two-stage control design. In the first stage, a sliding hyperplane is designed. This sliding function is then used as the training signal for a neural network controller designed in the second stage.

Design of the sliding hyperplane

First define the sliding function as:

$$s = \delta\sigma + \dot{\sigma}, \quad \delta > 0, \quad \delta\sigma(0) = -\dot{\sigma}(0) \tag{4.23}$$

where δ is a scalar,

$$\begin{aligned} \sigma &= \mathbf{H}\mathbf{x} \\ &= h_1 x_1 + h_2 x_2 + \cdots + x_n \end{aligned} \tag{4.24}$$

and the row vector $\mathbf{H} = [h_1, ..., h_{n-1}, 1]$ is designed so that the polynomial (4.24) has only left-half plane characteristic roots.

Assumption 4.2: *There exists a positive scalar* g_0 *satisfied:*

$$g_0 \le \mathbf{H}(\mathbf{B} + \Delta\mathbf{B}) \tag{4.25}$$

With the controllable canonical system (4.21), the equation (4.23) can be cast into a state-space form as:

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}s \tag{4.26}$$

where matrices $\mathbf{F} \in \mathbf{R}^{n\times n}$ and $\mathbf{G} \in \mathbf{R}^{n}$ are in the canonical form:

$$\mathbf{F}=\begin{bmatrix} 0 & 1 & . & . \\ . & 0 & 1 & . \\ . & . & . & . \\ -\delta h_1 & -\delta h_2 - h_1 & . & -\delta - h_{n-1} \end{bmatrix} \quad \mathbf{G}=\begin{bmatrix} 0 \\ . \\ 0 \\ 1 \end{bmatrix} \tag{4.27}$$

Theorem 4.1: *If* $\delta_i = -\lambda_n$ *and* $\mathbf{H}$ *is drawn directly from the following equation*

$$h_1 + \cdots + h_{n-1}s^{n-2} + s^{n-1} = (s-\lambda_1)\ldots(s-\lambda_{n-1})$$

then $\lambda_1, \lambda_2, \cdots, \lambda_n$ *are the eigenvalues of the system dynamics in the hyperplane* $s = 0$.

Proof: In the hyperplane $s = 0$, $\dot{\mathbf{x}} = \mathbf{F}\mathbf{x}$, and the eigenvalues of the system dynamics can be found from

$$\det[s\mathbf{I} - \mathbf{F}] = 0$$

$$\Leftrightarrow \det\begin{bmatrix} s & -1 & 0 & \cdots & 0 \\ 0 & s & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 \\ \delta h_1 & \delta h_2 + h_1 & \delta h_3 + h_2 & \cdots & s + \delta + h_{n-1} \end{bmatrix} = 0$$

$$\Leftrightarrow (s+\delta)\left(h_1 + \cdots + h_{n-1}s^{n-2} + s^{n-1}\right) = 0$$

Therefore, if $h_1 + \cdots + h_{n-1}s^{n-2} + s^{n-1} = (s-\lambda_1)\ldots(s-\lambda_{n-1})$ and $\delta = -\lambda_n$, then $\lambda_1, \lambda_2, \cdots, \lambda_n$ are the eigenvalues of the system dynamics in the hyperplane $s = 0$. ■

Remark 4.1: By defining the sliding function with the specific initial condition (4.23), the sliding function attains zero at the initial instant, $s(0) = 0$. If the control signal can keep system states on the hyperplane $s = 0$, then $\lambda_1, \lambda_2, \cdots, \lambda_n$ are the desired eigenvalues of the closed-loop system.

Remark 4.2: The desired poles $\lambda_1, \lambda_2, \cdots, \lambda_n$ of the closed-loop system always locate on the left-half plane, thus matrix $\mathbf{F}$ is a stable, and there exists a symmetric positive definite matrix $\mathbf{P} \in \mathbf{R}^{n \times n}$ satisfying the Lyapunov equation:

$$\mathbf{PF} + \mathbf{F}^T\mathbf{P} = -\mathbf{I} \tag{4.28}$$

Lemma 4.1: *For the dynamic system (4.26), the following relationship is always obtained:*

$$\|\mathbf{x}\| \le 2\|\mathbf{PG}\||s|. \tag{4.29}$$

Proof: Suppose that $\|\mathbf{x}\| > 2\|\mathbf{PG}\||s|$.

Choose a Lyapunov function $V_1 = \mathbf{x}^T\mathbf{Px}$.

The change of V_1 is obtained from Equations (4.26), (4.28) as:

$$\begin{aligned}\dot{V}_1 &= \dot{\mathbf{x}}^T\mathbf{Px} + \mathbf{x}^T\mathbf{P}\dot{\mathbf{x}} \\ &= -\mathbf{x}^T\mathbf{x} + 2\mathbf{x}^T\mathbf{PG}s\end{aligned}$$

If $\|\mathbf{x}\| > 2\|\mathbf{PG}\||s|$, then $\dot{V}_1 < 0$ and all system states $\mathbf{x}$ will converge to zero. Consequently, the inequality (4.29) is obtained. ■

Remark 4.3: The inequality (4.29) is often used in the popular boundary layer method for smoothing the sliding-mode control signal. In this continuous sliding mode controller, the system state is forced into the boundary layer $B(\mathbf{x}) = \{\mathbf{x} \| |s| \le \phi\}$, where ϕ denotes the thickness of the boundary layer. As a result, it is obtained from Lemma 4.1 that $\|\mathbf{x}\| \le 2\|\mathbf{PG}\|\phi$.

Also from Lemma 4.1, the system state asymptotically converges to zero if $s = 0$. Therefore, an NN controller designed in the next stage will drive the sliding variable s to zero.

Design of the neural network controller

The feedforward neural network defined in Section 4.3.1 is utilised in the neural controlled system, as shown in Figure 4.7.

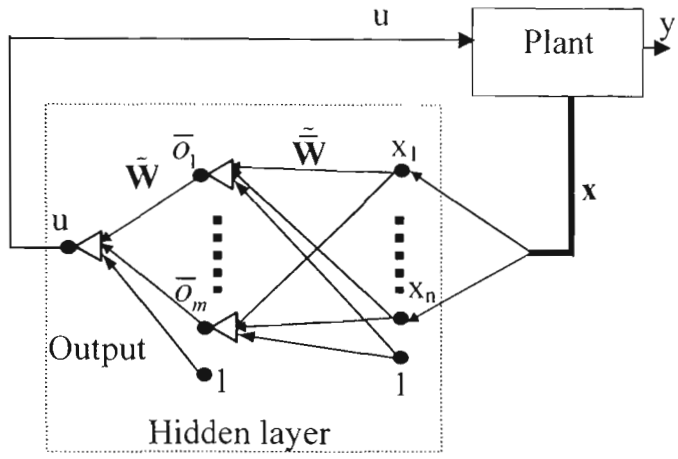


Figure 4.7: Structure of the proposed neural control system

Assumption 4.3: *Due to the physical constraints, the magnitudes of* $\mathbf{W}$ *and* $\bar{\mathbf{W}}$ *are assumed to be bounded by*

$$\|\mathbf{W}\| \le B_W, \qquad \|\bar{\mathbf{W}}\| \le B_{\bar{W}} \tag{4.30}$$

The output of the neural network is also the control signal u defined as:

$$u(\mathbf{x}) = \tilde{\mathbf{W}}\tilde{\mathbf{o}} = \sum_{i=1}^{m+1} W_i \bar{o}_i = \sum_{i=1}^{m} W_i f_{hi}\left(\sum_{j=1}^{n} \bar{W}_{ij} x_j + \bar{W}_{i(n+1)}\right) + W_{m+1} \tag{4.31}$$

As in Remark 4.3, the control objective is to regulate the sliding variable s to zero, thus the neural network can be trained on-line to minimise a cost function

$$V_2 = \tfrac{1}{2} s^2 \tag{4.32}$$

Substituting Equations (4.20) and (4.24) into Equation (4.23) yields:

$$s = \delta\sigma + \dot{\sigma} = \delta\mathbf{H}\mathbf{x} + \mathbf{H}\left(\mathbf{A} + \Delta\mathbf{A}\right)\mathbf{x} + \mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)u \tag{4.33}$$

$$\begin{aligned}\frac{\partial s}{\partial u} &= \frac{\partial}{\partial u}\left[\mathbf{H}\left(\mathbf{A} + \Delta\mathbf{A}\right)\mathbf{x} + \delta\mathbf{H}\mathbf{x} + \mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)u\right] \\ &= \mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)\end{aligned} \tag{4.34}$$

The gradients of V_2 with regard to the weight matrix or vector are calculated as follows:

$$\frac{\partial V_2}{\partial \tilde{\mathbf{W}}} = \frac{\partial V_2}{\partial s}\frac{\partial s}{\partial u}\frac{\partial u}{\partial \tilde{\mathbf{W}}} = s\mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)\tilde{\bar{\mathbf{o}}}^T \tag{4.35}$$

$$\frac{\partial V_2}{\partial \tilde{\bar{\mathbf{W}}}} = \frac{\partial V_2}{\partial s}\frac{\partial s}{\partial u}\frac{\partial u}{\partial \bar{\mathbf{o}}}\frac{\partial \bar{\mathbf{o}}}{\partial \tilde{\bar{\mathbf{W}}}} = s\mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T \tag{4.36}$$

where $\mathbf{F}_h' = \begin{bmatrix} f_{h1}' & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f_{hm}' \end{bmatrix}$, f_{hi}', $\left(i = 1, \ldots, m\right)$ is the derivative of $\bar{o}_i$ with regard to $\left(\sum_{j=1}^{n} \bar{W}_{ij} x_j + \bar{W}_{i(n+1)}\right)$.

The CFSMBP learning algorithm of the weight vector $\tilde{\mathbf{W}}$ and the weight matrix $\tilde{\bar{\mathbf{W}}}$ is presented as follows:

$$\begin{aligned}\dot{\tilde{\mathbf{W}}} &= -\frac{\eta}{\left\|\tilde{\bar{\mathbf{o}}}^T\right\|^2 + \left\|\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\right\|^2} s\tilde{\bar{\mathbf{o}}}^T \\ \dot{\tilde{\bar{\mathbf{W}}}} &= -\frac{\eta}{\left\|\tilde{\bar{\mathbf{o}}}^T\right\|^2 + \left\|\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\right\|^2} s\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\end{aligned} \tag{4.37}$$

where η is a learning rate to be chosen in (4.38).

Theorem 4.2: *Consider the closed-loop system consisting of the dynamics (4.20), the controller (4.31) and the training algorithm (4.37), if Assumptions 4.1, 4.2 and 4.3 are satisfied and η is chosen as*

$$\eta \geq \frac{k_{\max}}{g_0} C_s \tag{4.38}$$

where $k_{\max} = \|\mathbf{HA}\| + \|\mathbf{H}\|\left(\rho_A + \delta + 0.5\rho_B B_{\bar{W}} B_W\right) + 0.5bB_{\bar{W}} B_W$, $C_s = 2\delta\|\mathbf{PG}\| + \|\mathbf{H}\|^{-1}$, *then the closed-loop system is asymptotically stable.*

Proof: The derivative of the function (4.32) is attained as

$$\dot{V}_2 = \sum_{i=1}^{m+1} \frac{\partial V_2}{\partial W_i} \dot{W}_i + \sum_{i=1}^{m} \sum_{j=1}^{n+1} \frac{\partial V_2}{\partial \bar{W}_{ij}} \dot{\bar{W}}_{ij} + \sum_{j=1}^{n} \frac{\partial V_2}{\partial x_j} \dot{x}_j$$

From Equations (4.35), (4.36) and (4.37), it is obtained as:

$$\begin{aligned} \dot{V}_2 &= -\eta\left[\mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)\right] s^2 + s\frac{\partial s}{\partial \mathbf{x}}\dot{\mathbf{x}} \\ \dot{V}_2 &\leq -\eta\mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right) s^2 + |s|\left\|\frac{\partial s}{\partial \mathbf{x}}\right\|\|\dot{\mathbf{x}}\| \end{aligned} \tag{4.39}$$

From Equations (4.31) and (4.33), it is obtained as:

$$\frac{\partial s}{\partial \mathbf{x}} = \mathbf{H}\left(\mathbf{A} + \Delta\mathbf{A}\right) + \delta\mathbf{H} + \mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)\mathbf{W}\mathbf{F}_h'\bar{\mathbf{W}} \tag{4.40}$$

Because $f_{hi}' = \frac{d}{dv}\left(\frac{1-e^{-v}}{1+e^{-v}}\right) = \frac{1}{2}\left(1 - f_{hi}^2\right) \leq 0.5, \quad i = 1, \ldots, m$, and $\mathbf{HB} = b$,

$$\left\|\frac{\partial s}{\partial \mathbf{x}}\right\| \leq \|\mathbf{HA}\| + \|\mathbf{H}\|\left(\rho_A + \delta + 0.5\rho_B B_{\bar{W}} B_W\right) + 0.5bB_{\bar{W}} B_W = k_{\max}. \tag{4.41}$$

Multiplying the two sides of Equation (4.23) for $\frac{\mathbf{H}^T}{\|\mathbf{H}\|^2}$ yields another dynamic equation

$$\dot{\mathbf{x}} = -\delta\mathbf{x} + \frac{\mathbf{H}^T}{\|\mathbf{H}\|^2} s \tag{4.42}$$

Equation (4.42), combined with the inequality (4.29) leads to

$$\|\dot{\mathbf{x}}\| \le \left(2\delta\|\mathbf{PG}\| + \|\mathbf{H}\|^{-1}\right)|s| = C_s|s| \tag{4.43}$$

Substituting the inequalities (4.41) and (4.43) into the inequality (4.39) yields

$$\dot{V}_2 \le \left[-\eta\mathbf{H}(\mathbf{B}+\Delta\mathbf{B}) + k_{\max}C_s\right]|s|^2$$

From Assumption 4.2, if we can choose η as in the inequality (4.38), then $\dot{V}_2 < 0; s \neq 0$, and the sliding variable s will converge to zero by the Lyapunov stability method. When $s = 0$, from Lemma 4.1, the closed-loop system is asymptotically stable. ■

Remark 4.4: From Theorems 4.2, the NN controller with the on-line learning algorithms (4.37) can assure that the system state converges to zero and the system performances are satisfied by the design of the sliding hyperplane.

Theorm 4.2 also indicates that the number of hidden nodes does not affect system stability. However, the number of hidden nodes should be chosen from 2 to 7 in order to reduce the time associated with on-line computation.

4.3.4 Numerical results of the chattering-free sliding-mode-based neural network controller

Consider an uncertain linear system, as described in Coleman & Godbole (1994). The nominal motor plant is modelled by the following transfer function

System 1: $$G_1 = \frac{5}{s(s+1)(s+2)} \tag{4.44}$$

The two perturbed plant models are given by the following transfer functions

System 2: $$G_2 = \frac{2}{s(s-1)(s+1)} \tag{4.45}$$

System 3: $$G_3 = \frac{7}{(s+1)^2(s+2)} \tag{4.46}$$

The transfer function models are transferred into state space format as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$
$$y = \mathbf{C}\mathbf{x}$$

where $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -2 & -3 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ for System 1,

$\mathbf{A}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 2 & -1 \end{bmatrix}$, $\mathbf{B}_2 = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$, $\mathbf{C}_2 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ for System 2 , and

and $\mathbf{A}_3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -5 & -4 \end{bmatrix}$, $\mathbf{B}_3 = \begin{bmatrix} 0 \\ 0 \\ 7 \end{bmatrix}$, $\mathbf{C}_3 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ for System 3.

The whole motor plant can be modelled by an uncertain dynamic equation:

$$\dot{\mathbf{x}} = (\mathbf{A} + \Delta\mathbf{A})\mathbf{x} + (\mathbf{B} + \Delta\mathbf{B})u$$
$$y = \mathbf{C}\mathbf{x}$$

where $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -2 & -3 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ for the nominal system, and

$\Delta\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \pm 2 & \pm 4 & \pm 2 \end{bmatrix}$, $\Delta\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \pm 3 \end{bmatrix}$.

Suppose that the closed-loop eigenvalues are $(-2.54; -1.33 + j1.91; -1.33 - j1.91)$. From Theorem 4.1, a sliding function is designed as

$$s = \delta \mathbf{H}\mathbf{x} + \mathbf{H}\dot{\mathbf{x}},$$

where $\delta = 2.54$; $\mathbf{H} = [h_1, h_2, 1] = [5.417 \quad 2.66 \quad 1]$.

Apparently, Assumption 4.2 is always obtained with $\mathbf{H}(\mathbf{B} + \Delta\mathbf{B}) \geq \mathbf{H}\mathbf{B}_3 = 2 > 0$.

An FNN structure with $n = 3$, $m = 5$ and one output node is used as the neural controller, as described in Figure 4.7. The network weights are simply initialised at random small values. The on-line learning algorithm (4.37) is applied to minimise the cost function $V_2 = \frac{1}{2}s^2$.

From the simulations, the network weights vary in a range $[-3,3]$. From Theorem 4.2, one can choose $\eta \geq 89.9$. With $\eta = 109$ chosen, the output and control signals of the controlled systems are shown in Figure 4.8.

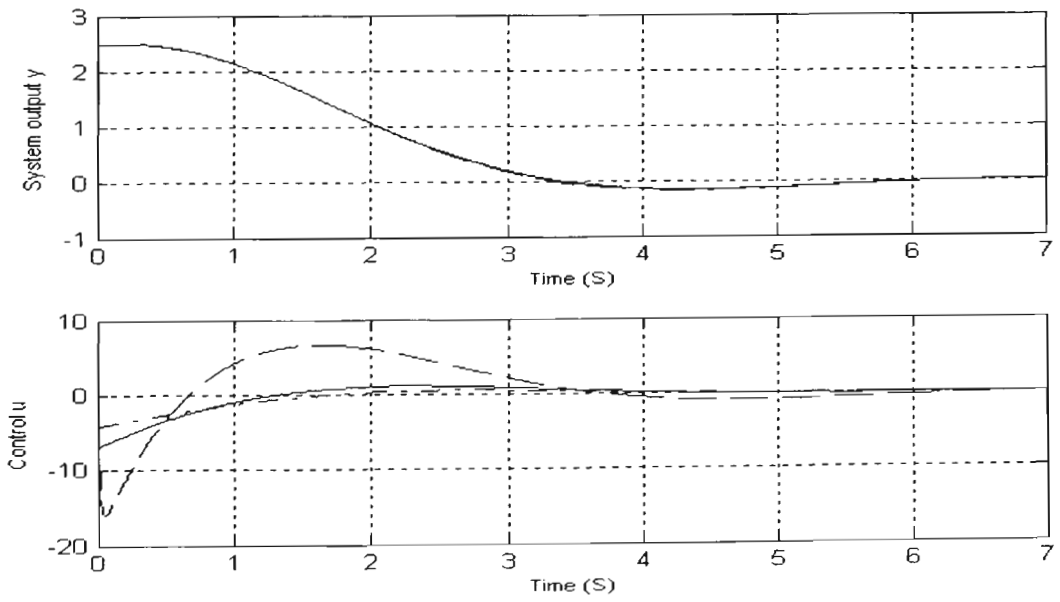


Figure 4.8: Output and control signals of System 1 (solid line), System 2 (dashed line) and System 3 (dash-dot line), using the chattering-free sliding-mode-based neural network controller.

To compare the simulation results, a state variable feedback controller can be designed using the pole placement method (Franklin et al., 2002), and the controller parameters are obtained for the nominal system as:

$$\mathbf{K} = [\,2.7518 \quad 2.0347 \quad 0.44\,]$$

This state variable feedback controller is then applied to control Systems 1, 2 and 3, and the output and control signals of the controlled systems are shown in Figure 4.9.

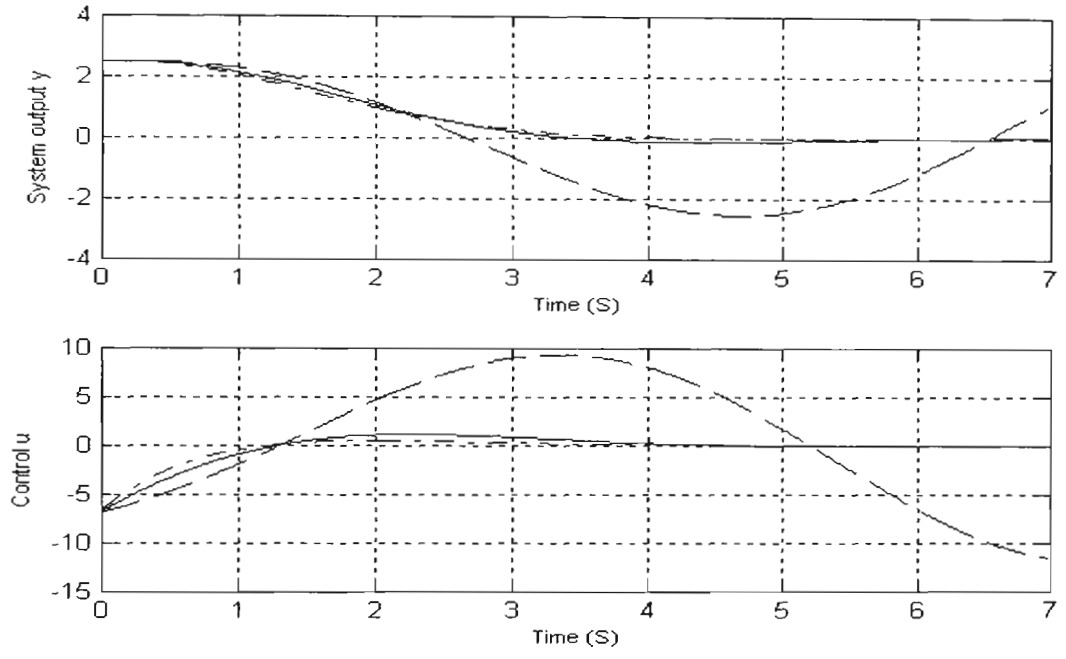


Figure 4.9: Output and control signals of System 1 (solid line), System 2 (dashed line) and System 3 (dash-dot line), using the state feedback controller.

Discussion:

In Figure 4.9, when controlling Systems 1 and 3, the state feedback controller assures the required performances, that is, a rise time of no more than 3.5 seconds and an overshoot of no more than 6%. However, when applied to System 2, the state feedback controller cannot stabilise the system, as shown in Figure 4.9. On the other hand, the proposed neural controller can guarantee the required performances of the closed-loop systems including Systems 1, 2 and 3, as described in Figure 4.8. This indicates the

robust capability of the system in the presence of parameter uncertainties when using the proposed neural controller in comparison to the state-feedback control system.

Although the CFSMBP algorithm, another version of Equation (4.16), is developed from the idea of Giordano et al. (2004), CFSMBP can completely avoid the chattering phenomenon. However, the CFSMBP-based neural network controller only stabilises the linear system with parameter uncertainties. When the system dynamics includes external disturbances, the on-line learning algorithm in Giordano et al. (2004) can further assure system stability and robustness. Therefore, CFSMBP needs to be upgraded to another robust sliding-mode-based learning algorithm, which will be presented in the next section.

4.4 Robust Sliding-mode-based Neural Networks for Control Applications

4.4.1 Robust sliding-mode-based learning algorithm

When the NN-based control system includes parameter uncertainties and external disturbances, the robustness of the learning algorithm becomes an important issue. The CFSMBP algorithm can face system parameter uncertainties, as described in Section 4.3.3. However, the included external disturbance requires the development of more robust algorithms. A novel learning algorithm is mathematically presented in the following.

Consider again the FNN, as shown in Figure 4.6. All the definition of the FNN in Section 4.3.1 are used in this section.

Assumption 4.4: *There exist some positive constants such that*

$$\begin{aligned} \|\dot{\mathbf{x}}\| \le B_{\dot{\mathbf{x}}}; \quad |\dot{y}_d| \le B_{\dot{y}_d} \\ \|\mathbf{W}\| \le B_W; \quad \|\bar{\mathbf{W}}\| \le B_{\bar{W}} \end{aligned} \tag{4.47}$$

Definition 4.1: (Sira-Ramirez & Morles, 1995) *A sliding motion is said to exist on a sliding surface* $s(e) = e(t) = 0$, *after time* t_h, *if the condition* $s(t)\dot{s}(t) = e(t)\dot{e}(t) < 0$ *is*

satisfied for all t in some nontrivial semi-open subinterval of time of the form $[t, t_h) \subset (-\infty, t_h)$.

It is desired to devise an adaptive learning algorithm for the NN such that the sliding-mode condition in Definition 4.1 is satisfied. Using the reaching law method, we then obtain the following results.

Theorem 4.3: *If Assumption 4.4 is satisfied, and the on-line learning algorithm for the FNN weights is chosen as*

$$\begin{aligned}\dot{\tilde{\mathbf{W}}} &= \frac{\left[-\eta e - \varepsilon \operatorname{sgn}(e)\right]}{\left\|\tilde{\bar{\mathbf{o}}}\right\|^2 + \left\|\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\right\|^2} \tilde{\bar{\mathbf{o}}}^T \\ \dot{\tilde{\mathbf{W}}} &= \frac{\left[-\eta e - \varepsilon \operatorname{sgn}(e)\right]}{\left\|\tilde{\bar{\mathbf{o}}}\right\|^2 + \left\|\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\right\|^2} \mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\end{aligned} \tag{4.48}$$

with $\eta > 0$ and ε being a sufficiently large positive constant satisfied by

$$\varepsilon > 0.5 B_{\mathbf{W}} B_{\mathbf{x}} B_{\tilde{\mathbf{W}}} + B_{\dot{y}_d}, \tag{4.49}$$

then, for an arbitrary initial condition $e(0)$, the learning error $e(t)$ converges to zero in finite time t_h estimated by

$$t_h \le \frac{1}{\eta} \ln \frac{\eta |e(0)| + \varepsilon - 0.5 B_{\mathbf{W}} B_{\mathbf{x}} B_{\tilde{\mathbf{W}}} - B_{\dot{y}_d}}{\varepsilon - 0.5 B_{\mathbf{W}} B_{\mathbf{x}} B_{\tilde{\mathbf{W}}} - B_{\dot{y}_d}}, \tag{4.50}$$

and a sliding motion is maintained on $e = 0$ for all $t > t_h$.

Proof: Consider a Lyapunov candidate given by

$$V = \frac{1}{2} e^2. \tag{4.51}$$

From Equations (4.51), (4.14) and (4.13), it is obtained as

$$\frac{\partial V}{\partial W_i} = e\bar{o}_i; \quad \frac{\partial V}{\partial \bar{W}_{ij}} = eW_i f'_{hi} x_j; \quad \frac{\partial V}{\partial x_j} = e\sum_{i=1}^{m} W_i f'_{hi} \bar{W}_{ij}; \quad \frac{\partial V}{\partial y_d} = -e.$$

The time derivative of V is obtained as:

$$\dot{V} = \sum_{i=1}^{m+1} \frac{\partial V}{\partial W_i} \dot{W}_i + \sum_{i=1}^{m} \sum_{j=1}^{n+1} \frac{\partial V}{\partial \bar{W}_{ij}} \dot{\bar{W}}_{ij} + \sum_{j=1}^{n} \frac{\partial V}{\partial x_j} \dot{x}_j + \frac{\partial V}{\partial y_d} \dot{y}_d$$

$$\dot{V} = \frac{e\left[-\eta e - \varepsilon \operatorname{sgn}(e)\right]}{\left\|\tilde{\bar{\mathbf{o}}}\right\|^2 + \left\|\mathbf{F}'_h \mathbf{W}^T \tilde{\mathbf{x}}^T\right\|^2} \left(\sum_{i=1}^{m+1} \bar{o}_i^2 + \sum_{i=1}^{m} \sum_{j=1}^{n+1} \left(W_i f'_{hi} x_j\right)^2 \right) + e\sum_{i=1}^{m} \sum_{j=1}^{n} W_i f'_{hi} \bar{W}_{ij} \dot{x}_j - e\dot{y}_d$$

$$\dot{V} = -\eta e^2 - \varepsilon |e| + e\sum_{i=1}^{m} f'_{hi} W_i \sum_{j=1}^{n} \bar{W}_{ij} \dot{x}_j - e\dot{y}_d$$

$$\dot{V} = -\eta e^2 - \varepsilon |e| + e\left(\mathbf{W}\mathbf{F}'_h \bar{\mathbf{W}} \dot{\mathbf{x}} - \dot{y}_d\right)$$

Because $f'_{hi} = \frac{d}{dv}\left(\frac{1-e^{-v}}{1+e^{-v}}\right) = \frac{1}{2}\left(1 - f_{hi}^2\right) \le 0.5, \quad i = 1, \ldots, m$, and from Assumption 4.4, it is obtained as:

$$\dot{V} \le -\eta e^2 - \varepsilon |e| + |e|\left(0.5 B_{\mathbf{W}} B_{\dot{\mathbf{x}}} B_{\bar{\mathbf{W}}} + B_{\dot{y}_d}\right) = -\eta e^2 - \left(\varepsilon - 0.5 B_{\mathbf{W}} B_{\dot{\mathbf{x}}} B_{\bar{\mathbf{W}}} - B_{\dot{y}_d}\right)|e|. \quad (4.52)$$

If ε is chosen in (4.49), then $\dot{V} < 0; \forall e \neq 0$, and the error function $e(t)$ converges to zero in a stable manner.

It can be shown that such a convergence occurs in finite time. The time derivative of $e(t)$ satisfies:

$$\dot{e} = \sum_{i=1}^{m+1} \frac{\partial e}{\partial W_i} \dot{W}_i + \sum_{i=1}^{m} \sum_{j=1}^{n+1} \frac{\partial e}{\partial \bar{W}_{ij}} \dot{\bar{W}}_{ij} + \sum_{j=1}^{n} \frac{\partial e}{\partial x_j} \dot{x}_j + \frac{\partial e}{\partial y_d} \dot{y}_d$$

$$\dot{e} = -\eta e - \varepsilon \operatorname{sgn}(e) + \left(\mathbf{W}\mathbf{F}'_h \bar{\mathbf{W}} \dot{\mathbf{x}} - \dot{y}_d\right)$$

$$\Rightarrow dt = \frac{\dot{e}\,\mathrm{sgn}(e)\,dt}{-\eta|e| - \varepsilon + \left(\mathbf{W}\mathbf{F}_h'\bar{\mathbf{W}}\dot{\mathbf{x}} - \dot{y}_d\right)\mathrm{sgn}(e)}$$

$$dt \le \frac{\dot{e}\,\mathrm{sgn}(e)\,dt}{-\eta|e| - \varepsilon + 0.5B_{\mathbf{W}}B_{\dot{\mathbf{x}}}B_{\bar{\mathbf{W}}} + B_{\dot{y}_d}}. \tag{4.53}$$

Integrating the two sides of (4.53) with regard to time, from $t = 0$ with initial condition $e(0)$ to $t = t_h$ with $e(t_h) = 0$, leads to

$$t_h \le \frac{1}{\eta}\ln\frac{\eta|e(0)| + \varepsilon - 0.5B_{\mathbf{W}}B_{\dot{\mathbf{x}}}B_{\bar{\mathbf{W}}} - B_{\dot{y}_d}}{\varepsilon - 0.5B_{\mathbf{W}}B_{\dot{\mathbf{x}}}B_{\bar{\mathbf{W}}} - B_{\dot{y}_d}}. \tag{4.54}$$

Evidently, for $\forall t < t_h$ and for the large constant ε chosen in (4.49), the sliding condition is satisfied as

$$e\dot{e} \le -\alpha e^2 - \left(\varepsilon - 0.5B_{\mathbf{W}}B_{\dot{\mathbf{x}}}B_{\bar{\mathbf{W}}} - B_{\dot{y}_d}\right)|e| < 0$$

and a sliding mode exists on $e(t) = 0$ for $t \ge t_h$. ■

Remark 4.5: In order to reduce the chattering phenomenon, the following approximation for the signum function has been adopted:

$$\mathrm{sgn}(e) \approx \frac{e}{|e| + \vartheta} \tag{4.55}$$

with ϑ being a small constant.

Remark 4.6: As the time derivative of V in (4.52) satisfies the sliding condition of the reaching law method, the algorithm (4.48) is named RMBP, which denotes the integration between the reaching law method and the backpropagation algorithm. On the other hand, the on-line learning algorithm proposed by Giordano et al. (2004) is derived from the integration between the standard SMC technique and BP. By inheriting the advantages of the reaching law method, the zero convergence of the learning error in the

RMBP algorithm is therefore faster than that in the learning algorithm proposed by Giordano et al. (2004).

4.4.2 Control applications of the robust sliding-mode-based neural networks

Consider the following continuous-time uncertain system:

$$\begin{aligned} \dot{\mathbf{x}} &= (\mathbf{A}+\Delta\mathbf{A})\mathbf{x}+(\mathbf{B}+\Delta\mathbf{B})u+\Theta d(t), \qquad \mathbf{x}(0)=\mathbf{x}_0 \\ y &= \mathbf{C}\mathbf{x} \end{aligned} \tag{4.56}$$

where $\mathbf{x}=[x_1 \quad \cdots \quad x_n]^T \in \mathbf{R}^n$ is the system state, $u \in \mathbf{R}$ is the control input, $\mathbf{A} \in \mathbf{R}^{n\times n}$ is the system matrix, $\mathbf{B} \in \mathbf{R}^{n\times 1}$ is the input matrix and $\mathbf{C} \in \mathbf{R}^{1\times n}$ is the output matrix so that $\mathbf{A},\mathbf{C}$ satisfy the observability condition, and $\mathbf{A},\mathbf{B}$ are in the controllable canonical form (4.21), $\Delta\mathbf{A} \in \mathbf{R}^{n\times n}, \Delta\mathbf{B} \in \mathbf{R}^{n\times 1}$ represent parameter uncertainties in the plant model, $\Theta \in \mathbf{R}^{n\times 1}$ is the disturbance input matrix, and $d(t)$ is the external disturbance.

Assumption 4.5: *Assume that the uncertain matrices* $\Delta\mathbf{A},\Delta\mathbf{B}$ *are bounded:*

$$\begin{aligned} \|\Delta\mathbf{A}\| &\le \rho_A \\ \|\Delta\mathbf{B}\| &\le \rho_B \end{aligned} \tag{4.57}$$

and the first-order derivative of the disturbance is bounded:

$$\|\Theta\dot{d}(t)\| \le D\,. \tag{4.58}$$

The control objective is to regulate the state $\mathbf{x}$ in (4.56) to zero, and this is obtained via a two-stage control design. In the first stage, a sliding hyperplane is derived from desired eigenvalues of the closed-loop system, as described in Theorem 4.1. This sliding function is then used as the training signal for a neural network controller designed in the second stage.

Similar to Section 4.3.3, a sliding function is defined, as in Equation (4.23). By the choice of the coefficients $h_1,\ldots,h_{n-1},1$, the system state $\mathbf{x}$ is forced to remain inside the subspace in which the inequality (4.29) is satisfied.

In the second stage, a neural controller structure, as shown in Figure 4.7, is used. The output of the neural network is also the control signal u defined as:

$$u(\mathbf{x}) = \tilde{\mathbf{W}}\tilde{\mathbf{o}} = \sum_{i=1}^{m+1} W_i \bar{o}_i = \sum_{i=1}^{m} W_i f_{hi}\left(\sum_{j=1}^{n} \bar{W}_{ij} x_j + \bar{W}_{i(n+1)}\right) + W_{m+1} \tag{4.59}$$

The sliding variable s is used as the training signal for the FNN. Therefore, the neural network can be trained on-line to minimise a cost function

$$V_3 = \tfrac{1}{2} s^2 \tag{4.60}$$

The RMBP learning algorithm of the weight vector $\tilde{\mathbf{W}}$ and the weight matrix $\tilde{\bar{\mathbf{W}}}$ are presented as follows:

$$\begin{aligned}
\dot{\tilde{\mathbf{W}}} &= -\frac{\eta s + \varepsilon \operatorname{sgn}(s)}{\left\|\tilde{\bar{\mathbf{o}}}^T\right\|^2 + \left\|\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\right\|^2} \tilde{\bar{\mathbf{o}}}^T \\
\dot{\tilde{\bar{\mathbf{W}}}} &= -\frac{\eta s + \varepsilon \operatorname{sgn}(s)}{\left\|\tilde{\bar{\mathbf{o}}}^T\right\|^2 + \left\|\mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T\right\|^2} \mathbf{F}_h' \mathbf{W}^T \tilde{\mathbf{x}}^T
\end{aligned} \tag{4.61}$$

where η is a standard learning rate and ε is an added robust learning rate.

