

Development of Neural Units with Higher-Order Synaptic Operations and their Applications to Logic Circuits and Control Problems

A Thesis

Submitted to the College of Graduate Studies and Research

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

In the

Department of Mechanical Engineering

and

Intelligent Systems Research Laboratory

University of Saskatchewan

By

Sanjeeva Kumar Redlapalli

Saskatoon, Saskatchewan, Canada

COPYRIGHT

The author has agreed that the Library, University of Saskatchewan, may make this thesis freely available for inspection. Moreover, I further agree that permission for copying this thesis in any manner, in whole or in part, for scholarly purposes. This may be granted either by the supervisor or the professors who supervised this thesis work or, in their absence, by the Head of the Department or the Dean of the College in which the thesis work was done. It is understood that due recognition will be given to the author for this thesis and to the University of Saskatchewan in any use of the material in this thesis. Copying or publication or any other use of the thesis for financial gain without approval by the University of Saskatchewan and the author's written permission is prohibited.

Requests for permission to copy or to make any other use of material in this thesis in whole or in part should be addressed to:

Head of the Department of Mechanical Engineering
University of Saskatchewan,
College of Engineering,
Saskatoon, Saskatchewan,
Canada, S7N 5A9.

ACKNOWLEDGMENTS

I wish to express my sincere thanks to Dr. Madan M. Gupta for his valuable guidance throughout this work. His encouragement and positive criticism have been mainly responsible for the success of this project. He has spent an enormous amount of time having many enlightening discussions with me on the project as well as guiding me in the preparation of this thesis and other materials for publication. His advice and help are greatly appreciated.

I would also like to express my sincere thanks to the advisory committee members, Professor Chris Zhang, Professor Allan Dolovich and Professor Saeid Habibi for their assistance.

I would like to extend my sincere appreciation for the visiting professor Dr. Zeng Guang Hou for his advice and assistance in the preparation of this thesis. I would also like to thank an exchange student at ISRL from Prague, Mr. Ivo Bukovsky, who is a Ph.D. student in the Department of Instrumentation and Control Engineering, Division of Automatic Control and Engineering Informatics, Czech Technical University in Prague, for his valuable suggestions on simulations.

Financial assistance provided partially by the Natural Sciences and Engineering Research Council (NSERC) is greatly acknowledged. I am also grateful to my parents for their financial support, during the initial and final phase of the project, without which this work would not have been possible.

I would like to take this opportunity to express my thanks to my friends who always provided constructive criticism and moral support throughout this work.

Sanjeeva Kumar Redlapalli.

DEDICATION

Dedicated to my grand mother and my beloved parents

Smt. Redlapalli Sulochana Devi and Sri. Redlapalli Anjaneyulu Setty

&

my family members

For their continuous love and unwavering support throughout this work

ABSTRACT

Neural networks play an important role in the execution of goal-oriented paradigms. They offer flexibility, adaptability and versatility, so that a variety of approaches may be used to meet a specific goal, depending upon the circumstances and the requirements of the design specifications. Development of higher-order neural units with higher-order synaptic operations will open a new window for some complex problems such as control of aerospace vehicles, pattern recognition, and image processing.

The neural models described in this thesis consider the behavior of a single neuron as the basic computing unit in neural information processing operations. Each computing unit in the network is based on the concept of an idealized neuron in the central nervous system (CNS). Most recent mathematical models and their architectures for neuro-control systems have generated many theoretical and industrial interests. Recent advances in static and dynamic neural networks have created a profound impact in the field of neuro-control.

Neural networks consisting of several layers of neurons, with linear synaptic operation, have been extensively used in different applications such as pattern recognition, system identification and control of complex systems such as flexible structures, and intelligent robotic systems. The conventional linear neural models are highly simplified models of the biological neuron. Using this model, many neural morphologies, usually referred to as multilayer feedforward neural networks (MFNNs), have been reported in the literature. The performance of the neurons is greatly affected when a layer of neurons are implemented for system identification, pattern recognition and control problems. Through simulation studies of the **XOR** logic it was concluded that the neurons with linear synaptic operation are limited to only linearly separable forms of pattern distribution. However, they perform a variety of complex mathematical operations when they are implemented in the form of a network structure. These networks suffer from various limitations such as computational efficiency and learning capabilities and moreover, these models ignore many salient features of the biological neurons such as

time delays, cross and self correlations, and feedback paths which are otherwise very important in the neural activity.

In this thesis an effort is made to develop new mathematical models of neurons that belong to the class of higher-order neural units (HONUs) with higher-order synaptic operations such as quadratic and cubic synaptic operations. The advantage of using this type of neural unit is associated with performance of the neurons but the performance comes at the cost of exponential increase in parameters that hinders the speed of the training process.

In this context, a novel method of representation of weight parameters without sacrificing the neural performance has been introduced. A generalised representation of the higher-order synaptic operation for these neural structures was proposed. It was shown that many existing neural structures can be derived from this generalized representation of the higher-order synaptic operation. In the late 1960's, McCulloch and Pitts modeled the stimulation-response of the primitive neuron using the threshold logic. Since then, it has become a practice to implement the logic circuits using neural structures. In this research, realization of the logic circuits such as **OR**, **AND**, and **XOR** were implemented using the proposed neural structures. These neural structures were also implemented as neuro-controllers for the control problems such as satellite attitude control and model reference adaptive control. A comparative study of the performance of these neural structures compared to that of the conventional linear controllers has been presented. The simulation results obtained in this research were applicable only for the simplified model presented in the simulation studies.

TABLE OF CONTENTS

COPYRIGHT	i
ACKNOWLEDGEMENTS	ii
DEDICATION	iii
ABSTRACT	iv
TABLE OF CONTENTS	vi
LIST OF FIGURES	ix
LIST OF TABLES	xiv

CHAPTER 1: Introduction

1.1 Biological Motivation	1
1.2 Neural Networks	3
1.2.1 Biological Neuronal Morphology	4
1.2.2 Neuron: The Basic Unit of the CNS	6
1.3 Thesis Objectives	8
1.4 Thesis Outline	10
1.5 Conclusion	11

CHAPTER 2: Neural Units with Linear Synaptic Operation (LSO)

2.1 Introduction	12
2.2 A Brief Description of the Neural Unit with LSO	12
2.2.1 Linear Classifier	15
2.3 Neural Models for Threshold Logic	17
2.3.1 Neural Model for XOR Logic Circuit	19
2.3.2 Simulation Studies for the XOR Logic Circuit with Neural Units with LSO	23
2.4 Neural Logic for XOR -Operation using Polynomial Discriminant Function	25
2.5 Application of Neural Units with LSO for Control Problems	27
2.6 Summary	28

CHAPTER 3: Development of Higher-Order neural Units with Quadratic and Cubic Synaptic Operation

3.1 Introduction	30
------------------	----

3.2 Development of Neural Unit with Quadratic Synaptic Operation (QSO)	31
3.2.1 Mathematical Model of the Neural Unit with Quadratic Synaptic Operation (QSO)	34
3.2.2 Learning and Adaptation Algorithm for the Neural Unit with Quadratic Synaptic Operation (QSO)	37
3.3 Development of Neural Unit with Cubic Synaptic Operation (CSO)	42
3.3.1 Structure and Mathematical Details of Neural Unit with CSO	42
3.4 General Methodology to Develop HONUs with Higher-Order Synaptic Operation	44
3.5 Summary	48

CHAPTER 4: Applications of Higher-Order Neural Units to Static Problems: Pattern Classification and Function Approximation

4.1 Introduction: Biological Motivation	50
4.2 Structure of the Neural Unit with QSO for Realizing the Logic Circuits	51
4.3 Learning Algorithm for Realizing the Logic Circuits	53
4.4 Realization of Logic Circuits using Neural Units with Quadratic Synaptic Operation (QSO)	55
4.4.1 Realization of XOR (Exclusive-OR) using a Neural Unit with Quadratic Synaptic Operation (QSO)	55
4.4.2 Realization of OR and AND Logic Circuits using a Neural Unit with QSO	62
4.5 How does the Neural Unit with QSO provide a Better Solution than the Neural Unit with LSO?	65
4.6 Mahalanobis Distance	68
4.7 Modified Mahalanobis Distance (MM-Distance)	69
4.8 Analysis of the Simulation Results	73
4.8.1 Exclusive-OR (XOR) Logic	73
4.8.2 OR Logic	75
4.8.3 AND Logic	77
4.9 Function Approximation	81
4.9.1 Simulation Studies	81

4.10 Conclusions	86
------------------	----

CHAPTER 5: Applications of Higher-Order Neural Units with Higher-Order Neural Synaptic Operations for Control Problems

5.1 Introduction: A Brief Review	88
5.2 HONUs for Control of the Linear Systems	89
5.2.1 Neural unit with QSO as a Neuro-Controller for Satellite Control Attitude Problem	90
5.2.2 Neural unit with CSO as a Neuro-Controller for Satellite Control Attitude Problem	92
5.3 Simulation Results	96
5.3.1 Simulation Results for Satellite Control (Linear Model)	98
5.3.2 Simulation Results for Satellite Control (Nonlinear Model)	102
5.4 HONUs for Control of Nonlinear Systems	104
5.5 Stability Analysis of Nonlinear Systems	106
5.5.1 Energy Method: Motion in a Potential Field	107
5.5.2 Lyapunov Function	109
5.5.3 Concept of Nullclines	111
5.6 Neural Nonlinear Controller	113
5.6.1 Development of New Damping Function	114
5.7 Non-Linear State Feedback Neuro-Controller for Control of unknown, varying Parameter and Structure, Nonlinear Dynamic System	117
5.8 Simulation Results	121
5.9 Conclusions	123

CHAPTER 6: Conclusions

6.1 Concluding Remarks	124
6.2 Contributions of the Thesis	126
6.3 Future Scope of the Research	129

LIST OF FIGURES

Figure 1.1 Layers of biological neurons arranged in a network depicting the flow of neuronal information in the forward direction as well as through inter and intra feedback direction.	5
Figure 1.2 A general mathematical model of the neuron with synaptic and somatic operation. The confluence operation compares the neural information with the past experience stored in the synaptic weights and the nonlinear activation function provides the bounded neural output.	6
Figure 1.3 Different activation functions used in the mathematical model of the biological neuron for bipolar input signals.	7
Figure 2.1 Basic mathematical model of an adaptive element: the neuron with linear synaptic operation.	14
Figure 2.2 Similarity measures between the vectors \mathbf{x}_a and \mathbf{w}_a	15
Figure 2.3 Block diagram representation of a neuron with linear synaptic operation.	17
Figure 2.4 $x_1 - x_2$ Attribute plane showing the pattern classification. The discriminant line L separates the patterns Class A and Class B.	17
Figure 2.5 Realization of neural logic for XOR operation using two AND neurons in stage 1 and one OR neuron in stage 2.	20
Figure 2.6 Geometrical view of the mapping operations for the XOR problem in stage 1.	21
Figure 2.7 Geometrical view of the mapping operations for the XOR problem in stage 2.	21
Figure 2.8 A two layered neural network with three neurons for realization of neural logic- XOR circuit using two AND and an OR neural unit.	22
Figure 2.9 A neural solution for the XOR problem obtained by BP learning algorithm with the learning rate of 0.8.	25
Figure 2.10 Neural unit mapped with inputs through nonlinearities (Higher-order synaptic operations).	26
Figure 2.11 Different nonlinear boundaries separating the patterns which are not	27

linearly separable (**XOR** operation).

- Figure 3.1 Structure of the biological neuron observed in the central nervous system (CNS). The soma of each neuron receives parallel inputs through its synapses and dendrites, and transmits the output via the axon to other neurons. 32
- Figure 3.2 Structure of the new neural unit with quadratic synaptic operation (QSO). 35
- Figure 3.3 Learning and adaptation scheme for the neural unit with QSO. 38
- Figure 3.4 Schematic representation of the backpropagation algorithm for the neural unit with QSO 41
- Figure 3.5 Schematic representation of the neural unit with CSO with synaptic and somatic operations. 42
- Figure 4.1 Schematic representation of the neural unit with QSO for realizing the logic circuits such as **OR**, **AND**, and Exclusive-OR (**XOR**). 52
- Figure 4.2 Learning and adaptation scheme for the realization of logic circuits. 54
- Figure 4.3 Desired output, $y_d(k)$ neural output, $y_n(k)$ and the error $e(k)$ with the learning iteration k . 59
- Figure 4.4 Hyperbolic boundary separating the patterns belonging to Class A and Class B for **XOR** logic with a single neural unit with QSO. 60
- Figure 4.5 Desired output, $y_d(k)$ neural output, $y_n(k)$ and the error $e(k)$ with the learning iteration k . 60
- Figure 4.6 Elliptical boundary separating the patterns belonging to Class A and Class B for the **XOR** logic with a single neural unit with QSO. 61
- Figure 4.7 Desired output, $y_d(k)$ neural output, $y_n(k)$ and the error $e(k)$ with the learning iteration k . 61
- Figure 4.8 Hyperbolic boundary separating the patterns belonging to Class A and Class B for the **XOR** logic with a single neural unit with QSO. 62
- Figure 4.9 Inverted hyperbolic boundary separating the patterns belonging to Class A and Class B for **OR** logic with a single neural unit with QSO. 64
- Figure 4.10 Parabolic boundary (nonlinear) separating the patterns belonging to Class A and Class B for **OR** logic with a single neural unit with QSO. 64

Figure 4.11 Hyperbolic boundary separating the patterns belonging to Class A and Class B for AND logic with a single neural unit with QSO.	65
Figure 4.12 The probability density function (pdf) of two Classes A1 and A2 with Bayesian Threshold S.	67
Figure 4.13 Mahalanobis distance (M-distance) from Class A1 and A2.	70
Figure 4.14 Three dimensional view of the M-distance.	70
Figure 4.15 Nonlinear decision boundaries separating the patterns belonging to Class A and Class B for the XOR logic with a single neural unit with QSO.	73
Figure 4.16 Nonlinear decision boundaries separating the patterns belonging to class A and class B for the OR logic with a single neural unit with QSO.	75
Figure 4.17 Nonlinear decision boundaries separating the patterns belonging to Class A and Class B for the AND logic with a single neural unit with QSO.	77
Figure 1.1 Critical bias for a hyperbolic decision boundary separating the patterns belonging to Class A and Class B	80
Figure 4.19 The learning scheme for functional approximation using a HONU.	82
Figure 4.20 Function approximation of a nonlinear function using a neural unit with QSO for Example 1 with different inputs such as sinusoidal, square, sawtooth and random signals.	83
Figure 4.21 Function approximation of a nonlinear function using a neural unit with QSO for Example 2 with a sinusoidal signal.	84
Figure 4.22 Function approximation of a nonlinear function using a neural unit with QSO for Example 3 with change in desired function during the simulation at $k = 500$	85
Figure 4.23 Function approximation of a nonlinear function using a neural unit with QSO for Example 4 with an arbitrary desired function.	86
Figure 5.1 Schematic representation of satellite control in its pitch plane (one dimensional view)	90
Figure 5.2 Schematic representation of the HONU's (the neural unit with QSO and the neural unit with CSO) as neural-controllers for the satellite attitude	91

control	
Figure 5.3 Structure of the neural unit with CSO for control applications.	92
Figure 5.4 Identification of a Plant using a neural unit with CSO	94
Figure 5.5 Block diagram of space vehicle (satellite control) system with nonlinear controller.	95
Figure 5.6 Adaptation of the neural unit with CSO as a nonlinear neuro-controller for satellite attitude control system.	96
Figure 5.7 Block diagram depicting different neural structures as neuro-controllers for a complex control system (satellite attitude control)	97
Figure 5.8 Step and error response of the satellite control with the three different neural controllers when the model is a first order system.	99
Figure 5.9 Step and error response of the satellite control with the three different neural controllers when the linear model is used.	100
Figure 5.10 Square input and the error response of the satellite control with the three different neural controllers when the linear model is used as MRAC.	101
Figure 5.11 Nonlinear damping function	102
Figure 5.12 Step input and the error response of the plant (satellite control) with different controllers when the nonlinear model is used as MRAC.	103
Figure 5.13 Level curves of the energy function and the solutions move along them. The equilibrium points are denoted by stars.	108
Figure 5.14 Phase plane curves from the potential. (a) Graph of a potential energy function $P(x)$. (b) A periodic orbit indicating absence of damping.	108
Figure 5.15 The slope field, flow of two solutions, and nullclines of closed loop control (5.18) in the phase plane.	112
Figure 5.16 The stability region for a nonlinear PD control system (5.23), area of decreasing energy and the nullclines.	112
Figure 5.17 Universal damping function	115
Figure 5.18 Different phases in the development of new damping function - universal damping function	116
Figure 5.19 Step response of a nonlinear system (Eqn. 5.29) with proposed neural controller.	119