Theorem 4.4: *Consider the closed-loop system consisting of the dynamics* (4.56), *the controller* (4.59) *and the training algorithm* (4.61), *if Assumptions 4.2, 4.3 and 4.5 are satisfied and* η, ε *is chosen as*

$$\begin{aligned}
\eta &\geq \frac{k_{\max}}{g_0} C, \\
\varepsilon &\geq \frac{\|\mathbf{H}\|}{g_0} D
\end{aligned} \tag{4.62}$$

where $k_{\max} = \|\mathbf{HA}\| + \|\mathbf{H}\|(\rho_1 + \delta + 0.5\rho_B B_{\bar{W}} B_W) + 0.5bB_{\bar{W}} B_W$, $C_s = 2\delta\|\mathbf{PG}\| + \|\mathbf{H}\|^{-1}$, *then the closed-loop system is asymptotically stable.*

Proof: Substituting Equation (4.56) into Equation (4.23) yields:

$$s = \delta\sigma + \dot{\sigma} = \delta\mathbf{Hx} + \mathbf{H}(\mathbf{A} + \Delta\mathbf{A})\mathbf{x} + \mathbf{H}(\mathbf{B} + \Delta\mathbf{B})u + \mathbf{H\Theta}d \tag{4.63}$$

$$\begin{aligned}\frac{\partial s}{\partial u} &= \frac{\partial}{\partial u}\left[\mathbf{H}(\mathbf{A} + \Delta\mathbf{A})\mathbf{x} + \delta\mathbf{Hx} + \mathbf{H}(\mathbf{B} + \Delta\mathbf{B})u + \mathbf{H\Theta}d\right] \\ &= \mathbf{H}(\mathbf{B} + \Delta\mathbf{B})\end{aligned} \tag{4.64}$$

$$\frac{\partial s}{\partial d} = \frac{\partial}{\partial d}\left[\mathbf{H}(\mathbf{A} + \Delta\mathbf{A})\mathbf{x} + \delta\mathbf{Hx} + \mathbf{H}(\mathbf{B} + \Delta\mathbf{B})u + \mathbf{HG}d\right] = \mathbf{H\Theta} \tag{4.65}$$

The gradients of V_3 with regard to the weight matrix or vector are calculated as:

$$\frac{\partial V_3}{\partial \tilde{\mathbf{W}}} = \frac{\partial V_3}{\partial s}\frac{\partial s}{\partial u}\frac{\partial u}{\partial \tilde{\mathbf{W}}} = s\mathbf{H}(\mathbf{B} + \Delta\mathbf{B})\tilde{\bar{\mathbf{o}}}^T \tag{4.66}$$

$$\frac{\partial V_3}{\partial \tilde{\bar{\mathbf{W}}}} = \frac{\partial V_3}{\partial s}\frac{\partial s}{\partial u}\frac{\partial u}{\partial \bar{\mathbf{o}}}\frac{\partial \bar{\mathbf{o}}}{\partial \tilde{\bar{\mathbf{W}}}} = s\mathbf{H}(\mathbf{B} + \Delta\mathbf{B})\mathbf{F}_h'\mathbf{W}^T\tilde{\mathbf{x}}^T \tag{4.67}$$

The derivative of the function (4.60) is obtained as

$$\dot{V}_3 = \sum_{i=1}^{m+1}\frac{\partial V_3}{\partial W_i}\dot{W}_i + \sum_{i=1}^{m}\sum_{j=1}^{n+1}\frac{\partial V_3}{\partial \bar{W}_{ij}}\dot{\bar{W}}_{ij} + \sum_{j=1}^{n}\frac{\partial V_3}{\partial x_j}\dot{x}_j + \frac{\partial V_3}{\partial d}\dot{d}$$

From Assumption 4.5 and Equations (4.61), (4.65), (4.66) and (4.67) , it is obtained as:

$$\begin{aligned}\dot{V}_3 &= -\eta\mathbf{H}(\mathbf{B} + \Delta\mathbf{B})s^2 - \varepsilon\mathbf{H}(\mathbf{B} + \Delta\mathbf{B})|s| + s\frac{\partial s}{\partial \mathbf{x}}\dot{\mathbf{x}} + s\frac{\partial s}{\partial d}\dot{d} \\ \dot{V}_3 &\le -\eta\mathbf{H}(\mathbf{B} + \Delta\mathbf{B})s^2 + |s|\left\|\frac{\partial s}{\partial \mathbf{x}}\right\|\|\dot{\mathbf{x}}\| - \mathbf{H}(\mathbf{B} + \Delta\mathbf{B})\varepsilon|s| + \|\mathbf{H}\|D|s|\end{aligned} \tag{4.68}$$

From Equations (4.59) and (4.63), it is obtained as:

$$\frac{\partial s}{\partial \mathbf{x}} = \mathbf{H}(\mathbf{A}+\Delta\mathbf{A}) + \delta\mathbf{H} + \mathbf{H}(\mathbf{B}+\Delta\mathbf{B})\mathbf{W}\mathbf{F}_h'\bar{\mathbf{W}} \tag{4.69}$$

Thus the inequalities (4.41) and (4.43) are obtained.

The inequality (4.68), combined with the inequalities (4.41) and (4.43) leads to

$$\dot{V}_3 < -\left[\eta\mathbf{H}(\mathbf{B}+\Delta\mathbf{B}) - k_{\max}C_s\right]|s|^2 - \left(\mathbf{H}(\mathbf{B}+\Delta\mathbf{B})\varepsilon - \|\mathbf{H}\|D\right)|s|$$

From Assumption 4.2, if we can choose η, ε as in the inequality (4.62), then $\dot{V}_3 < 0; s \neq 0$, and the sliding variable s will converge to zero by the Lyapunov stability method. When $s = 0$, from Lemma 4.1, the closed-loop system is asymptotically stable.

■

Remark 4.7: From Theorems 4.3, the NN controller with the on-line learning algorithms (4.61) can assure that the system state robustly converges to zero and the system performances are satisfied by the design of the sliding hyperplane.

Theorm 4.3 also implies that the number of hidden nodes does not affect system stability. However, the number of hidden nodes should be chosen from 2 to 7 in order to reduce the time associated with on-line computation.

In order to reduce the chattering phenomenon, the following approximation for the signum function has been adopted:

$$\operatorname{sgn}(e) \approx \frac{e}{|e| + \vartheta}$$

with ϑ being a small constant.

4.4.3 Numerical results of the robust sliding-mode-based neural network controller

Consider an uncertain linear system including three representational models, that is, Systems 1, 2 and 3 with added external disturbance $d(t)$. The whole motor plant can be modelled by an uncertain dynamic equation:

$$\dot{\mathbf{x}} = (\mathbf{A} + \Delta\mathbf{A})\mathbf{x} + (\mathbf{B} + \Delta\mathbf{B})u + \mathbf{\Theta}d(t)$$
$$y = \mathbf{C}\mathbf{x}$$

where $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -2 & -3 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$, $\mathbf{\Theta} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ for the nominal system, $\Delta\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \pm 2 & \pm 4 & \pm 2 \end{bmatrix}$, $\Delta\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \pm 3 \end{bmatrix}$,

and a small disturbance $d(t) = 5\sin(5t)$.

Suppose that the closed-loop eigenvalues are $(-2.54; -1.33 + j1.91; -1.33 - j1.91)$.

From Theorem 4.1, a sliding function is designed as $s = \delta\mathbf{H}\mathbf{x} + \mathbf{H}\dot{\mathbf{x}}$, where $\delta = 2.54$ and

$$\mathbf{H} = [h_1, h_2, 1] = [5.417 \quad 2.66 \quad 1].$$

Apparently, $\mathbf{H}(\mathbf{B} + \Delta\mathbf{B}) \geq \mathbf{H}\mathbf{B}_3 = 2 > 0$.

An FNN structure with $n = 3$, $m = 5$ and one output node is used as the neural controller, as described in Figure 4.7. The network weights are simply initialised at random small values. The on-line learning algorithm (4.61) is applied to minimise the cost function $V_3 = \frac{1}{2}s^2$. From Theorem 4.4, one can choose $\eta \geq 98.9$ and $\varepsilon \geq 67.72$. The approximation of signum function (4.55) is also used with $\vartheta = 0.0001$. With $\eta = 109$ and $\varepsilon = 70$ chosen, the output and control signals of the controlled systems are shown in Figure 4.10.

The chattering-free sliding-mode-based neural network controller is also applied to control this uncertain system. With $\eta = 109$ chosen, the output and control signals of the controlled systems are shown in Figure 4.11.

Moreover, the sliding-mode-control-based learning algorithm (SMC) in Giordano et al. (2004) is developed for training the neural control system presented in Section 4.4.2. With a learning rate $\alpha = 270$ chosen, system performances satisfy the requirements even in the presence of parameter uncertainties and disturbances, as shown in Figure 4.12.

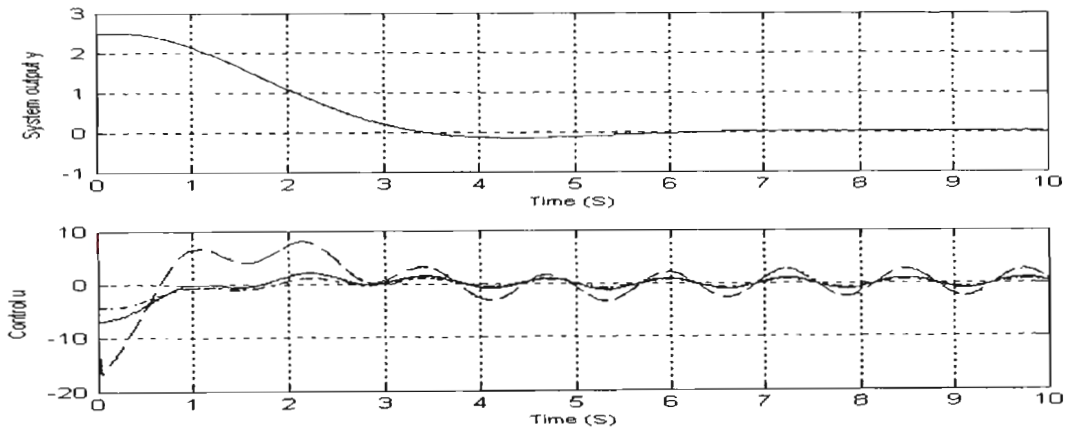


Figure 4.10: Output and control signals of System 1 (solid line), System 2 (dashed line) and System 3 (dash-dot line), using the robust sliding-mode-based neural network controller.

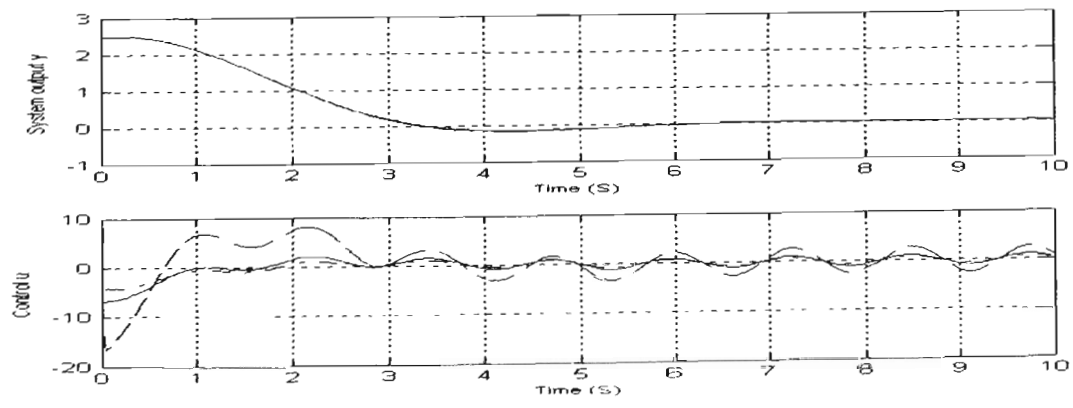


Figure 4.11: Output and control signals of System 1 (solid line), System 2 (dashed line) and System 3 (dash-dot line), using the chattering-free sliding-mode-based neural network controller.

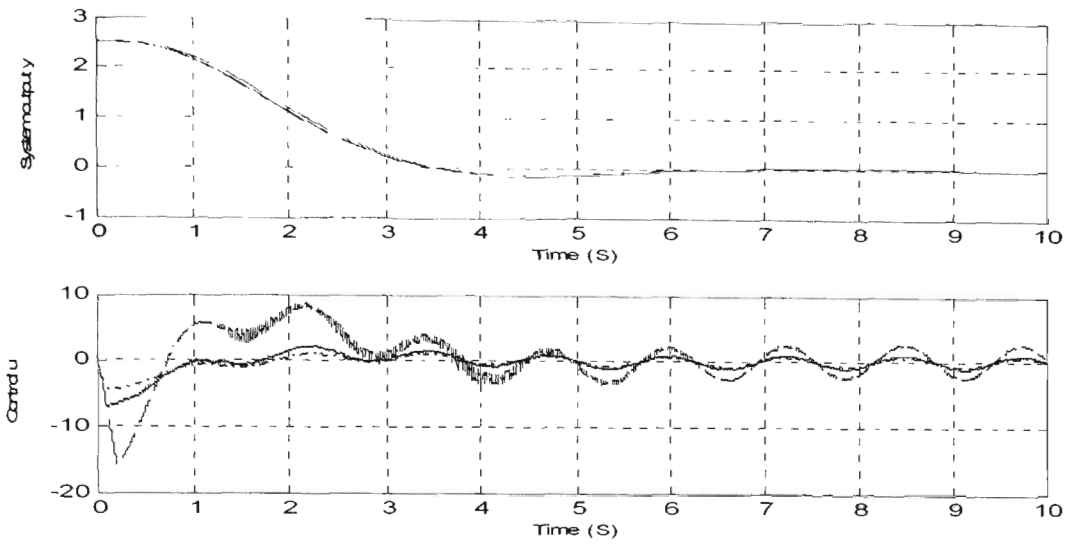


Figure 4.12: Output and control signals of System 1 (solid line), System 2 (dashed line) and System 3 (dash-dot line), using the SMC-based neural network controller.

As seen in Figures 4.10 and 4.11, the disturbance amplitude of 5 leads to no different results between the two methods. Therefore, for comparison purposes, a disturbance of $d(t) = 200\sin(5t)$ is utilised, although this disturbance eventually leads to a large control signal.

When the robust sliding-mode-based NN controller is utilised with $\eta = 109$ and $\varepsilon = 3000$, the output and control signals of the controlled systems are shown in Figure 4.13.

When the chattering-free sliding-mode-based NN controller is utilised with $\eta = 109$ chosen, the output and control signals of the controlled systems are shown in Figure 4.14.

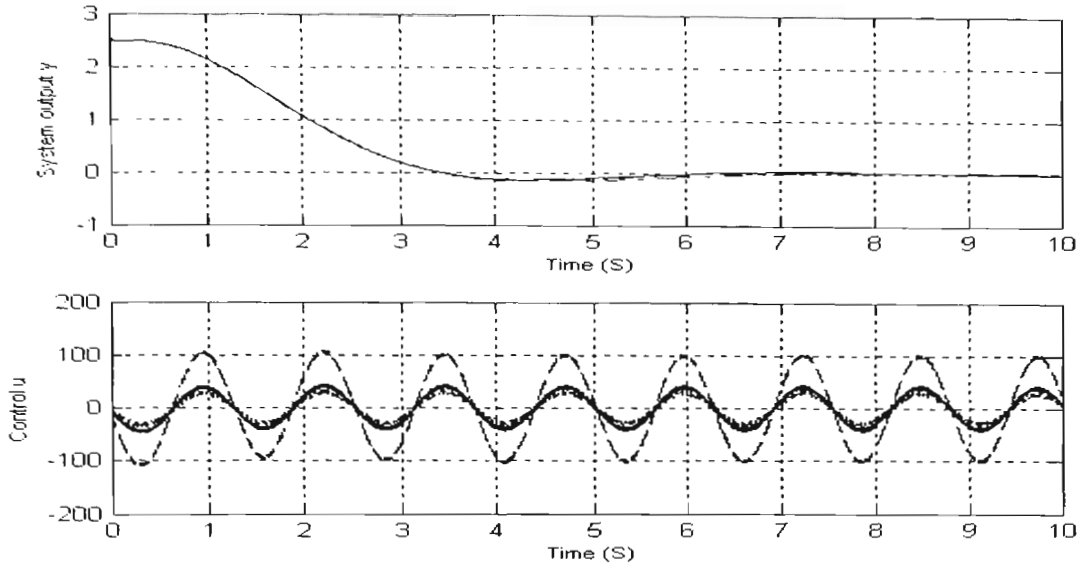


Figure 4.13: Output and control signals of System 1 (solid line), System 2 (dashed line) and System 3 (dash-dot line), using the robust sliding-mode-based neural network controller.

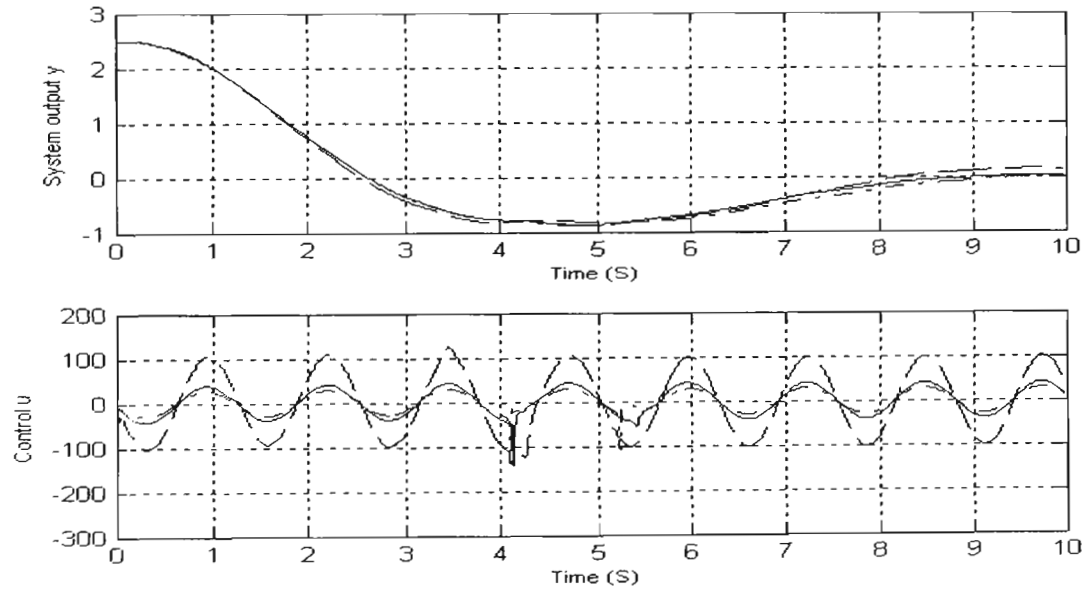


Figure 4.14: Output and control signals of System 1 (solid line), System 2 (dashed line) and System 3 (dash-dot line), using the chattering-free sliding-mode neural network controller.

Discussion:

From the results demonstrated in Figures 4.10 and 4.11, both the chattering-free sliding-mode-based neural network controller and the robust sliding-mode-based neural network controller robustly yield the required performances of the controlled systems in the presence of parameter uncertainties and external disturbances. With a severe disturbance, the controlled systems using the robust sliding-mode-based neural network still obtain the required output performances, as shown in Figure 4.13. Although the chattering-free sliding-mode-based neural network controller can stabilise the uncertain systems, the closed-loop systems have a maximum overshoot of 32%, as described in Figure 4.14. Therefore, the robust sliding-mode-based neural network controller shows superior performance in comparison to the chattering-free sliding-mode-based neural network controller.

As seen in Figures 4.10 and 4.12, the neural control system using the RMBP algorithm leads to lower chattering control signal than that system using the SMC algorithm. This indicates another advantage of the RMBP comparing to the learning algorithm proposed by Giordano et al. (2004).

4.5 Conclusion

In conclusion, several novel algorithms integrated between the SMC strategy and BP algorithm, named EABPM, CFSMBP and RMBP, are proposed in this chapter. The EABPM algorithm improves BP performances in convergence rate and global convergence. Example results in the XOR problem demonstrate the advantage of the proposed method. The EABPM algorithm is further applied in the neural network head-movement classifier, and the experiment results illustrate the effect of the proposed method.

When applied in control systems, FNNs with the CFSMBP, RMBP algorithms can work as the direct adaptive controllers. These neural controllers can assure closed-loop system stability in the present of parameter uncertainties or external disturbances. Some

example results in controlling two uncertain linear systems demonstrate the advantage of the proposed neural controller design techniques. In the next chapter, the chattering-free sliding-mode-based neural network is applied to control a class of nonlinear systems with transportation lag, and is implemented in a real-time Static VAR Compensator system. In Chapter 6, the robust sliding-mode-based neural network controller is developed for a class of multivariable systems with bounds of interconnections and disturbances, and is performed in a real-time Coupled Electric Drives apparatus CE8.

CHAPTER 5

NEURAL NETWORK CONTROLLER DESIGN FOR A CLASS OF UNCERTAIN SYSTEMS WITH TRANSPORTATION LAG

In this chapter, a neural-networks-based control design is developed for a class of uncertain systems with transportation lag, also known as input delay. A direct-feedback neural control scheme is developed for control of such a system. For training the neural network controller, a sliding function is designed from the pole placement method, and is used as the training signal. The CFSMBP on-line learning algorithm proposed in Chapter 4 is proven to stabilise the controlled system. Based on a novel training procedure, the neural network parameters can converge to optimal values. The approach is effective for control of such uncertain systems with transportation lag. Experiments for a real-time SVC system are implemented to validate the proposed approach.

5.1 Introduction

It is well known that input delays often exist in various engineering systems. The problem of controlling the systems with time delay in the control input becomes more complex. Therefore, a number of nonlinear control methods have been developed to solve this problem. Isidori and Astolfi (1992) presented a state feedback nonlinear H_∞ control approach to attenuate disturbance for an affine nonlinear system. However, the solution of the Hamilton-Jacobian-Isaacs equations in this method is only obtained for some special nonlinear systems. Basin et al. (2003) developed an optimal regulator based on the integral sliding-mode scheme for such linear delay systems. Chiang et al. (2004) also designed a robust fuzzy-model-based sliding-mode controller for uncertain nonlinear input-delay systems. However, the chattering problem often associated with the sliding mode schemes limits the applicability of these methods.

Recently, NNs have offered an attractive solution for this problem because of their nonlinear mapping and learning capability. In the literature, a variety of neural controller structures has been found for the controlled nonlinear systems (Hagan & Demuth, 1999). Among these approaches, the direct inverse neural controller (Miller et

al., 1990) has the simplest scheme. However, it encounters instable problem when noises or disturbances exist in the system. These problems are due to the lack of feedback. To overcome the problems, the direct feedback inverse neural control schemes have been utilised in Norgaard (1996) and recently in Nayeri et al. (2004) and Daosud et al. (2005). For training the inverse neural network controller, the data set is often generated by exciting the plant with random inputs within the operation region of the controlled system. Consequently, it is very difficult to choose the appropriate training inputs. Moreover, if the specialised inverse learning method (Psaltis et al., 1988) is applied, a trained neural network identifier must be utilised. This leads to significant efforts in the system identification.

My research aims to develop a novel training method for the direct-feedback neural control scheme. For optimising this neural controller's parameters, a genetic training algorithm was proposed by Ichikawa and Sawa (1992), but this approach is not possible to implement in the real-time systems. In my research, an FNN is first trained to approximate the nonlinear feedback controller, then replaces the feedback controller in the closed-loop system. The FNN is trained on-line, thus avoiding difficult choice of training data. A sliding function is defined and used as the training error without the need for unknown plant Jacobian or a trained neural network identifier. Both the BP and CFSMBP algorithms mentioned in Chapter 4 are utilised for on-line training of the network, which guarantee the system stability by the Lyapunov method.

Moreover, in Chiang et al. (2004), the uncertain nonlinear time-delay system can be approximated by the Takagi-Sugeno fuzzy model (Cao & Frank, 2000) as:

$$R^i : \begin{cases} \text{IF } x_n \text{ is } \Gamma_1^i \text{ and ... and } x_n \text{ is } \Gamma_n^i \\ \text{THEN } \dot{\mathbf{x}} = \mathbf{A}_i\mathbf{x} + \Delta\mathbf{A}_i\mathbf{x} + \mathbf{B}_i u\left(t-\tau_d\right) + \Delta\mathbf{B}_i u\left(t-\tau_d\right) \end{cases}$$

where R^i denotes the i^{th} fuzzy inference rule, Γ_j^i the fuzzy set in the i^{th} rule, $\mathbf{A}_i \in \mathbf{R}^{n\times n}, \mathbf{B}_i \in \mathbf{R}^{n\times 1}$ the constant matrices in controllable canonical form, and $\Delta\mathbf{A}_i, \Delta\mathbf{B}_i$ the real matrices representing uncertainties.

More generally, in this research, the uncertain nonlinear input-delay system is approximated by an uncertain discrete-time linear model

$$\mathbf{x}(k+1) = (\mathbf{A} + \Delta\mathbf{A})\mathbf{x}(k) + (\mathbf{B} + \Delta\mathbf{B})u(k)$$
$$y(k) = \mathbf{C}\mathbf{x}(k)$$

where $\mathbf{A}, \mathbf{B}$ is not in the controllable canonical form.

5.2 Static VAR Compensator as an Uncertain System with Transportation Lag

In modern power systems, power quality is a major concern. Recently, advanced reactive power compensators using power semiconductor switching devices have been applied for power factor correction, improvement of voltage regulation and increase of the transient stability margin of the power systems. One of these advanced equipments using power electronics devices is the Static VAR Compensator (SVC), which is designed for transmission lines and industry (Lockey & Philpott, 2002). In industry, many loads are partly inductive, which decreases the power factor of the electric power system. By varying its reactive power output, an SVC can maintain a virtually constant desired power factor and thereby reduce the losses in the power system.

The picture of an SVC system developed in the Faculty of Engineering, University of Technology, Sydney (UTS) is shown in Figure 5.1. This system was designed by Professor Hung Tan Nguyen. There are thyristor-switched capacitors and thyristor-controlled reactors. The capacitors are switched on to compensate the inductive component of the load, and the reactors are continuously controlled to correct the power factor. The SVC system also has a motor load with speed and torque control.

In order to obtain the system transfer functions, a number of step-response experiments are carried out. In these experiments, speed and force references of the motor load are kept constant, the capacitor bank is switched on and the reactive power absorption is controlled continuously by the thyristor-controlled reactors. With different references of input voltages, the system output voltages that are proportional to the power factors are measured and saved.

Figure 5.1: The Static VAR Compensator (SVC) system.

5.2.1 Experiments with different step inputs

In the first experiments, a step input of 3.5V is presented to the SVC system five times. With this input, the power factor $PF = \cos(\phi_v - \phi_i)$ attains approximately 0.90 with redundancy capacitive. Figure 5.2 shows five experiment results, where the black line represents the averaged response of these experiments.

After cancelling the offset of 3.41, we can obtain approximate models with the two transfer functions below:

$$G_{1p}(s) = \frac{0.275.e^{-0.11s}}{0.19s+1} \qquad G_{2p}(s) = \frac{0.305.e^{-0.09s}}{0.12s+1} \tag{5.1}$$

Figure 5.3 compares the system averaged response with the models' step responses. The modelling procedure of the SVC system is aided by the Matlab software, and a Matlab program is detailed in Appendix A.

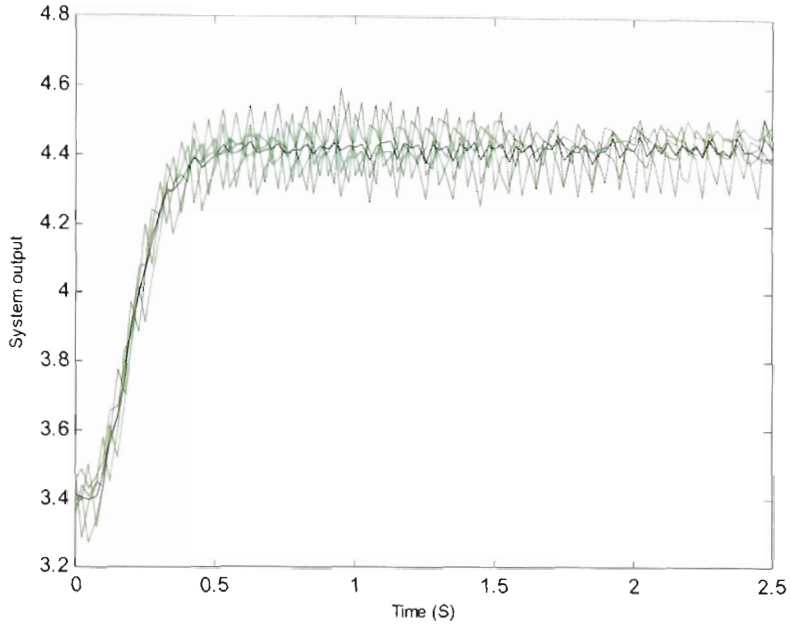


Figure 5.2: System step responses with an input of 3.5V.

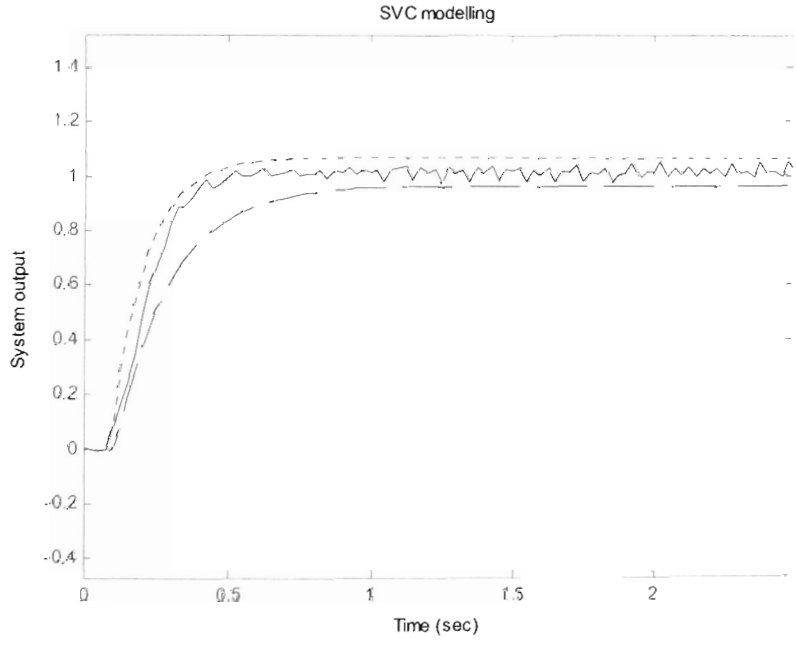


Figure 5.3: Step responses of the real-time system (solid line), model G_{1p} (dashed line) and model G_{2p} (dotted line) with an input of 3.5V.

Similarly, with different step inputs, several system models are obtained, as in Table 5.1.

Table 5.1: System transfer functions with different step inputs.

Input (V)	PF	offset	Transfer function 1	Transfer function 2
3.5	0.9 (C.)	3.41	$G_{1p}(s)=\frac{0.271.e^{-0.11s}}{0.19s+1}$	$G_{2p}(s)=\frac{0.31.e^{-0.09s}}{0.11s+1}$
3.7	0.95 (C.)	3.43	$G_{1p}(s)=\frac{0.271.e^{-0.11s}}{0.19s+1}$	$G_{2p}(s)=\frac{0.31.e^{-0.09s}}{0.11s+1}$
4.0	0.97 (C.)	3.4	$G_{1p}(s)=\frac{0.29.e^{-0.12s}}{0.2s+1}$	$G_{2p}(s)=\frac{0.32.e^{-0.09s}}{0.12s+1}$
4.2	0.99 (C.)	3.4	$G_{1p}(s)=\frac{0.3.e^{-0.11s}}{0.2s+1}$	$G_{2p}(s)=\frac{0.33.e^{-0.09s}}{0.12s+1}$
4.5	1 (C.)	3.43	$G_{1p}(s)=\frac{0.3.e^{-0.11s}}{0.2s+1}$	$G_{2p}(s)=\frac{0.32.e^{-0.08s}}{0.12s+1}$
4.8	1 (I.)	3.43	$G_{1p}(s)=\frac{0.31.e^{-0.11s}}{0.2s+1}$	$G_{2p}(s)=\frac{0.34.e^{-0.08s}}{0.12s+1}$
5.3	0.98 (I.)	3.43	$G_{1p}(s)=\frac{0.32.e^{-0.1s}}{0.19s+1}$	$G_{2p}(s)=\frac{0.35.e^{-0.07s}}{0.12s+1}$
5.5	0.96 (I.)	3.43	$G_{1p}(s)=\frac{0.32.e^{-0.1s}}{0.19s+1}$	$G_{2p}(s)=\frac{0.355.e^{-0.08s}}{0.11s+1}$
5.8	0.90 (I.)	3.44	$G_{1p}(s)=\frac{0.36.e^{-0.12s}}{0.15s+1}$	$G_{2p}(s)=\frac{0.385.e^{-0.09s}}{0.11s+1}$
6.0	0.80 (I.)	3.44	$G_{1p}(s)=\frac{0.39.e^{-0.12s}}{0.15s+1}$	$G_{2p}(s)=\frac{0.415.e^{-0.09s}}{0.11s+1}$
6.2	0.54 (I.)	3.5	$G_{1p}(s)=\frac{0.51.e^{-0.1s}}{0.22s+1}$	$G_{2p}(s)=\frac{0.56.e^{-0.09s}}{0.16s+1}$

5.2.2 Mathematical models of the Static VAR Compensator system

From 22 transfer functions above, we plot step responses of the system models with a step input of 1, as shown in Figure 5.4.

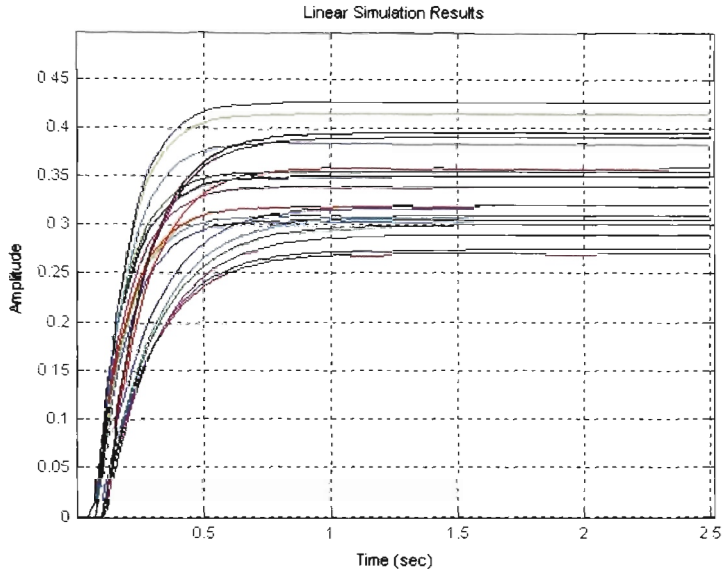


Figure 5.4: Model step responses with 1V input.

A general transfer function of the SVC system is obtained as:

$$G_p(s) = \frac{Y(s) - offset}{U(s)} = \frac{K.e^{-t_d s}}{\tau s + 1} \tag{5.2}$$

with a varying range of the system parameters as follows:

$$K = [0.29, 0.56]$$
$$\tau = [0.22, 0.11]$$
$$t_d = [0.07, 0.12]$$
$$offset = [3.4, 3.5].$$

Three following system models can represent the real-time SVC system, and are used for further study in this chapter.