- Figure 5.20 Neural unit with CSO in a dual mode, as a nonlinear neuro-controller 120
and as an identifier, performs state feedback control for optimal
performance.
- Figure 5.21 Nonlinear plant controlled by neural unit with CSO as state feedback 121
controller with constant parameters identified by the neural unit with
CSO.
- Figure 5.22 (b) Nonlinear plant controlled by neural unit with CSO as nonlinear 122
neural state feedback controller for unknown, unstable nonlinear plant
with variable parameters

LIST OF TABLES

Table 2.1 Correlation strength values for different similarity measure angles α	15
Table 2.2 Truth table for an XOR operation on binary inputs	19
Table 2.3 Truth table for neural stages 1 and 2 for realizing the XOR logic	22
Table 3.1 The number weights in polynomial networks (HONNs) when the order of the neuron is 2	36
Table 4.1 Truth Table for XOR Logic	56
Table 4.2 Initial and Final values of the Synaptic Weights for the XOR Logic	59
Table 4.3 Truth Table for OR and AND Logic	63
Table 4.4 Initial and Final values of the synaptic weights for the OR and AND Logic	63
Table 4.5 XOR Data Analysis	74
Table 4.6 OR Data Analysis	76
Table 4.7 AND Data Analysis	77
Table 4.8 Discriminant solutions for classifying the patterns of basic logic circuits	80

LIST OF SYMBOLS

Symbol	Meaning
\odot	Confluence operator
CNS	Central nervous system
CSO	Neural unit with cubic synaptic operation
d_e	Euclidean distance between new neural information \mathbf{x}_a and the stored knowledge \mathbf{w}_a
d	Distance between the reaction jets and the center of mass of the satellite
D	Dimension
e_{cso}	Error of the neuron with cubic synaptic operation
e_{lso}	Error of the neuron with linear synaptic operation
e_{qso}	Error of the neuron with quadratic synaptic operation
$e(k)$	Error signal in discrete time
E	Total energy
$E\{\cdot\}$	Expectation operator
$f(\mathbf{x})$	Damping function which is function of states
$\hat{f}[\cdot]$	Approximation of the function $f[\cdot]$
$f(x_1, x_2, x_3, \dots, x_n)$	Threshold function
F	Flow curve of the solution from the initial conditions (x_0, y_0)
g	Gain of the activation function
HONN	Higher-order neural network
HONU	Higher-order neural unit
J	Performance index
$J[\cdot]$	Performance or cost function

k	Discrete time index
k_p	Position gain
k_v	Velocity gain
k_{v1}	Shift gain in the universal damping function
$K.E$	Kinetic energy
LSO	Neural unit with linear synaptic operation
L_i	Discriminant line
M-distance	Mahalonobis distance
MM-distance	Modified Mahalonobis distance
n	Number of inputs
N	Order of the neuron
N_{ij}	Discriminant line equation (ij represents the neuron number in the network)
N^{th}	Order of the neuron
ND	Negative Definite
P	Potential energy
$p(x)$	Probability of an event x
$p(A_i / x)$	Probability of an event x belonging to one of the class A_i
PD	Positive Definite
QSO	Neural unit with quadratic synaptic operation
R	Set of real numbers
sgn	Sign function
S	Bayesian threshold
t	Continuous time index
v	Output of the synaptic operation
w_{ij}	Elements of the weight matrix
\mathbf{w}_a	Augmented synaptic weight matrix
$\mathbf{w}_{ai}^{(i)}$	Augmented weight matrix in a specific layer of the network

\mathbf{W}_a	Augmented synaptic weight matrix in HONUs
x_d	Desired location (target)
x_0, y_0	Initial condition
$\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$	Two states (position and velocity) of the satellite control
\mathbf{x}	Neural inputs from sensors
\mathbf{x}_a	Augmented neural input vector
$\mathbf{x}_a(k)$	Augmented neural input vector in discrete time
u	Control signal to the satellite
$y_d(k)$	Desired output
$y_n(k)$	Output of the neural unit
y_p	Plant (satellite) output
y_n	Output of the new neural units
y_{cso}	Output of the neuron with cubic synaptic operation
y_{lso}	Output of the neuron with linear synaptic operation
y_{qso}	Output of the neuron with quadratic synaptic operation
∞	Infinity symbol
$\phi[\cdot]$	Nonlinear activation function
\oint	Aggregation
α	Angle between the new neural information \mathbf{x}_a and the stored knowledge \mathbf{w}_a
μ	Learning rate in backpropagation algorithm
ε	Small positive number
\subseteq	Is a subset of (inclusion)
$\boldsymbol{\mu}$	Mean vector $[\mu_1 \ \mu_2]^T$
σ^2	Variance
Γ	Number of samples of the data

Ω	Covariance matrix defined in Mahalonobis distance
$ \Omega $	Determinant of the covariance matrix Ω
σ_{ij}	Elements of the covariance matrix Ω along the coordinates
Θ	Angular displacement of the satellite
$\dot{\Theta}$	Angular velocity of the satellite
$\ddot{\Theta}$	Angular acceleration of the satellite
$\phi[\cdot]$	Nonlinear activation function
$\phi'[\cdot]$	Slope of the nonlinear activation function
$\left[\sum_{i,j,l,m,\dots,\lambda^N}^n \right]^N$	Sigma tuner
$\langle \mathbf{W}_a, \mathbf{x}_a \rangle$	Inner product between the weight matrix \mathbf{W}_a and the augmented neural input vector \mathbf{x}_a

CHAPTER 1

Introduction

1.1 Biological Motivation

Biological systems are serving as inspirations for a variety of computational based learning systems. For example, biological knowledge has provided a great insight in the development of mathematical tools such as neural networks and genetic algorithms for the complex control problems. The research in this thesis conceptualizes the mathematical aspects of the basic building block of the central nervous system (CNS)-*the neuron*. From a computational standpoint, the CNS can be viewed as a parallel distributed system that has the capability to control a complex system over time. The primary objective of this thesis is to gain an insight in understanding the recognition, learning, and memory mechanisms of the CNS and to utilize these functions in the design and creation of the intelligent pattern classifiers and controllers.

Humans have been learning from '*Mother Nature*'. Since the evolution of machines, man has always dreamt of building machines that can emulate biological species like humans, birds etc with attributes such as locomotion, speech, and cognition (thinking, learning, memory, adaptation and intelligence). It has been a desire of system scientists to build a machine that can operate in an unstructured and uncertain environment with a high level of autonomy. To some extent, they have imitated birds and have created super sonic machines. In fact, man has been successful in implementing some of the attributes of the biological species such as human locomotion to transportation systems, human vision and speech for communication systems, human low-level cognition to computing systems. Now efforts are being made to imitate some of the attributes of cognition and intelligence-the higher cognitive faculty of the brain, and researchers are striving hard for the creation of intelligent systems; that is, a machine that has both autonomy and cognitive capabilities. The successful operation of a cognitive machine depends on its ability to adapt with a variety of unexpected events in its operating environment. By having machines possess such a level of autonomy, it would

be easy for the machines to learn higher-level cognitive tasks. Also, these machines would continue to adapt and perform the tasks with increasing efficiency even under changing and unpredictable environmental conditions.

The autonomous machines would be useful where direct human intervention is hazardous, tedious, or impossible. A hazardous task would be one where human intervention is in physical danger, such as in nuclear reactors, mining, and military operations. A tedious task would be one where a high level of concentration is required to perform tasks, such as programming the codes sitting in front of a computer where the human operator would be bored. A task impossible or difficult for humans would be the unmanned exploration of space, where a space craft is beyond the control of human intervention. For instance, spacecraft path planning is one of the major concerns in the design of autonomous vehicles for unstructured environments. It is very difficult to specify all functions *a priori* and in a deterministic way. Take, for instance, the Mars Mission- Spirit Rover. The vehicle was given high-level instructions (way points) and was equipped with smart cameras and laser sensors that would see the terrain. The information from the sensors was analyzed and catalogued in general classes. For each class a procedure was designed to accomplish the goal of moving from point A to point B. This brings a very different set of problems because the environment is complex and unpredictable. If the designed physical model does not capture the essentials of the environment, then the errors accumulate over time and the solution becomes impractical. Under these circumstances, the luxury of dictating the rules remotely is beyond our reach. It turns out that animals and humans do Spirit Rover-type tasks effortlessly.

Biological systems may be considered as a plausible source of motivation and framework for the design of autonomous machines. It provides motivation as well as gives several clues for the development of robust learning and adaptation algorithms in machines (Rao and Gupta 1994). In the present technology, lack of these robust and adaptive algorithms is due to that fact that the biological methods of processing information are different from conventional control techniques. The design procedures of the conventional control techniques are model based in the sense that the design methods involve the construction of an explicit mathematical model of the dynamic system to be controlled. Biological systems, on other hand, are non-model based and are quite

successful at dealing with uncertainty and complexity, and can smoothly coordinate many degrees of freedom during the execution of manipulative tasks in an unstructured environment.

Adaptive and neural control has gained renewed popularity in the past few decades, mainly emphasizing studies in the convergence of adaptive algorithms and in the stability of adaptive systems; that is, the systems considered are primarily systems described by differential (or difference) equations where the coefficients are (partially) unknown. In an attempt to enhance the applicability of adaptive control methods, learning control has been recently reintroduced in control literature; for example, (Gupta 1986), for learning methods in control with emphasis on neural networks.

1.2 Neural Networks

The conventional design methods of control systems involve the construction of a mathematical model describing the dynamic behaviour of the plant to be controlled and the application of analytical techniques to this model to derive a control law. Usually, such a mathematical model consists of a set of linear or nonlinear differential/difference equations, most of which are derived under some forms of approximation and simplification. These conventional techniques break down when a representative model is difficult to obtain due to uncertainty or sheer complexity. Modeling of a physical system for feedback control involves trade off between the simplicity of the model and its accuracy in matching the behaviour of the physical system. On the other hand, human operators do not always handle the system with a detailed mathematical model but they do with a qualitative feeling of the process, approximate reasoning and knowledge of the control process.

In the literature, two approaches are usually described to achieve satisfactory performance from a vaguely known dynamic plant. One approach is robust stabilizers or robust controllers and the other one is adaptive control. A Robust controller guarantees stability only if the actual system is a member of a class of systems that are close to the nominal plant. Application of adaptive control techniques has been slow as they require prior knowledge of the plant under control to determine the stability of the adaptive system. Since both approaches had some limitations, implementation of the conventional

adaptive methods and the robust stabilizers may be difficult or some times impossible (Ortega 1989). Detailed description of robust and adaptive control techniques may be found in Gupta [1986], and Narendra [1986].

The need to control complex systems under significant uncertainties has led to revaluations of the existing control methodologies. Evolution in the control paradigm has been fueled with two major concerns: the need to deal with increasingly complex systems, and the need to accomplish increasingly demanding design requirements with less precise knowledge of the plant and its environment. In these situations, it is almost mandatory for the control schemes to enforce learning and adaptive features (neural-networks). Neural and adaptive systems is a unique and a growing interdisciplinary field that considers adaptive, distributed, and mostly nonlinear systems-three of the ingredients found in biological systems. Neural and adaptive systems are used in many important engineering applications such as classification of patterns, system identification, signal enhancement, noise cancellation, prediction and control.

Neural network based controllers can be considered as a general class of adaptive controllers. The leading characteristic of neural and adaptive systems is their adaptivity, which brings a totally new system of design style. Instead of incorporating the *a priori* information from specifications, neural networks and adaptive systems use external data to automatically set their parameters. This means the neural systems are parametric. The neural-controller estimates the unknown information, and this information is used for future decisions and controls, thereby improving the performance of the control system. Neural networks, with their massive parallelism and ability to learn, offer good possibilities for improving techniques in control system, and may bring a bright future in the field of control system.

1.2.1 Biological Neuronal Morphology

In general, neural networks are described as connection models, parallel distributed processing units, or neuro-morphic systems (Rao and Gupta 1994). Neural networks consist of layers of neurons arranged in a set of rows and columns which perform some complex mathematical operations and mapping operations forming a complex pattern of neuronal layers. The neuronal inputs from the sensors are passed

through thread like structures called dendrites. The dendrites transmit the information to a synapse where it provides the confluence operation between the fresh neuronal information and the past experience, and sends a signal to the main body, soma of the neuron. This neural operation is termed as synaptic operation. The soma is the main body of the neuron. It receives all the signals from the synapses and provides an aggregation operation. If the aggregated value of the dendritic inputs exceeds a certain threshold, the neuron fires a signal along the axon (neural output). The firing of the neuron is associated with some nonlinear operation of the aggregated signal which is termed as somatic operation. A typical neural network structure is shown in Fig. 1.1.

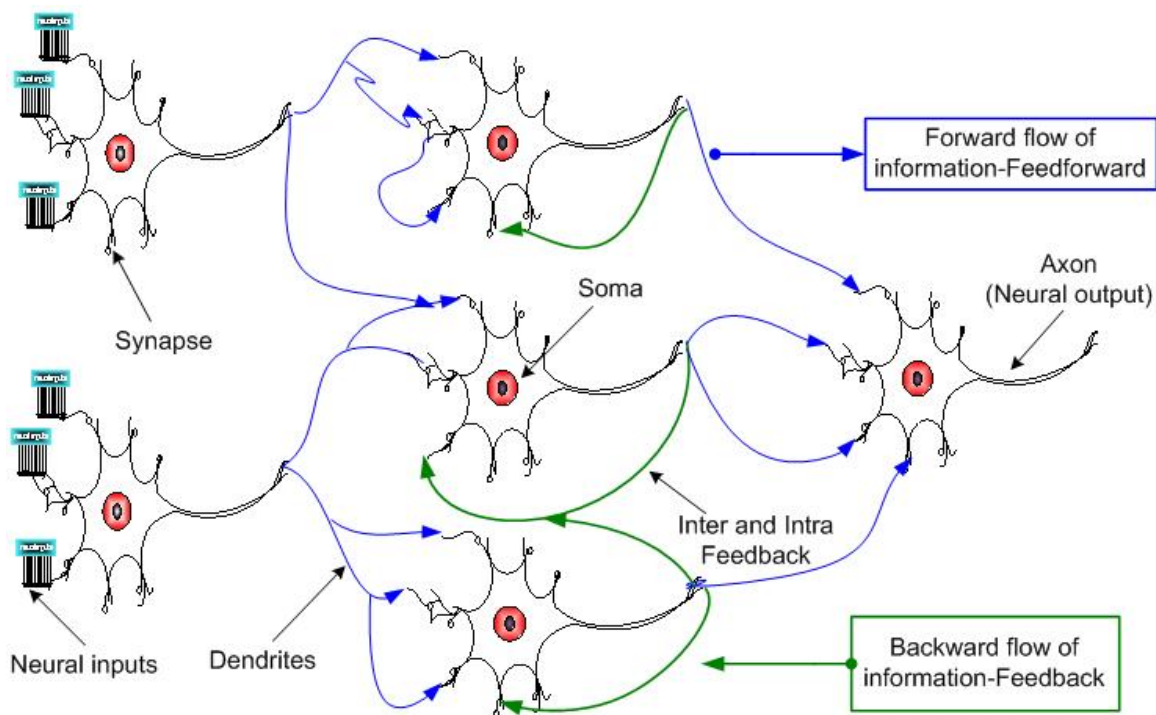


Figure 1.1 Layers of biological neurons arranged in a network depicting the flow of neuronal information in the forward direction as well as through inter and intra feedback direction.

The information in the network flows from one layer to another in the forward direction with continuous feedback evolving into a dynamical pyramid structure. The inputs from the input domain are mapped to the output domain through synaptic and somatic neuronal operations. These two neuronal operations play two distinct mathematical functions in a biological neuron. From the biological point of view, these two operations are physically

separate. However, from the mathematical point of view, the threshold function is shifted to synaptic operation for sheer simplicity.

1.2.2 Neuron: The Basic Unit of the CNS

In nature, the biological neurons are involved in various complex sensory, control and cognitive aspects of mathematical processing and in decision making processes. The discussions described in the existing literature often consider the behaviour of single neuron as the basic computing unit for processing neural information. A neural network consists of many interconnected identical simple processing units called *neurons*. Figure 1.2 shows a general mathematical model with confluence and somatic operations.

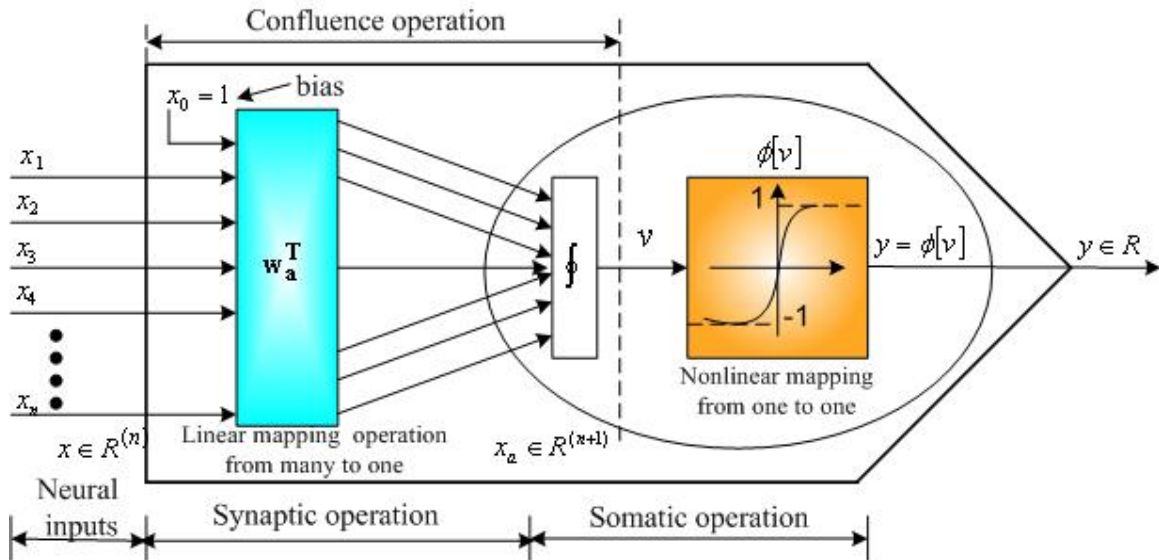


Figure 1.2 A general mathematical model of the neuron with synaptic and somatic operation. The confluence operation compares the neural information with the past experience stored in the synaptic weights and the nonlinear activation function provides the bounded neural output.

The synaptic operation provides the linear mapping from the neuronal inputs $\mathbf{x}_a \in R^{(n+1)}$ to $v \in R^1$ through the weight vector $\mathbf{w}_a^T \in R^{(n+1)}$, then the somatic operation performs nonlinear operation from $v \in R^1$ to $y \in R^1$ through an activation function $\phi[v]$. The somatic operation performs a nonlinear mapping through a nonlinear function called an activation function. There are different forms of activation functions such as linear, log

sigmoid, tansigmoid, bang-bang (hardlimiter), and radial basis functions. which are used in the mathematical model of the biological neuron. Figure 1.3 shows some of the most popular mapping functions employed in the neural networks.

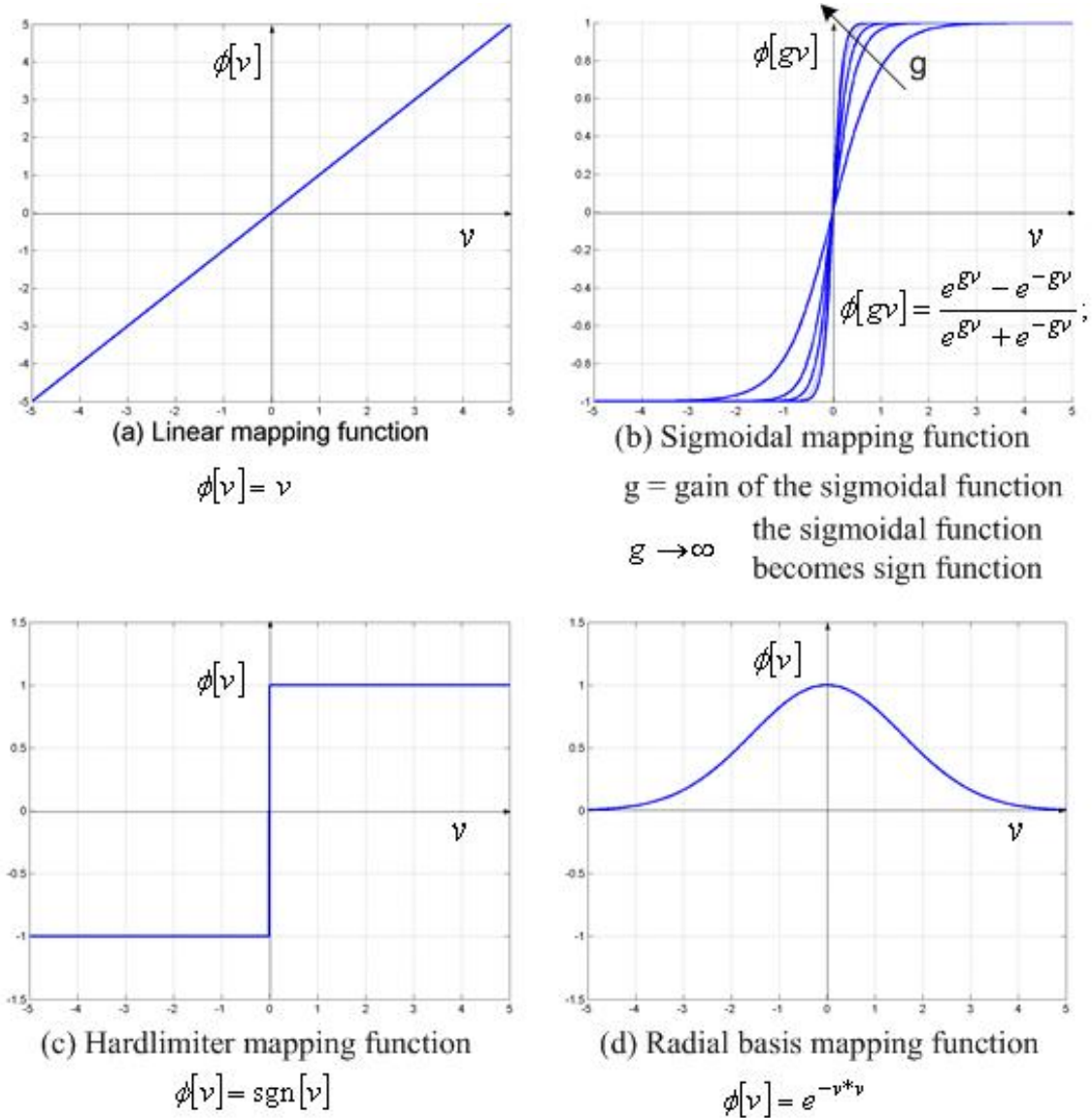


Figure 1.3 Different activation functions used in the mathematical model of the biological neuron for bipolar input signals.

The sigmoidal form is the most widely used activation function. However, the choice of nonlinear activation function in neural models depends on the nature of the problem under consideration.

1.3 Thesis Objectives

The neural networks consisting of the conventional neural units provide the neural output as a nonlinear function of the linear combination of the weighted neural inputs. These neural units have been successfully implemented in various applications such as pattern recognition, system identification, adaptive control, optimization and signal processing (Gupta et al. 2003, Rao 1994, Hopfield 1990, Kuroe et al. 1997). One of the most significant characteristics of neural networks is their ability to approximate arbitrary nonlinear functions. This ability of the neural networks has made them useful for modeling the nonlinear systems especially for the synthesis of nonlinear controllers (Song 2001). The performance of the neurons in the neural network depends on the following important factors

- a) Structure of the neuron; that is, static or dynamic models;
- b) Learning and adaptation algorithm such as backpropagation, quasi-Newton methods (the method of adjusting the neuron parameters);
- c) Type of activation function used in the mathematical model of the neuron; and
- d) Models of synaptic connections; that is, linear, quadratic, cubic...and Higher-order combinations of the neuronal inputs and the weights.

A considerable amount of research has been done focusing on the first three factors of the neurons. In the literature, most of the mathematical models of the neuron described incorporated modifications either in structure or learning and adaptation algorithms to improve the performance of the neuron. The selection of the nonlinear function in neural models necessitates a careful study of the problem. To some extent, the performance is influenced by activation function. Rao (Ph.D. thesis 1994) changed the slope of the activation function to affect the performance of the neurons. However, the performance of the neuron also depends on the model of the synaptic operation. In conventional neural models, the synaptic operation is modeled in such a way that the net input to the neural unit is just a linear summation of the weighted inputs. This is the commonly implemented form in most of the neural models developed. There is another set of neurons which consider multiplicative connections between the inputs and the neurons which closely resemble the neuronal structure shown in Fig. 1.1. These are called as higher-order neural

units which capture the nonlinear properties of the input pattern space. Rumelhart et al. (1986), Shin and Ghosh (1991), Heywood and Noakes (1995), and Homma and Gupta (2002) made extensive attempts to develop the higher-order neural units which have good storage, learning and computational properties. The performance of the neuron depends on the order of the inputs entering the neuron and the synaptic weights associated with them in the neural network. It has been found by Homma and Gupta (2002 b) that the Higher-order combination of the weighted inputs will yield the higher neural performance for complex problems. Villalobos and Merat (1995) have proposed a learning assessment method to optimize the feature shapes. However, one of the disadvantages encountered with Higher-order neural units is the combinatorial increase in weights with product terms; that is, a larger number of learning parameters (weights) are associated (Leda and Francis 1995).

In this thesis, a general method to develop Higher-order synaptic operation is presented in order to reduce the number of parameters without losing the Higher-order neural performance. The neurons with two levels of Higher-order neural synaptic operations are proposed. Using a novel general matrix form of the quadratic-operation, the Higher-order neural unit provides the output as a nonlinear function of the *quadratic combination* of the weighted input signals. The objectives of this thesis are as follows:

- To develop the concept of Higher-order neural units (HONUs) with Higher-order neural synaptic operation for control and pattern recognition problems based on the biological neuronal morphology;
- To propose the structure and general concept of a neural unit with N^{th} order synaptic operation for an N^{th} order HONU;
- To develop the learning and adaptive algorithms for N^{th} order neurons with higher-order neural synaptic operation;
- To validate the concept of the HONU by realizing the logic circuits such as Exclusive-OR (**XOR**), **OR**, **AND** circuits through simulation studies; and
- To apply these HONUs as neural controllers to linear and nonlinear systems such as satellite control, and to study the performance of these neural controllers through computer simulation studies.

1.4 Thesis Outline

In the following chapters, the mathematical foundation of the proposed neural structures of the HONUs such as the neural unit with quadratic synaptic operation (QSO) and the neural unit with cubic synaptic operation (CSO) and their potential for learning and control applications are presented. The neural unit with linear synaptic operation (LSO) which is a subset of the HONU is presented in Chapter 2. The structure and the mathematical modeling of the neural unit with LSO are discussed in this chapter. A nonlinear solution to **XOR** problem is presented along with the different applications of the neural unit with LSO for the control systems.

The concept of HONU is developed based on the structure of biological neurons in Chapter 3. Two HONUs are developed with higher-order synaptic operations. The structure, mathematical modeling and their implementation scheme for different applications are presented. A novel general matrix form of the quadratic-operation is developed. A general concept of the n^{th} order neural unit with n^{th} synaptic operation is developed based on the structure of the biological neuron.

The performance of the neural unit with QSO, as applied to pattern recognition problems, is demonstrated through simulation studies in Chapter 4. Basic logic circuits such as Exclusive-or (**XOR**), **AND**, and **OR** are realized using a single neural unit with QSO. A statistical perspective is provided to give a plausible explanation for the unique feature of the neural unit with QSO. This chapter also formulates the classification as the placement of discriminant functions in pattern space to minimize the probability of the classification error.

In chapter 5, the developed concept of HONUs is further strengthened by implementing these neural units as neuro controllers for the control of linear and nonlinear systems. A simple satellite attitude control problem is considered for simulation studies. In this chapter, a control technique called the model reference adaptive control using the HONUs is discussed. Some stability analysis approaches and stability results are presented. The fundamental concepts such as energy and lyapunov functions are used for the stability analysis of nonlinear systems. A new damping function called the

universal damping function is developed and implemented in the neuro controller for control of unknown parameter varying system.