System 1 with the nominal parameters has the transfer function:

$$G_p(s) = \frac{0.425.e^{-0.1s}}{0.165s+1}. \tag{5.3}$$

System 2 with upper-bound parameters has the transfer function:

$$G_p(s) = \frac{0.55.e^{-0.07s}}{0.11s+1}, \tag{5.4}$$

and System 3 with lower-bound parameters has the transfer function:

$$G_p(s) = \frac{0.29.e^{-0.12s}}{0.22s+1}. \tag{5.5}$$

5.2.3 State space dynamic equations

Applying the Maclaurin series expansion, $e^x = 1 + x + \frac{1}{2!}x^2 + \cdots \quad \left(0 \le |x| < \infty\right)$, for a small value of τ_d, the delay part in Equation (5.2) is approximated by:

$$e^{-t_d s} = \frac{e^{-\frac{t_d}{2}s}}{e^{\frac{t_d}{2}s}} \cong \frac{1 - \frac{t_d}{2}s + \frac{1}{2!}\left(\frac{t_d}{2}s\right)^2}{1 + \frac{t_d}{2}s + \frac{1}{2!}\left(\frac{t_d}{2}s\right)^2},$$

Therefore, Equation (5.3) becomes:

$$G_p(s) = \frac{0.425}{0.165s+1} \cdot \frac{1 - 0.05s + 0.00125s^2}{1 + 0.05s + 0.00125s^2} = \frac{0.425 - 0.02125s + 0.00053125s^2}{1 + 0.215s + 0.0095s^2 + 0.00020625s^3}. \tag{5.6}$$

The system (5.6) preceded by the zero-order holder is then transferred to the discrete-time dynamic equation with the sampling time $T = 0.025$ as follows:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}u(k) \\ y(k) &= \mathbf{C}\mathbf{x}(k) \end{aligned} \tag{5.7}$$

where $\mathbf{x}(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}$, and the parameters for System 1

$$\mathbf{A} = \begin{bmatrix} 0.16107 & 0.01327 & 0.00021 \\ -14.848 & 0.7723 & 0.022916 \\ -64.34 & -1.0153 & 0.9906 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.0166 \\ -2.3319 \\ 49.74 \end{bmatrix} \qquad (5.8)$$
$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Similarly, dynamic parameters for System 2 and 3 are obtained respectively as:

$$\mathbf{A}_2 = \begin{bmatrix} -0.03846 & 0.00964 & 0.00017 \\ -23.334 & 0.60013 & 0.02117 \\ -143.1 & -2.5838 & 0.97479 \end{bmatrix} \quad \mathbf{B}_2 = \begin{bmatrix} 0.012336 \\ -5.7088 \\ 196.96 \end{bmatrix} \qquad (5.9)$$
$$\mathbf{C}_2 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{A}_3 = \begin{bmatrix} 0.26792 & 0.01498 & 0.00023 \\ -11.164 & 0.8355 & 0.02352 \\ -37.838 & -0.56906 & 0.99483 \end{bmatrix} \quad \mathbf{B}_3 = \begin{bmatrix} 0.011348 \\ -1.0285 \\ 17.76 \end{bmatrix} \qquad (5.10)$$
$$\mathbf{C}_3 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

5.2.4 Remark

As shown from the experimental results, the SVC system is actually a nonlinear single-input-single-output system with transportation lag. Using an appropriate approximation, this real-time system can be modelled by an uncertain discrete-time linear system. Therefore, an NN-based control scheme is developed in the next section to solve the control problem of this system.

5.3 Advanced Neural Controller Design

Consider the following discrete-time uncertain system:

$$\begin{aligned} \mathbf{x}(k+1) &= (\mathbf{A}+\Delta\mathbf{A})\mathbf{x}(k)+(\mathbf{B}+\Delta\mathbf{B})u(k), \qquad \mathbf{x}(0)=\mathbf{x}_0 \\ y(k) &= \mathbf{C}\mathbf{x}(k) \end{aligned} \tag{5.11}$$

where $\mathbf{x}=[x_1 \ \cdots \ x_n]^T \in \mathbf{R}^n$ is the system state, $u \in \mathbf{R}$ is the control input, $\mathbf{A} \in \mathbf{R}^{n\times n}$ is the system matrix, $\mathbf{B} \in \mathbf{R}^{n\times 1}$ is the input matrix and $\mathbf{C} \in \mathbf{R}^{1\times n}$ is the output matrix so that $\mathbf{A}, \mathbf{B}, \mathbf{C}$ satisfy the controllability and observability conditions, and $\Delta\mathbf{A}, \Delta\mathbf{B}$ are the uncertain matrices.

Assumption 5.1: *The uncertain matrices* $\Delta\mathbf{A}, \Delta\mathbf{B}$ *are assumed to be bounded*

$$\begin{aligned} \|\Delta\mathbf{A}\| &\le \rho_A \\ \|\Delta\mathbf{B}\| &\le \rho_B \end{aligned} \tag{5.12}$$

The objective is to design a control law so that the output y tracks the constant reference r with the required performances.

From the chattering-free sliding-mode-based neural network controller design in Section 4.3, a neural controller structure, as shown in Figure 5.5, has been developed for the uncertain system (5.11).

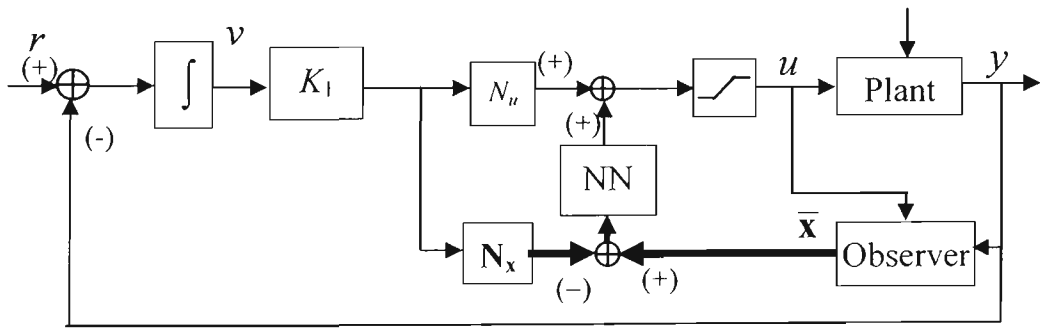


Figure 5.5: Structure of the chattering-free sliding-mode-based neural network controller system

In this system, the neural network (NN) is trained on-line by the CFSMBP algorithm. The constants N_u and $\mathbf{N}_x = [N_{x1}, N_{x2}, N_{x3}]^T$ are obtained from $\begin{bmatrix} \mathbf{N}_x \\ N_u \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & D \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$, and a small integral gain K_I is chosen to attain zero steady state error even in the presence of system parameters' uncertainties. This neural control system structure is more complicated than that proposed in Section 4.3, because the output y of the real-time system is required to track a constant reference input r. The on-line neural controller in Figure 5.5 has been applied in the SVC system, and system performances are even better than those of the neural control system using off-line neural controller presented in the following. However, the research in this chapter is to highlight the nonlinear mapping capability of the feedforward NNs.

From the neural control system in Figure 5.5, an off-line neural controller structure, as shown in Figure 5.6, is derived. The FNN consists of $(n+1)$ input nodes, $(m+1)$ hidden nodes, and one output node. The hyperbolic tangent sigmoid, $f_h(v) = \frac{1-e^{-v}}{1+e^{-v}}$, is used as the activation function for the hidden nodes and the linear function, $f_o(v) = v$, is used as the activation function for the output node.

The input vector $\mathbf{x}(t) = (x_1(k), ..., x_n(k))^T$ of the neural network is also the vector of system states $\mathbf{x}$ in (5.11). A constant input of 1, affecting the bias, is assigned to the vector of augment inputs $\tilde{\mathbf{x}} = (x_1(k), ..., x_n(k), 1)^T$. Let $\bar{\mathbf{W}} = \begin{bmatrix} \bar{W}_{11} & \cdots & \bar{W}_{1n} \\ \vdots & \ddots & \vdots \\ \bar{W}_{m1} & \cdots & \bar{W}_{mn} \end{bmatrix}$ be the weight matrix between the input layer and the hidden layer, and $\tilde{\bar{\mathbf{W}}} = \begin{bmatrix} \bar{W}_{11} & \cdots & \bar{W}_{1(n+1)} \\ \vdots & \ddots & \vdots \\ \bar{W}_{m1} & \cdots & \bar{W}_{m(n+1)} \end{bmatrix}$ be the matrix of augmented weights by including the bias weights components $\bar{W}_{i(n+1)}; i = 1, \ldots, m$.

The vector $\bar{\mathbf{o}} = (\bar{o}_1, ..., \bar{o}_m)^T ; \bar{o}_i = f_{hi}\left(\sum_{j=1}^{n} \bar{W}_{ij} x_j + \bar{W}_{i(n+1)}\right)$ is the vector of the output signals of the neurons in the hidden layer. A constant input of 1, affecting the bias, is assigned to the vector of augment outputs $\tilde{\bar{\mathbf{o}}} = (\bar{o}_1, ..., \bar{o}_m, 1)^T$. Let $\mathbf{W} = [W_1 \ \cdots \ W_m]$ be the weight vector between the hidden layer and the output layer, and $\tilde{\mathbf{W}} = [W_1 \ \cdots \ W_{m+1}]$ be the vector of augmented weights by including the bias weights component W_{m+1}.

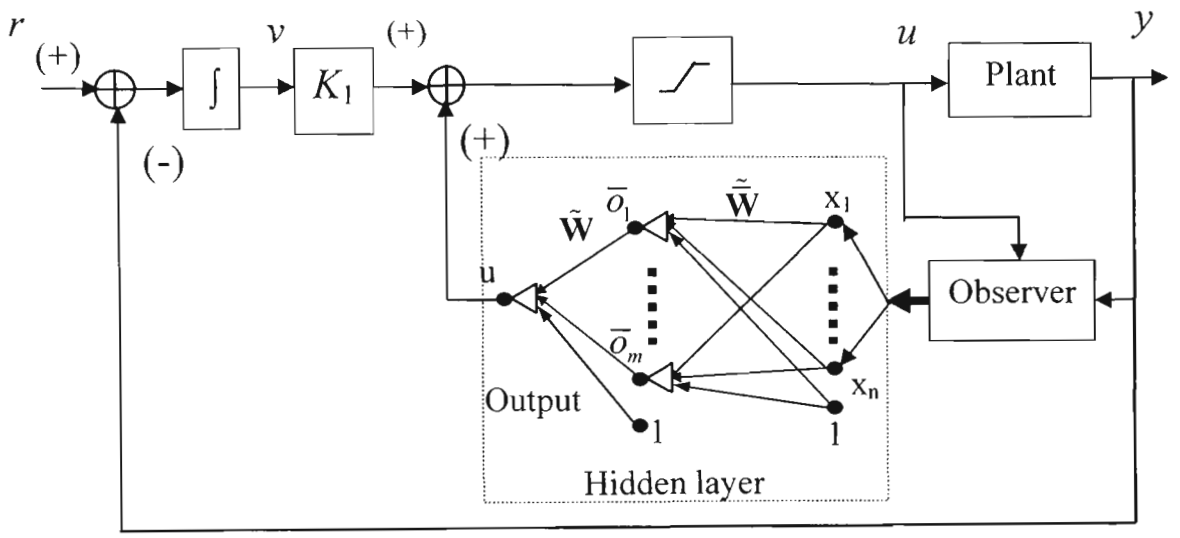


Figure 5.6: Structure of the proposed neural control system.

The output of the neural network is defined as:

$$u_{NN}(\mathbf{x}) = \tilde{\mathbf{W}}\tilde{\bar{\mathbf{o}}} = \sum_{i=1}^{m} W_i \bar{o}_i + W_{m+1} = \sum_{i=1}^{m} W_i f_{hi}\left(\sum_{j=1}^{n} \bar{W}_{ij} x_j(k) + \bar{W}_{i(n+1)}\right) + W_{m+1} \quad (5.13)$$

Because the system is required to respond to the step input r, the integral component, as shown in Figure 5.6, is utilised to attain robust tracking performances.

The integrator state equation is

$$v(k+1) = v(k) + r(k) - y(k), \quad (5.14)$$

The control input is given by

$$u(k) = u_{NN}(\mathbf{x}(k)) + K_1 v(k), \quad (5.15)$$

Therefore, the control objective is reduced to a more simple task, which is to develop a comprehensive training procedure so that the NN can approximate a nonlinear state feedback controller for the system (5.11).

The control design methodology is divided into four stages. Firstly, a sliding function is defined and used as the training error. This can avoid the need for unknown plant Jacobian or a trained neural network identifier. Secondly, the on-line training algorithm is applied into the NN controller such that the stability of the neural control system without reference input is guaranteed. Next, the on-line training process is repeated within a number of circles until the NN weights can converge to optimal values. Finally, the other parameters of the neural controller, as seen in Figure 5.6, are calculated, and the trained neural network controller is applied in simulations and experiments of the SVC system.

5.3.1 Design of the sliding hyperplane

Tsai et al. (2004) and Nguyen et al. (2004b) used a reduced-order sliding function as the training error for the neural controllers. Nevertheless, the gradient of V with regard to the NN weights includes an integral component, which results in difficult calculations of the weight updating rule.

Therefore, a novel sliding function is defined as:

$$s_k = \delta\sigma_k + \sigma_{k+1}, \quad \delta\sigma_k\big|_{k=0} = -\sigma_{k+1}\big|_{k=0}, \tag{5.16}$$

where δ is a scalar, and

$$\sigma_k = \mathbf{H}\mathbf{x}(k) = h_1 x_1(k) + h_2 x_2(k) + \cdots + h_n x_n(k), \tag{5.17}$$

where the vector $\mathbf{H} = [h_1, ..., h_{n-1}, h_n]$ is designed so that all roofs of the polynomial

$$h_1 + h_2 z + \cdots + h_n z^{n-1} = 0$$

lie within the unit circle in the z plane.

Since $\mathbf{A}, \mathbf{B}$ satisfy the controllability condition, there exists an invertible transformation matrix $\mathbf{T}$ and a new state:

$$\mathbf{z}(k) = \mathbf{T}^{-1}\mathbf{x}(k) \tag{5.18}$$

so that a controllable canonical system can be established as:

$$\mathbf{z}(k+1) = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\mathbf{z}(k) + \mathbf{T}^{-1}\mathbf{B}u(k). \tag{5.19}$$

The following theorem will show how the sliding function s_k may be found in an explicit form using the pole placement method.

Theorem 5.1: *For a row vector* $\mathbf{v} = [\alpha_1 \quad \cdots \quad \alpha_{n-1} \quad 1]$ *obtained directly from the following equation:*

$$\alpha_1 + \cdots + \alpha_{n-1} z^{n-2} + z^{n-1} = (z - z_1)\ldots(z - z_{n-1}),$$

if $\delta = -z_n$ *and* $\mathbf{H} = \mathbf{v}\mathbf{T}^{-1}$, *then* $z_1, z_2, \cdots, z_n$ *are the eigenvalues of the system dynamics in the hyperplane* $s_k = 0$.

Proof: Substituting Equations (5.17) and (5.19) into (5.16) yields

$$s_k = \delta\sigma_k + \sigma_{k+1} = \delta\mathbf{H}\mathbf{x}(k) + \mathbf{H}\mathbf{x}(k+1) = \delta\mathbf{H}\mathbf{T}\mathbf{z}(k) + \mathbf{H}\mathbf{T}\mathbf{z}(k+1). \tag{5.20}$$

If $\mathbf{H} = \mathbf{v}\mathbf{T}^{-1}$, then $\mathbf{v} = \mathbf{H}\mathbf{T}$, and Equation (5.20) becomes:

$$s_k = \delta\mathbf{v}\mathbf{z}(k) + \mathbf{v}\mathbf{z}(k+1). \tag{5.21}$$

With the controllable canonical system (5.19), the dynamic equation (5.21) can be cast into a state-space form as:

$$\mathbf{z}(k+1) = \mathbf{F}\mathbf{z}(k) + \mathbf{G}s_k \tag{5.22}$$

where the matrices $\mathbf{F} \in \mathbf{R}^{n \times n}$ and $\mathbf{G} \in \mathbf{R}^{n}$ are in the controllable canonical form:

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & . & . \\ . & 0 & 1 & . \\ . & . & . & . \\ -\delta\alpha_1 & -\delta\alpha_2 - \alpha_1 & . & -\delta - \alpha_{n-1} \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} 0 \\ . \\ 0 \\ 1 \end{bmatrix}. \tag{5.23}$$

In the hyperplane $s_k = 0$, the dynamic system (5.22) becomes:

$$\mathbf{z}(k+1) = \mathbf{F}\mathbf{z}(k),$$

and the eigenvalues $z_1, z_2, \cdots, z_n$ of the system dynamics can be found from:

$$\det[z\mathbf{I} - \mathbf{F}] = 0$$

$$\Leftrightarrow \det \begin{bmatrix} z & -1 & 0 & \cdots & 0 \\ 0 & z & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 \\ \delta\alpha_1 & \delta\alpha_2 + \alpha_1 & \delta\alpha_3 + \alpha_2 & \cdots & z + \delta + \alpha_{n-1} \end{bmatrix} = 0$$

$$\Leftrightarrow (z+\delta)\left(\alpha_1 + \cdots + \alpha_{n-1} z^{n-2} + z^{n-1}\right) = 0.$$

Therefore, if $\alpha_1 + \cdots + \alpha_{n-1} z^{n-2} + z^{n-1} = (z - z_1)\ldots(z - z_{n-1})$ and $\delta = -z_n$, then $z_1, z_2, \cdots, z_n$ are the eigenvalues of the system dynamics in the hyperplane $s_k = 0$. ■

Remark 5.1: The closed-loop system exhibits the desired dynamics behaviour after the sliding function attains zero. Therefore, if the control signal can keep the system states on the hyperplane $s_k = 0$, then $z_1, z_2, \cdots, z_n$ are the desired eigenvalues of the closed-loop system.

Remark 5.2: If the matrices $\mathbf{A}, \mathbf{B}$ are in the canonical form, that is, $\mathbf{T}$ is a unit matrix, then the sliding function s_k can be obtained directly from n desired eigenvalues of the closed-loop system.

Remark 5.3: The desired poles $z_1, z_2, \cdots, z_n$ of the closed-loop system are always located inside the unit circle in the z plane, thus matrix $\mathbf{F}$ in (5.23) is stable. If s_k is driven to zero by the controller action, then all system states $\mathbf{z}$ as well as $\mathbf{x}$ will converge to zero. This suggests that s_k can be used as a supervisor training signal for the neural network controller designed in the next stage. Because $\mathbf{F}$ is a stable matrix, there exists symmetric positive definite matrix $\mathbf{P} \in \mathbf{R}^{n \times n}$ satisfying the Lyapunov equation

$$\mathbf{F}^T \mathbf{P} \mathbf{F} - \mathbf{P} = -\mathbf{Q} \tag{5.24}$$

for any given symmetric positive matrix $\mathbf{Q}$.

Remark 5.4: The vector $\mathbf{H}$ chosen in Theorem 5.1 yields:

$$\mathbf{HB} = \mathbf{v}\mathbf{T}^{-1}\mathbf{B} = \begin{bmatrix} \alpha_1 & \cdots & \alpha_{n-1} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = 1$$

With $\mathbf{HB} > -|\mathbf{H}\Delta\mathbf{B}|$, the following assumption can be approached:

Assumption 5.2: There exist positive scalars g_0, g_1 satisfying

$$g_0 \le \mathbf{H}(\mathbf{B} + \Delta\mathbf{B}) \le g_1 \tag{5.25}$$

5.3.2 Application of the on-line learning algorithms with system stability guarantees

The aim of the training process is to converge the NN weights to optimal values, so the trained NN can approximate the nonlinear state feedback controller. For this training purpose, a novel neural control structure is introduced, as shown in Figure 5.7. The NN precedes the plant and receives input from the system state output.

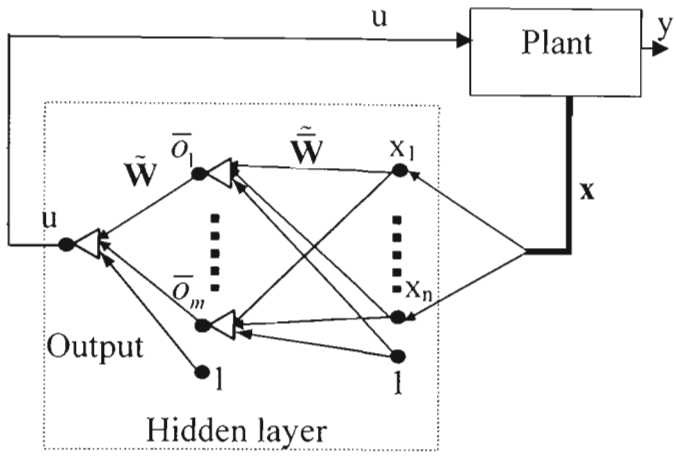


Figure 5.7: Structure of the neural control system for training process.

In contrast to the generalised inverse learning method (Psaltis et al., 1988), in the proposed training scheme, the neural controller on-line regulates the system state **x** from an arbitrary initial state to zero. Therefore, the on-line learning algorithm must guarantee system stability. Moreover, as in Remark 5.1, the closed-loop system exhibits the desired dynamics behaviour after the sliding function attains zero. Thus the training purpose is to minimise the sliding function s_k.

All network parameters are defined as in Section 5.3. The output of the neural network is as:

$$u_{NN}(\mathbf{x}) = \tilde{\mathbf{W}}\tilde{\bar{\mathbf{o}}} = \sum_{i=1}^{m} W_i \bar{o}_i + W_{m+1} = \sum_{i=1}^{m} W_i f_{hi}\left(\sum_{j=1}^{n} \bar{W}_{ij} x_j(k) + \bar{W}_{i(n+1)}\right) + W_{m+1} \quad (5.26)$$

Assumption 5.3: *Due to the physical constraints, the magnitudes of* $\mathbf{W}$ *and* $\bar{\mathbf{W}}$ *are assumed to be bounded by:*

$$\|\mathbf{W}\| \le B_W, \qquad \|\bar{\mathbf{W}}\| \le B_{\bar{W}} \tag{5.27}$$

Using s_k as the training error, the cost function is defined as:

$$V(k) = \tfrac{1}{2} s_k^2 \tag{5.28}$$

The CFSMBP learning algorithm of the weight vector $\tilde{\mathbf{W}}$ and the weight matrix $\tilde{\bar{\mathbf{W}}}$ are presented as follows:

$$\begin{aligned}
\Delta\tilde{\mathbf{W}}(k) &= -\frac{\eta}{\left\|\tilde{\bar{\mathbf{o}}}(k)^T\right\|^2 + \left\|\mathbf{F}_h'(k)\mathbf{W}(k)^T\tilde{\mathbf{x}}(k)^T\right\|^2} s_k \tilde{\bar{\mathbf{o}}}(k)^T \\
\Delta\tilde{\bar{\mathbf{W}}}(k) &= -\frac{\eta}{\left\|\tilde{\bar{\mathbf{o}}}(k)^T\right\|^2 + \left\|\mathbf{F}_h'(k)\mathbf{W}(k)^T\tilde{\mathbf{x}}(k)^T\right\|^2} s_k \mathbf{F}_h'(k)\mathbf{W}(k)^T\tilde{\mathbf{x}}(k)^T
\end{aligned} \tag{5.29}$$

where $\mathbf{F}_h'(k) = \begin{bmatrix} f_{h1}' & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f_{hm}' \end{bmatrix}$, f_{hi}', $(i = 1,...,m)$ is the derivative of $\bar{o}_i$ with regard to $\left(\sum_{j=1}^{n} \bar{W}_{ij} x_j(k) + \bar{W}_{i(n+1)}\right)$, and η is a positive scalar.

Proposition 5.1: *Consider that the closed-loop system consists of the dynamic system* (5.11)*, the controller* (5.26) *and the training algorithm (5.29), for an arbitrary positive scalar* β*, any symmetric positive matrix* $\mathbf{Q}$ *satisfied* $\mathbf{Q} - \beta^{-1}\mathbf{F}^T\mathbf{F} > 0$*, if Assumptions 5.1, 5.2 and 5.3 are satisfied and* η *is chosen as:*

$$0 < \frac{\vartheta_{\max}}{g_0} < \eta < \frac{1}{2g_1} + \frac{\vartheta_{\max}}{g_1}, \tag{5.30}$$

where $\vartheta_{\max} = k_{\max}\left[1+\sqrt{\dfrac{\lambda_{\max}\left(\beta\mathbf{G}^T\mathbf{P}^T\mathbf{P}\mathbf{G}+\mathbf{G}^T\mathbf{P}\mathbf{G}\right)}{\lambda_{\min}\left(\mathbf{Q}-\beta^{-1}\mathbf{F}^T\mathbf{F}\right)}}\right]1+\delta\|\mathbf{v}\|\Big],$

$$k_{\max} = \|\mathbf{A}\| + \rho_A + \delta + 0.5\left(\|\mathbf{B}\| + \rho_B\right) B_{\bar{W}} B_W,$$

then the closed-loop system is asymptotically stable.

Proof: See Appendix B.

The on-line backpropagation learning algorithms for the weight vector $\tilde{\mathbf{W}}$ and the weight matrix $\tilde{\bar{\mathbf{W}}}$ can also be obtained as follows:

$$\begin{aligned}\Delta\tilde{\mathbf{W}}(k) &= -\eta s_k \tilde{\bar{\mathbf{o}}}^T(k)\\ \Delta\tilde{\bar{\mathbf{W}}}(k) &= -\eta s_k \mathbf{F}_h' \mathbf{W}(k)^T \tilde{\mathbf{x}}(k)^T\end{aligned} \tag{5.31}$$

Proposition 5.2: *Consider that the closed-loop system consists of the dynamic system* (5.11)*, the controller* (5.26) *and the training algorithm* (5.31)*, for an arbitrary positive scalar* β*, any symmetric positive matrix* $\mathbf{Q}$ *satisfied* $\mathbf{Q}-\beta^{-1}\mathbf{F}^T\mathbf{F}>0$*, if Assumptions 5.1, 5.2 and 5.3 are satisfied and* η *is chosen as*

$$0 < \frac{\vartheta_{\max}}{g_0} < \eta < \frac{1}{2g_1} + \frac{\vartheta_{\max}}{g_1} \tag{5.32}$$

where $\vartheta_{\max} = k_{\max}\left[1+\sqrt{\dfrac{\lambda_{\max}\left(\beta\mathbf{G}^T\mathbf{P}^T\mathbf{P}\mathbf{G}+\mathbf{G}^T\mathbf{P}\mathbf{G}\right)}{\lambda_{\min}\left(\mathbf{Q}-\beta^{-1}\mathbf{F}^T\mathbf{F}\right)}}\right]1+\delta\|\mathbf{v}\|\Big],$

$$k_{\max} = \|\mathbf{A}\| + \rho_A + \delta + 0.5\left(\|\mathbf{B}\| + \rho_B\right) B_{\bar{W}} B_W,$$

then the closed-loop system is asymptotically stable.

Proof: See Appendix B.

Remark 5.5: With both the incremental BP and CFSMBP algorithms, the neural controller can work as a feedback adaptive controller and the controlled systems are asymptotically stable.

Another major issue is how to converge NN weights to optimal values. Therefore, a comprehensive training process will be developed in the next stage.

5.3.3 The training process for approximation of the nonlinear feedback controller

Choose Criterion Function

In the training process, the system state is initialised at arbitrary position $\mathbf{x}(0) = \mathbf{x}_0$. The neural network controller forces the state trajectory to stay on the sliding surface $s_k = 0$, then the system state will converge to zero with the required performance. With the defined sliding function, $s_k = 0$ in the initial instance. Because the initial weights of the neural networks are arbitrarily small, the sliding function s_k may have departed from zero. The online algorithm (5.29) or (5.31) will start to train the neural network to force s_k to zero. After system state $\mathbf{x}$ converges to zero, the training process is restarted with arbitrary initial system state $\mathbf{x}(0)$. If the neural network can exactly copy the nonlinear feedback controller for the uncertain system (5.11), $s_k = 0$ for all $t \geq 0$.

Therefore, a criterion function to stop the training phase is proposed as:

$$E = \frac{1}{T_m} \int_0^{T_m} 0.5 s(\tau)^2 \, d\tau \leq E_m \tag{5.33}$$

where T_m is a sufficient period such that the system state converges to zero, and E_m is a maximum tolerable error.

Data-processing

Because the input variables have very different orders of magnitude, it is recommended to rescale the data. In this way, more reliable predictions can be undertaken. The variables are rescaled to be included within the interval [-1, 1] by using the following equation:

$$x_i^{new}(k) = \frac{x_i^{old}(k)}{x_{i\max}} \tag{5.34}$$

where $x_i^{new}(k)$ and $x_i^{old}(k)$ are respectively the new and old values of the system state x_i at a sampling point k, and $x_{i\max}$ is the maximum amplitude of this state.

Training procedure

The proposed training method is shown in the following procedure.

1. (Initialisation) Initialise all weights to small random values $W_{mn}(0)$ and $\bar{W}_m(0)$. Choose a small positive learning rate η, sampling time T, maximum iteration $k_m = \frac{T_m}{T}$, maximum epoch value l_m and very small maximum tolerable error E_m. Set the iteration number $k = 0$, $E = 0$, set $\mathbf{x}(0)$ and $u(0)$, number of epochs $l = 0$.

2. (Forward propagation) Read the system state $\mathbf{x}(k+1)$ from the system (5.11), process the state inputs and propagate the inputs forward through the network to obtain the control signal:

$$u(k+1) = \tilde{\mathbf{W}}\tilde{\mathbf{o}} = \sum_{i=1}^{m} W_i o_i + W_{m+1} = \sum_{i=1}^{m} W_i f_{hi}\left(\sum_{j=1}^{n} \bar{W}_{ij} x_j(k) + \bar{W}_{i(n+1)}\right) + W_{m+1}$$

Calculate $s_k = \delta\sigma_k + \sigma_{k+1}$, $\delta > 0$, $\sigma_k = \mathbf{H}\mathbf{x}(k)$, $\sigma_{k+1} = \mathbf{H}\mathbf{x}(k+1)$,

$$E = E + 0.5s(k)^2.$$

3. (Backward propagation) Compute the changes of the weights for the next iteration using Equation (5.29) or (5.31).

   Update the weights

$$\tilde{\mathbf{W}}(k+1) = \tilde{\mathbf{W}}(k) + \Delta\tilde{\mathbf{W}}(k)$$
$$\tilde{\bar{\mathbf{W}}}(k+1) = \tilde{\bar{\mathbf{W}}}(k) + \Delta\tilde{\bar{\mathbf{W}}}(k)$$

4. (One epoch loop) Check whenever one training cycle has been finished. If $k \le k_m$, then set $k = k+1$, apply control signal $u(k+1)$ to the system (5.11), and go to step 2; otherwise, go to step 5.

5. (Total error checking): Check whenever the total error is acceptable within l_m epochs:

   if $\left(\frac{E}{k_m} \le E_m\right)$ or $(l \ge l_m)$ then terminate the training process and output the final weights;

   otherwise, set $E = 0, k = 0, l = l+1$, $\mathbf{x}(0)$ and $u(0)$, and go to Step 2.

Note that the training procedure for approximation of the NN controller is an incremental training method. Thus the batch learning approaches cannot be applied in this training procedure.

5.3.4 Calculation of the other parameters of the neural controller

After the three design stages above, the trained NN can now work as a direct feedback controller, as shown in Figure 5.5. In practice, not all the state variables are measured.

So an observer should be used to estimate system states. If the measurement of output variable y involves significant noise and is relatively inaccurate, then the use of a full-order observer results in a better system performance.

Assume that the state $\mathbf{x}$ is approximated by the state $\bar{\mathbf{x}}$ of the dynamic equation

$$\bar{\mathbf{x}}(k+1) = (\mathbf{A} - \mathbf{K}_e\mathbf{C})\bar{\mathbf{x}}(k) + \mathbf{B}u(k) - \mathbf{K}_e y(k) \tag{5.35}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are parameters of the nominal system (5.7) and the observer gain $\mathbf{K}_e$ can be calculated by the pole placement methods demonstrated in Franklin et al. (2002).

For the step input, $\mathbf{x}(k), u(k), v(k)$ approach the constant values $\mathbf{x}(\infty), u(\infty), v(\infty)$, respectively. Therefore, the integrator state in Equation (5.14) leads to:

$$v(\infty) = v(\infty) + r - y(\infty)$$

or $\quad y(\infty) = r$.

Clearly, there is no steady-state error in the output when a small value of integral gain K_I can be chosen.

5.4 Experimental Results

Consider again the SVC system model (5.7):

$$\begin{aligned} \mathbf{x}(k+1) &= (\mathbf{A} + \Delta\mathbf{A})\mathbf{x}(k) + (\mathbf{B} + \Delta\mathbf{B})u(k), \\ y(k) &= \mathbf{C}\mathbf{x}(k) \end{aligned}$$

The model parameters vary between three sets, as described in Equations (5.8), (5.9) and (5.10).

5.4.1 Sliding function design

Suppose that the closed-loop system performance has no more than 2% overshoot and a rise time of no more than 0.4 seconds. Thus a damping ratio $\zeta = 0.8$ will meet the overshoot requirement, and for this damping ratio, a rise time of 0.4 seconds suggests a natural frequency

$$\omega_n = 4\frac{1.8}{t_r} = 4\frac{1.8}{0.4} = 18.$$

Therefore two dominant poles are given by:

$$z_{1,2} = e^{T\left(-\zeta\omega_n \pm \omega_n\sqrt{1-\zeta^2}\right)} = e^{0.025(-14.4 \pm j10.8)} = 0.6724 \pm j0.18609$$

The last pole needs to be placed far to the left of the dominant pair, and we assume a factor of 2 in the respective undamped natural frequencies to be adequate. Thus the last desired pole is attained as:

$$z_3 = e^{T(-2\times 18)} = 0.40657$$

A sliding function for the nominal system, System 1, is designed as follows:

$$s(k) = \delta\sigma_k + \sigma_{k+1}; \quad \delta > 0 \tag{5.36}$$

where

$$\begin{aligned} \sigma_k &= \mathbf{H}\mathbf{x}(k), \quad \mathbf{H} = [h_1, h_2, h_3] \\ &= h_1 x_1(k) + h_2 x_2(k) + h_2 x_2(k) \end{aligned} \tag{5.37}$$

From Theorem 5.1, the sliding function parameters are obtained as:

$$\begin{aligned} \delta &= -z_3 = -0.4066 \\ \mathbf{H} &= [4.0583, -0.20269, 0.0092474]. \end{aligned}$$

5.4.2 Application of the on-line learning algorithms to the neural controller

A feedforward neural network controller, as illustrated in Figure 5.7 with four input nodes, four hidden nodes and one output node, is utilised.

The on-line learning algorithm (5.29) or (5.31) is used for training the network. The NN weights are initialised by random small values. From the experimental results, if the learning rate η is chosen as $0 < \eta < 0.55$, the neural control system is stable. This validates the theoretical development in Section 5.3.2. For the CFSMBP algorithm with the chosen learning rate $\eta = 0.1$, the system outputs and control inputs of the three representational systems are shown in Figure 5.8. Figure 5.9 shows the system performance results when the incremental BP algorithm is utilised with the chosen learning rate $\eta = 0.01$.

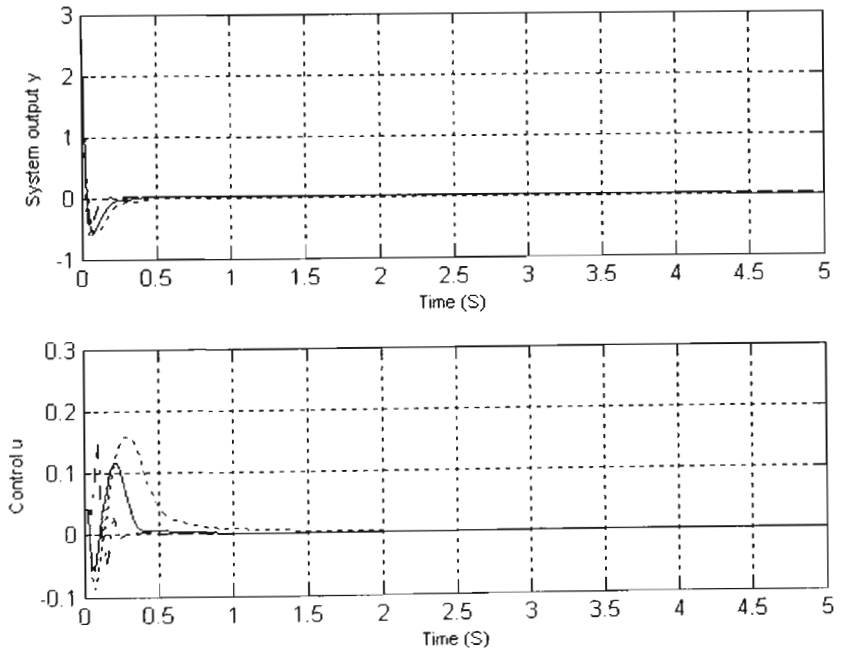


Figure 5.8: Output and control signals of System 1 (solid line), System 2 (dashed line) and System 3 (dash-dot line), using the CFSMBP algorithm.