Finally, the concluding remarks, the major contribution of the thesis, and suggested directions for future research are presented in Chapter 6. The major contributions of this thesis are as follows: (i) development of the HONUs for control and pattern recognition problems based on the structure of biological neuron, (ii) development of mathematical and structural models of the HONUs, (iii) application of HONUs for pattern recognition problems where basic logic circuits are realized using a single neural unit with QSO, (iv) development of new damping function named universal damping function for faster transient response. It is demonstrated through computer simulations that the neural structures with universal damping function developed in this thesis performed better compared to the conventional control techniques for control problems.

1.5 Conclusion

Neural networks play an important role in the execution of goal-oriented paradigms. They offer flexibility, adaptability and versatility, so that a variety of approaches may be used to meet a specific goal, depending upon the circumstances and the requirements of the design specifications. A brief review of the neural units with linear synaptic operation will be discussed in the next chapter. Development of higher-order neural units will open a new window for potential applications like control, pattern recognition, and image processing.

CHAPTER 2

Neural Units with Linear Synaptic Operation (LSO)

2.1 Introduction

Today, it is easy to build computers and other machines that can perform a variety of well defined tasks with celerity and reliability unmatched by humans. No humans with utmost cognizance can invert matrices or solve a system of differential equations at speeds rivaling these modern machines. However, no intelligent computer or machine can rival the human robust control mechanism (Widrow and Michael 1992). No doubt that these modern machines which are an extension of human muscular power, vision, and speech (car, aircrafts, robots, autonomous vehicles etc.) have brought luxury to human life but they are controlled by carbon-based computer- the brain (human intelligence and the human cognition). In this chapter, the mathematical details of a single neuron are described. These neural models emulate certain features of the biological neuron. One of the most important functions of the neuron is to make a decision. So, in this chapter, the neural models are implemented as decision makers.

The chapter is organized as follows: a brief introduction of the neural unit with LSO is presented in Section 2.2, neural models for threshold logics using neural unit with LSO are presented in Section 2.3. The neural logic for **XOR** operation using the neural unit with LSOs is shown in Section 2.3, the neural logic for **XOR** operation using polynomial discriminant functions is described in Section 2.4. Applications of neural unit with LSOs in control design are presented in Section 2.5. A brief summary is given in Section 2.6.

2.2 A Brief Description of the Neural Unit with LSO

The basic building block of all artificial neural networks, and most other adaptive systems, is the adaptive neuron. They communicate through a network of axons and synapses having a density of approximately 10^4 synapses per neuron. It is assumed that the modeling of the central nervous systems is based on the fact that the neurons communicate with each other by means of electrical impulses (Arbib 1987). The structure

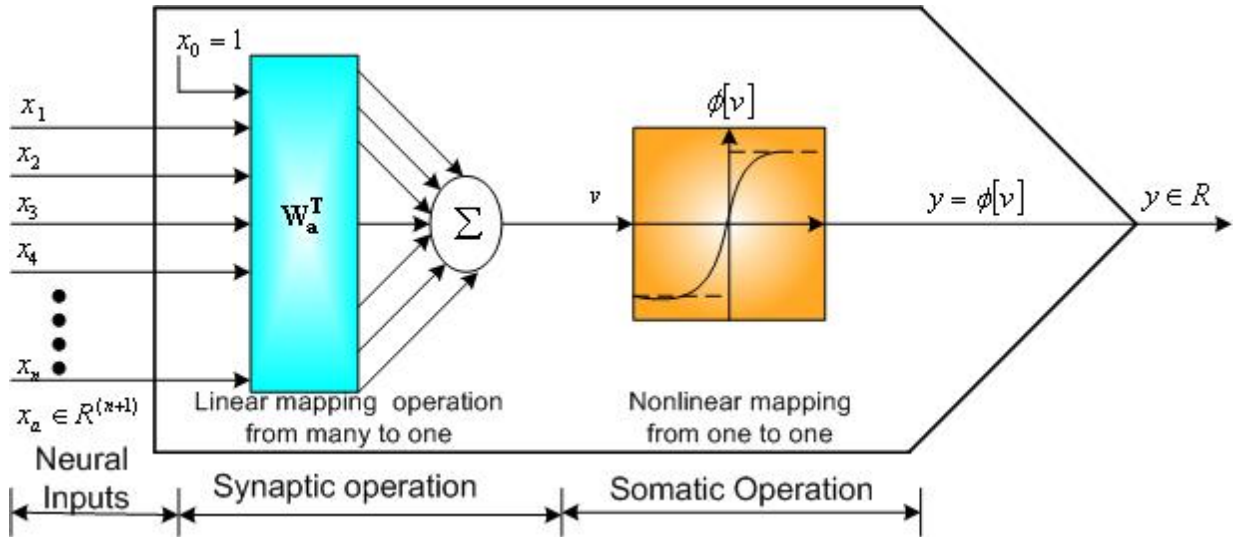
and mathematical model of the adaptive neuron is depicted in Fig. 2.1. The conventional model of neurons employed in control systems, pattern recognition and some other applications are linear in nature. Neural networks consisting of the linear neurons provide the neural out put as a nonlinear function of the weighted linear combination of the neural inputs. Let this element receive an input signal vector or input pattern vector \mathbf{x}_a and a desired response y_d , a special input used for learning. The components of the input vector are weighed by a set of coefficients \mathbf{w}_a , where the subscript 'a' stands for augmented notion of vectors which are defined as follows:

$$\mathbf{x}_a = [x_0, x_1, x_2, x_3, \dots, x_n]^T \in R^{n+1}, \quad x_0 = 1 \quad (2.1)$$

(augmented vector of neural inputs, where $x_0 = 1$ accounts for the threshold (bias)).

$$\mathbf{w}_a = [w_0, w_1, w_2, w_3, \dots, w_n]^T \in R^{n+1} \quad (2.2)$$

(augmented vector of synaptic weights including the threshold weight w_0)



$$\mathbf{x}_a = [x_0, x_1, x_2, x_3, \dots, x_n]^T \in R^{n+1}, \quad x_0 = 1 \text{ is the threshold.}$$

(augmented vector of neural inputs)

$$\mathbf{w}_a = [w_0, w_1, w_2, w_3, \dots, w_n]^T \in R^{n+1}$$

(augmented vector of synaptic weights)

Figure 2.1 Basic mathematical model of an adaptive element: *the neuron with linear synaptic operation.*

Thus, as illustrated in Fig. 2.1, the sum of the weighted inputs v can be expressed as a linear combination of neural inputs \mathbf{x}_a and the synaptic weights \mathbf{w}_a ; that is

$$\begin{aligned} v &= \mathbf{w}_a^T \mathbf{x}_a = \mathbf{x}_a^T \mathbf{w}_a \\ &= \langle \mathbf{w}_a, \mathbf{x}_a \rangle \text{ (inner product of two vectors } \mathbf{w}_a \text{ and } \mathbf{x}_a \text{)} \end{aligned} \quad (2.3)$$

The somatic operation provides a nonlinear mapping of the aggregated signal ' v ' yielding an output signal ' y_n '. Mathematically, the neural output, y_n can be represented as follows

$$\begin{aligned} y_n &= \phi[v] \\ &= \phi[\mathbf{w}_a^T \mathbf{x}_a] \end{aligned} \quad (2.4)$$

where $\phi[\cdot]$ is some nonlinear activation function with threshold w_0 . Equation 2.3 represents a measure of similarity between the neuronal input vector \mathbf{x}_a and the synaptic weight vector \mathbf{w}_a . There are two types of measure of similarity measures: (1) the inner product of the vectors \mathbf{x}_a and \mathbf{w}_a , and (2) the Euclidean distance between the vectors \mathbf{x}_a and \mathbf{w}_a . The similarity measures are shown in Fig.2.2.

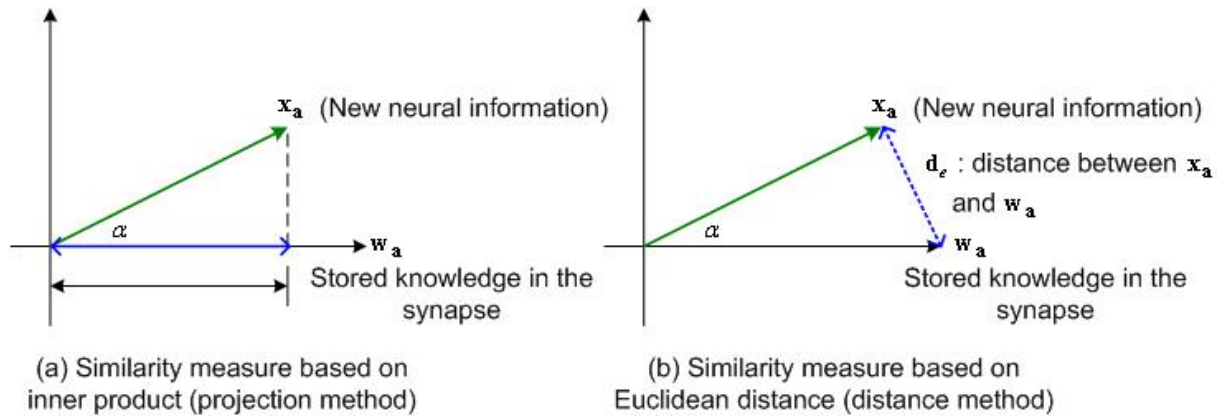
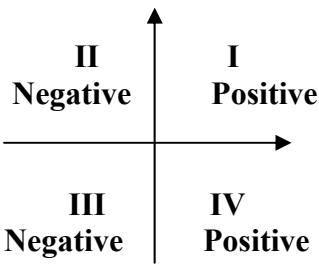


Figure 2.2 Similarity measures between the vectors \mathbf{x}_a and \mathbf{w}_a .

The correlation strength between the new information vector \mathbf{x}_a , and the stored knowledge vector \mathbf{w}_a depends on the angle α and its strength for particular values of α are tabulated in Table. 2.1. It is very difficult to explain and visualize the physical significance of the correlation strength between the weights and the neural inputs in a

network. However, it is possible to draw explicit correlation when a single neuron is implemented to make a decision.

Table 2.1 Correlation strength values for different similarity measure angles α

Angle α	Correlation	Similarity	Graphical representation
$\alpha = 0$	Positive	Close similarity (maximum)	
$\alpha = 90$	Zero	No similarity (minimum)	
$\alpha = 180$	Negative	No similarity	
$\alpha = 270$	Zero	No similarity	
$\alpha = 360$	Positive	Close similarity (maximum)	

Using this model, many neural morphologies, usually referred to as feedforward neural networks, have been reported in literature. These feedforward networks respond instantaneously to inputs because they possess no dynamic elements in their structure. Therefore, these neural structures are also called static neural networks or memory less networks; that is, they generate the output response determined by the present excitation (Zurada 1992). Extensions of these feedforward networks are the dynamic neural networks that incorporate feedback and dynamic elements in their structure. There are several dynamic neural structures based on different neural paradigms (Gupta and Rao 1994, Hopfield 1990). The neural networks either static or dynamic are implemented depending on the complexity of the problem. The following section describes the design of neural network classifiers for analyzing the threshold logic which were studied extensively in the 1960s.

2.2.1 Linear Classifier

The signals reaching the synapse and received by the dendrites are in the form of electrical impulses. The characteristic feature of the biological neuron is that the signals generated do not differ significantly in magnitude; that is, the signal in the nerve fiber is either absent or has the maximum value. It is assumed that the stimulus generates a train of pulses with a magnitude and a frequency. In other words, the information is transferred between the nerve cells by means of binary signals (Zurada 1992). In neural

networks, the inputs and outputs are often binary and are preferred to be ± 1 rather than the unsymmetrical '0' and '1'. With n binary inputs and one binary output, a single neuron shown in Fig. 2.1 is capable of implementing certain logic functions. There are 2^n possible input patterns. A general logic implementation would be capable of classifying each pattern as either +1 or -1, in accordance with the desired response. Thus, there are 2^{2^n} possible logic functions connecting n inputs to a single output. A single neuron is capable of realizing only the small subset of these functions, known as linearly separable logic functions. These are the set of logic functions that can be obtained with all possible settings of the weight values. In Figure 2.3, a two input neuron is shown. In Figure 2.4, all possible binary inputs for a two input neuron are shown in the pattern vector space. In this space, the coordinate axes are the components of the input pattern vector. The neuron separates the input patterns into two categories depending on the values of the input signal weights and the bias weight (threshold). A critical thresholding condition occurs when the output y_n equals zero.

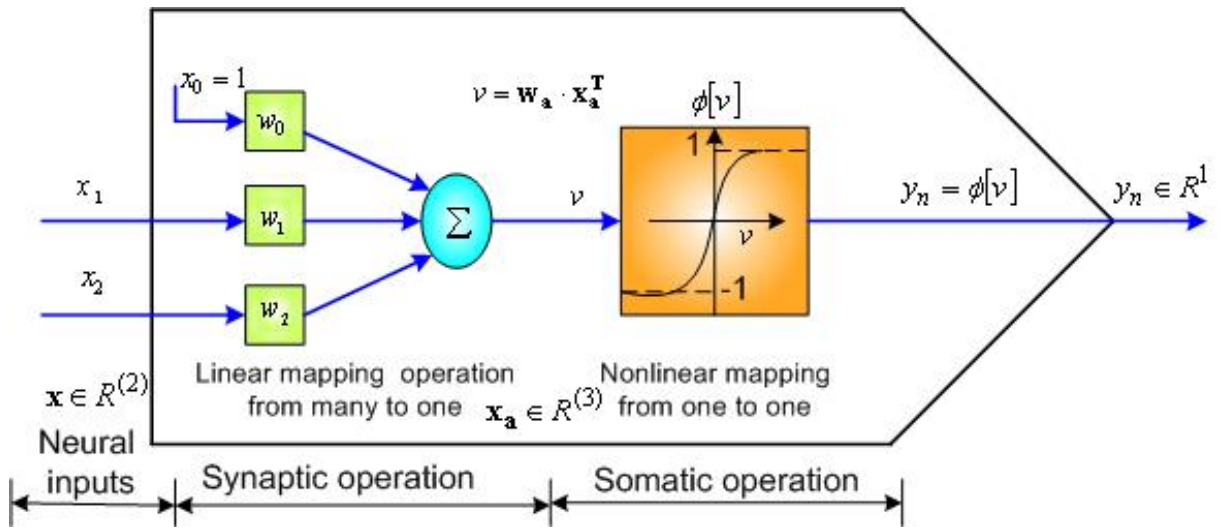


Figure 2.3 Block diagram representation of a neuron with linear synaptic operation.

$$\text{Synaptic operation: } v = \mathbf{x}_a^T \cdot \mathbf{w}_a = \mathbf{w}_a^T \cdot \mathbf{x}_a$$

$$\text{Somatic operation: } y_n = \phi[v] \in R$$

$$y_n = x_0 w_0 + w_1 x_1 + w_2 x_2 = 0 \quad (2.5)$$

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2} x_1 \quad (2.6)$$

The linear relationship is shown in the Fig. 2.4. It comprises of a separating line which has a slope

$$\text{Slope} = -\frac{w_1}{w_2}$$

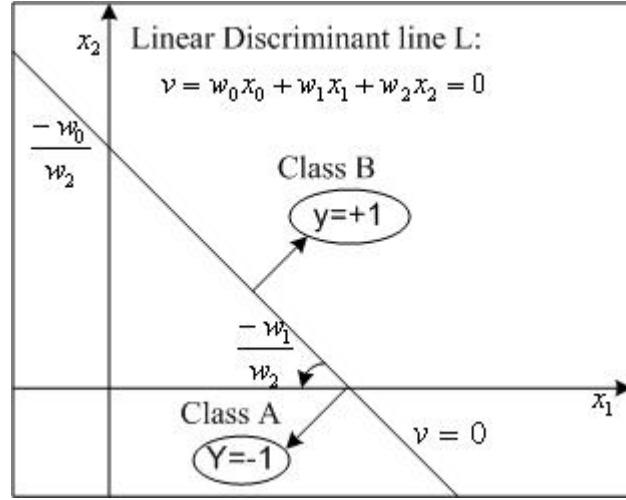


Figure 2.4 $x_1 - x_2$ attribute plane showing the pattern classification. The discriminant line L separates the patterns Class A and Class B.

and an intercept of

$$\text{Intercept} = -\frac{w_0}{w_2} \quad (2.7)$$

The line which separates the patterns appropriately is called the linear discriminant line L and is shown in Fig. 2.4. This mathematical model forms the basis of a neural network structure in contemporary neural computing.

2.3 Neural Models for Threshold Logic

On the basis of the highly simplified considerations of the biological neural systems, the first form of a neural model for the threshold logic is presented in this section. McCulloch and Pitts neural model is an element with n two-valued inputs $x_1, x_2, x_3, \dots, x_n \in \{-1, 1\}$ and a single two-valued output $y_n \in \{-1, 1\}$. The main goal in studying the threshold logic is to develop methods for identification and realization of threshold functions (logic functions). A switching function $y = f(x_1, x_2, x_3, \dots, x_n)$ is

said to be a threshold function if there exist weight coefficients $w_1, w_2, w_3, \dots, w_n$ and a threshold w_0 such that

$$y_n = f(x_1, x_2, x_3, \dots, x_n) = \text{sgn} \left(\sum_{i=1}^n w_i x_i + w_0 \right)$$

Designing neural model for threshold logic involves three steps

- Realization of switching function;
- Network synthesis; and
- Implementations of neural models for realizing the switching function.

In realizing the switching functions, the weights w_i should be assigned an appropriate real, positive, negative, or zero value. If the threshold function equality is satisfied, the switching function can be considered as linearly separable function. It is reported in literature that a single threshold element is sufficient to realize a switching function if the threshold function is linearly separable (Gupta et al 2003). For nonlinear separable functions, the threshold network requires more than one threshold element for realizing the given switching function. An effective approach to such a neural network synthesis is to decompose the non-threshold function into two or more terms, each of which will be a threshold function. For example consider a two-variable **XOR** function:

$$y = f(x_1, x_2) = x_1 \oplus x_2 = x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2, \quad x_1 x_2 \in \{-1, 1\} \quad (2.8)$$

If it can be realized with a single neural unit with weights w_0 , w_1 , and w_2 , then the output of the switching function is 1 for the input combinations $x_1 \bar{x}_2$ or $\bar{x}_1 x_2$, and -1 for the input combinations $x_1 x_2$ or $\bar{x}_1 \bar{x}_2$. Then the following inequalities should be satisfied; that is,

$$\begin{cases} w_1 \leq w_0 \\ w_2 \leq w_0 \end{cases}$$

and

$$w_1 + w_2 < 0$$

$$w_0 < 0$$

Obviously there is no such solution for which these contradictory inequalities are satisfied. Hence, the **XOR** logic function is not an ordinary threshold function that can be realized

by a single neural unit. The following section describes a method for realizing the **XOR** logic.

2.3.1 Neural Model for XOR Logic Circuit (Gupta et al. 2003)

For the binary state variables (x_1, x_2) , the logic **XOR** operation is defined as

$$y = x_1 \oplus x_2 = [\bar{x}_1 \text{ AND } x_2] \text{ OR } [x_1 \text{ AND } \bar{x}_2] \quad (2.9)$$

(Two **AND** operations in parallel followed by one **OR** operation.)

Alternatively, the **XOR** operation can also be defined as

$$y = x_1 \oplus x_2 = [x_1 \text{ OR } x_2] \text{ AND } [\bar{x}_1 \text{ OR } \bar{x}_2] \quad (2.10)$$

(Two **OR** operations in parallel followed by one **AND** operation.)

Thus, the **XOR** logic provides two classes of output which are defined as

$$\text{Class A: } A_1 \cup A_2 = \{[-1, 1] \cup [1, -1]\} \rightarrow +1 \quad (2.11a)$$

$$\text{Class B: } B_1 \cup B_2 = \{[-1, -1] \cup [1, 1]\} \rightarrow -1 \quad (2.11b)$$

These two classes of neural outputs are defined in Table 2.2

Table 2.2 Truth table for an XOR operation on binary inputs

Neural Inputs		Neural Outputs
x_1	x_2	$y = x_1 \oplus x_2$
-1	-1	-1: Class B ₁
-1	1	1: Class A ₁
1	-1	1: Class A ₂
1	1	-1: Class B ₂
Class A = Class A ₁ \cup Class A ₂		
Class B = Class B ₁ \cup Class B ₂		

There are four different neural methods for implementing the **Exclusive-OR (XOR)** logic operation. The first two methods use the classical **OR**, **AND** and **NOT** operation, and the last-two methods use the nonlinear neural operations.

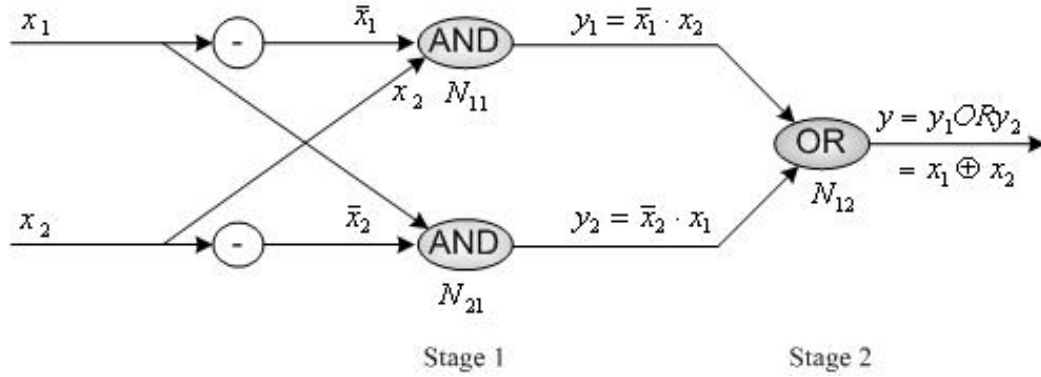


Figure 2.5 Realization of neural logic for **XOR** operation using two **AND** neurons in stage 1 and one **OR** neuron in stage 2.

It is obvious from Equations (2.9) and (2.10) that **XOR** switching function needs two stages (layers) of neurons: the first stage will have two neurons in parallel, and the second, the output stage, will have one neuron. Figure 2.5 shows the implementation of the **XOR** neural machine using the relation (2.9), and Figure 2.6 shows the geometrical view of the mapping operations over these stages. Each neuron provides a mapping from two inputs to a single output. Neuron N_{11} (row 1, column 1), the first **AND** neuron in stage 1, and N_{21} (row 2, column 1), the second **AND** neuron in stage 1, provide the mapping operation as shown in Table 2.3. Figure 2.6 shows the geometrical view of the mapping operations over these stages. The nonlinear mapping shown in Figures 2.6, and 2.7 results from the operations of two parallel **AND** neurons in the first stage followed by a single **OR** neuron in the second stage.

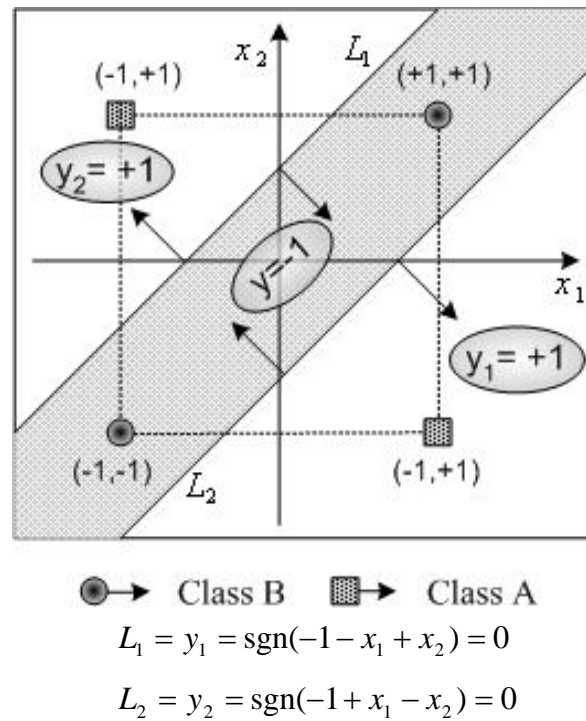


Figure 2.6 Geometrical view of the mapping operations for the **XOR** problem in stage 1.

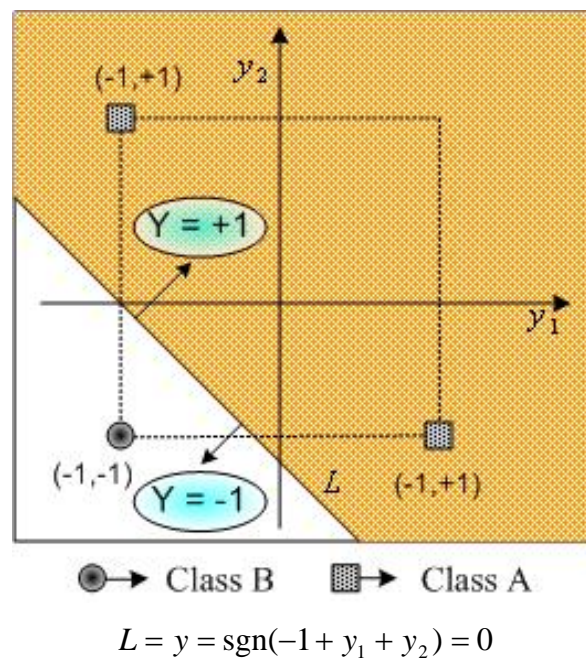


Figure 2.7 Geometrical view of the mapping operations for the **XOR** problem in stage 2.

Table 2.3 Truth table for neural stages 1 and 2 for realizing the XOR logic

Neural Inputs		Intermediate Stages				Neural Output
x_1	x_2	Points in the $x_1 - x_2$ plane (Fig. 2.6)	$y_1 = \bar{x}_1 \cdot x_2$	$y_2 = x_1 \cdot \bar{x}_2$	Points in the $y_1 - y_2$ plane (Fig. 2.7)	$y = y_1 \text{ OR } y_2$
-1	1	A ₁ : (-1, 1)	1	-1	A ₁ : (1, -1)	1: Class A ₁
1	-1	A ₂ : (1, -1)	-1	1	A ₂ : (-1, 1)	1: Class A ₂
-1	-1	B ₁ : (-1, -1)	-1	-1	B ₁ : (-1, -1)	-1: Class B ₁
1	1	B ₂ : (1, 1)	-1	-1	B ₂ : (-1, -1)	-1: Class B ₂
Class A = Class A ₁ \cup Class A ₂						
Class B = Class B ₁ \cup Class B ₂						

These points are shown in the $x_1 - x_2$ attribute plane in Fig 2.6 and in the modified $y_1 - y_2$ attribute plane in Fig 2.7. The neural operation of Table 2.3 and Fig 2.7 are illustrated in detail using the two stage neural circuit given in Fig. 2.8.

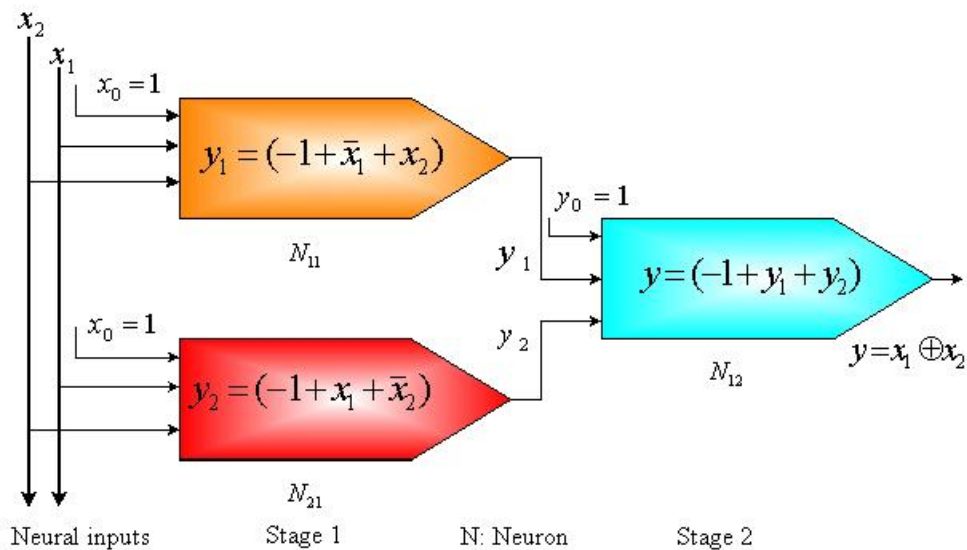


Figure 2.8 A two layered neural network with three neurons for realization of neural logic- XOR circuit using two AND and an OR neural unit.

The neural operation implemented in this neural network may be treated as a static, nonlinear, and discontinuous mapping from binary input space to the binary output space with preprogrammed weight parameters. The network is not associated with any sort of dynamics except that the information is feedback during the training process.