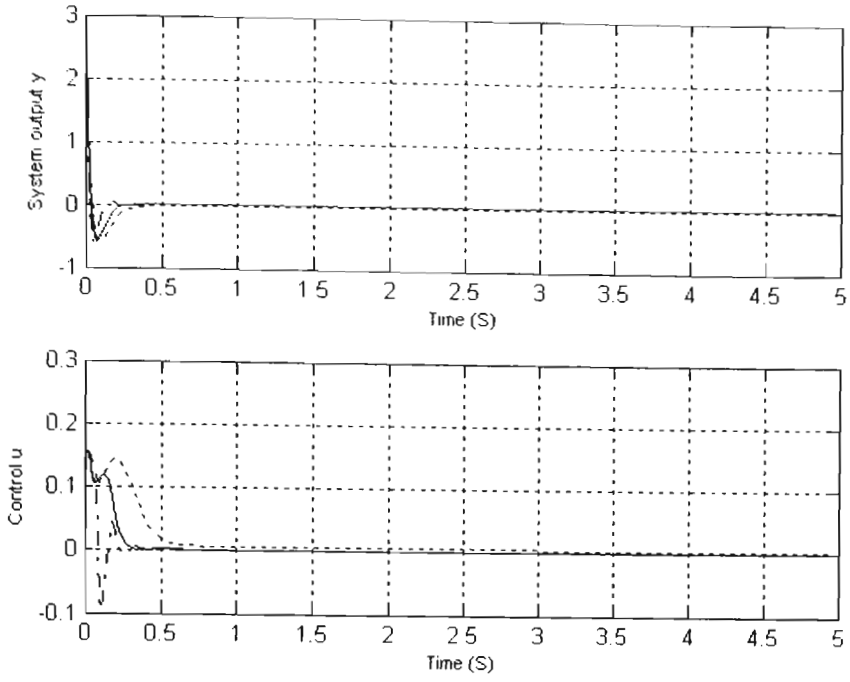


Figure 5.9: Output and control signals of System 1 (solid line), System 2 (dashed line) and System 3 (dash-dot line), using the BP algorithm.

5.4.3 Training process

Experiment 1:

In the first experiments, the NN controller with the CFSMBP learning algorithm (5.29) and $\eta = 0.1$ is applied to the nominal system 1.

From simulations in Section 5.4.2, we can choose $T_m = 2$ seconds. We perform some simulations from Section 5.4.2 again with different initial states, and the maximum amplitudes of three states are estimated as:

$$x_{1m} = 3.5; \quad x_{2m} = 50; \quad x_{3m} = 480$$

The initial weights are drawn at random between -0.1 and 0.1. The maximum tolerable error $E_m = 10^{-4}$ can be chosen. After hundreds of iterations, the training process is converged, and the NN weights approach optimal parameters. Figure 5.10 shows the

output and sliding function responses of the controlled system, using the trained NN controller with its optimal weights.

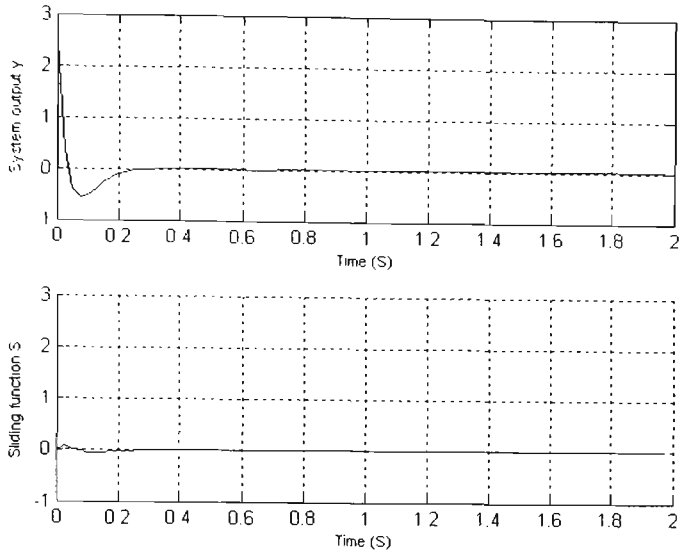


Figure 5.10: System output and sliding function responses when the trained neural controller with optimal parameters is utilised.

Experiment 2:

In these experiments, both the CFSMBP and incremental BP algorithms are applied for training the neural controller in order to compare the convergence rates.

A first set of weights is randomly generated and kept constant while the learning rate is tuned to obtain the fastest possible convergence. Then 29 additional initial random weight settings are chosen, and each of the algorithms is executed using these sets of weights with the optimal learning rate obtained from the tuning procedure. Table 5.2 show the experimental results for a 4-4-1 network, where η^* means the optimal learning rate, % means the percentage of global convergence, which counts the percentage of runs (over 30 runs) that successfully converge within $l_m = 3\times10^4$ e pochs, R means the average number of epochs to converge over the converged runs.

Table 5.2: Performance comparison for training the neural controller.

Leaning algorithm	η^*	R	%
Incremental BP	0.12	1221.47	100
CFSMBP	0.34	725.3	100

Experiment 3:

In these experiments, the NN controller with the on-line learning algorithm (5.29) or (5.31) is applied to the nonlinear SVC system represented by a bank of linear models (5.2). The NN in these experiments is trained to approximate a nonlinear feedback controller for the nonlinear system (5.11).

The training procedure in this case is the same as that in Section 5.3.3, but the system state space dynamics (5.8), (5.9) and (5.10) are sequentially used for the training process. The initial weights are drawn at random between -0.1 and 0.1. The number of hidden nodes is increased from 2 to 10. The maximum tolerable error is set to $E_m = 3 \times 10^{-4}$.

Similar to the first experiments, an optimal learning rate is obtained from the tuning procedure using the first set of weights, then 29 additional initial random weight settings are chosen, and the proposed algorithm is executed using these sets of weights with the optimal learning rate.

Tables 5.3 and 5.4 show the experimental results when the CFSMBP algorithm and BP algorithms are utilised, respectively.

Table 5.3: Performance for training the neural controller using the CFSMBP learning algorithm and different numbers of hidden nodes.

No. of hidden nodes	η^*	R	%
2	0.08	463.83	100
3	0.1	336.71	100
4	0.1	348.23	100
5	0.09	341.42	100
6	0.12	312.43	100
7	0.09	359.7	100
8	0.09	371.68	100
9	0.09	372.74	100

Table 5.4: Performance for training the neural controller using the BP learning algorithm and different numbers of hidden nodes.

No. of hidden nodes	η^*	R	%
2	0.02	1126.87	100
3	0.03	687.4	100
4	0.03	688.4	100
5	0.02	1016.43	100
6	0.02	994.33	100
7	0.015	1287.8	100
8	0.015	1252.93	100
9	0.01	1885.9	100

From Table 5.3, we can choose $m = 6$ for the best result. By the training procedure in the second experiments, a 4-6-1 FNN is trained to approximate the nonlinear feedback controller for the three representational systems: System 1 in (5.8), System 2 in (5.9) and System 3 in (5.10). The detail program of the training process can be found in Appendix C. After training, the network parameters are obtained as

$$\tilde{\mathbf{W}} = [0.50661 \quad -0.83856 \quad 1.39899 \quad 0.020495 \quad -0.72583 \quad -0.008715]$$

$$\tilde{\tilde{\mathbf{W}}} = \begin{bmatrix} 0.447966845 & -0.317514887 & 0.176749624 & 0.047171784 \\ -0.783156952 & 0.419020834 & -0.341294348 & 0.001924934 \\ 1.249068789 & -0.758633954 & 0.608523292 & -0.005631910 \\ 0.059103632 & 0.029943808 & -0.024164411 & 0.001681880 \\ -0.639445163 & 0.371328805 & -0.362590229 & -0.004127458 \end{bmatrix}$$

These parameters will then be used in the next stage.

5.4.4 The neural controller design and implementation in the real-time system

The observer poles can be chosen to be faster than the controller poles by a factor of 5. In this case, the desired poles of the observer are chosen as:

$$z_{1,2,3} = e^{T_s[-72+54*i \quad -72-54*i \quad -90]} = [0.036202 + 0.16129*i \quad 0.036202 - 0.16129*i \quad 0.1054].$$

Using the Ackermann formula, the observer gain can be designed as:

$$\mathbf{K}_e = [k_{e3} \quad k_{e2} \quad k_{e1}]^T = [1.7462 \quad 103.115 \quad 2206.8345]^T$$

A small integral gain is chosen $K_I = 0.17$.

The design procedure of the controller parameters is written in a Matlab programme, which is detailed in Appendix D.

With the NN controller parameter obtained in the previous stage, some simulation results for the dynamic systems is shown in Figure 5.11. Figure 5.11 shows the closed-loop system outputs and control inputs of the three representational systems.

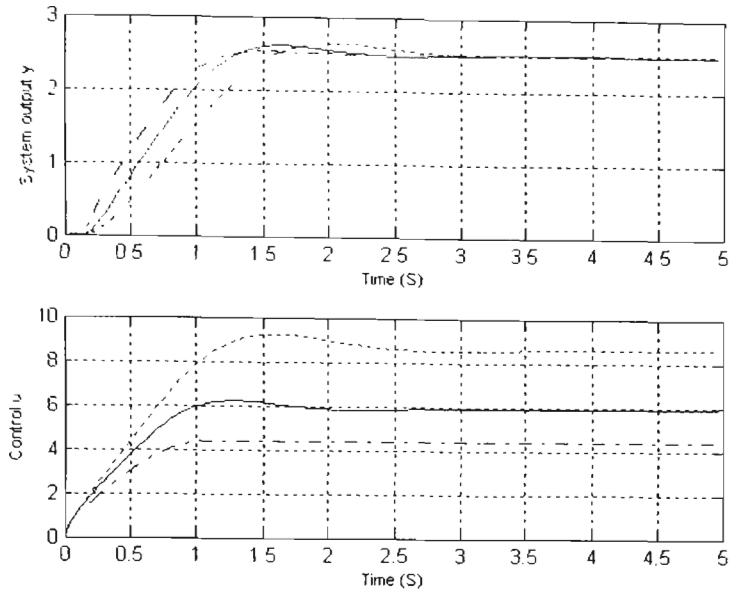
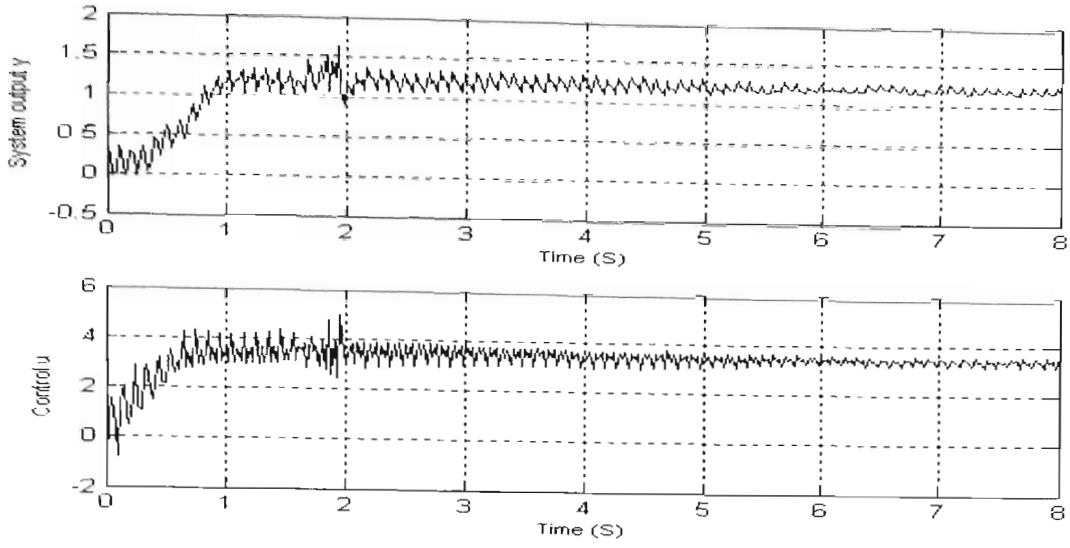


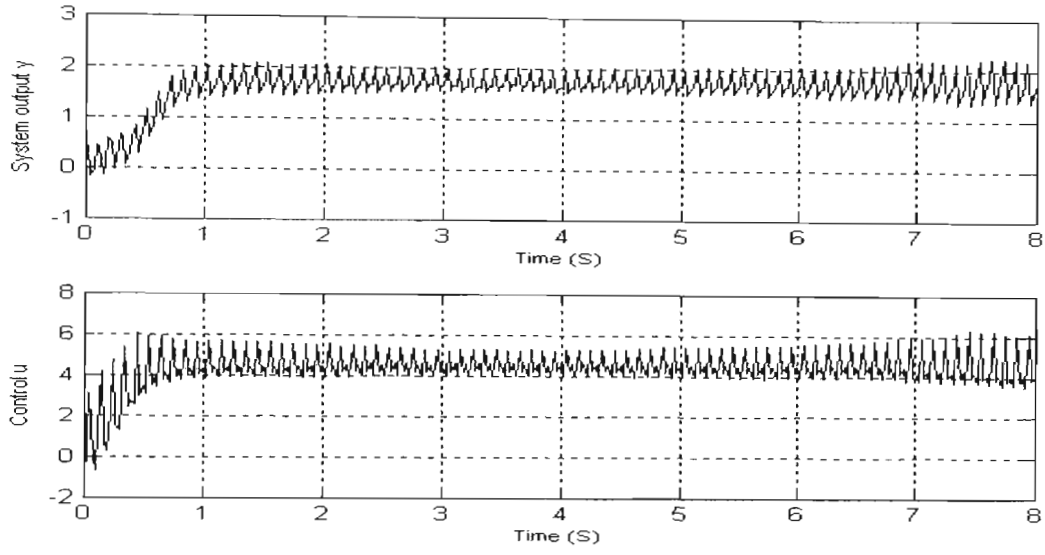
Figure 5.11: Output and Control Signals of the system 1 (solid line), System 2 (dashed line) and System 3 (dash-dot line) using the proposed neural controller.

In the real-time experiment, we use the software Turbo Pascal, applied to all controllers and hardware. The program includes many individual modules for graphic interface, reading/writing ADC/DAC card, applying controller algorithms and others. An engineer can monitor the system states or controller signal via specific windows, revise the parameters of the controller and windows online. The timer module offers interrupt request at every fixed sampling time, thus the controller algorithm is calculated and implemented in real time while the system states and controller signal are displaying.

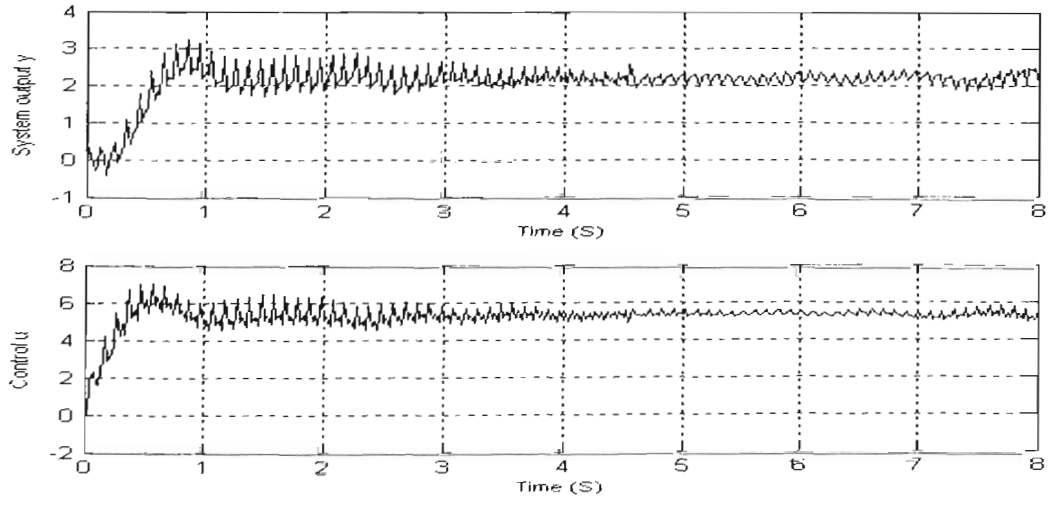
For the NN controller, the network and observer parameters are left exactly as in Sections 5.4.3 and 5.4.4, except for a slight change of the integral gain $K_1 = 0.22$. Figure 5.12 displays the system outputs and control signals of the neural controlled system with different reference inputs.



(a)



(b)



(c)

Figure 5.12: The output and control input signals of the SVC system, using the neural controller designed with the reference inputs of 1.2 (a), 1.7 (b) and 2.2 (c).

In order to compare the control performance, a PID controller is designed for the time-delay SVC system as:

$$G_c(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right) \tag{5.38}$$

where K is the proportional gain, T_I the integral gain and T_D the differential gain.

From the Ziegler-Nichols Rules (Nguyen, 2004), the PID controller parameters can be determined from the reaction rate R and the transportation lag L.

$$\begin{aligned} R &= \frac{K}{\tau}; \quad L = t_d \\ K_p &= \frac{1.2}{RL}; \quad T_I = 2L; \quad T_D = 0.5L \end{aligned} \tag{5.39}$$

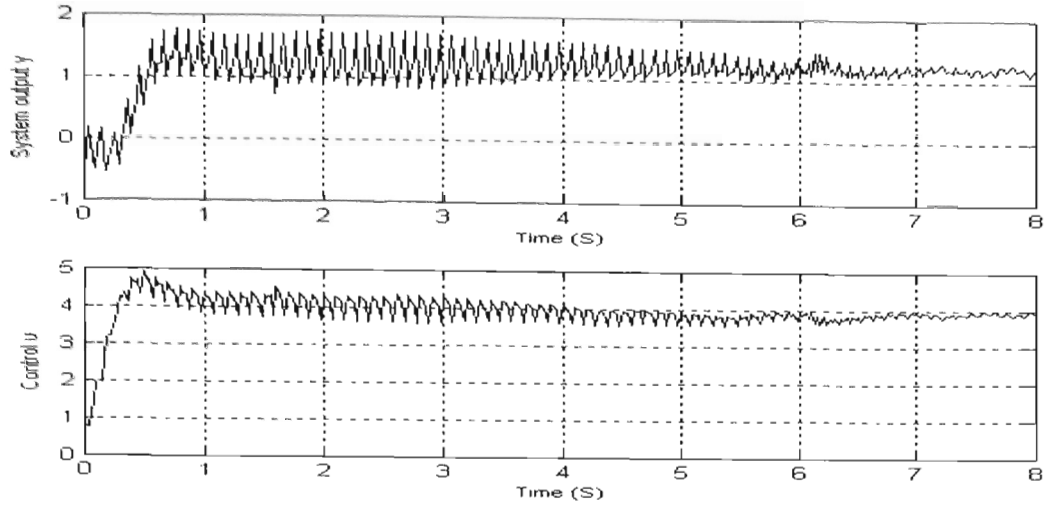
An improved Ziegler-Nichols rule can also be applied to maximise the integral gain

$$\begin{aligned} K_p &= \frac{1}{K}\min\left(0.4\frac{\tau}{t_d}, 0.25\right) \\ K_I &= \frac{K_p}{T_I} = \frac{1}{Kt_d}\max\left(0.1\frac{\tau}{t_d}, 0.5\right) \end{aligned} \tag{5.40}$$

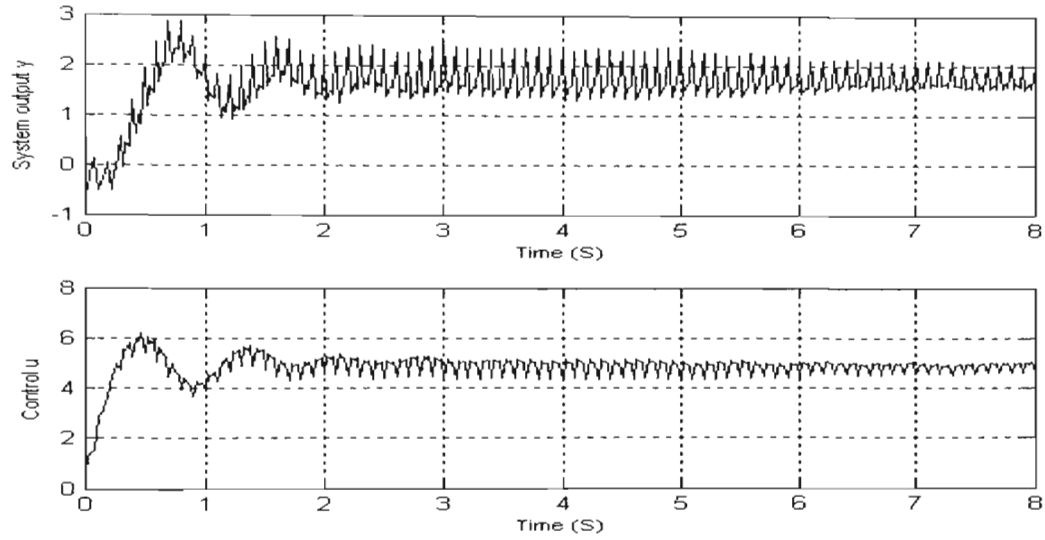
With the nominal system, the PID controller's parameters can be attained as

$$\begin{aligned} K_p &= \frac{1}{0.425}\min\left(0.4\frac{0.165}{0.1}, 0.25\right) = \frac{0.25}{0.425} = 0.59 \\ K_I &= \frac{K_p}{T_I} = \frac{1}{0.425\times 0.1}\max\left(0.1\frac{0.165}{0.1}, 0.5\right) = \frac{0.5}{0.425\times 0.1} = 11.765 \\ K_D &= K_P T_D = 0.59\times 0.05 = 0.0295 \end{aligned}$$

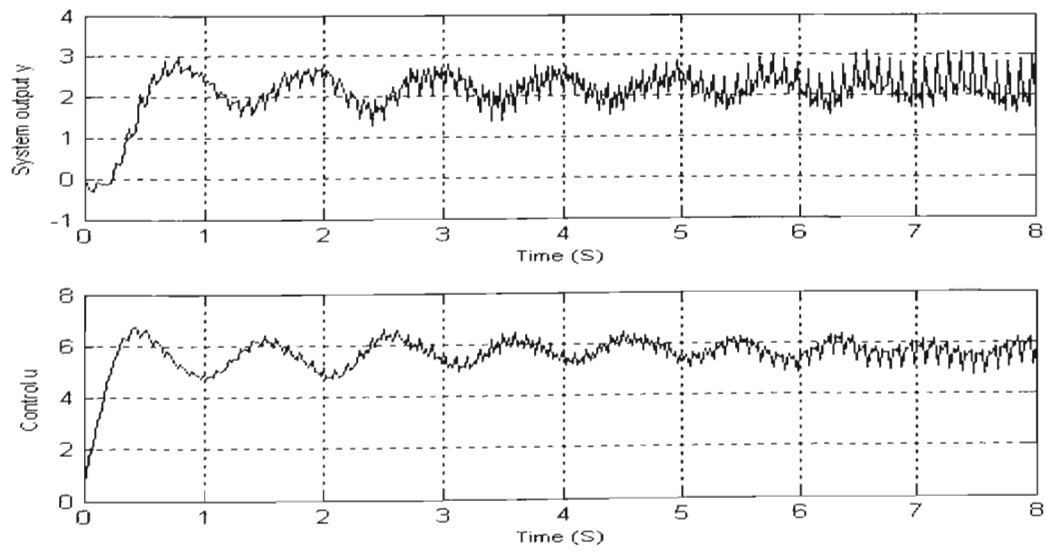
In the real-time experiment, the parameters of the PID controller are appropriately tuned to obtain the best performance. The optimal values $K_P = 0.59$, $K_I = 0.358$ and $K_I = 2$ were finally chosen. Figure 5.13 shows the experimental results of the PID controlled system with different reference inputs.



(a)



(b)



(c)

Figure 5.13: The output and control input signals of the SVC system using the PID controller designed with the reference inputs of 1.2 (a), 1.7 (b) and 2.2 (c).

5.5 Discussion and conclusion

As seen in Figures 5.8 and 5.9, with the on-line learning neural controller, the system outputs always converge to zero. Therefore, both the on-line BP and CFSMBP algorithms can guarantee the system stability.

It is very promising, from the results illustrated in Figure 5.10, that $s_k \approx 0$ for all $t \geq 0$. This means that the neural network can exactly copy the state feedback controller. As a consequence, the closed-loop system exhibits the desired dynamic behaviour on the predefined sliding hyperplane.

The experimental results in Table 5.2 showed that the CFSMBP and BP algorithms both assure global convergence of the training process. In addition, the proposed CFSMBP algorithm outperforms the BP in terms of the convergence rate.

Table 5.3 and 5.4 demonstrate the converged property of the training process for the NN controller approximation. The simulation results show that the number of hidden nodes does not affect the convergence of the proposed method. Moreover, the CFSMBP algorithms can approximately converge twice as fast as the BP algorithm does.

From the experiment results shown in Figures 5.12 and 5.13, output performances of the neural control system and the PID control system have been reported in Table 5.5.

With the reference input of 1.2, no overshoot is observed in either the NN and PID control systems, and the rise times of the two systems are smaller than or equal to 0.4 seconds. In this case, the PID controller and the neural controller both guarantee the required performance.

For a step input of 1.7, the NN controller still assures the best performance, whereas the PID controller results in an overshoot of 29%. Therefore, the neural controller shows superior performance in comparison to the PID controller.

When the system must respond to a step input of 2.2, the system nonlinearity becomes more severe. Thus both of the NN and PID control systems have a maximum overshoot

of 22.7%. However, the neural controller successfully maintains the zero steady-state error after 2.4 seconds. On the contrary, the PID controller fails to track the system output to the reference input. This indicates the robustness capability of the system in the presence of uncertain nonlinearity when using the NN controller in comparison to the PID-based control system.

Table 5.5: Performance comparison between the NN and PID control systems with different reference inputs.

Control method	Reference input (V)	Rise time (s)	Max. overshoot (%)	Settling time (s)
PID	1.2	0.28	0	0.8
NN	1.2	0.40	0	1
PID	1.7	0.25	29	1.7
NN	1.7	0.35	0	1
PID	2.2	0.27	22.7	Large
NN	2.2	0.38	22.7	2.4

In conclusion, this chapter presents the neural network controller design methodology for a class of uncertain SISO systems, particularly in the time-delay nonlinear Static VAR Compensator system. Following the direct feedback neural controller structure, a novel sliding function is designed from the pole placement method, and is used as the neural network's training signal. A new training process for the neural network is proposed, in which the neural network works as a direct adaptive feedback controller. The incremental BP learning algorithm and the proposed CFSMBP on-line learning algorithm are both proven to stabilise the controlled system. Apparently, in the training phase, the CFSMBP approach outperforms the BP in term of convergence speed. At the end of the training process, the NN can approximate a nonlinear feedback controller. The proposed training procedure also eliminates the requirement of a careful choice of the training input signals, which is the shortcoming of the existing training procedure for the neural network identifier. Fruitful experimental results for the real-time Static VAR Compensator system can validate the effectively proposed method.

CHAPTER 6

DECENTRALISED NEURAL NETWORK CONTROLLER DESIGN FOR A CLASS OF INTERCONNECTED UNCERTAIN NONLINEAR SYSTEMS

In this chapter, a decentralised controller using neural networks is designed for a class of uncertain large-scale nonlinear systems with high-order interconnections and disturbances. The novel decentralised NN-based controller is developed from a sliding-mode-based feedback controller. Using a new predefined sliding surface, the robust sliding-mode-based neural network is trained to stabilise the system states on the sliding surface. Thus performance of the controlled system is guaranteed. Experimental results for a Coupled Electric Drives CE8 system show that the real-time neural controller has been implemented successfully.

6.1. Introduction

Controller design for complex systems with high levels of uncertainties is a current major challenge. When dealing with large-scale nonlinear systems, such as communication networks, satellite constellations, robotics systems and powered wheelchairs, the decentralised adaptive control schemes are usually used. Ikeda and Siljak (1980) showed that the linear interconnected system is decentralised and stabilisable by choosing appreciate gains of the local feedback controller. In Ioannou (1985), the direct model reference adaptive control schemes were presented to solve the decentralised control problem for the large-scale systems with first-order interconnections. An indirect adaptive controller is also introduced in Ossman (1989) for the interconnected systems. In these approaches, the global stability proof requires that the matrix **M** involving the bounds of interconnections is positive definite. This condition, however, cannot easily be checked a priori in the adaptive case, since the **M**-matrix entries depend on the unknown system parameters (Ossman, 1989). This problem was solved in Gavel and Siljak (1989) by using an alternative high-gain stabilisation technique with strictly matching uncertainties. Nevertheless, this adaptive method requires the known subsystem dynamics.

Large-scale systems with high-order interconnections were first considered by Shi and Singh (1992), who have made simpler assumptions about the strength of the interconnections. Although the method could handle unbounded interconnections and unknown parameters of subsystems, the dynamic subsystems were required in linear form. Wen and Soh (1997) relaxed the relative degree limitations of the linear subsystem and the structure condition of interconnections by using the integrator backstepping technique, but the use of infinite memory for the modelling error limits the applicability of the scheme. In Jain and Khorrami (1997), a decentralised adaptive controller was designed for a class of large-scale nonlinear system with known bounds of interconnections and disturbances. The approach, however, requires parameterisation of the system dynamics in linear in the parameter form. This is also the main drawback of most centralised adaptive control schemes.

Recently, a number of researchers have used neural networks to relax many restrictive assumptions mentioned above in the design of the decentralised controllers. Spooner and Passino (1999) proposed decentralised direct and indirect adaptive control schemes for uncertain nonlinear systems with restrictions on the interconnections. For each subsystem, a feedback linearising controller based on the sliding-mode control technique was designed, and a radial basic neural network was utilised to approximate the feedback controller. The NN update law in Spooner and Passino (1999) eventually has the same form of the sliding-mode-based learning algorithms. In Da (2000), a new type of the controller, fuzzy neural networks sliding-mode controller, was proposed for a class of large-scale system with unknown bounds of high-order interconnections and disturbances. In each subsystem, a sliding mode regulator of system error is passed back through a neural identifier for tuning the neural controller parameters. However, these methods are still restricted in the real-time applications.

In this chapter, the SMC-based feedback control and NN-based control approaches are studied for a class of uncertain nonlinear large-scale systems with bounds on the interconnections and disturbances. Utilising techniques from Nguyen et al. (2003) and Seshagiri and Khalil (2005), a sliding-mode-based state feedback controller is first developed for this uncertain nonlinear large-scale system. However, this control scheme often leads to a larger amplitude of control input, which in turn distorts system performance. To overcome this problem, a novel neural control structure is proposed.

The whole system has a parallel structure such that each subsystem has one FNN acting as a direct inverse adaptive controller. A novel sliding function is defined and used for training the neural controller. The NNs' weights are updated on-line by the RMBP learning algorithm, as mentioned in Chapter 4. The neural controller using the RMBP learning algorithm can stabilise the system states on the sliding surface. System performance is therefore satisfied by the appropriate choice of the sliding function parameters. The proposed method is tested in a real-time Coupled Electric Drives system.

6.2 Coupled Electric Drives CE8 as an Interconnected Uncertain Nonlinear System

Coupled drives systems are very common in industrial systems, such as textile processes, paper mills, rolling mills, wire manufacturing plants and every process requiring a continuous production line. In these systems, the material speed and tension controlled by drive systems is frequently required to adjust within defined limits. The coupled electric drives apparatus is a typical multivariable nonlinear system with interactions between subsystems. In Lee et al. (1991), the coupled drives apparatus was represented by two single-input single-output systems with the interactions within the multivariable considered as measurable disturbances. In each subsystem, the effect of the interactions was reduced by a self-tuning controller based on a feedforward paradigm and the generalised minimum variance control strategy. Also in Kocijan et al. (1997), the coupled drives apparatus was decomposed into two equivalent single-input single-output channels with disturbance inputs representing interaction influents. Using the Individual Channel Analysis and Design technique (Kocijan et al., 1997), each compensator was designed for each channel to meet the channel specification. More generally, in this research, the coupled electric drives system is considered as a large-scale nonlinear system with interconnections and disturbances.

In the Coupled Electric Drives CE8 system, as shown in Figure 6.1, an elastic belt loops around the two pulleys and an intermediate jockey pulley. The two pulleys are driven by two DC motors while the jockey pulley is suspended vertically by a spring. The tension of the elastic belt is provided by the extension of the spring and it is measured by a potentiometer. A pulse sensor on the jockey pulley produces a signal that is proportional

to the magnitude of the velocity. Because all parts of the elastic belt, as shown in Figure 6.1, are not identical to the connection of the belt, the tension output always suffers from a small vibration. The main object of this project is to control both tension and velocity effectively.

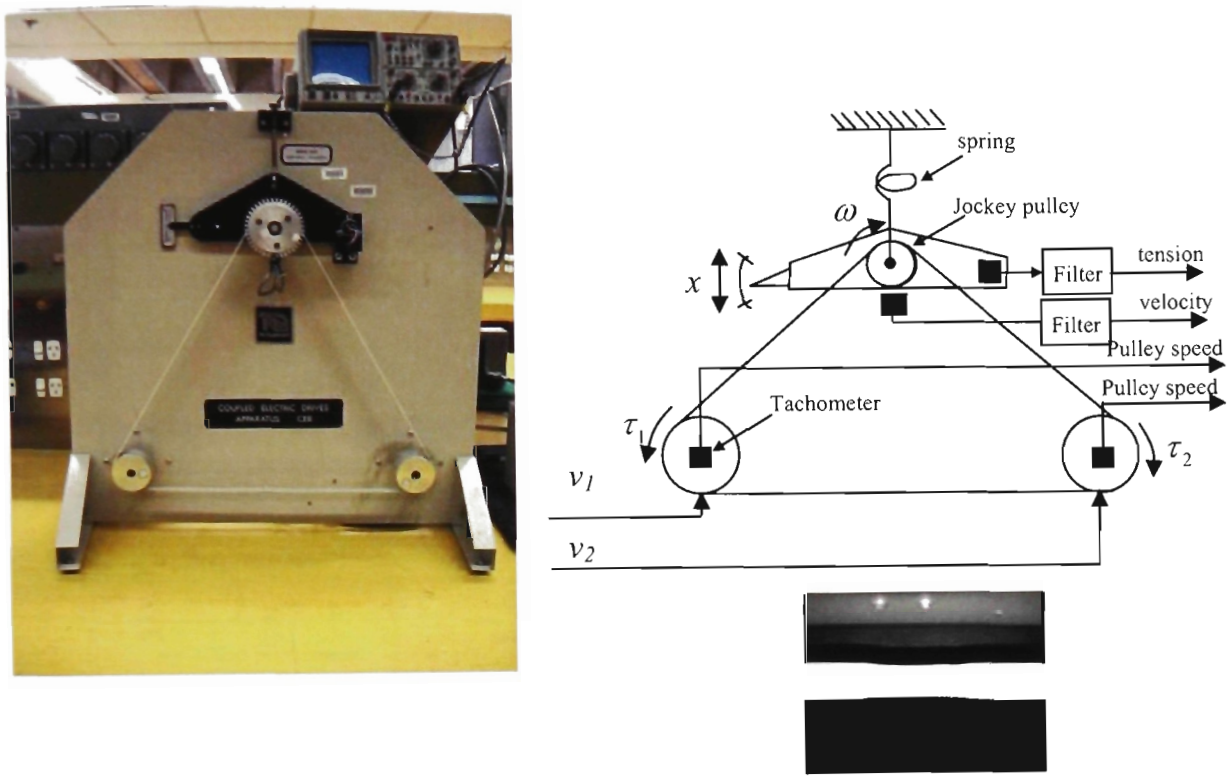


Figure 6.1: The Coupled Electric Drives CE8 System and a part of the elastic belt.