2.3.2 Simulation Studies for the XOR Logic Circuit with Neural Units with LSO

The simulation studies of **XOR** circuit implements the neuronal learning and adaptation capabilities. In this case three neurons are used as shown in Fig. 2.9 incorporating the backpropagation (BP) learning method. The augmented weight vectors associated with the neuron N_{11} , N_{21} , and N_{12} may be denoted as

$$\begin{aligned}\mathbf{w}_{a1}^{(1)} &= [w_{10}^{(1)} \quad w_{11}^{(1)} \quad w_{12}^{(1)}]^T \\ \mathbf{w}_{a2}^{(1)} &= [w_{20}^{(1)} \quad w_{21}^{(1)} \quad w_{22}^{(1)}]^T \\ \mathbf{w}_{a1}^{(2)} &= [w_{10}^{(2)} \quad w_{11}^{(2)} \quad w_{12}^{(2)}]^T\end{aligned}$$

and the input vectors for layers 1 and 2 are respectively

$$\begin{aligned}\mathbf{x}_a &= [x_0 \quad x_1 \quad x_2]^T, \quad x_0 = 1 \\ \mathbf{y}_a &= [y_0 \quad y_1 \quad y_2]^T, \quad y_0 = 1\end{aligned}$$

where $x_0 = 1$ and $y_0 = 1$ are bias terms. The input-output equations of the neurons are given by Eqns (2.3) and (2.4); that is,

$$\begin{aligned}y_1 &= \phi(\mathbf{w}_{a1}^{(1)} \cdot \mathbf{x}_a^T) \\ y_2 &= \phi(\mathbf{w}_{a2}^{(1)} \cdot \mathbf{x}_a^T) \\ y &= \phi(\mathbf{w}_{a1}^{(2)} \cdot \mathbf{y}_a^T)\end{aligned}$$

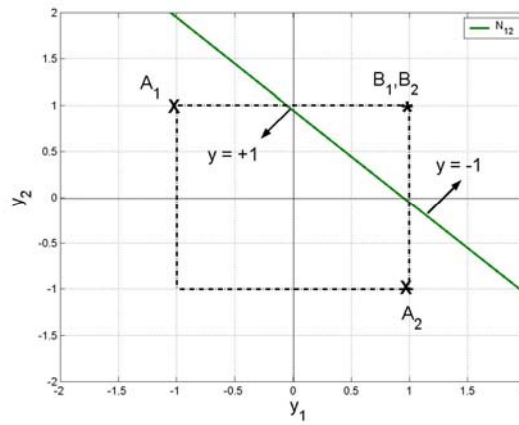
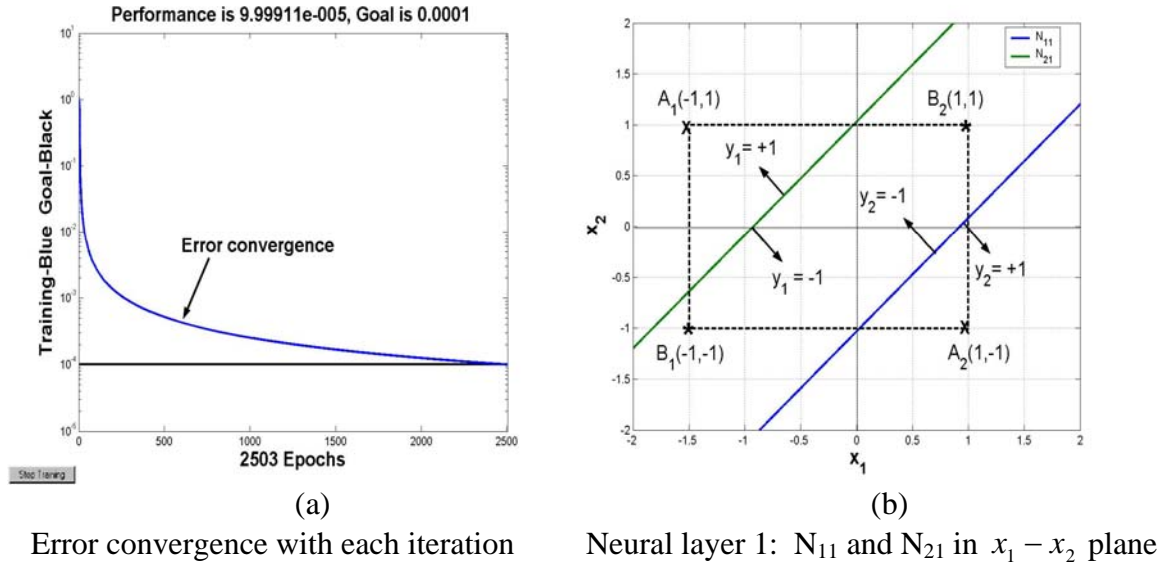
where $\phi(\cdot)$ is the sigmoidal nonlinear activation function. The learning rate for the simulation studies should be chosen in the range of 0.1 to 1.1 beyond which the learning phase could not be ensured (Gupta et al, 2003). The learning rate is the gain of the adaptable parameters of the network which determine the stability and speed of convergence during training. A learning rate of 0.8 was chosen for the simulation study and the initial weight values are randomly chosen by the Nguen-Widrow method. The

simulation results are shown in Fig. 2.8. The network took 2503 epochs to classify the patterns belonging to class A and B. The resulting weight vectors are

$$\mathbf{w}_1^{(1)} = [1.9467 \quad -2.1059 \quad 1.8854]^T$$

$$\mathbf{w}_2^{(1)} = [1.9677 \quad 2.1298 \quad -1.9100]^T$$

$$\mathbf{w}_1^{(2)} = [2.6688 \quad -2.8269 \quad -2.8244]^T$$



(a) N_{11} = Discriminant line L_{11} : $1.9467x_0 - 2.1059x_1 + 1.8854x_2$

(b) N_{21} = Discriminant line L_{21} : $1.9667x_0 + 2.1298x_1 - 1.9100x_2$

(c) N_{12} = Discriminant line L_{12} : $2.6688x_0 - 2.8269x_1 - 2.8244x_2$

Figure 2.9 A neural solution for the **XOR** problem obtained by BP learning algorithm with the learning rate of 0.8.

The convergence of the error with each epoch is shown in Fig. 2.8 (a). The learning stopped when the absolute value of the error $|e(i)| \leq 0.0001$ for $i = 2$ A_i and B_i . It took 4692 iterations for solving the same problem when the weights are initialized manually than the weights initialized by the Nguen-Widrow method (Gupta et al, 2003). The BP algorithm can take varying amounts of time to solve this problem depending on the choice of weights and the learning rate. One more problem with the BP algorithm is that the error will converge very fast initially and tend to slack down as it approaches the desired tolerance limit (in this case 10^{-04}). This is due to the fact that the learning algorithm encounters more local minima as the performance curve is a nonlinear curve. The performance of the neural network depends on the learning algorithm employed, learning rate, network structure and the problem it self. The performance indicates the real computing power of network structure. It can be improved by incorporating different fast learning algorithms such as BP with momentum, quasi-Newton techniques etc. or by changing the network structure. In this thesis, an effort is made to improve the performance of the network by modifying the existing neural structure.

2.4 Neural Logic for XOR-Operation using Polynomial Discriminant Function

The linear classifier is limited in its capacity and, of course, it is limited to only linearly separable forms of pattern discrimination. Design of neural network classifiers becomes far more involved and intriguing when requirements for the membership in categories become complicated. More sophisticated classifiers with higher capacities are nonlinear. There are two types of nonlinear classifiers (Widrow and Michael 1992):

1. Fixed preprocessor network connected to a single adaptive neuron
2. Multielement feedforward neural network (Madline)

Nonlinear functions of the applied inputs to single adaptive neuron will yield nonlinear decision boundaries. Useful nonlinearities include the polynomial functions. Consider the nonlinear classifier as shown in Fig. 2.10. The synaptic operation is a combination of linear and quadratic weighted combinations of the neural inputs. The sum of weighted

inputs v can be expressed as a nonlinear combination of the neural inputs x_a and the synaptic weights w_a .

$$v = w_0 + w_1x_1 + w_{11}x_1x_1 + w_{12}x_1x_2 + w_2x_2 + w_{22}x_2x_2 = 0 \quad (2.12a)$$

And the neural output y_n

$$y_n = \phi[v] \quad (2.12b)$$

or

$$y_n = \text{sgn}[f(w_a, x_a)] \quad (2.12c)$$

where $\phi(\cdot)$ is a nonlinear activation function which can yield a nonlinear discriminant surface of the shape shown in Fig. 2.11. With the proper choice of weights, the separating boundary pattern space can be established as shown in Fig. 2.11. This represents a solution for the **XOR** problem.

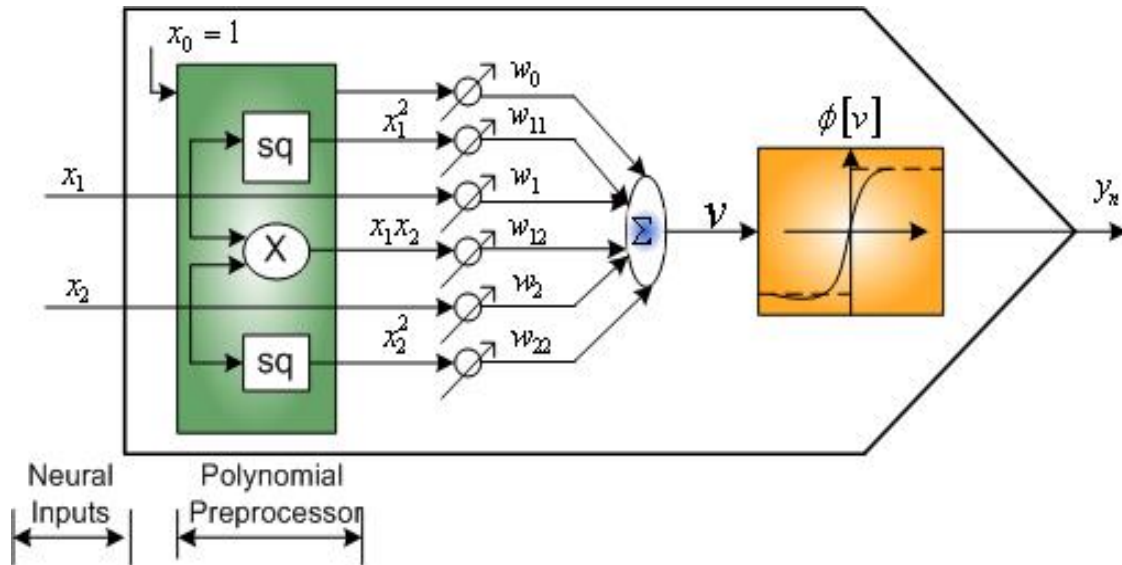


Figure 2.10 Neural unit mapped with inputs through nonlinearities (higher-order synaptic operations).

Of course, all linearly separable functions can also be realized using higher-order synaptic operations. This type of nonlinearities can be generalized for more than two inputs and higher degree polynomial functions of the inputs (Specht 1967). From Fig. 2.11 it is clear that only one adaptive neuron is sufficient to separate the two classes A and B appropriately.

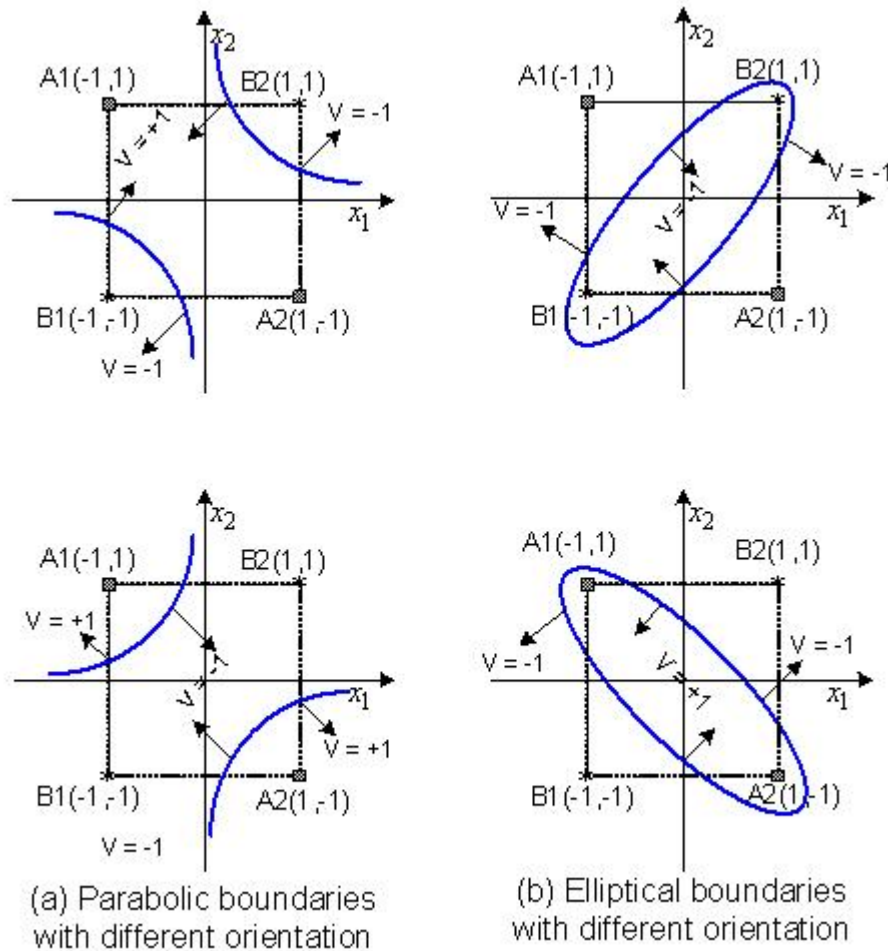


Figure 2.11 Different nonlinear boundaries separating the patterns which are not linearly separable (**XOR** operation).

2.5 Application of Neural Units with LSO for Control Problems

Based on the understanding of neuro-biological control aspects, the desire to develop simple models of neuronal structures has evolved into a promising area of research for many complex control problems in engineering industries. In the preceding sections, brief descriptions of single and multilayer feedforward network structures were introduced. These neural networks are called as static, feedforward, or non-recurrent neural networks. Such networks have no dynamic memory as the response of the network depends on its current inputs and the value of synaptic weights. Indeed, it is a well established fact that the feedforward neural networks can approximate nonlinear functions

to any desired degree of accuracy. This attribute of feedforward neural networks has motivated many researchers to utilize them as modern resourceful tools to model the dynamic systems. However, these networks suffer from many limitations (Hopfield 1990). The neural architectures with feedback have been introduced for various applications to overcome these limitations. These networks are called dynamic neural networks or recurrent neural networks. The dynamics in neural networks or neural computing does provide some functional basis of the cerebellum and its associated circuitry. In recent years, dynamic neural networks and recurrent networks have emerged as important components, which have proven to be extremely successful in system modeling (identification), adaptive control, signal processing and optimization problems (Widrow, Winter and Baxter, Hopfield 1982, Hopfield and Tank 1985, Narendra and Parthasarathy 1989, Gupta et al. 2003, Rao 1994). These networks are important because many of the systems that are modeled in the real world are dynamic and nonlinear. Narendra and Parthasarathy developed new mathematical models for the identification and control of complex nonlinear dynamical systems with unknown parameters. Gupta and Rao (1994) have developed neural structures which are useful in control and identification of unknown nonlinear systems. There are several dynamic neural structures based on different neural paradigms and the list goes on. With the parallel growth in the field of fuzzy logic, many new neural models encompassing the principles of neural networks and fuzzy logic are developed. Although the static, dynamic, and fuzzy-neural networks are being used in many control and machine vision applications, the basic neural models remain a feeble imitation of the biological counterparts.

2.6 Summary

In this chapter, a brief description of a neuron with linear synaptic operation was presented. The basic mathematical model and structure of the adaptive element, the neuron, were outlined briefly. Neural networks have been used in different applications such as pattern recognition, system identification and control of complex systems such as flexible structures, and intelligent robotic systems. In order to explain the concepts and nuances associated with linear neuron, a simple pattern classification problem (**XOR** logic) is studied thoroughly. Through simulation studies of the **XOR** logic it was

concluded that the neurons with linear synaptic operation were limited to only linearly separable forms of pattern distribution. However, they perform a variety of complex mathematical operations when they are implemented in the form of a network structure. These networks suffer from various limitations such as computational efficiency, learning capabilities (Hopfield 1990). These limitations are motivating many researchers around the globe to develop novel neural morphologies with better learning and adaptive capabilities that can closely mimic biological neurons.

CHAPTER 3

Development of Higher-Order Neural Units with Quadratic and Cubic Synaptic Operations

3.1 Introduction

The basic concepts of learning and adaptation in the field of control systems were introduced in the early sixties and many extensions and advances have been made since then. However, advances in understanding the physiology of biological control have spurred the interest of system scientists to explore the field of neuro-control systems. Due to the complex and diverse behaviour of the biological neurons, it is extremely difficult to compress their complicated characteristics into a model. However, recent mathematical models and the architectures of neuro-control systems have generated many theoretical and industrial interests. Towards this goal, a mathematical model of the biological neuron, named as the *neural unit with linear synaptic operation* (neural unit with LSO), or simply a neuron, was developed (Gupta and Rao 1994). Arranging neurons in layers or stages is supposed to mimic the layered structure of a certain portion of the brain. The most commonly used neural network architecture is the multilayer neural network (MFNN) with an error backpropagation algorithm. Although static, dynamic, and fuzzy neural networks are used in many control and machine vision applications, the basic neural models remain feeble imitations of their biological counterparts (Gupta and Rao 1994). In the previous chapter, it was shown that for solving a simple pattern classification problem requires a two layered linear neural network to realize the patterns. Many factors affect the learning performance of the MFNNs and must be dealt within order to have a successful learning process. The choice of initial weights, learning rate, network size and the learning database are the critical parameters that influence the performance of the MFNN. A good choice of these parameters will speed up the learning process to achieve the desired target. However, the MFNNs suffer from many limitations (Hopfield 1990, Gupta et al. 2003, and Principe et al. 2000). On the other hand, a new class of neural networks with higher-order neural units (HONUs) may provide a better solution which

decreases the training time, thereby improving the network efficiency (Lilly and Reid 1993, Leda and Francis 1995).

The objective of this research was to reduce the number of parameters of the HONUs without sacrificing the higher-order neural performance. In this chapter, two novel higher-order neural units are introduced; that is, the *neural unit with quadratic synaptic operation* (QSO) and the *neural unit with cubic synaptic operation* (CSO) are developed for pattern classification, control problems and other applications. The term “*quadratic and cubic*” are applied to these neural units in the sense that the output of the synaptic operation is an aggregation of the weighted *nonlinear combinations* of the neural inputs. The structure and mathematical details of the neural unit with QSO and the neural unit with CSO are given in Section 3.2, and 3.3. The learning and adaptation algorithm for both neural units are discussed in respective Sections. A general methodology for developing the HONUs with higher-order neural synaptic operation is presented in Section 3.4. A brief summary is given in Section 3.5.

3.2 Development of Neural Unit with Quadratic Synaptic Operation (QSO)

Computational neural networks can accommodate many inputs in parallel and encode the information in a distributed fashion. The learning capacity of a neural unit depends on its structure and the properties of its component elements (Rumelhart et al. 1986). The structure of the biological neuron, shown in Fig. 3.1, manifests that it can process linear as well as nonlinear combinations of the weighted neural inputs. These neural structures belong to the class of HONUs which are the basic building blocks of the higher-order neural networks (HONNs). The higher-order weighted combination of the inputs will yield higher neural performance as they require fewer training passes and a smaller training set to achieve the generalization over the input domain. Further, in HONUs, the neural inputs exploit cross and self correlations between them and building such specific knowledge into the network structure results in “pretrained” network. The pretrained network doesn’t need more iterations to learn the transformations (Lilly and Reid 1993, Kuroe et al. 1997).

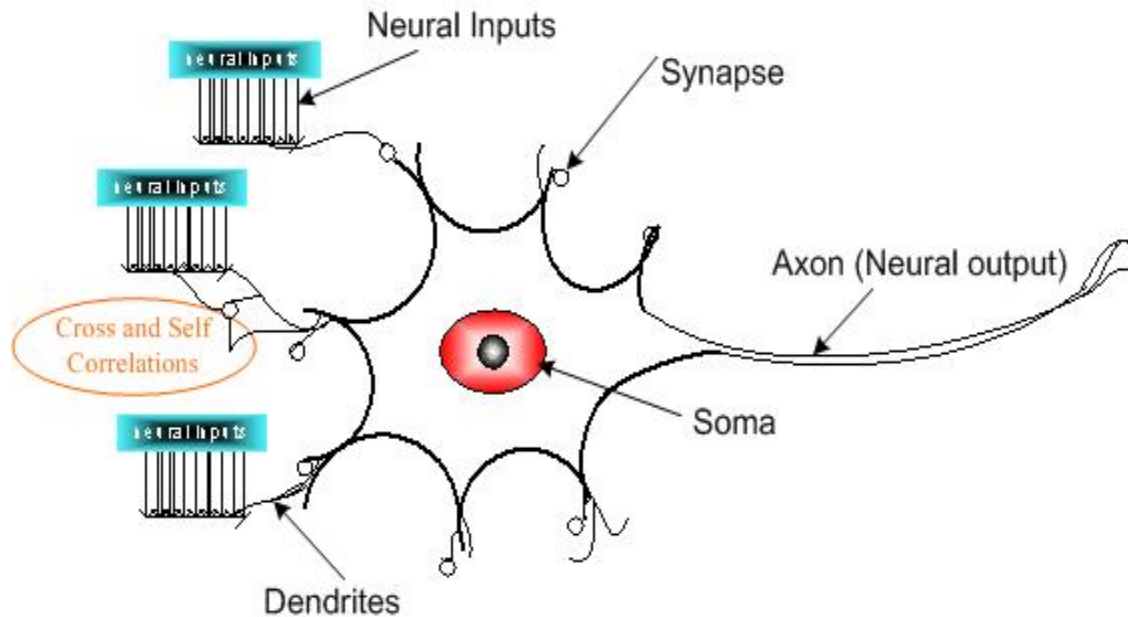


Figure 3.1 Structure of the biological neuron observed in the central nervous system (CNS). The soma of each neuron receives parallel inputs through its synapses and dendrites, and transmits the output via the axon to other neurons.

Note that a HONU contains all the linear and nonlinear correlations terms of the input components to the order of N . A generalized structure of the HONU is a polynomial network that includes the weighted sums of products of selected input components with an appropriate power. This type of network is called a sigma-pi network (S-PNNs) (Rumelhart et al. 1986). The synaptic operation of the S-PNNs creates the product of the selected input components computed with power operation while the conventional neural units compute the synaptic operation as a weighted sum of all the neural inputs. Since S-PNNs result in exponential increase in the number of parameters, some modified forms of S-PNNs were introduced by Shin and Ghosh (1991) which involve smaller number of weights than the HONNs. They are called as pi-sigma network (PSN) where the synaptic operation is the product of the weighted sum of all nonlinear correlations terms of the input components to the order N . Another type of HONNs called the ridge polynomial neural network (RPNN) were introduced by Shin and Ghosh (1991). However, the problem encountered with these HONNs is the combinatorial increase of the weight

numbers; that is, as the input size increases, the number of weights in HONNs increases exponentially (Zhengquan and Siyal 1998).

In Chapter 2 Section 2.4, a nonlinear solution was provided for the realization of the **XOR** circuit using a HONU shown in Fig. 3.1. Consider the discriminant equation of the HONU given in Section 2.4 by Eqn. (2.12 a)

$$v = w_0 + w_1x_1 + w_{11}x_1x_1 + w_{12}x_1x_2 + w_2x_2 + w_{22}x_2x_2 = 0 \quad (3.1)$$

The above equation belongs to the general second order equation which is of the form

$$H(x, y) = Ax^2 + By^2 + Cxy + Dx + Ey + F = 0 \quad (3.2)$$

where the coefficients A,B,C,D,E,F are real constants. When at least one of A, B, and C is nonzero, the above equation is referred to as the general second degree equation in two variables x and y . This can be expressed in a general quadratic form in terms of matrices, column and row vectors. Consider the Eqn. (3.2) with $\mathbf{X}_{(1 \times 3)} = [1 \ x \ y]$ as a column vector and a symmetric matrix $\mathbf{\Lambda}_{(3 \times 3)}$ which is expressed as $\mathbf{X}_{(1 \times 3)} \mathbf{\Lambda}_{(3 \times 3)} \mathbf{X}_{(3 \times 1)}^T$ and expanded as shown below

$$H(x, y) = [1 \ x \ y] \begin{bmatrix} F & D/2 & E/2 \\ D/2 & A & C/2 \\ E/2 & C/2 & B \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \quad (3.3)$$

Similarly expressing the Eqn. (3.1) in the quadratic form as $v = \mathbf{x}_a \mathbf{W}_a \mathbf{x}_a^T$ where $\mathbf{x}_a = [x_0 \ x_1 \ x_2] \in R^3$, is the augmented vector of neural inputs including bias, $x_0 = 1$, threshold (bias) of the neuron and \mathbf{W}_a is the weight matrix; that is,

$$v = [x_0 \ x_1 \ x_2] \begin{bmatrix} w_{00} & w_{01/2} & w_{02/2} \\ w_{01/2} & w_{11} & w_{12/2} \\ w_{02/2} & w_{12/2} & w_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \quad (3.4)$$

Weight matrix \mathbf{W}_a is of the same form as $\mathbf{\Lambda}_{(3 \times 3)}$ and a careful observation to the following elements of the matrix $\mathbf{\Lambda}_{(3 \times 3)}$ will provide an interesting result; that is, the following elements have the same magnitude.

$$\mathbf{\Lambda}_{(1 \times 2)} = \mathbf{\Lambda}_{(2 \times 1)}$$

$$\mathbf{\Lambda}_{(1 \times 3)} = \mathbf{\Lambda}_{(3 \times 1)}$$

$$\Lambda_{(2 \times 3)} = \Lambda_{(3 \times 2)}$$

Since the magnitude of these elements is same, it is indeed a prudent choice to consider the upper triangle or the lower triangle of the weight matrix which results in less number of parameters (weights) in the HONU. Taking this fact into consideration, a new general mathematical model of the neural unit is described in the following section.

3.2.1 Mathematical Model of the Neural Unit with Quadratic Synaptic Operation (QSO)

In this chapter, a novel neural unit called the neural unit with QSO is introduced. The mathematical model of the neural unit with QSO with n -dimensional neural inputs, $\mathbf{x}(t) \in R^n$, and a single neural output, $y_n(t) \in R^1$, is shown in Fig. 3.2. The neural inputs processed by the neural unit with QSO are summation of weighted linear and quadratic combination of inputs. The augmented neural input vector is defined as

$$\begin{aligned} \mathbf{x}_a &= [x_0, x_1, x_2, \dots, x_{n-1}, x_n]^T \in R^{n+1}, \\ &= [1, x_1, x_2, \dots, x_{n-1}, x_n]^T \\ x_0 &= 1 \text{ is the threshold.} \end{aligned} \quad (3.5)$$

The preprocessor of the neural unit with QSO generates a nonlinear combination of inputs (quadratic, cubic, etc.) depending on the requirement of the problem. The synaptic operation of the neural unit with QSO performs a new quadratic operation using an augmented weight matrix $\mathbf{W}_a \in R^{(n+1) \times (n+1)}$ given by Eqn. (3.6)

$$\begin{aligned} v(k) &= \mathbf{x}_a^T \mathbf{W}_a \mathbf{x}_a \in R^1 \\ \mathbf{x}_a &= [x_0, x_1, x_2, \dots, x_{n-1}, x_n]^T \in R^{n+1} \\ \mathbf{W}_a &= \begin{bmatrix} w_{00} & w_{01} & w_{02} & \dots & w_{0n} \\ 0 & w_{11} & w_{12} & \dots & w_{1n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \ddots & w_{(n-1)n} \\ 0 & 0 & 0 & 0 & w_{nn} \end{bmatrix} \in R^{(n+1) \times (n+1)} \end{aligned} \quad (3.6)$$

The neural output, $y_n(k)$, is given by a nonlinear function of the synaptic output, $v(k)$, as

$$y_n(k) = \phi[v(k)] \in R^1 \quad (3.7)$$

where $\phi(\cdot)$ is the somatic nonlinear activation function.

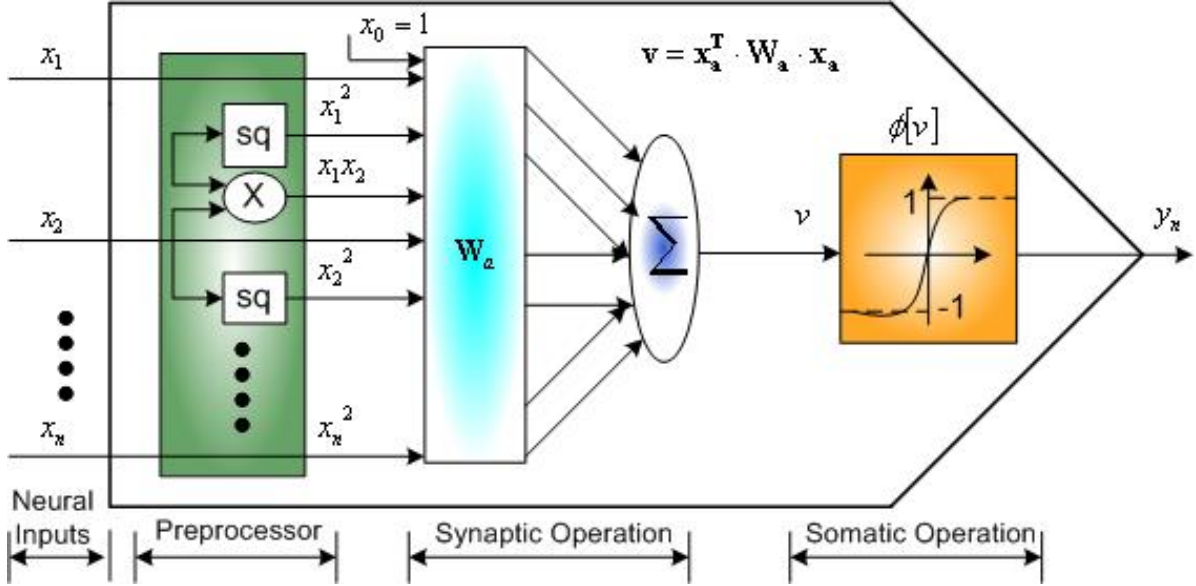


Figure 3.2 Structure of the new *neural unit with quadratic synaptic operation (QSO)*.