Let ω denote the pulley angular speed, x the displacement of the spring, v_1, v_2 the input voltages of motor 1 and motor 2, respectively. To decrease the effect of interactions between two subsystems, a pre-compensator (6.1) is often utilised, as shown in Figure 6.2.

$$\begin{aligned} v_1 &= \frac{1}{2}(u_\omega + u_x) \\ v_2 &= \frac{1}{2}(u_\omega - u_x) \end{aligned} \tag{6.1}$$

where u_ω, u_x are the control signals for the tension and speed subsystems

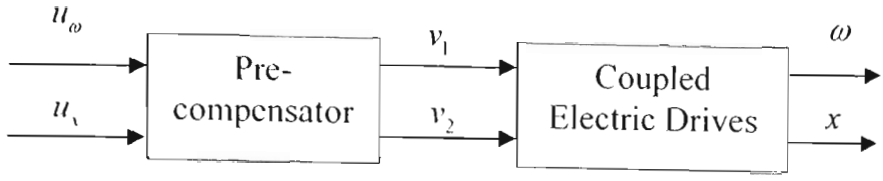
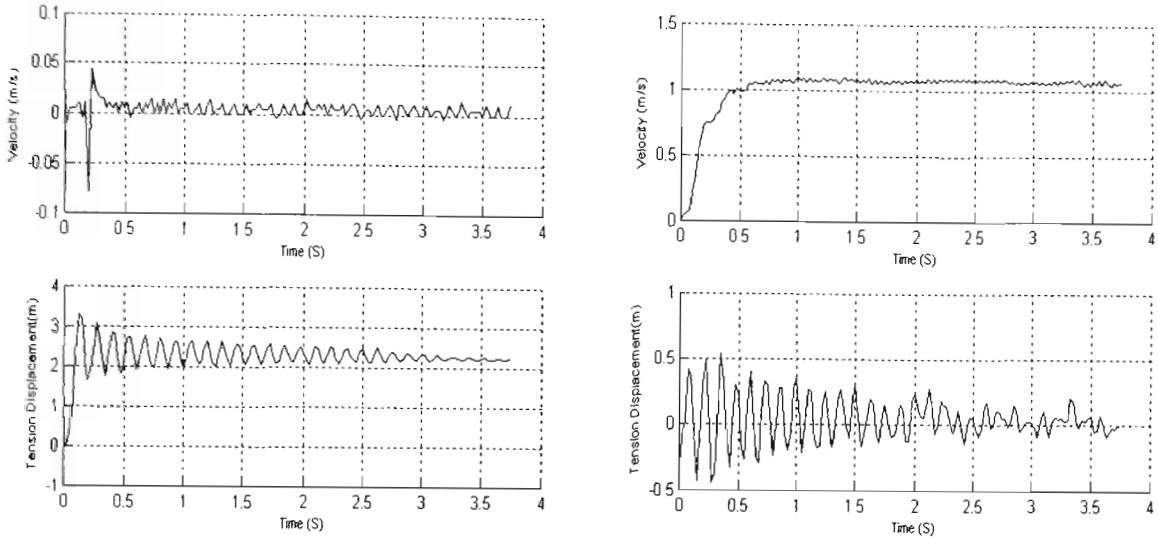


Figure 6.2: Diagram of the Coupled Electric Drives system with the pre-compensator.



(a) **(b)**

Figure 6.3: Step responses of the open-loop system with the control inputs (a) $u_\omega = 0$, $u_x = 3$ and (b) $u_\omega = 3$, $u_x = 0$.

In order to model the system, some step-response experiments are performed. Figure 6.3 shows the experiment results for two cases: u_ω=0, u_x=3 and u_ω=3, u_x=0. As shown in Figure 6.3 (a), although the velocity reference input u_ω is set to zero, the velocity output is different from zero. Similarly, the tension displacement is oscillated as in Figure 6.3 (b) when the tension reference input u_x equals zero. These results show that the coupling effects between two subsystems still exist. Consequently, the real Coupled Electric Drives system is actually a large-scale system with interconnections and disturbances. It is also observed from the experiment results that the velocity subsystem is a first-order system and the tension subsystem has a fourth-order dynamic equation. This is similar to the mathematical model of the coupled electric drives systems described in Wellstead (1979).

The system dynamics have a general form as:

$$\text{Speed subsystem}\begin{cases}\dot{\omega} = f_{\omega}(\omega) + b_{\omega}(\omega)u_{\omega} + z_{\omega}(\mathbf{X}) + d_{\omega}(\omega, t) \\ y_{\omega} = \omega\end{cases} \tag{6.2}$$

$$\text{Tension subsystem}\begin{cases}x^{(1)} = f_x(\mathbf{x}) + b_x(\mathbf{x})u_x + z_x(\mathbf{X}) + d_x(\mathbf{x}, t) \\ y_x = x\end{cases} \tag{6.3}$$

where $\mathbf{x} = \begin{bmatrix} x & \dot{x} & x^{(2)} & x^{(3)} \end{bmatrix}^T$ is the state vector of the tension subsystem, $\mathbf{X} = \begin{bmatrix} \omega & \mathbf{x}^T \end{bmatrix}^T$ the system state, $f_{\omega}(\omega)$, $f_x(\mathbf{x})$ the unknown continuous function, $b_{\omega}(\omega)$, $b_x(\mathbf{x})$ the unknown control gain function, z_x, z_{ω} the strength of interconnections from the other subsystems, d_x, d_{ω} the disturbances, u_{ω}, u_x the control inputs of the velocity and tension subsystems, respectively.

In the next section, some control strategies are proposed to solve the control problem of this class of uncertain interconnected systems.

6.3 Advanced Neural Controller Design

Consider a large-scale system which is composed of N interconnected subsystems $q_i; i = 1, ..., N$. Each subsystem q_i may be represented as (Da, 2000):

$$q_i\begin{cases} \dot{x}_{i1} = x_{i2} \\ \dot{x}_{i2} = x_{i3} \\ \quad \vdots \\ \dot{x}_{in_i} = f_i(\mathbf{x}_i) + g_i(\mathbf{x}_i)u_i(t) + z_i(\mathbf{X}) + d_i(\mathbf{x}_i, t) \\ y_i = x_{i1}\end{cases} \tag{6.4}$$

where $\mathbf{x}_i = \begin{bmatrix} x_{i1}, ..., x_{in_i} \end{bmatrix}^T \in R^{n_i}$ denotes the state vector of q_i, $u_i \in R$ the control input of q_i, y_i the system output, $f_i(\mathbf{x}_i)$ the unknown continuous function, $g_i(\mathbf{x}_i)$ the unknown control gain function, $z_i(\mathbf{X})$ the strength of interconnections from the other

subsystems, $d_i(\mathbf{x}_i,t)$ the uncertainties and disturbances of the subsystem q_i, $\mathbf{X}=\left[\mathbf{x}_1^T,....,\mathbf{x}_N^T\right]^T \in R^n, n=\sum_{i=1}^{N} n_i$ the state vector of the whole system.

Assumption 6.1: *Full state vectors of the system are measurable.*

Assumption 6.2: *The sign of* $g_i(\mathbf{x}_i)$ *is known and there exist some positive constants* $g_{i0}, g_{i1} > 0$ *such that* $g_{i0} \le |g_i(\mathbf{x}_i)| \le g_{i1};\ i=1,...,N$.

Without losing generality, $g_i(\mathbf{x}_i)$ *is assumed to be positive.*

Assumption 6.3: $|f_i(\mathbf{x}_i)| \le F_i(\mathbf{x}_i); i=1,...,N$, *where* F_i *is a non-negative function, and there exist some positive constants* B_{f_i} *such that* $\|\nabla f_i(\mathbf{x}_i)\| \le B_{f_i};\ i=1,...,N$.

Assumption 6.4: $|d_i(\mathbf{x}_i,t)| \le D_i(\mathbf{x}_i); i=1,...,N$, *where* D_i *is a non-negative function, and there exist some positive constants* B_{d_i}, B_{t_i} *such that*

$$\left\|\frac{\partial d_i(\mathbf{x}_i)}{\partial \mathbf{x}_i}\right\| \le B_{d_i};\ \left|\frac{\partial d_i(\mathbf{x}_i)}{\partial t}\right| \le B_{t_i};\quad i=1,...,N.$$

Assumption 6.5: *The interconnections are bounded by a* p^{th}*-order polynomial in states, that is, there exist non-negative numbers* ζ_{ij}^k *and* B_{z_i} *such that*

$$|z_i(\mathbf{X})| \le \sum_{k=0}^{p}\sum_{j=1}^{N}\zeta_{ij}^k\|\mathbf{x}_j\|^k;\ \|\nabla z_i(\mathbf{X})\| \le B_{z_i};\ i=1,...,N.$$

Assumption 6.6: *Assume that the reference* $y_{id}(t)=x_{id}(t)$ *has the property*

$$\lim_{t\to\infty} y_{id}(t) = y_{iss} \text{ and } \lim_{t\to\infty} y_{id}^{(j)}(t) = 0 \text{ for } 1 \le j \le n-1.$$

and the external disturbance approaches a constant limit d_{iss}, *that is,* $\lim_{t\to\infty} d_i(\mathbf{x}_i, t) = d_{iss}$. *For every* (y_{iss}, d_{iss}); $i = 1,...,N$, *there exist a unique equilibrium point* $\mathbf{x}_{idss} = [y_{iss}, 0, ..., 0]^T$. $\mathbf{X}_{ss} = [\mathbf{x}_{1ss}^T, ..., \mathbf{x}_{Nss}^T]^T$ *and unique control signal* $u_{iss} = u_{iss}(y_{iss}, d_{iss})$; $i = 1,..., N$ *such that* $0 = f_i(\mathbf{x}_{idss}) + g_i(\mathbf{x}_{idss})u_{iss} + z_i(\mathbf{X}_{ss}) + d_{iss}$ *and* $y = y_{iss}$; $i = 1,..., N$.

Assumption 6.6 means that there exists an equilibrium point $\mathbf{e}_i = 0$; $i = 1,..., N$ and steady-state control input which maintain the equilibrium.

The research objective is to design a decentralised robust control scheme for the system (6.4) so that the state $\mathbf{x}_i = [x_{i1}, ..., x_{im_i}]^T$ can track the desired state $\mathbf{x}_{id} = [x_{id}, ..., x_{id}^{(n_i-1)}]^T$, and the tracking error $\mathbf{e}_i = \mathbf{x}_i - \mathbf{x}_{id} = [e_{i1}, ..., e_{in_i}]^T \in R^{n_i}$ can converge to zero.

6.3.1 Sliding-mode-based state feedback controller design

For the subsystem q_i, define a sliding function $s_i(\mathbf{e}_i)$ in the state space R^{n_i}

$$s_i(\mathbf{e}_i) = \mathbf{H}_i \mathbf{e}_i = h_{i1}e_{i1} + \cdots + h_{i(n_i-1)}e_{i(n_i-1)} + e_{in_i} \tag{6.5}$$

where the column constant vector $\mathbf{H}_i = [h_{i1}, ..., h_{i(n_i-1)}, 1]$ is chosen so that the polynomial $s^{n_i-1} + h_{i(n_i-1)}s^{n_i-2} + \cdots + h_{i1}$ is Hurwitz. Thus tracking error $\mathbf{e}_i$ will converge to zero if $s_i(\mathbf{e}_i) = 0$; $i = 1,..., N$.

By choosing a Lyapunov candidate $V_{i1} = \frac{1}{2}s_i^2$, if the control law u_i is designed so that $\dot{V}_i < 0; s_i \neq 0$, then s_i will asymptotically converge to zero.

The control law u_i can be designed as:

$$u_i = -K_i \rho\left(\frac{s_i}{\phi_i}\right); \phi_i > 0; i = 1,..., N \tag{6.6}$$

with $\rho\left(\frac{s_i}{\phi_i}\right)$ is a saturated integral-proportional function, and is defined as:

$$\rho\left(\frac{s_i}{\phi_i}\right) = \begin{cases} 1 & \text{if} \quad s_i > \phi_i \\ \frac{s_i}{\phi_i} + \frac{1}{T_i}\int_{t_0}^{t} \frac{s_i}{\phi_i} dt & \text{if} \quad |s_i| \le \phi_i \\ -1 & \text{if} \quad s_i < -\phi_i \end{cases} \tag{6.7}$$

where T_i is the integral time constant and t_0 is the initial time when system states enter the boundary layer $B_i(\mathbf{x}_i) = \{\mathbf{x}_i \| s_i | \le \phi_i\}$.

The next lemma shows that all system state trajectories will be driven toward the sliding surface $s_i = 0$.

Lemma 6.1: *For the system* (6.4) *with control law* (6.6), *if Assumptions 6.1–6.5 are satisfied and* K_i *is chosen as*

$$K_i = \frac{1}{g_{i0}}\left[\left|\sum_{j=1}^{n_i-1} h_{ij} e_{i(j+1)} - x_{id}^{(n_i)}\right| + F_i(\mathbf{x}_i) + \sum_{k=0}^{p}\sum_{j=1}^{N} \zeta_{ij}^{k} \|\mathbf{x}_j\|^k + D_i(\mathbf{x}_i) + \eta_i\right], \tag{6.8}$$
$$\eta_i > 0; \quad i = 1,..., N$$

then s_i *is forced to zero in a finite time.*

Proof: Consider again the Lyapunov function $V_{i1} = \frac{1}{2} s_i^2$.

When $|s_i| > \phi_i$, $\text{sat}\left(\frac{s_i}{\phi_i}\right) = \text{sgn}(s_i)$, and the derivatives of s_i and V_{i1} are obtained as:

$$\dot{s}_i = \sum_{j=1}^{n_i-1} h_{ij} e_{i(j+1)} + \dot{e}_{im_i}$$
$$= \sum_{j=1}^{n_i-1} h_{ij} e_{i(j+1)} + f_i(\mathbf{x}_i) - g_i(\mathbf{x}_i) K_i \operatorname{sgn}(s_i) + z_i(\mathbf{X}) + d_i(\mathbf{x}_i) - x_{id}^{(n_i)}$$

$$\dot{V}_{i1} = s_i \dot{s}_i = s_i \left[\sum_{j=1}^{n_i-1} h_{ij} e_{i(j+1)} + f_i(\mathbf{x}_i) + z_i(\mathbf{X}) + d_i(\mathbf{x}_i) - x_{id}^{(n_i)} \right] - g_i(\mathbf{x}_i) K_i |s_i|.$$

If Assumptions 6.2–6.5 are satisfied and

$$K_i = \frac{1}{g_{i0}} \left[\left| \sum_{j=1}^{n_i-1} h_{ij} e_{i(j+1)} - x_{id}^{(n_i)} \right| + F_i(\mathbf{x}_i) + \sum_{k=0}^{p} \sum_{j=1}^{N} \zeta_{ij}^{k} \left\| \mathbf{x}_j \right\|^k + D_i(\mathbf{x}_i) + \eta_i \right], \ \eta_i > 0,$$

then $\dot{V}_{i1} \le 0$. and all the subsystem state $\mathbf{x}_i$ starting from any initial position will be forced into a boundary layer $B_i(\mathbf{x}_i) = \{\mathbf{x}_i \,|\, |s_i| \le \phi_i\}$ in a finite time.

When $|s_i| \le \phi_i$, there are three possible cases for any value T_i.

- If $\dot{\rho} = \dot{s}_i + \frac{1}{T_i} s_i = 0$, s_i will tend to zero with convergence rate $\frac{1}{T_i}$.

- If $\dot{\rho} = \begin{cases} \dot{s}_i + \frac{1}{T_i} s_i < 0 \text{ for all } s_i > 0 \\ \dot{s}_i + \frac{1}{T_i} s_i > 0 \text{ for all } s_i < 0 \end{cases}$, it leads to $\dot{V}_{i1} < -\frac{2}{T_i} V_{i1}$. So the Lyapunov stability condition is satisfied and $s_i = 0$ is the asymptotically stable point of the system.

- If $\dot{\rho} = \begin{cases} \dot{s}_i + \frac{1}{T_i} s_i > 0 \text{ for all } s_i > 0 \\ \dot{s}_i + \frac{1}{T_i} s_i < 0 \text{ for all } s_i < 0 \end{cases}$, this means that $\begin{bmatrix} \rho \text{ increases on all } s_i > 0 \\ \rho \text{ decreases on all } s_i > 0 \end{bmatrix}$.

As a result, the inequalities below can be obtained after a finite time (Salas & Hill, 1990):

$$\rho = \begin{cases} s_i + \dfrac{1}{T_i}\int_{t_0}^{t} s_i dt \geq \phi_i & \text{for } s_i > 0 \\ s_i + \dfrac{1}{T_i}\int_{t_0}^{t} s_i dt \leq -\phi_i & \text{for } s_i < 0 \end{cases}.$$

The control law (6.6) with K_i chosen in (6.8) consequently causes the condition $\dot{V}_{i1} < 0;\ s_i \neq 0$ to be satisfied, and the system state will go to the sliding surface $s_i = 0$ in a finite time. ■

Proposition 6.1: *For the system* (6.4) *with control law* (6.6), *if Assumptions 6.1–6.6 are satisfied and* K_i *is chosen as in* (6.8), *then the equilibrium point* $\mathbf{e}_i = 0;\quad i = 1,...,N$ *of the closed-loop system is asymptotically stable.*

Proof:

- When $|s_i| > \phi_i$, $\text{sat}\left(\dfrac{s_i}{\phi_i}\right) = \text{sgn}(s_i)$, as in the proof of Lemma 6.1, the condition $\dot{V}_{i1} < 0$ is satisfied, and all the subsystem state $\mathbf{x}_i$ starting from any initial position will be forced into a boundary layer $B_i(\mathbf{x}_i) = \{\mathbf{x}_i \,\|s_i| \leq \phi_i\}$ in a finite time.

- When $|s_i| \leq \phi_i$, define $\varsigma_i = \left[e_i,...,e_{i(n_i-1)}\right]^T$,

$$\mathbf{F}_i = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \\ -h_{i1} & -h_{i2} & \cdots & -h_{i(n_i-1)} \end{bmatrix},\quad \mathbf{G}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

The system (6.5) can be represented by

$$\dot{\varsigma}_i = \mathbf{F}_i \varsigma_i + \mathbf{G}_i s_i \tag{6.9}$$

With the vector $\mathbf{H}_i$ chosen, the matrix $\mathbf{F}_i$ is stable. Thus there exists a symmetric positive definite matrix $\mathbf{P}_i \in \mathbf{R}^{(n_i-1)\cdot(n_i-1)}$ satisfying the Lyapunov equation

$$\mathbf{P}_i \mathbf{F}_i + \mathbf{F}_i^T \mathbf{P}_i = -\mathbf{I}_i \tag{6.10}$$

where $\mathbf{I}_i \in \mathbf{R}^{(n_i-1)\times(n_i-1)}$ is a unit matrix.

Choose a Lyapunov function $V_{i2}(\varsigma_i) = \varsigma_i^T \mathbf{P}_i \varsigma_i$,

if $\forall \|\varsigma_i\| \geq 2\|\mathbf{P}_i \mathbf{G}_i\| |s_i|$, then $\dot{V}_{i2} = -\varsigma_i^T \varsigma_i + 2\varsigma_i^T \mathbf{P}_i \mathbf{G}_i s_i \leq -\|\varsigma_i\|^2 \leq 0$, and all subsystem errors ς_i will converge to zero.

When $\|\varsigma_i\| < 2\|\mathbf{P}_i \mathbf{G}_i\| |s_i|$, because $|s_i| \leq \phi_i$, all the subsystem errors $\varsigma_i = \left[e_i, ..., e_{i(n_i-1)}\right]^T$ will be forced into the region:

$$\Omega_i = \left\{\|\varsigma_i\| \leq 2\|\mathbf{P}_i \mathbf{G}_i\| |\phi_i|\right\} \cap \left\{|s_i| \leq \phi_i\right\}.$$

From Lemma 6.1, s_i is forced to zero, thus $\varsigma_i = \left[e_i, ..., e_{i(n_i-1)}\right]^T$ tends to zero, as described in the inequality $\|\varsigma_i\| \leq 2\|\mathbf{P}_i \mathbf{G}_i\| |s_i|$. From the dynamic equation (6.4), the final error state, $e_{in_i} = \dot{x}_{i(n_i-1)} - x_{id}^{(n_i-1)} = \dot{e}_{i(n_i-1)}$, consequently tends to zero.

When $\mathbf{e}_i = 0$ and $s_i = \mathbf{H}_i \mathbf{e}_i = 0$, $\dot{V}_{i2} = 0$. Therefore, $\mathbf{e}_i = 0$ is the largest invariant set inside the region Ω_i.

The application of the invariance principle (Khalil, 2002) shows that the equilibrium point $\mathbf{e}_i = 0; \quad i = 1, ..., N$ is asymptotically stable. ■

Remark 6.3: The proposed sliding-mode-based feedback controller can maintain the system stability, attenuates the tracking errors and completely eliminates the chattering of control forces.

However, the values of K_i, as chosen in (6.8), are often large and distort system performance. Using the learning capability of neural networks, the neural controller's parameters initialised to small values can be on-line updated, thus the above disadvantage will be avoided. Additionally, motivated from the saturated integral-proportional function, a new sliding function for training the neural controller is designed in the next section.

6.3.2 Neural network controller design

Equation (6.4) can be rewritten as

$$\underset{i=1,\ldots,N}{q_i} \begin{cases} \dot{\mathbf{x}}_i = \mathbf{A}_i\mathbf{x}_i + \mathbf{B}_i\left[f_i(\mathbf{x}_i) + g_i(\mathbf{x}_i)u_i(t) + z_i(\mathbf{X}) + d_i(\mathbf{x}_i,t)\right] \\ y_i = \mathbf{C}_i\mathbf{x}_i \end{cases} \tag{6.11}$$

where $\mathbf{A}_i = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix}$; $\mathbf{B}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$; $\mathbf{C}_i = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$.

Design the sliding surfaces

First define new sliding functions as:

$$\begin{aligned} &\overline{s}_i = \delta_i s_i + \dot{s}_i; \quad \delta_i > 0; \quad \delta_i s_i(0) = -\dot{s}_i(0) \\ &i = 1,\ldots,N \end{aligned} \tag{6.12}$$

with $\delta_i > 0$, and s_i is defined as in (6.5):

$$s_i(\mathbf{e}_i) = \mathbf{H}_i\mathbf{e}_i = h_{i1}e_{i1} + \cdots + h_{i(n_i-1)}e_{i(n_i-1)} + e_{in_i}$$

where the row constant vector $\mathbf{H}_i = \left[h_{i1}, \ldots, h_{i(n_i-1)}, 1 \right]$ is chosen so that the polynomial $s^{n_i-1} + h_{i(n_i-1)} s^{n_i-2} + \cdots + h_{i1}$ is Hurwitz.

Remark 6.4: By defining the sliding function with the specific initial condition (6.12), the sliding function attains zero at the initial instance, $\bar{s}_i(0) = 0$. If the control signal can keep system states on the surfaces $\bar{s}_i = 0$, then the closed-loop system behaviour is characterised by the sliding surfaces' dynamics.

Equation (6.11), added (6.5) and (6.12) leads to:

$$\bar{s}_i = \delta_i \mathbf{H}_i \mathbf{e}_i + \mathbf{H}_i \dot{\mathbf{e}}_i \tag{6.13}$$

$$\dot{\mathbf{e}}_i = \bar{\mathbf{F}}_i \mathbf{e}_i + \bar{\mathbf{G}}_i \bar{s}_i \tag{6.14}$$

where the matrices $\bar{\mathbf{F}}_i \in \mathbf{R}^{n_i \times n_i}$ and $\bar{\mathbf{G}}_i \in \mathbf{R}^{n_i}$ are in the canonical form:

$$\bar{\mathbf{F}}_i = \begin{bmatrix} 0 & 1 & . & . \\ . & 0 & 1 & . \\ . & . & . & . \\ -\delta h_{i1} & -\delta h_{i2} - h_{i1} & . & -\delta - h_{i(n_i-1)} \end{bmatrix}; \quad \bar{\mathbf{G}}_i = \begin{bmatrix} 0 \\ . \\ 0 \\ 1 \end{bmatrix}. \tag{6.15}$$

Remark 6.5: With the chosen vector $\mathbf{H}_i$ and $\delta_i > 0$, the matrix $\bar{\mathbf{F}}_i$ in (6.15) is stable, thus there exists a symmetric positive definite matrix $\bar{\mathbf{P}}_i \in \mathbf{R}^{n_i \times n_i}$ satisfying the Lyapunov equation

$$\bar{\mathbf{P}}_i \bar{\mathbf{F}}_i + \bar{\mathbf{F}}_i^T \bar{\mathbf{P}}_i = -\mathbf{I}_i \tag{6.16}$$

where $\mathbf{I}_i \in \mathbf{R}^{n_i \times n_i}$ is a unit matrix.

Lemma 6.2: *For* the *dynamic system* (6.14), *the following relationship is always obtained:*

$$\|\mathbf{e}_i\| \le 2\|\bar{\mathbf{P}}_i\bar{\mathbf{G}}_i\||\bar{s}_i| \tag{6.17}$$

Proof: Suppose that $\|\mathbf{e}_i\| > 2\|\bar{\mathbf{P}}_i\bar{\mathbf{G}}_i\||\bar{s}_i|$

Choose a Lyapunov function $V_{i3} = \mathbf{e}_i^T\bar{\mathbf{P}}_i\mathbf{e}_i$.

The time derivative of V_{i3} is obtained from (6.14) and (6.16) as:

$$\dot{V}_{i3} = \dot{\mathbf{e}}_i^T\bar{\mathbf{P}}_i\mathbf{e}_i + \mathbf{e}_i^T\bar{\mathbf{P}}_i\dot{\mathbf{e}}_i = -\mathbf{e}_i^T\mathbf{e}_i + 2\mathbf{e}_i^T\bar{\mathbf{P}}_i\bar{\mathbf{G}}_i\bar{s}_i$$

If $\|\mathbf{e}_i\| > 2\|\bar{\mathbf{P}}_i\bar{\mathbf{G}}_i\||\bar{s}_i|$, then $\dot{V}_{i3} < 0$ and all the tracking errors $\mathbf{e}_i$ will converge to zero.

Consequently, the inequality (6.17) is obtained. ■

Remark 6.6: Lemma 6.2 means that tracking errors $\mathbf{e}_i$ asymptotically converge to zero if $s_i = 0$. This suggests that $\bar{s}_i$ can be used as a supervisor training signal for the neural network controller, which is designed in the next stage.

Design the decentralised neural network controller

The whole system has a parallel structure; each subsystem has one feedforward NN controller, as shown in Figure 6.4. For each subsystem q_i, each feedforward neural network consists of (n_i+1) input nodes, (m_i+1) hidden nodes, and one output node. The hyperbolic tangent sigmoid, $f_h(v) = \dfrac{1-e^{-v}}{1+e^{-v}}$, is used as the activation function for the hidden nodes and the linear function, $f_o(v) = v$, is used as the activation function for the output node. For the i^{th} subsystem, a superscript i is appended to the name of neural network's weights.

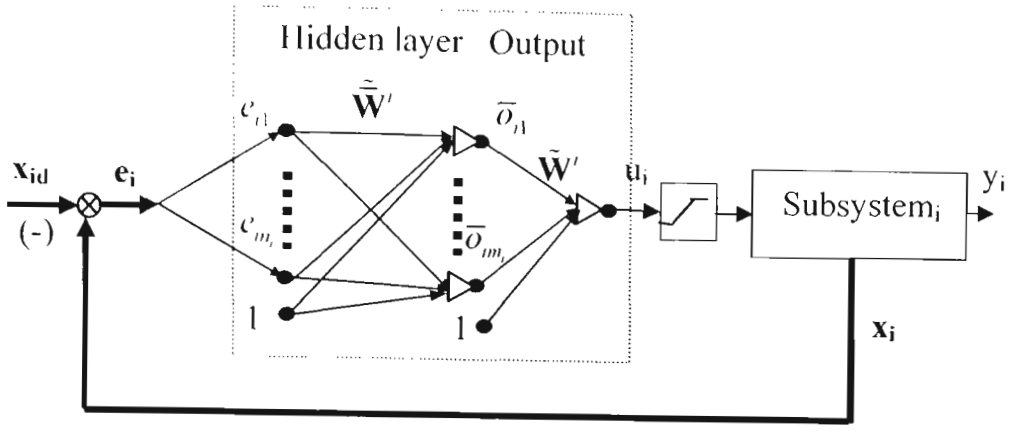


Figure 6.4: Structure of the i^{th} neural control subsystem.

The input vector $\mathbf{e}_i = \left[e_{i1},...,e_{im_i}\right]^T$ of the neural network is also the vector of the tracking error $\mathbf{e}_i$ of the system (6.4). A constant input of 1, affecting the bias, is assigned to the vector of augment inputs $\tilde{\mathbf{e}}_i = \left[e_{i1},...,e_{im_i},1\right]^T$. Let $\bar{\mathbf{W}}^i = \begin{bmatrix} \bar{W}^i_{11} & \cdots & \bar{W}^i_{1n_i} \\ \vdots & \ddots & \vdots \\ \bar{W}^i_{m1} & \cdots & \bar{W}^i_{mn_i} \end{bmatrix}$ be the weight matrix between the input layer and the hidden layer, and $\tilde{\bar{\mathbf{W}}}^i = \begin{bmatrix} \bar{W}^i_{11} & \cdots & \bar{W}^i_{1(n_i+1)} \\ \vdots & \ddots & \vdots \\ \bar{W}^i_{m1} & \cdots & \bar{W}^i_{m(n_i+1)} \end{bmatrix}$ be the matrix of augmented weights by including the bias weights components $\bar{W}^i_{j(n_i+1)}; j = 1,\ldots,m$.

The vector $\bar{\mathbf{o}}_i = \left[\bar{o}_{i1},...,\bar{o}_{im_i}\right]^T$; $\bar{o}_{kj} = f_{ij}\left(\sum_{k=1}^{n_i} \bar{W}^i_{jk} x_k + \bar{W}^i_{j(n_i+1)}\right); j = 1,...,m_i$ is the vector of the output signals of the neurons in the hidden layer. A constant input of 1, affecting the bias, is assigned to the vector of augment outputs $\tilde{\bar{\mathbf{o}}}_i = \left[\bar{o}_{i1},...,\bar{o}_{im_i},1\right]^T$. Let $\mathbf{W}^i = \left[W^i_1 \quad \cdots \quad W^i_m\right]$ be the weight vector between the hidden layer and the output layer, and $\tilde{\mathbf{W}}^i = \left[W^i_1 \quad \cdots \quad W^i_{m+1}\right]$ be the vector of augmented weights by including the bias weights component W^i_{m+1}.

Assumption 6.7: *Due to the physical constraints, the magnitudes of* $\tilde{\mathbf{W}}$ *and* $\tilde{\bar{\mathbf{W}}}$ *are assumed to be bounded by*:

$$\left\|\mathbf{W}^{i}\right\| \le B_{W^{i}}, \qquad \left\|\bar{\mathbf{W}}^{i}\right\| \le B_{\bar{W}^{i}}, \qquad \left\|\tilde{\mathbf{W}}^{i}\right\| \le B_{\tilde{W}^{i}}.$$

The output of the neural network in the i^{th} subsystem is also the control signal u_i as:

$$u_i(\mathbf{e}_i) = \tilde{\mathbf{W}}^{i}\tilde{\bar{\mathbf{o}}}_i = \sum_{j=1}^{m_i} W^{i}{}_{j}\bar{o}_{ij} + W^{i}{}_{m_i+1} = \sum_{j=1}^{m_i} W^{i}{}_{j} f_h\left(\sum_{k=1}^{n_i} \bar{W}^{i}{}_{jk} e_{ik} + \bar{W}^{i}{}_{j(m_i+1)}\right) + W^{i}{}_{m_i+1} \quad (6.18)$$

Substituting Equations (6.5), (6.11) into (6.12) leads to:

$$\begin{aligned}\bar{s}_i &= \delta_i \mathbf{H}_i \mathbf{e}_i + \mathbf{H}_i\left\{\mathbf{A}_i\mathbf{x}_i + \mathbf{B}_i\left[f_i(\mathbf{x}_i) + g_i(\mathbf{x}_i)u_i(t) + z_i(\mathbf{X}) + d_i(\mathbf{x}_i,t)\right] - \dot{\mathbf{x}}_{id}\right\}\\ \bar{s}_i &= \left(\delta_i\mathbf{H}_i + \mathbf{H}_i\mathbf{A}_i\right)\mathbf{e}_i + g_i(\mathbf{x}_i)u_i + \\ &\quad + \mathbf{H}_i\left\{\mathbf{A}_i\mathbf{x}_{id} + \mathbf{B}_i\left[f_i(\mathbf{x}_i) + z_i(\mathbf{X}) + d_i(\mathbf{x}_i,t)\right] - \dot{\mathbf{x}}_{id}\right\}.\end{aligned} \quad (6.19)$$

Therefore,
$$\frac{\partial \bar{s}_i}{\partial u_i} = g_i. \quad (6.20)$$

The cost function for training the i^{th} network is defined as:

$$V_{i4} = \frac{1}{2}\left(\bar{s}_i\right)^2 \quad (6.21)$$

The proposed online learning algorithm of the weight vector $\tilde{\mathbf{W}}^{i}$ and the weight matrix $\tilde{\bar{\mathbf{W}}}^{i}$ is presented as follows:

$$\begin{aligned}\dot{\tilde{\mathbf{W}}}^{i} &= -\frac{\left[\eta_i\bar{s}_i + \varepsilon_i\,\mathrm{sgn}\left(\bar{s}_i\right)\right]}{\left\|\tilde{\bar{\mathbf{o}}}_i^{T}\right\|^2 + \left\|\mathbf{F}'_{hi}\,\mathbf{W}^{iT}\tilde{\mathbf{x}}_i^{T}\right\|^2}\,\tilde{\bar{\mathbf{o}}}_i^{T}\\ \dot{\tilde{\bar{\mathbf{W}}}}^{i} &= -\frac{\left[\eta_i\bar{s}_i + \varepsilon_i\,\mathrm{sgn}\left(\bar{s}_i\right)\right]}{\left\|\tilde{\bar{\mathbf{o}}}_i^{T}\right\|^2 + \left\|\mathbf{F}'_{hi}\,\mathbf{W}^{iT}\tilde{\mathbf{x}}_i^{T}\right\|^2}\,\mathbf{F}'_{hi}\,\mathbf{W}^{iT}\tilde{\mathbf{x}}_i^{T}\end{aligned} \quad (6.22)$$

where $\mathbf{F}'_{lm} = \begin{bmatrix} f'_{l1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f'_{lm_l} \end{bmatrix}$, f'_{lj}, $\left(j = 1,\ldots,m_l\right)$ is the derivative of $\bar{o}_{lj}$ with regard to $\left(\sum_{k=1}^{n_l} \bar{W}'_{jk} x_k + \bar{W}'_{j(n_l+1)}\right)$, η_l, ε_l are some positive scalars to be chosen in the next theorem.

The learning algorithm (6.22) is mentioned in Chapter 4, and is named RMBP.

Theorem 6.1: *Consider the closed-loop system consisting of* (6.4); *each subsystem* q_l *uses the controller (6.18) and the training algorithm (6.22), if Assumptions 6.1–6.5 and 6.7 are satisfied and* η_i, ε_i *is chosen as:*

$$\eta_i \geq \left(\frac{\left\|\delta_i \mathbf{H}_i + \bar{\mathbf{H}}_l\right\|}{g_0} + 0.5 B_W B_{\bar{W}} + B_{f_l} + B_{d_l}\right)\left(2\delta_i \left\|\bar{\mathbf{P}}_i \bar{\mathbf{G}}_l\right\| + \frac{1}{\left\|\mathbf{H}_l\right\|}\right), \tag{6.23}$$

$$\varepsilon_i \geq \left(\frac{\left|\bar{\mathbf{H}}_l \dot{\mathbf{x}}_{ld} - \mathbf{H}_i \ddot{\mathbf{x}}_{id}\right|}{g_0} + \left(B_{f_l} + B_{d_l}\right)\left\|\dot{\mathbf{x}}_{id}\right\| + B_{t_l} + B_{z_l}\left\|\dot{\mathbf{X}}\right\|\right), \tag{6.24}$$

where $\bar{\mathbf{H}}_l = \begin{bmatrix} 0 & h_1 & \cdots & h_{n_l-1} \end{bmatrix}$, *then the closed-loop system is asymptotically stable.*

Proof: For each subsystem q_l, the derivative of the function (6.21) is obtained as:

$$\dot{V}_{i4} = \sum_{j=1}^{m_l+1} \frac{\partial V_{l4}}{\partial W_j^i} \dot{W}_j^l + \sum_{j=1}^{m_l} \sum_{k=1}^{n_l+1} \frac{\partial V_{l4}}{\partial \bar{W}'_{jk}} \dot{\bar{W}}'_{jk} + \sum_{j=1}^{n_l} \frac{\partial V_{i4}}{\partial e_{ij}} \dot{e}_{ij}$$
$$+ \sum_{j=1}^{n_l} \frac{\partial V_{l4}}{\partial x_{idj}} \dot{x}_{idj} + \sum_{j=1}^{n_l} \frac{\partial V_{l4}}{\partial \dot{x}_{idj}} \ddot{x}_{idj} + \sum_{j=1}^{\aleph} \frac{\partial V_{l4}}{\partial x_j} \dot{x}_j + \frac{\partial V_{l4}}{\partial f_i\left(\mathbf{x}_l\right)} + \frac{\partial V_{i4}}{\partial d_i\left(\mathbf{x}_i, t\right)}$$

where x_j is the j^{th} components of vector $\mathbf{X} = \left[\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T\right]^T \in R^n$, $\aleph = \sum_{i=1}^{N} n_i$.

It is obtained from (6.18), (6.20) and (6.21) that

$$\frac{\partial V_{i4}}{\partial W_j^i} = \frac{\partial V_{i4}}{\partial \bar{s}_i}\frac{\partial \bar{s}_i}{\partial u_i}\frac{\partial u_i}{\partial W_j^i} = \bar{s}_i g_i \bar{o}_{ij}$$

$$\frac{\partial V_{i4}}{\partial \bar{W}_{jk}^i} = \frac{\partial V_{i4}}{\partial \bar{s}_i}\frac{\partial \bar{s}_i}{\partial u_i}\frac{\partial u_i}{\partial \bar{W}_{jk}^i} = \bar{s}_i g_i W_j^i f_{hi}' x_{jk}$$

$$\frac{\partial V_{i4}}{\partial e_{ij}} = \bar{s}_i \left(\delta_i h_{ij} + h_{i(j-1)} + g_i \sum_{k=1}^{m_i} W_k^i f_{hk}' W_{kj}^i \right) \text{ with } h_{i(-1)} = 0 .$$

$$\frac{\partial V_{i4}}{\partial x_{idj}} = \bar{s}_i h_{i(j-1)} ; \ \frac{\partial V_{i4}}{\partial \dot{x}_{idj}} = -\bar{s}_i h_{ij} ; \ \frac{\partial V_{i4}}{\partial x_j} = \bar{s}_i g_i \left[z_i \right]'_{x_j}$$

$$\frac{\partial V_{i4}}{\partial f_i(\mathbf{x}_i)} = \frac{\partial V_{i4}}{\partial \bar{s}_i}\frac{\partial \bar{s}_i}{\partial f_i(\mathbf{x}_i)} = \bar{s}_i g_i \nabla f_i(\mathbf{x}_i)\dot{\mathbf{x}}_i$$

$$\frac{\partial V_{i4}}{\partial d_i(\mathbf{x}_i,t)} = \frac{\partial V_{i4}}{\partial \bar{s}_i}\frac{\partial \bar{s}_i}{\partial d_i(\mathbf{x}_i,t)} = \bar{s}_i g_i \left[\frac{\partial d_i(\mathbf{x}_i,t)}{\partial \mathbf{x}_i} \right] \dot{\mathbf{x}}_i + \bar{s}_i g_i \frac{\partial d_i(\mathbf{x}_i,t)}{\partial t}$$

Therefore,

$$\begin{aligned}\dot{V}_{i4} = & -\bar{s}_i g_i \left[\eta_i \bar{s}_i + \varepsilon_i \operatorname{sgn}(\bar{s}_i) \right] + \bar{s}_i \left(\delta_i \mathbf{H}_i + \bar{\mathbf{H}}_i + g_i \mathbf{W}^i \mathbf{F}_{hi}' \bar{\mathbf{W}}^i \right) \dot{\mathbf{e}}_i \\ & + \bar{s}_i \left(\bar{\mathbf{H}}_i \dot{\mathbf{x}}_{id} - \mathbf{H}_i \ddot{\mathbf{x}}_{id} \right) + \bar{s}_i g_i \left[\nabla f_i(\mathbf{x}_i) + \frac{\partial d_i(\mathbf{x}_i,t)}{\partial \mathbf{x}_i} \right] \dot{\mathbf{x}}_i + \bar{s}_i g_i \frac{\partial d_i(\mathbf{x}_i,t)}{\partial t} + \bar{s}_i g_i \nabla z_i(\mathbf{X}) \dot{\mathbf{X}}\end{aligned}$$

$$\begin{aligned}\dot{V}_{i4} = & -\bar{s}_i g_i \left[\eta_i \bar{s}_i + \varepsilon_i \operatorname{sgn}(\bar{s}_i) \right] + \bar{s}_i \left\{ \delta_i \mathbf{H}_i + \bar{\mathbf{H}}_i + g_i \mathbf{W}^i \mathbf{F}_{hi}' \bar{\mathbf{W}}^i + g_i \left[\nabla f_i(\mathbf{x}_i) + \frac{\partial d_i(\mathbf{x}_i,t)}{\partial \mathbf{x}_i} \right] \right\} \dot{\mathbf{e}}_i \\ & + \bar{s}_i \left\{ \bar{\mathbf{H}}_i \dot{\mathbf{x}}_{id} - \mathbf{H}_i \ddot{\mathbf{x}}_{id} + g_i \left[\nabla f_i(\mathbf{x}_i) + \frac{\partial d_i(\mathbf{x}_i,t)}{\partial \mathbf{x}_i} \right] \dot{\mathbf{x}}_{id} \right\} + \bar{s}_i g_i \frac{\partial d_i(\mathbf{x}_i,t)}{\partial t} + \bar{s}_i g_i \nabla z_i(\mathbf{X}) \dot{\mathbf{X}}\end{aligned}$$

$$\begin{aligned}\dot{V}_{i4} \le & -\eta_i g_i |\bar{s}_i|^2 - \varepsilon_i g_i |\bar{s}_i| + |\bar{s}_i| \left\| \delta_i \mathbf{H}_i + \bar{\mathbf{H}}_i + g_i \mathbf{W}^i \mathbf{F}_{hi}' \bar{\mathbf{W}}^i + g_i \left[\nabla f_i(\mathbf{x}_i) + \frac{\partial d_i(\mathbf{x}_i,t)}{\partial \mathbf{x}_i} \right] \right\| \|\dot{\mathbf{e}}_i\| \\ & + |\bar{s}_i| \left(\left| \bar{\mathbf{H}}_i \dot{\mathbf{x}}_{id} - \mathbf{H}_i \ddot{\mathbf{x}}_{id} + g_i \left[\nabla f_i(\mathbf{x}_i) + \frac{\partial d_i(\mathbf{x}_i,t)}{\partial \mathbf{x}_i} \right] \dot{\mathbf{x}}_{id} \right| + g_i \left| \frac{\partial d_i(\mathbf{x}_i,t)}{\partial t} \right| + g_i \|\nabla z_i(\mathbf{X})\| \|\dot{\mathbf{X}}\| \right)\end{aligned} \tag{6.25}$$

where $\bar{\mathbf{H}}_i = \mathbf{H}_i \mathbf{A}_i = \begin{bmatrix} 0 & h_1 & \cdots & h_{n_i-1} \end{bmatrix}$.

Multiplying the two sides of Equation (6.13) for $\frac{\mathbf{H}_i^T}{\|\mathbf{H}_i\|^2}$ yields another dynamics equation:

$$\dot{\mathbf{e}}_i = -\delta_i \mathbf{e}_i + \frac{\mathbf{H}_i^T}{\|\mathbf{H}_i\|^2} \bar{s}_i \tag{6.26}$$

Equation (6.26), combined with the inequality (6.17) yields:

$$\|\dot{\mathbf{e}}_i\| \le \left(2\delta_i \|\bar{\mathbf{P}}_i \bar{\mathbf{G}}_i\| + \frac{1}{\|\mathbf{H}_i\|} \right) |\bar{s}_i| . \tag{6.27}$$

The inequality (6.27) added to (6.25) yields:

$$\begin{aligned} \dot{V}_{i2} \le & -\left[\eta_i - \left\| \frac{\delta_i \mathbf{H}_i + \bar{\mathbf{H}}_i}{g_i} + \mathbf{W}' \mathbf{F}_{hi}' \bar{\mathbf{W}}' + \nabla f_i(\mathbf{x}_i) + \frac{\partial d_i(\mathbf{x}_i, t)}{\partial \mathbf{x}_i} \right\| \left(2\delta_i \|\bar{\mathbf{P}}_i \bar{\mathbf{G}}_i\| + \frac{1}{\|\mathbf{H}_i\|} \right) \right] g_i |\bar{s}_i|^2 \\ & -\left[\varepsilon_i - \left(\left| \frac{\bar{\mathbf{H}}_i \dot{\mathbf{x}}_{id} - \mathbf{H}_i \ddot{\mathbf{x}}_{id}}{g_i} \right| + \left\| \nabla f_i(\mathbf{x}_i) + \frac{\partial d_i(\mathbf{x}_i, t)}{\partial \mathbf{x}_i} \right\| \|\dot{\mathbf{x}}_{id}\| + \left| \frac{\partial d_i(\mathbf{x}_i, t)}{\partial t} \right| + \|\nabla z_i(\mathbf{X})\| \|\dot{\mathbf{X}}\| \right) \right] g_i |\bar{s}_i| \end{aligned} \tag{6.28}$$

Assumption 6.2–6.5 and 6.7 and the inequality $f_{hi}' = \frac{1}{2}\left(1 - f_{hi}^2\right) \le 0.5, \quad i = 1, \ldots, m_i$ leads to:

$$\begin{aligned} \dot{V}_{i2} \le & -\left[\eta_i - \left(\frac{\|\delta_i \mathbf{H}_i + \bar{\mathbf{H}}_i\|}{g_0} + 0.5 B_W B_{\bar{W}} + B_{f_i} + B_{d_i} \right) \left(2\delta_i \|\bar{\mathbf{P}}_i \bar{\mathbf{G}}_i\| + \frac{1}{\|\mathbf{H}_i\|} \right) \right] g_i |\bar{s}_i|^2 \\ & -\left[\varepsilon_i - \left(\frac{|\bar{\mathbf{H}}_i \dot{\mathbf{x}}_{id} - \mathbf{H}_i \ddot{\mathbf{x}}_{id}|}{g_0} + \left(B_{f_i} + B_{d_i} \right) \|\dot{\mathbf{x}}_{id}\| + B_{t_i} + B_{z_i} \|\dot{\mathbf{X}}\| \right) \right] g_i |\bar{s}_i| \end{aligned}$$

If η_i is chosen in (6.23) and ε_i is chosen in (6.24), then $\dot{V}_{i2} < 0; \bar{s}_i \neq 0$, and the sliding variable $\bar{s}_i$ will converge to zero by the Lyapunov stability theorem.

When $\bar{s}_i = 0$, the inequality (6.17) is satisfied. Therefore, from Lemma 6.2, the closed-loop system is asymptotically stable. ■

Remark 6.7: In order to reduce the chattering phenomenon, the following approximation for the signum function has been adopted:

$$\operatorname{sgn}(e) \approx \frac{e}{|e| + \vartheta} \tag{6.29}$$

with ϑ being a small positive constant.

6.4 Experimental Results

In this section, a neural controller for the Coupled Electric Drives CE8 system, as shown in Figure 6.1, is designed and implemented to validate the proposed method.

6.4.1 High-gain observer design

For the tension subsystem, the state estimation is an important consideration and high-gain observer is constructed through

$$\begin{aligned} \dot{\hat{\mathbf{x}}} &= \mathbf{A}\hat{\mathbf{x}} + \mathbf{L}(y_x - \hat{y}_x) \\ y_x &= \mathbf{C}\hat{\mathbf{x}} \end{aligned} \tag{6.30}$$

where $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$, $\hat{\mathbf{x}}$ denotes the state estimate, $\mathbf{L} = \begin{bmatrix} \frac{a_1}{\mu} & \frac{a_2}{\mu^2} & \frac{a_3}{\mu^3} & \frac{a_4}{\mu^4} \end{bmatrix}^T$ the observer gains, and μ small positive scalar to assure small observer errors (Khalil, 2002).

The positive constant vector $[a_1, a_2, a_3, a_4]$ is chosen such that all roofs of the polynomial $s^4 + a_1 s^3 + a_2 s^2 + a_3 s + a_4 = 0$ are μ times the desired poles of the observer.

For the tension subsystem, suppose that the closed-loop system performance has no more than 2% overshoot and a rise time of no more than 1 second. Thus a damping ratio $\zeta = 0.8$ will meet the overshoot requirement, and for this damping ratio, a rise time of 4 seconds suggests a natural frequency $\omega_n = 4 \times \frac{1.8}{4} = 1.8$. The observer poles can be chosen to be faster than the desired poles by a factor of 5:

$$s_{1,2} = 7.2 \pm j5.4; \quad s_{3,4} = -9.$$

Thus with $\mu = 0.033$, the positive constant vector $[a_1, a_2, a_3, a_4]$ is obtained as:

$$[a_1, a_2, a_3, a_4] = [0.09431, 0.458687, 1.0692, 1]. \tag{6.31}$$

Discretisating the system (6.30) using the zero-order hold method (Franklin et al., 2002), with the sampling time $T = 0.025$, leads to the applicable observer equation:

$$\begin{aligned} \hat{\mathbf{x}}(k+1) &= \boldsymbol{\Gamma}\hat{\mathbf{x}}(k) + \boldsymbol{\Lambda}\left(y(k) - \hat{y}(k)\right) \\ \hat{y}(k) &= \boldsymbol{\Pi}\hat{\mathbf{x}}(k) \end{aligned} \tag{6.32}$$

where

$$\boldsymbol{\Gamma} = \begin{bmatrix} 1 & 0.025 & 0.0003125 & 0.0000026 \\ 0 & 1 & 0.025 & 0.0003125 \\ 0 & 0 & 1 & 0.025 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad \boldsymbol{\Lambda} = \begin{bmatrix} 0.294278 \\ 22.023423 \\ 1007.30993 \\ 21080.6622 \end{bmatrix};$$

$$\boldsymbol{\Pi} = [1 \quad 0 \quad 0 \quad 0].$$

6.4.2 Sliding-mode-based state feedback controller

From the system dynamics (6.2) and (6.3), the sliding functions for the speed and tension subsystems are defined as:

$$\begin{aligned} &\text{Speed subsystem } \{s_w = e_\omega \\ &\text{Tension subsystem } \{s_x = \mathbf{H}_x \mathbf{e}_x = h_{x1}e_{x1} + h_{x2}e_{x2} + h_{x3}e_{x3} + h_{x4}e_{x4} \end{aligned} \tag{6.33}$$

where the speed error $e_\omega = \omega - \omega_d$, ω_d is the speed reference input, $\mathbf{e}_x = \mathbf{x} - \mathbf{x}_d = [e_{x1} \quad e_{x2} \quad e_{x3} \quad e_{x4}]^T$ is the tracking error vector, $\mathbf{x}_d = [x_{d1} \quad x_{d2} \quad x_{d3} \quad x_{d4}]^T = [y_{xd} \quad 0 \quad 0 \quad 0]^T$ is the desired state and y_{xd} the desired tension input.

The vector value $\mathbf{H}_x$ is appropriately chosen so that the system performance is satisfied.

From the real-time experiments in the discrete-time domain, the appropriate value of the vector $\mathbf{H}_x$ is obtained as:

$$\mathbf{H}_x = [1.7784 \quad 0.075036 \quad -0.015123 \quad 0.001]. \tag{6.34}$$

To compare the control performance, a continuous sliding-mode controller using the popular boundary layer method is designed for the speed and tension subsystem as:

$$\begin{aligned} u_\omega &= K_\omega \text{sat}\left(\frac{s_\omega}{\phi_\omega}\right) \\ u_x &= K_x \text{sat}\left(\frac{s_x}{\phi_x}\right) \end{aligned} \tag{6.35}$$

where $\text{sat}\left(\frac{s}{\phi}\right) = \begin{cases} 1 & \text{if} \quad s > \phi \\ \frac{S}{\phi} & \text{if} \quad |s| \le \phi \\ -1 & \text{if} \quad s < -\phi \end{cases}$, and ϕ is the thickness of the boundary layer.

Figure 6.5 shows the experiment results of the controlled system with the reference inputs $r_\omega = 2$; $r_x = 2$ and the controller parameters $K_\omega = 2.0$; $\phi_\omega = 1.0$; $K_x = 2.0$; $\phi_x = 0.5$.

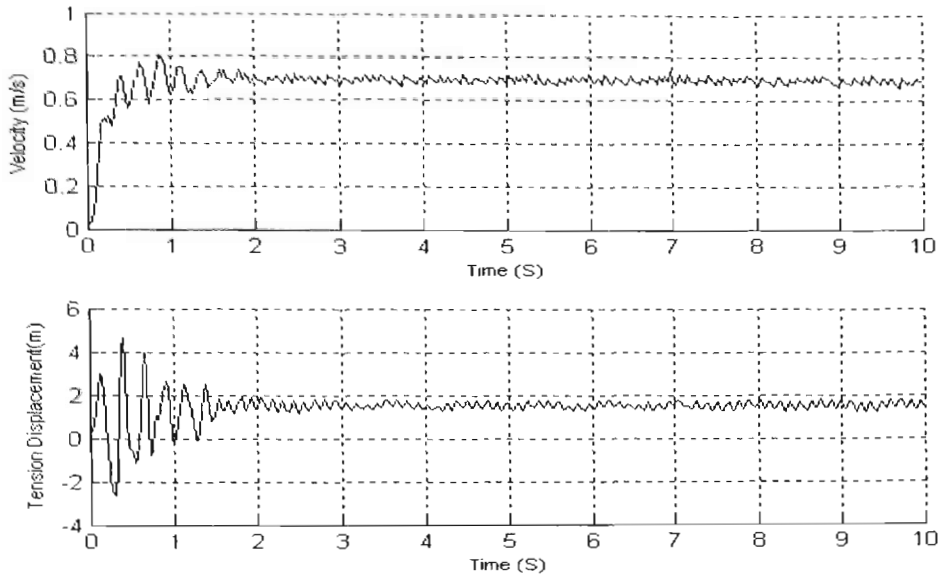


Figure 6.5: The system outputs using the continuous sliding-mode controller with the reference inputs $r_\omega = 2;\ r_x = 2$.

When the saturated function is replaced by the saturated integral-proportional function, the control law becomes:

$$u_\omega = K_\omega \rho\left(\frac{s_\omega}{\phi_\omega}\right)$$
$$u_x = K_x \rho\left(\frac{s_x}{\phi_x}\right) \tag{6.36}$$

where the saturated integral-proportional function is defined in (6.7).

In the experiment, the proposed controller (6.36) is applied to the speed and tension subsystems. With the reference inputs $r_\omega = 2;\ r_x = 2$ and the controller parameters $K_\omega = 2.0;\ \phi_\omega = 1.0;\ K_x = 2.0;\ \phi_x = 0.5;\ T_\omega = \frac{1}{0.09}$ and $T_x = \frac{1}{0.02}$, the controlled system outputs are plotted in Figure 6.6.

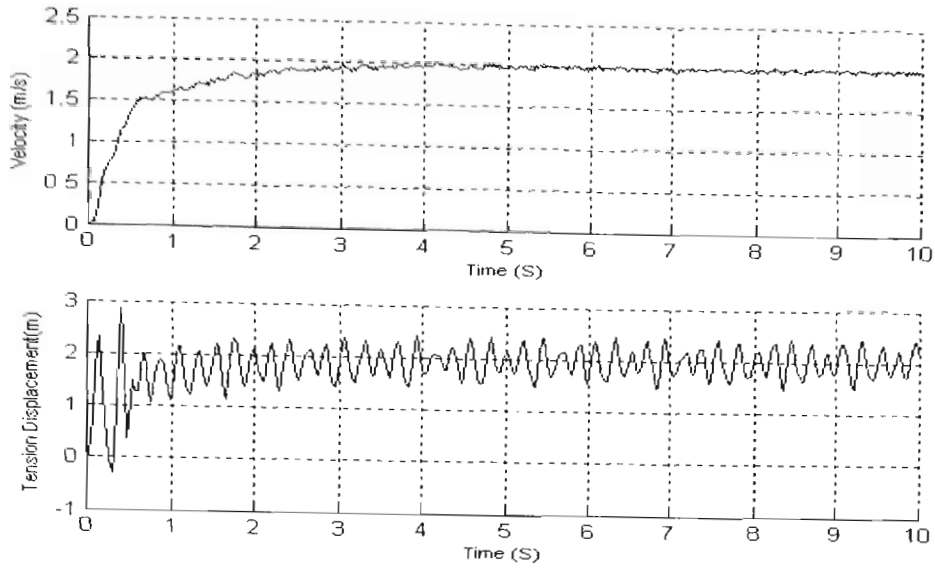


Figure 6.6: The system outputs using the sliding-mode-based feedback controller with the reference inputs $r_\omega = 2;\ r_x = 2$.

6.4.3 The NN controller

The sliding functions for the speed and tension subsystems are defined in the discrete-time domain as:

$$\overline{s}_\omega(k) = \delta_\omega e_\omega(k) + e_\omega(k+1) \tag{6.37}$$

$$\overline{s}_x(k) = \delta_x s_x(k) + s_x(k+1);\quad s_x(k) = \mathbf{H}_x \mathbf{e}_x(k) \tag{6.38}$$

In the real-time system working in the discrete-time domain, $\mathbf{H}_x$ is chosen as in (6.34), and $\delta_\omega = -0.7305$ and $\delta_x = -0.9305$ can be chosen.

Two parallel neural network controllers are designed for the whole Coupled Electrical Drives system, as shown in Figure 6.7. A neural network with two input nodes, three hidden nodes and one output node is designed for the speed subsystem and a neural network controller with five input nodes, seven hidden nodes and one output node is designed for the tension subsystem. The NNs' weights are initialised by very small random values.

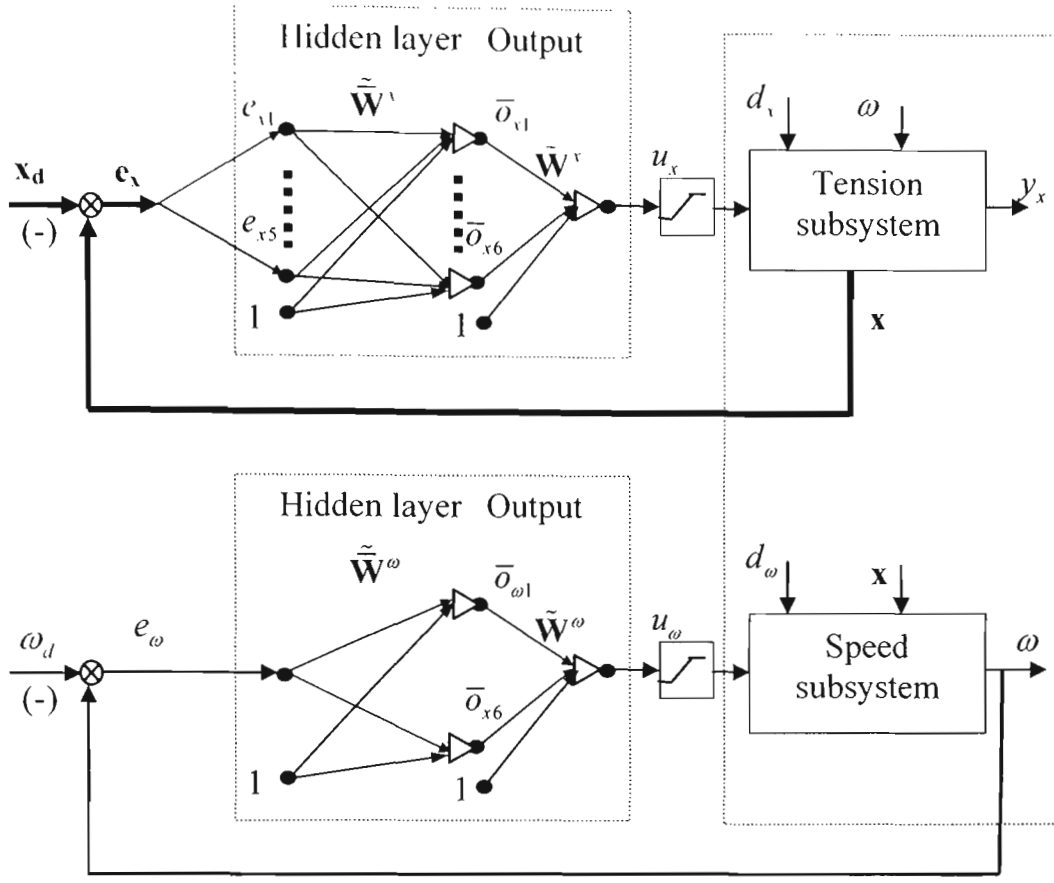


Figure 6.7: Structure of the whole Coupled Electrical Drives system using the decentralised neural network controller.

The on-line training algorithms (6.22) in the discrete-time domain become:

$$\Delta\tilde{\mathbf{W}}'(k) = -\frac{\left[\eta_i \bar{s}_i(k) + \varepsilon_i \operatorname{sgn}\left(\bar{s}_i(k)\right)\right]}{\left\|\tilde{\bar{\mathbf{o}}}_i(k)^T\right\|^2 + \left\|\mathbf{F}'_{hi}(k)\mathbf{W}'(k)^T \tilde{\mathbf{x}}_i(k)^T\right\|^2} \tilde{\bar{\mathbf{o}}}_i(k)^T$$

$$\Delta\tilde{\bar{\mathbf{W}}}'(k) = -\frac{\left[\eta_i \bar{s}_i(k) + \varepsilon_i \operatorname{sgn}\left(\bar{s}_i(k)\right)\right]}{\left\|\tilde{\bar{\mathbf{o}}}_i(k)^T\right\|^2 + \left\|\mathbf{F}'_{hi}(k)\mathbf{W}'(k)^T \tilde{\mathbf{x}}_i(k)\right\|^2} \mathbf{F}'_{hi}(k)\mathbf{W}'(k)^T \tilde{\mathbf{x}}_i(k)$$

With the approximation of the signum function $\operatorname{sgn}(e) \approx \frac{e}{|e| + 0.01}$, and the parameters $\eta_\omega = 0.02;\ \varepsilon_\omega = 0.03;\ \eta_x = 0.02;\ \varepsilon_x = 0.022$, the training algorithm is successfully implemented in the real-time Coupled Electric Drives system.

Figures 6.8 and 6.9 indicate the experimental results of the controlled system under the designed neural controller with different reference inputs.

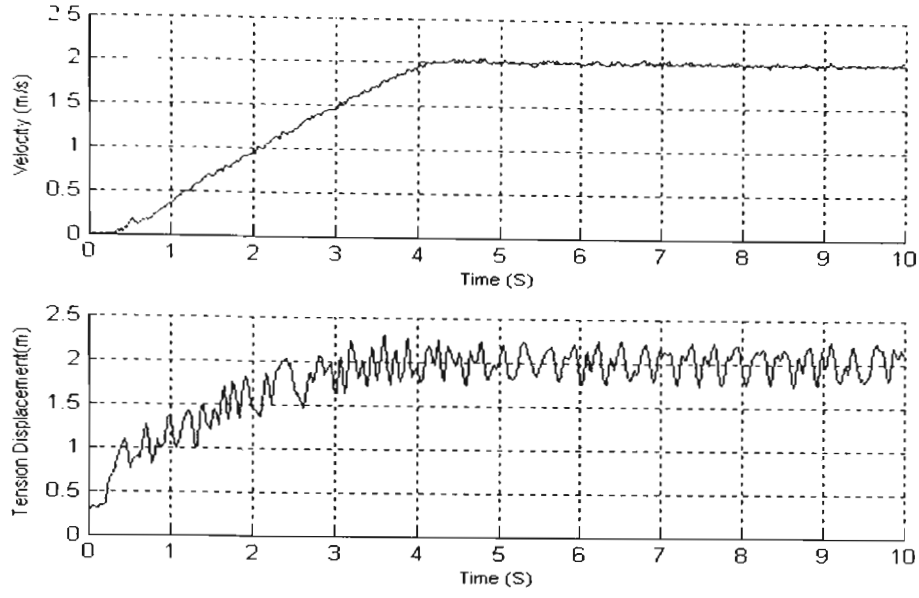


Figure 6.8: The system outputs using the proposed neural controller with the reference inputs $r_\omega = 2;\ r_x = 2$.

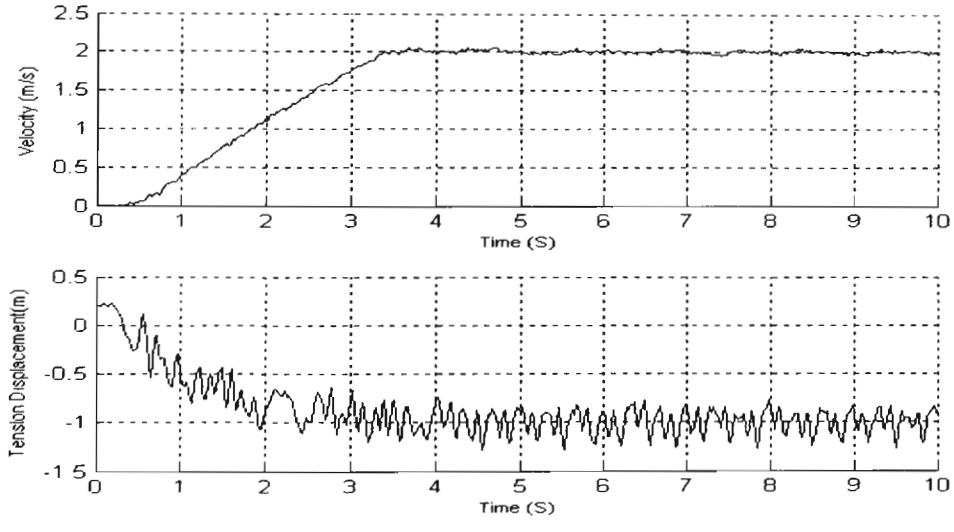


Figure 6.9: The system outputs using the proposed neural controller with the reference inputs $r_\omega = 2;\ r_x = -1$.

6.5 Discussion and conclusion

From the experimental results shown in Figures 6.5, 6.6 and 6.8, output performances of the three control systems have been reported in Table 6.1.

Table 6.1: Performance comparison among the three control systems

Control method	Subsystem	Rise time	Max. overshoot (%)	Settling time
The continuous sliding-mode controller	Speed	—	—	false
	Tension	—	—	false
The sliding-mode-based feedback controller	Speed	3.7	0	3.9
	Tension	3.5	45	3.6
The proposed neural controller	Speed	3.9	0	4.1
	Tension	3.4	0	3.5

In the real-time experiments, when the standard sliding-mode controller is utilised, the high-frequency component of control signals causes severe oscillations of system outputs. This chattering problem can be decreased if a conventional equivalent control term is added. However, the calculation of the equivalent control term requires the exact mathematical model, which is difficult to obtain for the Coupled Electrical Drives. With the continuous sliding-mode controller (6.35), steady state errors always exist, that is $e_{\omega}(\infty) = 1.3$ and $e_{x}(\infty) = 0.22$, although the control inputs are smooth. This trade-off between the smoothness of control signals and the control accuracy is often mentioned in the variable structure system theory. Apparently, because the system's mathematical model is unknown, both the standard sliding-mode controller and the continuous sliding-mode controller cannot effectively be implemented in the real-time Coupled Electric Drives system.

By using the saturated integral-proportional function in the control law, the system outputs can track the reference inputs after a rise time of 3.7 seconds. However, the control signal described in (6.36) is large, and leads to a large overshoot of 45% in the transition response of the tension output. Comparing to the experimental results of the continuous sliding-mode controller, the sliding-mode-based state feedback controller can improve system performance with minimum tracking errors. Nevertheless, the system's overshoot performance is unsatisfactory when the sliding-mode-based state feedback controller is used.

Using the proposed neural controller, the control system outputs can track their setting points after 3.9 seconds, while the overshoot performance is good. Moreover, system performances of the neural control system are still good even with different reference inputs, as shown in Figure 6.8 and 6.9. As the signum function has been approximated by Equation (6.29), the effect of interconnections and disturbances results in an acceptable limit of tension output. These experimental results show the effective applicability of the proposed neural controller for a large-scale system with interconnections and disturbances.

In conclusion, in this chapter, a sliding-mode-based feedback controller and a neural network-based controller have been designed for a class of uncertain nonlinear large-scale system with high-order interconnections and disturbances. Although the sliding-mode-based feedback controller can guarantee the closed-loop system stability, the associated system performance is not satisfactory. For the NN-based controller, a novel sliding function has been designed and used for training the neural network. When the RMBP on-line learning algorithm is utilised in the neural controller, system states are kept in the sliding surface. Therefore, the system stability is guaranteed and the performance specification is satisfied by the appropriate choice of sliding function parameters. Experimental results in a real-time Coupled Electric Drives illustrated the effectiveness of the proposed methods.

CHAPTER 7

CONCLUSION AND RECOMMENDATIONS FOR FUTURE RESEARCH

7.1 Conclusion

The main objective of this research, as mentioned in Chapter 1, was to develop novel learning algorithms for feedforward neural networks so that the problems, which occur when the feedforward neural networks using the backpropagation algorithm are applied in classification and control, can be overcome. This objective has been achieved through the combination of the sliding mode techniques and the backpropagation algorithm, which improves the convergence rate, global convergence capability and robustness of the backpropagation algorithm.

The first chattering-free sliding-mode-based learning algorithm for the feedforward neural networks was developed to obtain fast and global convergence when the FNNs were applied in classification, particularly in the head-movement classifier for wheelchair control. The second chattering-free sliding-mode-based algorithm was designed for on-line training of a direct feedback neural controller such that the controlled system's stability was guaranteed. Moreover, the training procedure for the neural controller was proposed to avoid the requirement of the careful choice of training inputs in the current identification methods. The third robust sliding-mode-based algorithm for on-line training of a decentralised neural network controller was designed to assure the stability and robustness of a class of uncertain nonlinear large-scale systems with interconnections and disturbances.

In order to obtain an overview of the development of neural networks and their learning algorithms and applications, an initial historical perspective was presented in Chapter 2. A critical analysis, contained in Chapter 2, showed the disadvantages of the existing learning algorithms for improving convergence speed and global convergence

capability, and also pointed out a gap in the literature, which can be developed further to an effective algorithm in this thesis. The review also highlighted the difficulties experienced in the use of neural networks for modelling system dynamics and the lack of stability guarantees in most of the adaptive neural control systems. The analysis exhibits an opening capability of the sliding-mode-based learning algorithms in neural control applications.

Background information and research relating to the backpropagation learning algorithm and its variations, especially the use of the variable structure system theory in the neural network leaning algorithms, are provided in Chapter 3. A systematic analysis based on the Lyapunov method explained the slow convergence rate of BP and the possibility of getting trapped in the local minima of the BP and second-order gradient algorithms, and highlighted the efficacy of use of the VSS in the NNs' learning algorithms. This background research is the foundation of and motivation for the development of some novel sliding-mode-based training algorithms in this thesis.

From the chattering-free sliding-mode control strategy, an extension of the adaptive BP algorithm with a momentum term added, named EABPM, was developed to avoid the shortcomings of the adaptive BP algorithm. The proposed EABPM algorithm obtains fast and global convergence properties in the sense of the Lyapunov method. Moreover, the simple implementation and less computational complexity are the advantages of the EABPM algorithm. The EABPM was also applied successfully in the head-movement neural classifier for wheelchair control, as presented in Chapter 4. Experiment results showed that the EABPM always had 100% global convergence capability, obtained the average converged epochs of 423.84, which was the best result, compared with the BPM, ABP, RPROP and MGFPROP algorithms, and also improved more than 5% of the classification accuracy in comparison with the other algorithms. This indicated that the EABPM algorithm outperformed the other algorithms in term of convergence rate, global convergence capability and classification accuracy.

Another novel learning algorithm, named CFSMBP, was developed from the integration between the chattering-free sliding-mode control technique and the backpropagation algorithm, and was appropriate for application in the control area. Solving control problems of the uncertain continuous linear systems, as described in Chapter 4, the

feedforward neural network trained by the CFSMBP algorithm can work as a direct adaptive feedback controller, and guarantees system stability based on the Lyapunov synthesis. Moreover, a novel sliding function was designed and used as the training signal, thus avoiding the problem of unknown plant's Jacobian, or the need for a trained neural network identifier. In an example with an uncertain linear system, the proposed neural controller robustly guaranteed the required performances of the closed-loop system.

The above neural controller was further developed to establish the neural network controller design methodology for a class of uncertain single-input single-output systems, particularly in the time-delay nonlinear Static VAR Compensator system, as presented in Chapter 5. A review of control methods for nonlinear systems with transportation lag, contained in this chapter, illustrated the difficulties and inadequacies of the current solutions, and provided an approach for approximating the nonlinear controller of the time-delay systems. This modelling approach was verified in a real-time Static VAR Compensator system, as described in Section 5.2.

The neural control design method, as proposed in Chapter 5, comprised four stages. Firstly, a novel sliding function was designed from the pole placement method, and was used as the neural network's training signal. Secondly, the feedforward neural network trained by the increment BP learning algorithm or the proposed CFSMBP learning algorithm, which acted as a direct adaptive feedback controller, was proven to stabilise the closed-loop system. Next, a new training procedure for the neural network controller was developed so that the network parameters converge to the optimal values. Finally, the other parameters of the neural controller were calculated, and incorporated with the optimal neural network to control the uncertain systems with transportation lag.

Experimental results for the real-time Static VAR Compensator system, provided in Chapter 5, showed that both the BP and CFSMBP algorithms can guarantee the system's stability. In the training process, the BP and CFSMBP both assure the global convergence of the network, as indicated in Section 5.4. This means the neural network always obtains its optimal weights, and the required system performance is satisfied. Results also concluded that the number of hidden nodes did not affect the global convergence capability of the proposed method. Moreover, the CFSMBP algorithm

always outperforms the BP in term of convergence rate. This training procedure can avoid the difficult choice of the training input signal associated with the convenient training procedure for the neural network identifiers, as discussed in Chapter 2. After training the network, implementation results indicated that the proposed neural controller can robustly stabilise the real-time nonlinear system with transportation lag, while the PID controller, in some cases, failed to track the system output to the reference input.

Furthermore, a robust learning algorithm, named RMBP, was developed from the reaching law method integrated with the BP algorithm. This on-line learning algorithm offered a simple and robust adaptation approach for control applications. As proven in Chapter 4, the direct adaptive feedback controller using the feedforward neural network with the RMBP algorithm assured the stability and robustness of a class of uncertain continuous linear system with disturbances. Simulation results, also provided in Chapter 4, indicated that the RMBP was more robust than the CFSMBP in term of performance.

The method was further developed for controlling the large-scale nonlinear systems, as presented in Chapter 6. A decentralised neural network controller structure was proposed so that each feedforward neural network was utilised as a direct inverse adaptive controller for each subsystem. Motivated from a sliding-mode-based feedback controller, novel sliding functions were defined and utilised as the training signals. The RMBP algorithm was then applied for on-line updating the network parameters, which guaranteed asymptotic stability of the closed-loop system by the Lyapunov method. In the real-time implementation for a two-input two-output Coupled Electric Drives CE8 system, the continuous sliding-mode controller always caused steady-state errors. Although the sliding-mode-based feedback controller could reduce the output errors, system performance was distorted with a large overshoot of 45%. The proposed decentralised neural controller using the RMBP algorithm could stabilise the system and the system performance was satisfied, that is, a rise time of 3.9 seconds and no overshoot. These real-time experimental results demonstrated the effectiveness and feasiblity of the proposed neural control method.

7.2 Recommendations for Future Research

Further research should involve two directions. The first direction concerns obtaining a unified learning algorithm from the three neural learning algorithms proposed in this thesis. As discussed in Chapter 4, the reaching law method is the most general scheme among other sliding-mode control techniques, and converges the output error to zero in the fastest finite time. Therefore, the RMBP algorithm derived from the reaching law method is the optimal choice for the unified learning algorithm. If the RMBP algorithm replaces the CFSMBP algorithm in the proposed neural controller for the delay-input systems, the training process, as described in Chapter 5, will converge faster and more robustly. In classification problems, the RMBP algorithm, in the same manner of the robust Minowski-r BP algorithm (Hanson & Burr, 1988), can also improve the robust property of the BP algorithm in noisy environments. However, the RMBP is an incremental learning algorithm while the EABPM is a batch learning approach. Therefore, the application of RMBP in the head-movement neural classifiers needs further research and validation. The development of a unified learning algorithm for the advanced neural network controllers and classifiers is a topic currently under investigation.

Another critical point in this research is to design the parameters of the sliding function, which in turn decides the system performance. For linear systems, these parameters of the sliding function were designed by the pole placement method, as presented in Chapters 4. However, for nonlinear systems, this is a very challenging problem in control engineering. Some studies have already developed an adaptive approach for design of the sliding surface (Bekiroglu, 1996). Therefore, it is possible to utilise a neural network to learn the nonlinear sliding surface. Nevertheless, if a neural network is used for identifying the sliding function, the neural control system becomes the conventional direct adaptive neural control scheme with a trained neural identifier (Ng, 1997). To avoid the challenge of choosing appropriate input signals over the operational range of the control system, a novel training procedure for FNN to approximate the nonlinear sliding surface should be developed. Based on the training procedure obtained

in Chapter 5, the development of an advanced neural approach for designing an optimal sliding surface for nonlinear control systems should be feasible.

APPENDIX A

A.1 Matlab Program for Modelling the SVC System

File name: StVrEx01.m

```
function out=StVaEx01(filename1,filename2,filename3,filename4,filename5)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%   Program to get model of the SVC system     %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Validate input args
if nargin==0, error('Not enough input arguments.'); end

% get filename
if ~isstr(filename1), error('Filename must be a string.'); end

% do some validation
if isempty(filename1), error('Filename must not be empty.'); end

% make sure file exists
if ~isequal(exist(filename1), 2), error('File not found.'), end

% open the file
fid = fopen(filename1,'r');
if fid == (-1)
   error(['Could not open file ', filename1 ,'.']);
end
A = fscanf(fid,'%f%f%f%f%f%f%f',[7,inf]);
fclose(fid);

[m,n]=size(A);
[St,ERRMSG] = sprintf('The array is %dx%d.',m,n);
disp(St);
for i=1:(n-400),
   t(i)=A(1,i);
   in11(i)=A(2,i);
   in2(i)=A(3,i);
   in3(i)=A(4,i);
   in4(i)=A(5,i);
   out1(i)=A(6,i);
   out2(i)=A(7,i);
end
plot(t,in11,'g');
```

```
xlabel('Time (S)','FontSize',8);
ylabel('System output','FontSize',8);
hold;
%grid on;
pause;
% Validate input args
if nargin==0, error('Not enough input arguments.'); end

% get filename
if ~isstr(filename2), error('Filename must be a string.'); end

% do some validation
if isempty(filename2), error('Filename must not be empty.'); end

% make sure file exists
if ~isequal(exist(filename2), 2), error('File not found.'), end

% open the file
fid = fopen(filename2,'r');
if fid == (-1)
   error(['Could not open file ', filename2 ,'.']);
end
A = fscanf(fid,'%f%f%f%f%f%f%f',[7,inf]);
fclose(fid);

[m,n]=size(A);
[St,ERRMSG] = sprintf('The array is %dx%d.',m,n);
disp(St);
for i=1:(n-400),
   t(i)=A(1,i);
   in12(i)=A(2,i);
   in2(i)=A(3,i);
   in3(i)=A(4,i);
   in4(i)=A(5,i);
   out1(i)=A(6,i);
   out2(i)=A(7,i);
end
plot(t,in12,'g');
xlabel('Time (S)','FontSize',8);
ylabel('System output','FontSize',8);
%hold;
%grid on;
pause;
% Validate input args
if nargin==0, error('Not enough input arguments.'); end

% get filename
if ~isstr(filename3), error('Filename must be a string.'); end

% do some validation
```

```
if isempty(filename3), error('Filename must not be empty.'); end

% make sure file exists
if ~isequal(exist(filename3), 2), error('File not found.'), end

% open the file
fid = fopen(filename3,'r');
if fid == (-1)
   error(['Could not open file ', filename3 ,'.']);
end
A = fscanf(fid,'%f%f%f%f%f%f%f',[7,inf]);
fclose(fid);

[m,n]=size(A);
[St,ERRMSG] = sprintf('The array is %dx%d.',m,n);
disp(St);
for i=1:(n-400),
   t(i)=A(1,i);
   in13(i)=A(2,i);
   in2(i)=A(3,i);
   in3(i)=A(4,i);
   in4(i)=A(5,i);
   out1(i)=A(6,i);
   out2(i)=A(7,i);
end
plot(t,in13,'g');
xlabel('Time (S)','FontSize',8);
ylabel('System output','FontSize',8);
%hold;
%grid on;
pause;
% Validate input args
if nargin==0, error('Not enough input arguments.'); end

% get filename
if ~isstr(filename4), error('Filename must be a string.'); end

% do some validation
if isempty(filename4), error('Filename must not be empty.'); end

% make sure file exists
if ~isequal(exist(filename4), 2), error('File not found.'), end

% open the file
fid = fopen(filename4,'r');
if fid == (-1)
   error(['Could not open file ', filename4 ,'.']);
end
A = fscanf(fid,'%f%f%f%f%f%f%f',[7,inf]);
fclose(fid);
```

```
[m,n]=size(A);
[St,ERRMSG] = sprintf('The array is %dx%d.',m,n);
disp(St);
for i=1:(n-400),
   t(i)=A(1,i);
   in14(i)=A(2,i);
   in2(i)=A(3,i);
   in3(i)=A(4,i);
   in4(i)=A(5,i);
   out1(i)=A(6,i);
   out2(i)=A(7,i);
end
plot(t,in14,'g');
xlabel('Time (S)','FontSize',8);
ylabel('System output','FontSize',8);
%hold;
%grid on;
pause;

% Validate input args
%if nargin==0, error('Not enough input arguments.'); end

% get filename
if ~isstr(filename5), error('Filename must be a string.'); end

% do some validation
if isempty(filename5), error('Filename must not be empty.'); end

% make sure file exists
if ~isequal(exist(filename5), 2), error('File not found.'), end

% open the file
fid = fopen(filename5,'r');
if fid == (-1)
   error(['Could not open file ', filename5 ,'.']);
end
A = fscanf(fid,'%f%f%f%f%f%f%f',[7,inf]);
fclose(fid);

[m,n]=size(A);
[St,ERRMSG] = sprintf('The array is %dx%d.',m,n);
disp(St);
for i=1:(n-400),
   t(i)=A(1,i);
   in15(i)=A(2,i);
   in2(i)=A(3,i);
   in3(i)=A(4,i);
   in4(i)=A(5,i);
   out1(i)=A(6,i);
```

```
   out2(i)=A(7,i);
end
plot(t,in15,'g');
xlabel('Time (S)','FontSize',8);
ylabel('System output','FontSize',8);
%hold;
grid on;
pause;


 B=[in11;in12;in13;in14;in15];
 H=mean(B);
 plot(t,H,'k-');

xlabel('Time (S)','FontSize',8);
ylabel('System output','FontSize',8);
grid on;
hold;
pause;

clf;
[k]=length(H)
 for i=1:k,
   t(i)=A(1,i);
   out1(i)=A(6,i);
   md(i)=H(i)-3.41; % Cancel offset
end
plot(t,md,'k-');
xlabel('Time (S)','FontSize',8);
ylabel('System output','FontSize',8);
hold;
%pause;
%plot(t,out1);
pause;
hs1=tf(0.305,[0.12 1],'InputDelay',0.09)

hs2=tf(0.275,[0.19 1],'InputDelay',0.11)

lsim(hs1,out1,t);lsim(hs2,out1,t);
pause;
```

APPENDIX B

B.1 Chapter 5 Proofs

Let us first introduce the following Lemmas.

Lemma 5.1: *For the real vectors or matrices* $\mathbf{M},\mathbf{N}$ *with appropriate dimension, we have*

$$\mathbf{M}^T\mathbf{N}+\mathbf{N}^T\mathbf{M}\le\beta\mathbf{M}^T\mathbf{M}+\beta^{-1}\mathbf{N}^T\mathbf{N}$$

for any positive constant β.

Proof: *See* (Wang & Cheng, 1992).

Lemma 5.2: *For the dynamic system* (5.22), *the following relationship is always obtained:*

$$\|\mathbf{z}(k)\|\le\sqrt{\frac{\lambda_{\max}\left(\beta\mathbf{G}^T\mathbf{P}^T\mathbf{P}\mathbf{G}+\mathbf{G}^T\mathbf{P}\mathbf{G}\right)}{\lambda_{\min}\left(\mathbf{Q}-\beta^{-1}\mathbf{F}^T\mathbf{F}\right)}}|s_k|, \tag{B.1}$$

where β *is an arbitrary positive constant, and a symmetric positive matrix* $\mathbf{Q}$ *is chosen so that* $\left(\mathbf{Q}-\beta^{-1}\mathbf{F}^T\mathbf{F}\right)$ *is positive definite.*

Proof: Suppose that $\|\mathbf{z}(k)\|>\sqrt{\frac{\lambda_{\max}\left(\beta\mathbf{G}^T\mathbf{P}^T\mathbf{P}\mathbf{G}+\mathbf{G}^T\mathbf{P}\mathbf{G}\right)}{\lambda_{\min}\left(\mathbf{Q}-\beta^{-1}\mathbf{F}^T\mathbf{F}\right)}}|s_k|$

Choose a Lyapunov function

$$V_1(k)=\mathbf{z}^T(k)\mathbf{P}\mathbf{z}(k).$$

The change of V_1 is obtained from Equations (5.22), (5.24) and Lemma 5.1

$$\begin{aligned}
\Delta V_1(k) &= V_1(k+1) - V_1(k) = \mathbf{z}(k+1)^T \mathbf{P}\mathbf{z}(k+1) - \mathbf{z}(k)^T \mathbf{P}\mathbf{z}(k) \\
&= \left(\mathbf{F}\mathbf{z}(k) + \mathbf{G}s_k\right)^T \mathbf{P}\left(\mathbf{F}\mathbf{z}(k) + \mathbf{G}s_k\right) - \mathbf{z}(k)^T \mathbf{P}\mathbf{z}(k) \\
&= \mathbf{z}(k)^T \left(\mathbf{F}^T\mathbf{P}\mathbf{F} - \mathbf{P}\right)\mathbf{z}(k) + s_k\mathbf{G}^T\mathbf{P}^T\mathbf{F}\mathbf{z}(k) + \mathbf{z}(k)^T\mathbf{F}^T\mathbf{P}\mathbf{G}s_k + s_k\mathbf{G}^T\mathbf{P}\mathbf{G}s_k \\
&\le -\mathbf{z}(k)^T\mathbf{Q}\mathbf{x}(k)\mathbf{z}(k) + \beta s_k\mathbf{G}^T\mathbf{P}^T\mathbf{P}\mathbf{G}s_k + \beta^{-1}\mathbf{z}(k)^T\mathbf{F}^T\mathbf{F}\mathbf{z}(k) + s_k\mathbf{G}^T\mathbf{P}\mathbf{G}s_k \\
&\le -\lambda_{\min}\left(\mathbf{Q} - \beta^{-1}\mathbf{F}^T\mathbf{F}\right)\left\|\mathbf{z}(k)\right\|^2 + \lambda_{\max}\left(\beta\mathbf{G}^T\mathbf{P}^T\mathbf{P}\mathbf{G} + \mathbf{G}^T\mathbf{P}\mathbf{G}\right)s_k^2
\end{aligned}$$

If $\left\|\mathbf{z}(k)\right\| > \sqrt{\dfrac{\lambda_{\max}\left(\beta\mathbf{G}^T\mathbf{P}^T\mathbf{P}\mathbf{G} + \mathbf{G}^T\mathbf{P}\mathbf{G}\right)}{\lambda_{\min}\left(\mathbf{Q} - \beta^{-1}\mathbf{F}^T\mathbf{F}\right)}}\left|s_k\right|$, then $\Delta V_1 < 0$ and all system states $\mathbf{z}$ will converge to zero. Consequently, the inequality (B.1) is obtained. ■

Proof of Proposition 5.1: Substituting Equations (5.11) and (5.17) into (5.16) yields

$$s_k = \delta\sigma_k + \sigma_{k+1} = \delta\mathbf{H}\mathbf{x}(k) + \mathbf{H}\left(\mathbf{A} + \Delta\mathbf{A}\right)\mathbf{x}(k) + \mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)u(k) \tag{B.2}$$

$$\begin{aligned}
\frac{\partial s_k}{\partial u} &= \frac{\partial}{\partial u}\left[\mathbf{H}\left(\mathbf{A} + \Delta\mathbf{A}\right)\mathbf{x}(k) + \delta\mathbf{H}\mathbf{x}(k) + \mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)u(k)\right] \\
&= \mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)
\end{aligned} \tag{B.3}$$

The gradients of V with respect to the weight matrix or vector are calculated as:

$$\frac{\partial V}{\partial \tilde{\mathbf{W}}(k)} = \frac{\partial V}{\partial s_k}\frac{\partial s_k}{\partial u}\frac{\partial u}{\partial \tilde{\mathbf{W}}(k)} = s_k\mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)\tilde{\bar{\mathbf{o}}}(k)^T \tag{B.4}$$

$$\frac{\partial V}{\partial \tilde{\bar{\mathbf{W}}}(k)} = \frac{\partial V}{\partial s_k}\frac{\partial s_k}{\partial u}\frac{\partial u}{\partial \bar{\mathbf{o}}(k)}\frac{\partial \bar{\mathbf{o}}(k)}{\partial \tilde{\bar{\mathbf{W}}}(k)} = s_k\mathbf{H}\left(\mathbf{B} + \Delta\mathbf{B}\right)\mathbf{F}_h'(k)\mathbf{W}(k)^T\tilde{\mathbf{x}}(k)^T \tag{B.5}$$

As demonstrated in Salas & Hill (1990) and Ng (1997), the change of function V in (5.28) can be obtain as:

$$\Delta V = \sum_{i=1}^{m}\frac{\partial V}{\partial \bar{W}_i}\Delta\bar{W}_i + \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{\partial V}{\partial W_{ij}}\Delta W_{ij} + \sum_{j=1}^{n}\frac{\partial V}{\partial x_j}\Delta x_j \tag{B.6}$$

Substituting Equations (5.29) , (B.4) and (B.5) in to Equation (B.6) leads to:

$$\begin{aligned}\Delta V &= -\eta \mathbf{H}\left(\mathbf{B}+\Delta\mathbf{B}\right) s_k^{\,2} + s_k \frac{\partial s_k}{\partial \mathbf{x}(k)} \Delta \mathbf{x}(k) \\ \Delta V &\le -\eta \mathbf{H}\left(\mathbf{B}+\Delta\mathbf{B}\right) s_k^{\,2} + \left|s_k\right| \left\| \frac{\partial s_k}{\partial \mathbf{x}(k)} \right\| \left\| \Delta \mathbf{x}(k) \right\|\end{aligned} \tag{B.7}$$

From Equation (5.16), added Equations (5.11) and (5.26), it is obtained as:

$$\frac{\partial s_k}{\partial \mathbf{x}(k)} = \mathbf{H}\left(\mathbf{A}+\Delta\mathbf{A}\right) + \delta\mathbf{H} + \mathbf{H}\left(\mathbf{B}+\Delta\mathbf{B}\right)\mathbf{W}\mathbf{F}_h'\bar{\mathbf{W}} \tag{B.8}$$

Because $f_{hi}' = \dfrac{d}{dv}\left(\dfrac{1-e^{-v}}{1+e^{-v}}\right) = \dfrac{1}{2}\left(1-f_{hi}^2\right) \le 0.5, \quad i = 1,\ldots,m$,

$$\left\| \frac{\partial s_k}{\partial \mathbf{x}(k)} \right\| \le \left\|\mathbf{H}\right\| \left[\left\|\mathbf{A}\right\| + \rho_A + \delta + 0.5\left(\left\|\mathbf{B}\right\| + \rho_B\right) B_{\bar{W}} B_W \right] = \left\|\mathbf{H}\right\| k_{\max} \tag{B.9}$$

where $k_{\max} = \left\|\mathbf{A}\right\| + \rho_A + \delta + 0.5\left(\left\|\mathbf{B}\right\| + \rho_B\right) B_{\bar{W}} B_W$.

Multiplying two sides of Equation (5.16) for $\dfrac{\mathbf{H}^T}{\left\|\mathbf{H}\right\|^2}$ to yield another dynamic equation

$$\mathbf{x}(k+1) = -\delta\mathbf{x}(k) + \frac{\mathbf{H}^T}{\left\|\mathbf{H}\right\|^2} s_k \,. \tag{B.10}$$

Equation (B.10), combined with the inequality (B.1) and Equation (5.18) leads to

$$\left\|\Delta\mathbf{x}(k)\right\| \le \left[\sqrt{\frac{\lambda_{\max}\left(\beta\mathbf{G}^T\mathbf{P}^T\mathbf{P}\mathbf{G} + \mathbf{G}^T\mathbf{P}\mathbf{G}\right)}{\lambda_{\min}\left(\mathbf{Q} - \beta^{-1}\mathbf{F}^T\mathbf{F}\right)}} \frac{\left|1+\delta\right|}{\left\|\mathbf{T}^{-1}\right\|} + \frac{1}{\left\|\mathbf{H}\right\|} \right] \left|s_k\right| . \tag{B.11}$$

Note from Theorem 5.1 that

$$\|\mathbf{H}\| \le \|\mathbf{v}\|\|\mathbf{T}^{-1}\|. \tag{B.12}$$

The inequality (B.7), added (B.9), (B.11) and (B.12) yields

$$\Delta V \le -2\left[\eta\mathbf{H}(\mathbf{B}+\Delta\mathbf{B}) - k_{\max}\sqrt{\frac{\lambda_{\max}\left(\beta\mathbf{G}^T\mathbf{P}^T\mathbf{P}\mathbf{G}+\mathbf{G}^T\mathbf{P}\mathbf{G}\right)}{\lambda_{\min}\left(\mathbf{Q}-\beta^{-1}\mathbf{F}^T\mathbf{F}\right)}}|1+\delta\|\mathbf{v}\| + k_{\max}\right]V.$$

The function V will globally converge to zero if the following inequality is satisfied

$$0 < \eta\mathbf{H}(\mathbf{B}+\Delta\mathbf{B}) - \vartheta_{\max} < \frac{1}{2},$$

where $\vartheta_{\max} = k_{\max}\left[\sqrt{\frac{\lambda_{\max}\left(\beta\mathbf{G}^T\mathbf{P}^T\mathbf{P}\mathbf{G}+\mathbf{G}^T\mathbf{P}\mathbf{G}\right)}{\lambda_{\min}\left(\mathbf{Q}-\beta^{-1}\mathbf{F}^T\mathbf{F}\right)}}|1+\delta\|\mathbf{v}\|+1\right]$.

From Assumption 5.2, if the learning rate η satisfies:

$$0 < \frac{\vartheta_{\max}}{g_0} < \eta < \frac{1}{2g_1} + \frac{\vartheta_{\max}}{g_1},$$

then the sliding variable s_k will globally converge to zero. The on-line training algorithms of the neural network will stop when the sliding variable s_k reaches zero. When $s_k = 0$, from Lemma 5.2, the closed-loop system is asymptotically stable. ■

Proof of Proposition 5.2: Similar to the proof of Proposition 5.1, the gradient terms (B.4) and (B.5) are obtained, and the change of function V can be obtained as in (B.6).

Substituting Equations (5.31), (B.4) and (B.5) in to Equation (B.6) leads to:

$$\Delta V = -\eta\mathbf{H}(\mathbf{B}+\Delta\mathbf{B})\left(\left\|\tilde{\mathbf{o}}(k)^T\right\|^2 + \left\|\mathbf{F}_h'(k)\mathbf{W}(k)^T\tilde{\mathbf{x}}(k)^T\right\|^2\right)s_k^2 + s_k\frac{\partial s_k}{\partial\mathbf{x}(k)}\Delta\mathbf{x}(k)$$

Because $\left\|\tilde{\tilde{\mathbf{o}}}(k)^{l'}\right\|^2 + \left\|\mathbf{F}_h^{l}(k)\mathbf{W}(k)^{l'}\,\tilde{\mathbf{x}}(k)^{l'}\right\|^2 \geq 1$,

$$\Delta V \leq -\eta \mathbf{H}(\mathbf{B}+\Delta\mathbf{B}) s_k^{\,2} + |s_k| \left\| \frac{\partial s_k}{\partial \mathbf{x}(k)} \right\| \|\Delta \mathbf{x}(k)\|$$

As the proof of Proposition 5.1, if we can choose η as in the inequality

$$0 < \frac{\vartheta_{\max}}{g_0} < \eta < \frac{1}{2g_1} + \frac{\vartheta_{\max}}{g_1},$$

then V converges to zero. Consequently the closed-loop system is asymptotically stable.

■

APPENDIX C

C.1 C Program for Training the Neural Network Controller

File name: DSVC_WS1.CPP

```
/* Program simulate SVC sytem using output state variable feedback controller
 Writer: Tri VM Nguyen
 Date: 29 November 2005
 version: 0.1
 Describe: Proposed algorithm for approximating  NN controller,
 optimal learning rate is used and 30 runs for different initial weights

 Control law:   Neural Control System without Observer and reference input
 3 model is used to train the NN controller to check if V->0 or not?

 -> Good result of sliding function defined.

 Observing V by graphic functions


*/

 #include <stdlib.h>
 #include <conio.h>
 #include<stdio.h>
 #include<math.h>
 #include  <graphics.h>
#define TileX 640.0
#define  TileY  480.0
 #define PI 3.14159255

const  N=4;  // state variables = no. of input node.
const  M=6;  // no. of hidden node
const  L=1; // no. of output node
const Lm=10000;     // max epochs

char str[25];
int sig = 5; /* significant digits */
float Un,ymax,xmax,x;
int n;
int Xm,Ym;
int  KDDoHoa(void);
void  VeTruc(void);
void  VeHam(float Un,int n,float ymax,float xmax);

int DongDoHoa(void);
```

```
double sat(double x,double v);
double baf(double v);
double dbaf(double v);
double purelin(double v);
   /*-----------------------------------------------*/
void main()
 {
  FILE *fp,*fp1;
  char fa[40]="weight.txt";
   char fa2[40]="fweight.txt";
  char fa1[40]="SVC_WS12.txt";
  char ch;
 int i,j,dem;
 long k,STEP,count,l,count2;
 double
alpha,beta,Xk_1[N],Xk[N],Xk1[N],Xb[N],Xb1[N],Xc[2],Xc1[2],Nu,Nr,Uk,T,T1,T2,Ki,
e,e1;
 double  yk,rk,z0k[N],y0k[N],W1k1[M][N];
 double  net1k[M],netz1k[M],z1k[N],y1k[M],es1k[M];
 double  W2k1[M],y2k,z2k,net2k,netz2k,es2k ;
 double  Sk,Sk_1,Sk1,dSk,V,J1,delta,HB,k0,Vk,Vk1,g0;
 //Coefficients for Data Processing//
 double DP[N-1]={3.5,50,480};
 // Sliding function coefficients//
 double  H[N-1]={4.0583,-0.2027,0.0092};
// double  H[N-1]={9.40518,-0.28799,0.00346};
// double Nx[N]={55.175034,55.175034,55.175034};
 double E,Em,temp,W2k[M],W1k[M][N];
  randomize();

fp1=fopen(fa1,"w");
fflush(fp1);
printf("\nStarting");
for (dem=1;dem<=1;dem++)
{

  // Initial conditions
    fp=fopen(fa,"w");
   if (fp==NULL)
      perror("Error in data file");
   fflush(fp);

  for(i=0;i<M;i++)
     {
         temp=random(1000);
          x=(500-temp)/5000.0;
           W2k[i]=x;
          fprintf(fp,"%f\n",x);
      }
```

```
  for (i=0;i<(M-1);i++)
    for (j=0;j<N;j++)
      {
          temp=random(1000);
           x=(500-temp)/5000.0;
            W1k[i][j]=x;
           fprintf(fp,"%f\n",x);
       }
  fclose(fp);


  KDDoHoa();

  T=0.025;  //simulation timestep
  STEP=80; // no. of step
  alpha=0.1;  // learning rate
  beta=0.12;
   // Initial conditions
   g0=1;
   rk=0;
  // count=2;
 //    delta=-0.637628;
   delta=-0.4066;
   Em=0.0003;     // error threshold
   l=1;         // epochs


  do
  {
    E=0;     // error function
  for (count2=0;count2<3;count2++)
  {
   count=random(3);
   VeTruc();
   temp=random(10);
   Xk[0]=5-temp;
   Xk[1]=0;      // Initial states
   Xk[2]=0;
    V=0; yk=0;
 // Initial control signal
  if (count==0)
   {
    //nominal System
  Uk=-(delta*H[0]+H[0]*0.161-H[1]*14.848-H[2]*64.34)*Xk[0]/(H[0]*0.0166-
H[1]*2.332+H[2]*49.74);
    }
   if (count==1)
     {
   // System 2
```

```
    Uk=-(delta*H[0]-H[0]*0.038-H[1]*23.334-H[2]*143.102)*Xk[0]/(H[0]*0.012-
H[1]*5.709+H[2]*196.959);
     }
  if (count==2)
          {
    // Sytem 3
          Uk=-(delta*H[0]+H[0]*0.268-H[1]*11.164-H[2]*37.838)*Xk[0]/(H[0]*0.011-
H[1]*1.0285+H[2]*17.76);
           }


 for (k=0;k<STEP;k++)
  {
   //Dynamic relations
  if (count==0)
   {
////Nominal Parameter//////
   Xk1[0] = 0.1607*Xk[0]+0.01327*Xk[1]+0.00021*Xk[2]  +0.0166*Uk;
    Xk1[1] = -14.848*Xk[0]+0.7723*Xk[1]+0.022916*Xk[2]-2.3319*Uk;
    Xk1[2] = -64.34*Xk[0]-1.015*Xk[1]+0.9906*Xk[2] +49.74*Uk;
    yk=Xk[0];
   }
  if (count==1)
    {
  //// Upper bound parameter///////
   Xk1[0] = -0.03846*Xk[0]+0.00964*Xk[1]+0.00017*Xk[2]  +0.012336*Uk;
    Xk1[1] = -23.334*Xk[0]+0.60013*Xk[1]+0.02117*Xk[2]  -5.7088*Uk;
    Xk1[2] = -143.102*Xk[0]-2.5838*Xk[1]+0.97479*Xk[2] +196.959*Uk;
    yk=Xk[0];
     }
  if (count==2)
          {
 ////Under bound parameter///////
   Xk1[0] = 0.26792*Xk[0]+0.01498*Xk[1]+0.00023*Xk[2]  +0.011348*Uk;
   Xk1[1] = -11.164*Xk[0]+0.8355*Xk[1]+0.02352*Xk[2]-1.0285*Uk;
   Xk1[2] = -37.838*Xk[0]-0.56906*Xk[1]+0.99483*Xk[2] +17.76*Uk;
   yk=Xk[0];
           }


    /////// Calculate error signals for training//////////////
         Sk=0;Sk1=0;
         for (i=0;i<(N-1);i++)
           {
             Sk=Sk+H[i]*(Xk[i]);
            Sk1=Sk1+H[i]*(Xk1[i]);
           }
        V=delta*Sk+Sk1;
        E=E+0.5*V*V;
//        printf("\nJ=%f",J);getch();
```

```
////////Present kth input pattern////////////
 for(i=0;i<(N-1);i++)
  {
  y0k[i]=-(Xk1[i])/DP[i];
  }
  y0k[N-1]=1;
  //////Calculate hidden layer input and output/////////
      for (i=0;i<(M-1);i++)
       {
       net1k[i]=0;
       for(j=0;j<N;j++)
         {
          net1k[i]=net1k[i]+W1k[i][j]*y0k[j];
         }
       y1k[i]=baf(net1k[i]);
      }
   y1k[M-1]=1;
////////Calculate output layer input and output///////
       net2k=0;
       for(j=0;j<M;j++)
         {
          net2k=net2k+W2k[j]*y1k[j];
         }
       y2k=purelin(net2k);

       Uk=y2k;
  ///// calculate the new weights of hidden-to-output layer/////

   if (V!=0)
   {
/*      // For back-propagation learning algorithm
       for (i=0;i<M;i++)
            W2k1[i]=W2k[i]-beta*V*y1k[i]/g0;
      for (i=0;i<(M);i++)
       for (j=0;j<N;j++)
         W1k1[i][j]=W1k[i][j]-beta*V*dbaf(net1k[i])*W2k[i]*y0k[j]/g0;
*/
            // For proposed exponential learning algorithm
      T1=0;
      for (i=0;i<M;i++)
        {
         T1=T1+y1k[i]*y1k[i];
        }
       T2=0;
       for (i=0;i<(M-1);i++)
        for (j=0;j<N;j++)
         {
            T2=T2+(dbaf(net1k[i])*W2k[i]*y0k[j])*(dbaf(net1k[i])*W2k[i]*y0k[j]);
         }
```

```
    if ((T1+T2)!=0)
        {
         for (i=0;i<M;i++)
          {
                W2k1[i]=W2k[i]-alpha*V*y1k[i]/(g0*(T1+T2));
          }

         for (i=0;i<(M-1);i++)
          for (j=0;j<N;j++)
           {
                W1k1[i][j]=W1k[i][j]-
alpha*V*dbaf(net1k[i])*W2k[i]*y0k[j]/(g0*(T1+T2));
           }

        } // of if T1+T2

    } // of if V

 ymax=6;xmax=STEP;
 VeHam(V,k,ymax,xmax);
        // Preparing for next step
   for(i=0;i<M;i++)
     W2k[i]=W2k1[i];
  for (i=0;i<(M-1);i++)
   for (j=0;j<N;j++)
     W1k[i][j]=W1k1[i][j];

 for (i=0;i<(N-1);i++)
   {
    Xk[i]=Xk1[i];
   }

 } // of FOR k

} // of For count
 l+=1;
 E=E/(3*STEP);
// ymax=60;xmax=Lm;
// VeHam(E,l,ymax,xmax);
// getch();
}
while ((E>=Em)&&(l<=Lm));

   for(i=0;i<M;i++)
     fprintf(fp1,"%0.9f\n",W2k[i]);
   for (i=0;i<(M-1);i++)
    {
     for (j=0;j<N;j++)
         fprintf(fp1,"%0.9f\t",W1k[i][j]);
```

```
      fprintf(fp1,"\n");
     }
  fprintf(fp1,"%d\t%d\n",dem,l-1);
  fprintf(fp1,"%f\n",E);
  fprintf(fp1,"\n");
  printf("\n%d\t%d\t%f",dem,l-1,E);

}          // For dem
fclose(fp1);


//  setcolor(RED);
//  gcvt(ymax, sig, str);
//  outtextxy(15,10,str);
   DongDoHoa();
printf("\n%d\t%d\t%f",dem,l-1,E);
printf("\nFine");
getch();

return;
 }

   /*-------------------------------------------------*/
         double sat(double x,double v)
         {
         double tg;
         if (x>v) tg=v;
                  else  if (x<-v)  tg=-v;
                      else   tg=x;
         return(tg);
         }

/*-----------------------------------*/
int  KDDoHoa(void)
{
   int   driver = DETECT,mode;
   int  error_code;

 initgraph(&driver,&mode,"c:\\tc\\bgi");
 error_code=graphresult();
 if (error_code!=grOk)
   return(-1);
   /*khong phai man hi`nh EGA hoac VGA */
 if ((driver!=EGA)&&(driver!=VGA))
   {
    closegraph();
    return 0;
   }
 return(1);
}
```

```
/*-----------------------------------*/
void  VeTruc(void)
{

  setbkcolor(CYAN);
  /* Mo cua so va` cho phep ve ca ngoai cua so  */
  Xm=getmaxx();Ym=getmaxy();
  setviewport(0,0,Xm,Ym,0);
  clearviewport();
  setcolor(LIGHTRED);
  outtextxy(15,Ym/2+10,"(0,0)");
  setcolor(RED);
  /* chuyen con tro ve goc va` ve truc X*/
  moveto(5,Ym/2);
  lineto(Xm-10,Ym/2);
  outtextxy((Xm-10),Ym/2,"x");


  /* chuyen con tro ve goc va` ve truc Y*/
  moveto(10,Ym-10);
  lineto(10,10);
  outtextxy(15,5,"y");

  setcolor(BLUE);
  settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
  outtextxy(200,Ym-40,"Do thi Y=F(x)");
  moveto(10,Ym/2);
  setcolor(YELLOW);

}
/*-----------------------------------*/
void  VeHam(float Un,int n,float ymax,float xmax)
{
  int x,y;

   x=10+(int)(n*TileX/xmax+0.5);
   y=Ym/2-(int)(Un*TileY/(ymax*2)+0.5);
   lineto(x,y);
   moveto(x,y);


}

int DongDoHoa(void)
{
   getch();
   closegraph();
   return(1);
}
/*-------------------------*/
```

```
double baf(double v)
{
 double tg;
 tg=-1;
 if (v>-709)
   tg=(1-exp(-v))/(1+exp(-v));
 return(tg);
}
/*---------------------------*/

double dbaf(double v)
{
 double tg,tg1;
 tg=0;
 if (v>-709)
  {
   tg1=exp(-v);
   tg=2*tg1/((1+tg1)*(1+tg1));
  }
 return(tg);
}
/*---------------------------*/

double purelin(double v)
{
 return(v);
}
```

APPENDIX D

D.1 Matlab Program for Design of The Neural Network Controller

File name: DiscreteStaticVarContr.m

```
function out=DiscreteStaticVarContr()
%----------Pole placement-------
% Determination of state feedback gain matrix K by use of transformation
% matrix T
% ** Enter matrices A and B**
% Note that X=[x3;x2;x1]
format short g
% Upper parameter system 2
td=0.07
a=0.55*[td*td/8 -td/2 1]
b=0.11*[td*td/8  +td/2 1]
c=[b(1) b(2) b(3) 0] + [0   td*td/8  +td/2 1]
numeratorGs=a
denominatorGs=c
pause;
[A2,B2,C2,D2]=tf2ss(numeratorGs,denominatorGs)
step(A2,B2,C2,D2)
N=[C2'  A2'*C2'  (A2')^2*C2'];
rank(N)
pause;
%**since the rank of N is 3, design of the observer is possible**
%% Obtain the coefficients of the characteristic polynamial |sI-A|. This
%% can be done by entering statement poly(A)
JA=poly(A2)
a1=JA(2);a2=JA(3);a3=JA(4);
%*** Define matries W and T as follows***
W=[a2 a1 1;a1 1 0;1 0 0]
Q=W*N'
% Transfer system into the observable canonical form
F=Q*A2*inv(Q)
G=Q*B2
H=C2*inv(Q)
J=D2
step(F,G,H,J)
pause;

A2=F;B2=G;C2=H;D2=J;
sysSSc=ss(A2,B2,C2,D2);
Ts=0.025;
sysSSd=c2d(sysSSc,Ts,'zoh');
[Phi2,Gam2,H2]=ssdata(sysSSd)
```

```
pause;
%under bound parameter system 3
td=0.12
a=0.29*[td*td/8 -td/2 1]
b=0.22*[td*td/8  +td/2 1]
c=[b(1) b(2) b(3) 0] + [0   td*td/8  +td/2 1]
numeratorGs=a
denominatorGs=c
pause;
[A3,B3,C3,D3]=tf2ss(numeratorGs,denominatorGs)
step(A3,B3,C3,D3)
N=[C3'  A3'*C3'  (A3')^2*C3'];
rank(N)
pause;
%**since the rank of N is 3, design of the observer is possible**
%% Obtain the coefficients of the characteristic polynamial |sI-A|. This
%% can be done by entering statement poly(A)
JA=poly(A3)
a1=JA(2);a2=JA(3);a3=JA(4);
%*** Define matries W and T as follows***
W=[a2 a1 1;a1 1 0;1 0 0]
Q=W*N'
% Transfer system into the observable canonical form
F=Q*A3*inv(Q)
G=Q*B3
H=C3*inv(Q)
J=D3
step(F,G,H,J)
pause;
A3=F;B3=G;C3=H;D3=J;
sysSSc=ss(A3,B3,C3,D3);
Ts=0.025;
sysSSd=c2d(sysSSc,Ts,'zoh');
[Phi3,Gam3,H3]=ssdata(sysSSd)

pause;
% Nominal parameter System 1
td=0.1
a=0.425*[td*td/8 -td/2 1]
b=0.165*[td*td/8  +td/2 1]
c=[b(1) b(2) b(3) 0] + [0   td*td/8  +td/2 1]
numeratorGs=a
denominatorGs=c
pause;
[A,B,C,D]=tf2ss(numeratorGs,denominatorGs)
step(A,B,C,D)

%*** Define the observability matrix N and check its rank***
N=[C'  A'*C'  (A')^2*C'];
```

```
rank(N)
pause;
%**since the rank of N is 3, design of the observer is possible**
%% Obtain the coefficients of the characteristic polynamial |sI-A|. This
%% can be done by entering statement poly(A)
JA=poly(A)
a1=JA(2);a2=JA(3);a3=JA(4);
%*** Define matries W and T as follows***
W=[a2 a1 1;a1 1 0;1 0 0]
Q=W*N'
% Transfer system into the observable canonical form
F=Q*A*inv(Q)
G=Q*B
H=C*inv(Q)
J=D
step(F,G,H,J)
pause;
A=[F(3,3) F(3,2) F(3,1)
  F(2,3) F(2,2) F(2,1)
  F(1,3) F(1,2) F(1,1)]
B=[G(3);G(2);G(1)]
C=[H(3) H(2) H(1)]
D=J
%A=F;B=G;C=H;D=J;
sysSSc=ss(A,B,C,D);
Ts=0.025;
sysSSd=c2d(sysSSc,Ts,'zoh');
[Phi,Gam,H]=ssdata(sysSSd)

pause;
%*** Define the controllability matrix M***
Md=[Gam  Phi*Gam  Phi^2*Gam];
%*** Check the rank of matrix M***
rank(Md)
pause;
%**since the rank of Md is 3, arbitrary pole placement is possible**
%% Obtain the coefficients of the characteristic polynamial |sI-A|. This
%% can be done by entering statement poly(A)
JA=poly(Phi)
a1=JA(2);a2=JA(3);a3=JA(4);
%*** Define matries W and T as follows***
W=[a2 a1 1;a1 1 0;1 0 0];
T=Md*W


tr=0.2;
ce=0.8;
M=exp(-pi*ce/sqrt(1-ce^2))
wn=2*1.8/tr
el=ce*wn
```

```
wd=wn*sqrt(1-ee^2)

P=[-el+wd*i;-el-wd*i;-2*wn]
%Pd=[exp(-el+wd*i);exp(-el-wd*i);exp(-el+wd*i);exp(-el-wd*i)]
Pd=exp(Ts*P)
Kd=acker(Phi,Gam,Pd)
%Design H of the sliding mode function
rh=[-el+wd*i;-el-wd*i]
rdh=exp(Ts*rh)
Hp=poly(rdh)
H1p=[Hp(3) Hp(2) Hp(1)]
H2p=H1p*inv(T)
HB=H2p*Gam
H3p=exp(Ts*(-2*wn))*H2p


% Calculate input gain for zero steady-state error to a step command at x1
I=eye(3);
G=[Phi-I  Gam;H  0]
N=inv(G)*[0;0;0;1]
Nu=N(4)
Nx=[N(1);N(2);N(3)]
Nr=Nu + Kd*Nx

pause;
%K=acker(A,B,P)
% Full order Observer
%*** Define the observability matrix N and check its rank***
N=[H'  Phi'*H'  (Phi')^2*H'];
rank(N)
pause;
%**since the rank of N is 3, design of the observer is possible**

% Using acker() command in Matlab
Pe=exp(Ts*[-5*el+5*wd*i;-5*el-5*wd*i;-5*wn])
Ke=acker(Phi',H',Pe)'
temp=[Phi-Ke*H]

pause;
```

BIBLIOGRAPHY

1988. *DARPA Neural Network Study*, MIT Lincoln Laboratory, Lexington, MA.

Albus, J.S. 1975, 'A new approach to manipulator control: The celebellar model articulation controller', *Trans. ASME, J. Dyn. Sys, Meas. Control*, vol. 97, pp. 220–227.

Anderson, J.A. 1972, 'A simple neural network generating an interactive memory', *Mathematical Biosciences*, vol. 14, pp. 197–220.

Baba, N. 1989, 'A New Approach for Finding the Global Minimum of Error Function of Neural Networks', *Neural Networks*, vol. 2, pp. 367–373.

Barnard, E. 1992, 'Optimization for Training Neural Nets', *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 232–240.

Barto, A.G. 1990, *Neural Networks for Control*, MIT Press, Cambridge, MA.

Basin, M., Gonzalez, J.R., Acosta, P. and Fridman, L. 2003, 'Robust Integral Sliding mode Regulator for Linear Systems with Time delay in Control Input', *Proc. American Control Conference, Denver, CO.*, pp. 2138–2143. .

Battiti, R. 1989, 'Accelerated backpropagation learning: Two optimization methods', *Complex Syst*, vol. 3, pp. 331–342.

Battiti, R. and Masulli, F. 1990, 'BFGS optimization methods for backpropagation: Automatic parameter tuning and faster convergence', *INNC 90, Paris, International Neural Network Conference*, vol. 2, pp. 757–760.

Baum, E.B. and Wilczek, F. 1988, 'Supervised learning of probability distributions by neural networks', in D.Z. Anderson (ed.), *Neural Information Processing Systems*, American Institute of Physics, New York.

Becker, S. and Le Cun, Y. 1989, 'Improving the convergence of back-propagation learning with second order methods', *Proc. the 1988 Connectionist Models Summer School*, ed. G.H. D. Touretzky, and T. Sejnowski, Eds., San Mateo, CA, pp. 29–37.

Bekiroglu, N. 1996, 'Adaptive sliding surface design for sliding mode control systems', PhD dissertation thesis, Bogazici University, Bogazici, Turkey.

Bianchini, M., Gori, M. and Maggini, M. 1994, 'On the Problem of Local Minima in Recurrent Neural Networks', *IEEE Trans. Neural Networks*, vol. 5, pp. 167–177.

Bishop, C.M. 1995, *Neural networks for pattern recognition*, Oxford University Press, New York.

Bonissone, P.P., Chen, Y.T., Goebel, K. and Khedkar, P.S. 1999, 'Hybrid Soft Computing Systems: Industrial and Commercial Applications', *Proc. of IEEE*, vol. 87, no. 9, pp. 1641–1667.

Breemen, A.J.N.V. and Veelenturf, L.P.J. 1996, 'Neural Adaptive Feedback Linearization Control', *Journal A*, vol. 37, pp. 65–71.

Broomhead, D.S. and Lowe, D. 1988, 'Multivariable functional interpolation and adaptive network', *Complex Syst*, vol. 2, pp. 321–355.

Brunelli, R. 1994, 'Training Neural Nets Through Stochastic Minimization', *Neural Networks*, vol. 7, no. 9, pp. 1405–1412.

Burrascano, P. 1991, 'A norm selection criterion for the generalized delta rule', *IEEE Trans. Neural Networks*, vol. 2, no. 1, pp. 125–130.

Cao, Y.Y. and Frank, P.M. 2000, 'Analysis and Synthesis of Nonlinear Time-Delay Systems Via Fuzzy Control Approach', *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 2, pp. 200–211.

Cetin, B.C., Burdick, J.W. and Barhen, J. 1993, 'Global descent replaces gradient descent to avoid local minima problem in learning with artificial neural networks', *IEEE Inter. Conf. Neural Networks*, vol. II, San Francisco, pp. 836–842.

Chen, S. and Billings, S.A. 1992, 'Neural networks for nonlinear dynamic system modelling and identification', *Int. J. Control*, vol. 56, no. 2, pp. 319–346.

Chen, V.C. and Pao, Y.H. 1989, 'Leaning control using neural networks', *Proc. Inter. Conference Robotics and Automation*, vol. 3, pp. 1448–1453.

Chiang, C.-C., Shen, K.-M. and Tsai, M.-Y. 2004, 'Robust fuzzy-model-based sliding mode controller for uncertain nonlinear input-delay systems', *Proc. 2004 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2243–2248.

Cichocki, A. and Unbehauen, R. 1993, *Neural Networks for Optimization and Signal Processing*, John Wiley and Sons, New York.

Coleman, C.P. and Godbole, D. 1994, 'A comparison of Robustness: Fuzzy, PID, & Sliding Mode Control', *Proc. of the 1994 IEEE 3rd International Fuzzy Systems Conference, Orlando, Florida*, pp. 1654–1659.

Coyle, E.D. 1995, 'Electronic Wheelchair Controller Designed for Operating by Hand-Operated Joystick, Ultrasonic Non-Contact Head Control and Utterance from a Small Word-Command Vocabulary', *IEE Colloquium (Digest)*, vol. 55, pp. 3/1–3/4.

Crisman, E.E., Loomis, A., Shaw, R. and Laszewski, Z. 1991, 'Using the Eye Wink Control Interface to Control a Powered Wheelchair', *Proc. the Annual International Conference of the IEEE Engineering in Medicine and Biology Society* vol. 13, pp. 1821–1822.

Cybenko, G. 1989, 'Approximations by superpositions of sigmoidal functions', *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314.

Da, F. 2000, 'Decentralized Sliding Mode Adaptive Controller Design Based on Fuzzy Neural Networks for Interconnected Uncertain Nonlinear Systems', *IEEE Trans. Neural Networks*, vol. 11, no. 2, pp. 1471–1480.

Da, F. and Song, W. 2003, 'Fuzzy Neural Networks for Direct Adaptive Control', *IEEE Trans. Ind. Electron.*, vol. 50, no. 3, pp. 507–513.

Daosud, W., Thitiyasook, P., Arpornwichanop, A., Kitisupakorn, P. and Hussain, M.A. 2005, 'Neural network inverse model-based controller for the control of a steel pickling process', *Computers and Chemical Engineering*, vol. 29, pp. 2110–2119.

Efe, M.O. and Kaynak, O. 2000, 'Stabilizing and robustifying the learning mechanisms of artificial neural networks in control engineering applications', *Int. J. Intell. Syst.*, vol. 15, no. 5, pp. 365–388.

Efe, M.O., Kaynak, O. and M.Wilamowski, B. 2000, 'Stable training of computationally intelligent systems by using variable structure systems technique', *IEEE Trans. Ind. Electron.*, vol. 47, pp. 487–496.

Efe, M.O., Kaynak, O., Wilamowski, B.M. and Yu, X. 2003, 'A Robust On-line Leaning Algorithm for Intelligent Control Systems', *Inter. J. Adapt. Control and Signal Processing*, vol. 17, pp. 489–500.

Emelyanov, S. 1967, *Variable Structure Control Systems*, Nauka, Moscow.

Emelyanov, S.V. 1959, 'Control of first order delay systems by means of an astatic controller and nonlinear correction', *Autom. Remote Control*, no. 8, pp. 983–991.

Ertugrul, M., Kaynak, O. and Sabanovic, A. 1995, 'A comparison of various VSS techniques on the control of automated guided vehicles', *Proc. IEEE ISIE'95*, vol. 2, pp. 837–842.

Fahlman, S.E. 1988, 'Fast learning variations on backpropagation: An empirical study', *Proc. the 1988 Connectionist Models Summer School*, ed. G.H. D. Touretzky, and T. Sejnowski, Eds., Pittsburgh, San Mateo, CA, pp. 38–51.

Finnoff, W., Hergert, F. and Zimmermann, H.G. 1993, 'Improving Model Selection by Nonconvergent Methods', *Neural Networks*, vol. 6, pp. 771–783.

Fletcher, R. 1987, *Practical Methods of Optimization*, Wiley, Chichester, U. K.

Franklin, G.F., Powell, J.D. and A.Emami-naeini 2002, *Feedback control of dynamic system*, Prentice Hall, New Jersey.

Funahashi, K. 1989, 'On the Approximation Realization of Continuous Mappings by Neural Networks', *Neural Networks*, vol. 2, no. 3, pp. 183–192.

Gao, W. and Hung, J.C. 1993, 'Variable structure control of non-linear systems: a new approach', *IEEE Trans. Ind. Electron.*, vol. 40, pp. 45–55.

Gavel, D. and Siljak, D. 1989, ' Decentralized adaptive control: structural conditions for stability', *IEEE Transactions of Automtic Control*, vol. 34, no. 4, pp. 413–426.

Giordano, V., Topalov, A.V., Kaynak, O. and Turchiano, B. 2004, 'Sliding-mode approach for on-line neural identification of robotic manipulators', *Pro. the 5th Asian Control Conference, Melbourne, Australian, 2004* pp. 2070–2076.

Grossberg, S. 1976, 'Adaptive pattern classification and universal recording: I. Parallel development and coding of neural feature detectors', *Biological Cybernetics*, vol. 23, pp. 121–134.

Hagan, M. and Demuth, H. 1999, 'Neural Networks for Control', *American Control Conference, June, 1999, San Diego*, pp. 1642–1656.

Hagan, M.T., Demuth, H.B. and Beale, M. 1995, *Neural network design*, PWS Publisher, Boston.

Hagan, M.T., Demuth, H.B. and Jesus, O.D. 2002, 'An introduction to the use of neural networks in control systems', *International Journal of Robust and Nonlinear Control*, vol. 12, pp. 959–985.

Hagan, M.T. and Menhaj, M.B. 1994, 'Training feedforward networks with the Marquardt algorithm', *IEEE Trans. Neural Networks*, vol. 5, pp. 989–993.

Hanson, S.J. and Burr, D.J. 1988, 'Minkowski-r back-propagation: Learning in connectionist models with non-Euclidean error signals', in D.Z. Anderson (ed.), *Neural Information Processing Systems*, American Institute of Physics, New York, pp. 348–357.

Hayakawa, T., Haddad, W.M., Bailey, J.M. and Hovakimyan, N. 2005, 'Passivity-Based Neural Network Adaptive Output Feedback Control for Nonlinear Nonnegative Dynamical Systems', *IEEE Trans. Neural Networks*, vol. 16, no. 2, pp. 387–398.

Haykin, S. 1995, *Neural Networks: A Comprehensive Foundation*, vol. 2, Prentice-Hall, New York.

He, S. 2002, 'Neural Adaptive Control of Nonlinear Multivariable System with Application to a Class of Inverted Pendulums', *Inter. J. Neural Syst.*, vol. 12, no. 5, pp. 411–424.

Hebb, D.O. 1949, *The Organization of Behaviour*, Wiley, New York.

Hert, J., Krogh, A. and Palmer, R.G. 1991, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood, CA.

Hinton, G.E. and Sejnowski, T.J. 1986, *Learning and Relearning in Boltzmann machines*, MIT Press, Cambridge, MA.

Hopfield, J.J. 1982, 'Neural networks and physical systems with emergent collective computational abilities', *Proc. the National Academy of Sciences*, vol. 79, pp. 2554–2558.

Hornik, K., Stinchcombe, M. and White, H. 1989, 'Multilayer feedforward networks are universal approximators', *Neural Networks*, vol. 2, pp. 359–366.

Hovakimyan, N., Nardi, F., Calise, A. and Kim, N. 2002, 'Adaptive Output Feedback Control of Uncertain Nonlinear Systems Using Single-Hidden-Layer Neural Networks', *IEEE Trans. Neural Networks*, vol. 13, no. 6, pp. 1420–1431.

Hung, J.Y., Gao, W. and Hung, J.C. 1993, 'Variable Structure Control: A Survey', *IEEE Trans. Ind. Electron.*, vol. 40, no. 1, pp. 2–22.

Hunt, K.J. and Sbarbaro, D. 1991, 'Neural networks for nonlinear internal model control', *IEE Proc. Contr. Theory and Applicat.*, vol. 138, pp. 431–438.

Ichikawa, Y. and Sawa, T. 1992, 'Neural Network Application for Direct Feedback Controllers', *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 224–231.

Ikeda, M. and Siljak, D.D. 1980, 'On decentralized stabilizable large-scale systems', *Automatica*, vol. 16, pp. 331–334.

Ioannou, P. 1985, 'Decentralized Adaptive Control of Interconnected Systems', *IEEE Trans. Automat. Contr.*, vol. 31, no. 4, pp. 291–298.

Isidori, A. and Astolfi, A. 1992, 'Disturbance Attenuation and H∞-Control Via Measurement Feedback in Nonlinear Systems', *IEEE Trans. Automat. Contr.*, vol. 37, no. 9, pp. 1283–1293.

Itkis, U. 1976, *Control System of Variable Structure.*, Wiley, New York.

Jacobs, R.A. 1988, 'Increased rates of convergence through learning rate adaptation', *Neural Networks*, vol. 1, no. 4, pp. 295–308.

Jagannathan, S. 2001, 'Control of a Class of Nonlinear Discrete-Time Systems Using Multilayer Neural Networks', *IEEE Trans. Neural Networks*, vol. 12, no. 5, pp. 1113–1120.

Jain, S. and Khorrami, F. 1997, 'Decentralized Adaptive Output Feedback design for large-Scale Nonlinear Systems', *IEEE Trans. Automat. Contr.*, vol. 42, no. 5, pp. 729–735.

Jervis, T.T. and Fitzgerald, W.J. 1993, *Optimization schemes for neural networks*, Trumpington Street, Cambridge.

Jin, L., Nikiforuk, P.N. and Gupta, M.M. 1993, 'Direct Adaptive Output Tracking Control Using Multilayer neural Networks', *IEE Proc. D*, vol. 40, no. 6, pp. 393–398.

Jordan, M.I. and Rumelhart, D.E. 1992, 'Forward models: supervised learning with a distal teacher', *Cognitive Science*, vol. 16, pp. 307–354.

Joseph, T. and Nguyen, H. 1998, 'Neural network control of wheelchairs using telemetric head movement', *Proc.the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 5, pp. 2731–2733.

Kawato, M., Furukawa, K. and Suzuki, R. 1987, 'A hierarchical neural network model for control and learning of voluntary movement', *Biological Cybernetics*, vol. 57, pp. 169–185.

Kaynak, O., Erbatur, K. and Ertugrul, M. 2001, 'The Fusion of Computationally Intelligent Methodologies and Sliding-Mode Control—A Survey', *IEEE Trans. Ind. Electron.*, vol. 48, no. 1, pp. 4–17.

Khalil, H. 2002, *Nonlinear System*, Prentice Hall, Upper Saddle River, NJ

Kocijan, J., O'Reilly, J. and Leithead, W.E. 1997, 'An Integrated Undergraduate Teaching laboratory Approach to Multivariable Control', *IEEE TRans. Education*, vol. 40, no. 4, pp. 266–272.

Kohonen, T. 1972, 'Correlation matrix memories', *IEEE Trans. on Computers*, vol. 21, pp. 353–359.

Kosko, B. 1987, 'Adaptive bi-directional associative memories', *Appl. Opt.*, vol. 26, pp. 4947–4960.

Lee, T.-H., Lim, K.-W. and Lai, W.-C. 1991, 'Real-time Multivariable Self-Tuning Controller Using a Feedforward Paradigm with Application to a Coupled Electric-Drive Pilot Plant', *IEEE Trans. Ind. Electron.* , vol. 38, no. 4, pp. 137–242.

Lee, Y., Oh, S.H. and Kim, M.W. 1993, 'An analysis of premature saturation in back propagation learning', *Neural Networks*, vol. 6, pp. 719–728.

Lockey, B. and Philpott, G. 2002, 'Static Var Compensators: A Solution to the Big Motor/Weak System Problem?' *IEEE Industry Applications Magazine*, vol. 8, no. 2, pp. 43–49.

Makram-Ebeid, S., Sirat, J.A. and Viala, J.R. 1989, 'A Rationalized Error Back-Propagation Learnign Algorithm', *Proc. International Joint Conference on Neural Networks*, vol. 2, pp. 373–380.

Matyas, J. 1965, 'Random optimization', *Automation and Remote Control*, vol. 26, pp. 246–253.

Mazo, M., Rodriguez, F.J., Lazoro, J.L., Urena, J., Garcia, J.C., Santiso, E. and Revenga, P.A. 1995, 'Electronic Control of a Wheelchair Guided by Voice Commands', *Control Eng. Practice*, vol. 3, pp. 665–674.

McCulloch, W. and Pitts, W. 1943, 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133.

Meireles, M.R.G., Almeida, P.E.M. and Simões, M.G. 2003, 'A Comprehensive Review for Industrial Applicability of Artificial Neural Networks', *IEEE Trans. Ind. Electron.*, vol. 50, no. 3, pp. 585–601.

Miller, W.T., Sutton, R.S. and Werbos, P.J. 1990, *Neural network for control*, MIT Press, Cambridge, MA.

Minsky, M. and Papert, S. 1969, *Perceptrons*, MIT Press, Cambridge, MA.

Moller, M.F. 1993, 'A scaled conjugate gradient algorithm for fast supervised learning', *Neural Networks*, vol. 6, no. 525–533.

Moody, J. 1989, *Fast-learning in multi-resolution hierarchies*, vol. 1, Morgan Kaufman, San Mateo, CA.

Narendra, K.S. and Parthasarathy, K. 1990, 'Identification and Control of Dynamical Systems Using Neural Networks', *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4–27.

Nayeri, M.R.D., Alasty, A. and Daneshjou, K. 2004, 'Neural optimal control of flexible spacecraft slew maneuver', *Acta Astronautica*, vol. 44, pp. 817–827.

Ng, G.W. 1997, *Application of Neural Networks to Adaptive Control of Nonlinear Systems*, J. Wiley, New York.

Ng, S.-C., Cheung, C.-C. and Leung, S.-H. 2004, 'Magnified Gradient Function With Deterministic Weight Modification in Adaptive Learning', *IEEE Trans. Neural Networks*, vol. 15, no. 6, pp. 1411–1423.

Nguyen, D.-k. 1998, 'Sliding-mode control: advanced design techniques', University of Technology, Sydney, Sydney, Australia.

Nguyen, D.H. and Widrow, B. 1990, 'Neural Networks for Self-Learning Control Systems', *IEEE Control Syst. Mag.*, vol. 10, pp. 18–23.

Nguyen, H.T. 2004, *Analogue and Digital Control*, Faculty of Engineering, University of Technology, Sydney.

Nguyen, H.T., King, L.M. and Knight, G. 2004a, 'Real-time Head Movement System and Embedded Linux Implementation for the Control of Power Wheelchair', *Proc.the 26th Annual International Conference of the IEEE EMBS, San Francisco, CA, USA*, pp. 4892–4895.

Nguyen, T.V.M. 2005, 'Neural Network-based Sliding Mode Control for Linear Systems with Unmatched Time-varying Uncertainties', *2005 Research Showcase*, Faculty of Engineering, UTS.

Nguyen, T.V.M., Ha, Q.P. and Nguyen, H.T. 2003, 'A Chattering-Free Variable Structure Controller for Tracking of Robotic Manipulators', *Proc. of the Australian Conference on Robotics and Automation (ACRA 2003), Brisbane, Australia.*, no. 2, pp. 1–6.

Nguyen, T.V.M., Nguyen, H.T. and Ha, Q.P. 2004b, 'Sliding Mode Neural Controller for Nonlinear Systems with Higher-Order and Uncertainties', *Proc. IEEE CIS'04 & RAM'04*, pp. 1026–1031.

Norgaard, M. 1996, 'System Identification and Control with Neural Networks', Ph.D. thesis thesis, Technical University of Denmark.

Ooyen, A.V. and Nienhuis, B. 1992, 'Improving the convergence of the backpropagation algorithm', *Neural Networks*, vol. 5, pp. 465–471.

Ossman, K.A. 1989, 'Indirect adaptive control for interconnected systems', *IEEE Trans. Automt. Contr.*, vol. 34, no. 8, pp. 908–911.

Paker, D. 1982, *Learning logic*, Department of Electrical Engineering, Stanford University, Stanford, CA.

Parlos, A.G., Fernandez, B., Atiya, A.F., Muthusami, J. and Tsai, W.K. 1994, 'An accelerated leaning algorithm for multilayer perceptron networks', *IEEE Trans. Neural Networks*, vol. 5, no. 3, pp. 493–397.

Parma, G.G., Menezes, B.R.D. and Braga, A.P. 1999, 'Neural Networks Learning with Sliding Mode Control: The Sliding Mode Backpropagation Algorithm.' *International Journal of Neural Systems*, vol. 9, no. 3, pp. 187–193.

Plaut, D.S., Nowlan, S. and Hinton, G. 1986, *Experiments on Learning by Back Propagation*, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

Poggio, T. and Girosi, F. 1990, 'Networks for approximation and learning', *Proc.the IEEE*, vol. 78, pp. 1481–1497.

Psaltis, D., Sideris, A. and Yamamura, A.A. 1988, 'A multilayer neural network controller', *IEEE Control Syst. Mag.*, vol. 8, no. 1, pp. 17–21.

Riedmiller, M. and Braun, H. 1993, 'A direct adaptive method for faster backpropagation learning: The RPROP algorithm', *in Proc. Int. Conf. Neural Networks*, vol. 1, pp. 586–591.

Rockland, R.H. and Reisman, S. 1998, 'Voice activated wheelchair controller', *Proc.the IEEE 24th Annual Northeast Bioengineering Conference*, pp. 128–129.

Rosenblatt, F. 1958, 'The perceptron: A Probabilistic model for information storage and organization in the brain', *Psychological Review*, vol. 65, pp. 386–408.

Rumelhart, D.E. and McClelland, J.L. 1986, *Parallel Distributed Processing:" explorations in the microstructure of cognition*, vol. 1, MIT Press, London.

Saerens, M. and Soquet, A. 1991, 'Neural controller based on back-propagation algorithm', *IEE Proc.Radar and Signal Processing*, vol. 138, no. 1, pp. 55–62

Salas, S.L. and Hill, E. 1990, *Calculus: one and several variables*, Wiley, New York.

Salomon, R. 1998, 'Evolutionary Algorithms and Gradient Search: Similarities and Differences', *IEEE Trans. Evolutionary Computation*, vol. 2, no. 2, pp. 45–55.

Sanner, R.M. and Slotine, J.-J.E. 1992, 'Gaussian Networks for Direct Adaptive Control', *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 837–863.

Schimeisser, G. and Seamone, W. 1979, 'An assitive equipment controller for quadriplegics', *Johns Hopkins Med. Journal*, vol. 143, no. 3, pp. 84–88.

Seshagiri, S. and Khalil, H.K. 2005, 'Robust Output Feedback Regulation of Minimum-phase Nonlinear Systems Using Conditional Integrators', *Automatica*, vol. 41, pp. 43–54.

Shi, L. and Singh, S.K. 1992, 'Decentralized adaptive controller design for large-scale systems with high order interconnections', *IEEE Trans. Automat. Contr.*, vol. 37, no. 8, pp. 1106–1118.

Sira-Ramirez, H. and Morles, E.C. 1995, 'A sliding mode strategy for adaptive learning in adalines', *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 1001–1012.

Slotine, J.J. and Sastry, S.S. 1983, 'Tracking control of nonlinear systems using sliding surfaces with application to robot manipulators', *Int. J. Control*, vol. 38, pp. 465–492.

Solis, F.J. and Wets, J.B. 1981, 'Minimization by random search techniques', *Mathematics of Operations Research*, vol. 6, pp. 19–30.

Soloway, D. and Haley, P.J. 1996, 'Neural generalized predictive control', *Proc. 1996 IEEE International Symposium on Intelligent Control*, pp. 277–281.

Song, Q. 1998, 'Robust training algorithm of multilayered neural network for identification of nonlinear dynamic systems', *IEE Proc. Control Theory and Applications – Part D*, vol. 145, no. 1, pp. 41–46.

Song, Q., Xiao, J. and Soh, Y.C. 1999, 'Robust Backpropagation Training Algorithm for Multilayer Neural Tracking Controller', *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 1133–1141.

Spooner, J.T. and Passino, K.M. 1999, 'Decentralized adaptive control of nonlinear systems using radial basis neural networks', *IEEE Trans. Automt. Contr.*, vol. 44, no. 11, pp. 2050–2057.

Sutton, R.S. and Barto, A.G. 1998, *Introduction to reinforcement learning*, MIT Press, Cambridge, MA.

Szu, H. 1987, 'Nonconvex optimization by fast simulated annealing', *Proc. IEEE*, vol. 75, pp. 1538–1540.

Tanaka, K. 1996, 'An Approach to Stability Criteria of Neural Network Control Systems', *IEEE Trans. Neural Networks*, vol. 7, no. 3, pp. 629–642.

Tang, Z. and Koehler, G.J. 1994, 'Deterministic Global Optimal FNN Training Algorithms', *Neural Networks*, vol. 7, no. 2, pp. 301–311.

Taylor, P.B. and Nguyen, H.T. 2003, 'Performance of a head-movement interface for wheelchair control', *Proc.the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2, pp. 1590–1593.

Tew, A.I. 1988, 'The Oxford Optical Pointer: A Direction-sensing Device with Proportional Electric Output', *Med. & Biol. Eng. & Comp.*, vol. 26, pp. 68–74.

Tollenaere, T. 1990, 'SuperSab: fast adaptive backpropagation with good scaling properties', *Neural Networks*, vol. 3, no. 5, pp. 561–573.

Treadgold, N.K. and Gedeon, T.D. 1998, 'Simulated annealing and weight decay in adaptive learning: The SARPROP algorithm', *IEEE Trans. Neural Networks*, vol. 9, pp. 662–668.

Tsai, C.-H., Chung, H.-Y. and Yu, F.-M. 2004, 'Neuro-Sliding Mode Control With Its Applications to Seesaw Systems', *IEEE Trans. Neural Networks*, vol. 15, no. 10, pp. 124–134.

Utkin, V. and Shi, J. 1996, 'Integral Sliding Mode in Systems Operating under Uncertainty Condition', *Proc.the 35th Conference on Decision and Control, Kobe, Japan*, pp. 4592–4596.

Utkin, V.I. 1977, 'Survey Paper - Variable Structure System with Sliding Modes.' *IEEE Trans. Automat. Contr.*, vol. AC-22, no. 2, pp. 212–222.

Varshney, K. and Panigrahi, P.K. 2005, 'Artificial neural network control of a heat exchanger in a closed flow air circuit', *Applied Soft Computing*, vol. 5, pp. 441–465.

Venugopal, K.P., Sudhakar, R. and Pandya, A.S. 1995, 'An Improved Scheme for Direct Adaptive Control of Dynamical Systems using Backpropagation Neural Networks', *Circuits, System and Signal Process*, vol. 14, no. 2, pp. 213–236.

Vitela, J.E. and Reifman, J. 1997, 'Premature saturation in backpropagation networks: Mechanism and necessary conditions', *Neural Networks*, vol. 10, no. 4, pp. 721–735.

Vogl, T.P., Mangis, J.K., Zigler, A.K., Zink, W.T. and Alkon, D.L. 1988, 'Accelerating the convergence of the backpropagation method', *Biological Cybernetics*, vol. 59, pp. 256–264.

Wang, W.J. and Cheng, C.F. 1992, 'Stabilising controller and observer synthesis for uncertain large-scale systems by the Riccati equation approach', *IEE Proc.* , vol. 139, pp. 72–78.

Weigand, A.S., Huberman, B.A. and Rumelhart, D.E. 1990, 'Predicting the future: a connectionist approach', *Int. J. Neural systems*, vol. 1, no. 3, pp. 193–209.

Weir, M.K. 1991, 'A method for self-determination of adaptive learning rates in back propagation', *Neural Networks*, vol. 4, no. 3, pp. 371–379

Wellstead, P.E. 1979, *Introduction to physical system modelling*, Control System Centre, University of Manchester, Academic Press, Manchester, UK.

Wen, C. and Soh, Y.C. 1997, 'Decentralized Adaptive Control using Integrator Backstepping', *Automatica*, vol. 33, no. 9, pp. 1719–1724.

Werbos, P.J. 1974, 'Beyond regression: New tools for prediction and analysis in the behavioural sciences', Ph. D dissertation thesis, Harvard Uni., Cambridge, MA.

Wessels, L.F.A. and Barnard, E. 1992, 'Avoiding false local minima by proper initialization of connections ', *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 899–905.

White, H. 1989, 'Learning in artificial neural networks: A statistical perspective', *Neural Computation*, vol. 1, pp. 425–464.

Whitley, D., Starkweather, T. and Bogart, C. 1990, 'Genetic algorithms and neural networks: Optimizing connections and connectivity', *Parallel Computing*, vol. 14, pp. 347–361.

Widrow, B. and Hoff, M.E. 1960, 'Adaptive Switching Circuit', *1960 IRE WESCON Convention Record, New York: IRE Part 4*, pp. 96–104.

Zak, S.H. and Sira-Ramirez, H. 1990, 'On the adaptation algorithms for feedforward neural networks', in D.A.F.a.V. Komkov (ed.), *Theoretical Aspects of Industrial Design*, SIAM, New York, pp. 116–133.

Zhang, Q.G. and Benveniste, A. 1992, 'Wavelet Networks', *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 889–898.

Zhang, Y., Sen, P. and Hern, G.E. 1995, 'An on-line trained adaptive neural controller', *IEEE Control Syst. Mag.*, vol. 15, no. 5, pp. 67–75.