This nonlinear function maps the synaptic output $v(k) \in [-\infty, \infty]$ to a bounded neural output. Many different forms of mathematical nonlinear functions can be used to model the activation function. Typically, the sigmoidal activation function is a widely used nonlinear activation function. An example of such a function $\phi(\cdot)$ is given as

$$\phi(gv) = \frac{e^{gv} - e^{-gv}}{e^{gv} + e^{-gv}}$$

where $g > 0$ is a constant value which determines the slope of the $\phi(v)$, the activation gain of the activation function. It is clear from the Fig. 3.2 that the augmented neural inputs to the neuron are $x_0, x_1, x_2, \dots, x_{n-1}, x_n \in R^{n+1}$, $x_0 = 1$, and the higher-order inputs are generated within the neuron. For simplicity, the discrete time variable k is not represented in the Fig. 3.2. This structure encapsulates the basic features and the structure of the biological neuron shown in Fig. 3.1. The weights w_{ij} and w_{ji} , $i, j \in \{0, 1, 2, \dots, n\}$ in the augmented matrix \mathbf{W}_a yield the same quadratic term $x_i x_j$ or $x_j x_i$. Therefore, an upper triangle (or a lower triangle) of the augmented weight matrix \mathbf{W}_a is sufficient to

generate the discriminant equation which has the quadratic form. The upper triangle matrix can give the general quadratic discriminant equation as

$$v(k) = \mathbf{x}_a^T \mathbf{W}_a \mathbf{x}_a = \sum_{i=0}^n \sum_{j=i}^n w_{ij} x_i(k) x_j(k) \in R^1, \quad x_0 = 1 \quad (3.8)$$

A careful observation of the augmented weight matrix \mathbf{W}_a reveals that the conventional neural units with linear synaptic operation are a subset of the neural unit with QSO. For example, the first row of the novel weight matrix \mathbf{W}_a , that is, the row vector

$$[w_{00} \quad w_{01} \quad w_{02} \quad \cdots w_{0n}] \in R^{(n+1) \times (n+1)}$$

can produce the weighted linear combination $v(k) = \sum_{j=0}^n w_{0j} x_j(k)$ of the neural inputs.

The total number of weights involved in this structure are $(n+1) \times (n+2)/2$, where n is equal to the number of inputs. A comparison of the number of weights of different types of HONNs is given in Table 3.1.

Table 3.1 The number weights in polynomial networks (HONNs) when the order of the neuron is 2

Type of the HONNs	Order of the Neuron N = 2	
	General formula for the total number of parameters	Number of parameters for $n = 2$
Neural Unit with QSO	$(n+1) \times (n+2) / 2$	6
PSN	$(n+1)^{(n+1)}$	27
RPNN	$N(N+1) \times (n+1) / 2$	9

The above table shows that when the networks have the same higher-order terms, the weights of the neural unit with QSO are significantly less than number of the weights in other HONNs. Therefore, the developed neural structure with a quadratic combination of the neural inputs and weights is more general and can improve the network efficiency significantly. The other HONUs like the neural unit with cubic synaptic operation can be expressed with different combinations of the higher-order synaptic operations.

3.2.2 Learning and Adaptation Algorithm for the Neural Unit with Quadratic Synaptic Operation (QSO)

Developing a learning and adaptation rule involves optimizing the parameters of the neural structure; that is, the adaptable weights w_{ij} and the slope of the nonlinear activation function $\phi'[v(k)]$ (Gupta 1970). Learning is an iterative process in which the control sequence is modified in such a way that the neural output approaches the desired output $y_d(k) \in R^1$ as closely as possible. The equivalency of the input pattern sequence or desired response $y_d(k) \in R^1$ and the output $y_n(k)$ is a convenient condition for testing the learning process. Generally, the performance of a system is measured by defining an index called the error function, $J[e(k)]$, (or the cost function or performance index) (Gupta and Rao 1994). Defining the performance index is an optimizing process which finds a quantitative measure of the network performance. The performance index is small when the network performs well and is large when the network performs poorly.

The learning and adaptation algorithm for the neural unit with QSO is developed in discrete time (k). The learning scheme is shown in Fig. 3.3. Let k denote the discrete time steps, $k = 1, 2, 3 \dots$ and $y_d(k) \in R^1$ be the desired output signal corresponding to the neural input vector $\mathbf{x}_a(k) \in R^{(n+1)}$ at the k^{th} time step. Backpropagation algorithm using the gradient or steepest descent method can be used to derive the learning and adaptation algorithm. The error signal $e(k)$ is the difference between the desired output $y_d(k)$ and the neural output $y_n(k)$ and is defined as

$$e(k) = y_d(k) - y_n(k) \quad (3.5)$$

where

$y_n(k)$ = neural output

$y_d(k)$ = desired output

$e(k)$ = error between the desired output and the neural output

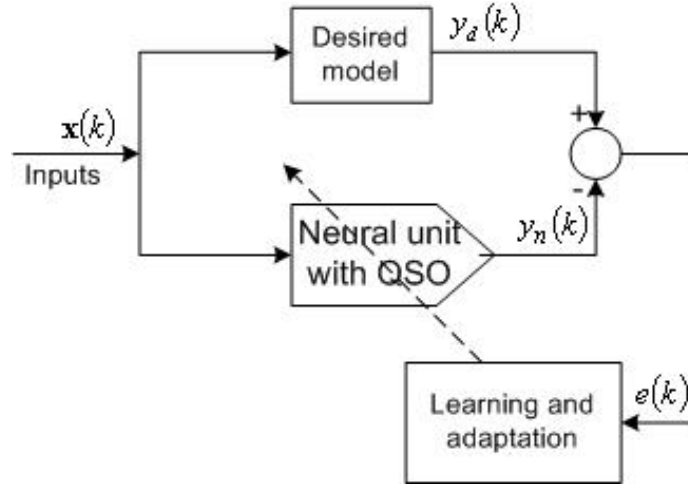


Figure 3.3 Learning and adaptation scheme for the neural unit with QSO.

In this iterative process, the control sequence is modified in each learning iteration to cause the neural output $y_n(k)$ to approach the desired output $y_d(k)$. To achieve this, a performance index is defined as

$$J[e(k)] = E\{F(e(k))\} \quad (3.6)$$

where E is the expectation operator. A commonly used form of the $F(e(k))$ in Eqn. (3.6) is the squared function of the error; that is, the performance index is given as

$$J[e(k)] = \frac{1}{2} E\{e^2(k)\} \quad (3.7)$$

The error function can be minimized using the following adaptation algorithm by adapting the weight matrix \mathbf{W}_a as

$$\mathbf{W}_a(k+1) = \mathbf{W}_a(k) + \Delta\mathbf{W}_a(k) \quad (3.8)$$

where $\Delta\mathbf{W}_a(k)$ denotes the change in the weight matrix. The changes in the weight matrix are proportional to the negative of the gradient of the error function $J[e(k)]$. Hence, $\Delta\mathbf{W}_a(k)$ is given as

$$\Delta\mathbf{W}_a(k) = -\mu \frac{\partial J[e(k)]}{\partial \mathbf{W}_a(k)} \quad (3.9)$$

where $\mu > 0$ is the learning rate. The value of μ determines the stability and the speed of convergence of the adaptive algorithm to optimal weights. The partial derivatives of the

error function, $\frac{\partial \mathbf{J}[e(k)]}{\partial \mathbf{W}_a(k)}$, with respect to the weights can be calculated using the chain rule

of derivatives as shown below

$$\frac{\partial \mathbf{J}[e(k)]}{\partial \mathbf{W}_a(k)} = \left[\frac{\partial \mathbf{J}}{\partial w_{00}}, \frac{\partial \mathbf{J}}{\partial w_{01}}, \frac{\partial \mathbf{J}}{\partial w_{02}}, \dots \dots \frac{\partial \mathbf{J}}{\partial w_{nn}} \right]^T$$

From the definitions of the performance index, $\mathbf{J}[e(k)]$ and the error signal $e(k)$, the gradient of the performance index with respect to the weight vector is obtained as follows

$$\frac{\partial \mathbf{J}[e(k)]}{\partial \mathbf{W}_a(k)} = \frac{1}{2} E \left[\frac{\partial (y_d(k) - y_n(k))^2}{\partial \mathbf{W}_a(k)} \right] \quad (3.10)$$

$$= E \left[e(k) \left\{ -\frac{\partial y_n(k)}{\partial \mathbf{W}_a(k)} \right\} \right] \quad (3.11)$$

$$= E \left[e(k) \left\{ -\frac{\partial \phi[v(k)]}{\partial \mathbf{W}_a(k)} \right\} \right] \\ = E \left[-e(k) \left\{ \frac{\partial \phi[v(k)]}{\partial v(k)} \frac{\partial v(k)}{\partial \mathbf{W}_a(k)} \right\} \right] \quad (3.12)$$

Note that, since

$$v(k) = \mathbf{x}_a^T(k) \mathbf{W}_a(k) \mathbf{x}_a(k)$$

and

$$\frac{\partial v(k)}{\partial \mathbf{W}_a(k)} = \frac{\partial [\mathbf{x}_a^T(k) \mathbf{W}_a(k) \mathbf{x}_a(k)]}{\partial \mathbf{W}_a(k)} = \mathbf{x}_a^T(k) \mathbf{x}_a(k)$$

Substituting this result in Eqn. (3.12) gives

$$= E \left[-e(k) \left\{ \phi'[v(k)] \frac{\partial v(k)}{\partial \mathbf{W}_a(k)} \right\} \right] \\ = -e(k) \phi'[v(k)] \mathbf{x}_a^T(k) \mathbf{x}_a(k) \quad (3.13)$$

where $\phi'[v(k)]$ is the slope of the nonlinear activation function used in the neural unit

with QSO. Therefore, $\frac{\partial \mathbf{J}[e(k)]}{\partial \mathbf{W}_a(k)}$ can be expressed using Eqn. (3.13) as

$$\frac{\partial \mathbf{J}[e(k)]}{\partial \mathbf{W}_a(k)} = -e(k) \cdot \phi'[v(k)] \cdot \mathbf{x}_a^T(k) \cdot \mathbf{x}_a(k) \quad (3.14)$$

Substituting this result in Eqn. (3.9) gives the changes in the weight matrix which is given by

$$\begin{aligned} \Delta \mathbf{W}_a(k) &= -\mu \frac{\partial \mathbf{J}[e(k)]}{\partial \mathbf{W}_a(k)} \\ \Delta \mathbf{W}_a(k) &= -\mu \cdot -e(k) \cdot \phi'[v(k)] \cdot \mathbf{x}_a^T(k) \cdot \mathbf{x}_a(k) \\ &= \mu \cdot e(k) \cdot \phi'[v(k)] \cdot \mathbf{x}_a^T(k) \cdot \mathbf{x}_a(k) \end{aligned} \quad (3.15)$$

Note that, taking the average of changes for the input vectors and the changes in the weights, $\Delta \mathbf{W}_a(k)$, provides the strength of the cross-correlation between the error $e(k)$ and the corresponding neural input terms $\mathbf{x}_a^T(k) \cdot \mathbf{x}_a(k)$. Using this gradient estimate with steepest descent method provides a tool for minimizing the mean square error $e^2(k)$. Thus the updating algorithm for the augmented weight vector is given by

$$\begin{aligned} \mathbf{W}_a(k+1) &= \mathbf{W}_a(k) + \Delta \mathbf{W}_a(k) \\ \mathbf{W}_a(k+1) &= \mathbf{W}_a(k) + \mu \cdot e(k) \cdot \phi'[v(k)] \cdot \mathbf{x}_a^T(k) \cdot \mathbf{x}_a(k) \end{aligned} \quad (3.16)$$

The implementation scheme of Eqn. (3.16) is shown in Fig. 3.4. Usually the nonlinear activation function is chosen as sigmoidal function; that is, a hyperbolic tangent function $\tanh(s)$. In this case the derivative $\phi'(v(k))$ is given by

$$\begin{aligned} \phi'(v(k)) &= \frac{\partial(\tanh(v(k)))}{\partial(v(k))} \\ &= 1 - \tanh^2(v(k)) = 1 - \phi^2(v(k)) \end{aligned}$$

Thus the Eqn. (3.16) can be rewritten as

$$\mathbf{W}_a(k+1) = \mathbf{W}_a(k) + \mu \cdot e(k) \cdot (1 - \phi^2(v(k))) \cdot \mathbf{x}_a^T(k) \cdot \mathbf{x}_a(k) \quad (3.17)$$

Without loss of generality, the above algorithm can be converted into time domain.

Following observations are made with reference to the adaptation algorithm for the parameters of the neural unit with QSO

- The desired model is an entity representing a physical reality as in the case of system identification and pattern classification problems (Rao 1994);
- $\phi'[v(k)]$ is the slope of the nonlinear activation function and can be considered as a gain of the changes in the weights;
- μ is the learning rate which determines the speed and the convergence of the adaptive algorithm to the optimal values;
- The weight matrix contains $((n+1) \times (n+2)/2)$, $n = [2, 3, \dots, n]$ number of parameters and;
- An upper (or lower) triangular weight matrix is sufficient to generate the nonlinear decision boundary.

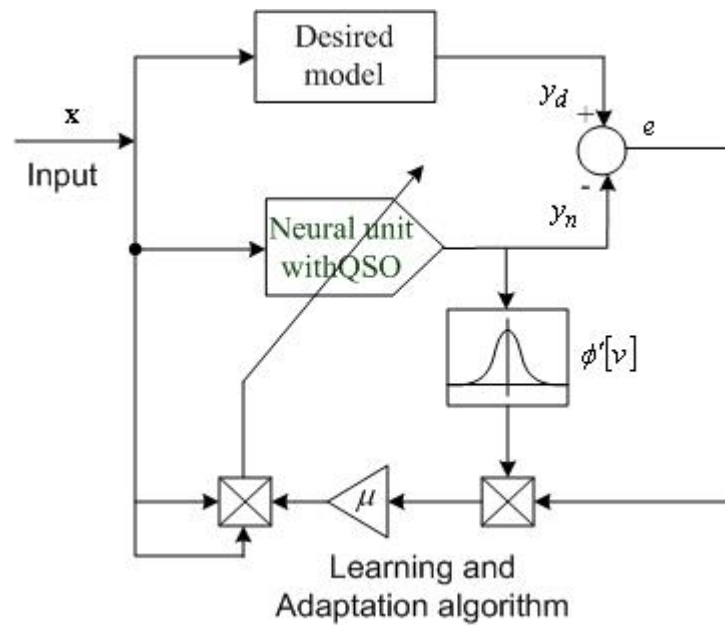


Figure 3.4 Schematic representation of the backpropagation algorithm for the neural unit with QSO

3.3 Development of Neural Unit with Cubic Synaptic Operation (CSO)

The higher-order neural units (HONUs) may be used in conventional feedforward neural network structures as the hidden units to form HONNs. These networks are proved to have good computational, storage, pattern recognition capabilities, and learning properties, and are realizable in hardware (Taylor and Commbes 1993). The higher correlation between the neural inputs is considered to improve the approximation and generalization capabilities of the neural networks (Gupta et al 2003). The following section describes another HONU called the neural unit with cubic synaptic operation (CSO).

3.3.1 Structure and Mathematical Details of Neural Unit with CSO

The structure of the neural unit with CSO is shown in Fig. 3.5. Defining the augmented vector of neural inputs and neural weights, the synaptic operation can be expressed as

$$v(k) = \sum_{i=0}^n \sum_{j=i}^n \sum_{l=j}^n w_{ijl} x_i x_j x_l \quad (3.18)$$

$$= w_{000}x_0x_0x_0 + w_{001}x_0x_0x_1 + \dots + w_{nn(n-1)}x_nx_nx_{n-1} + w_{nnn}x_nx_nx_n.$$

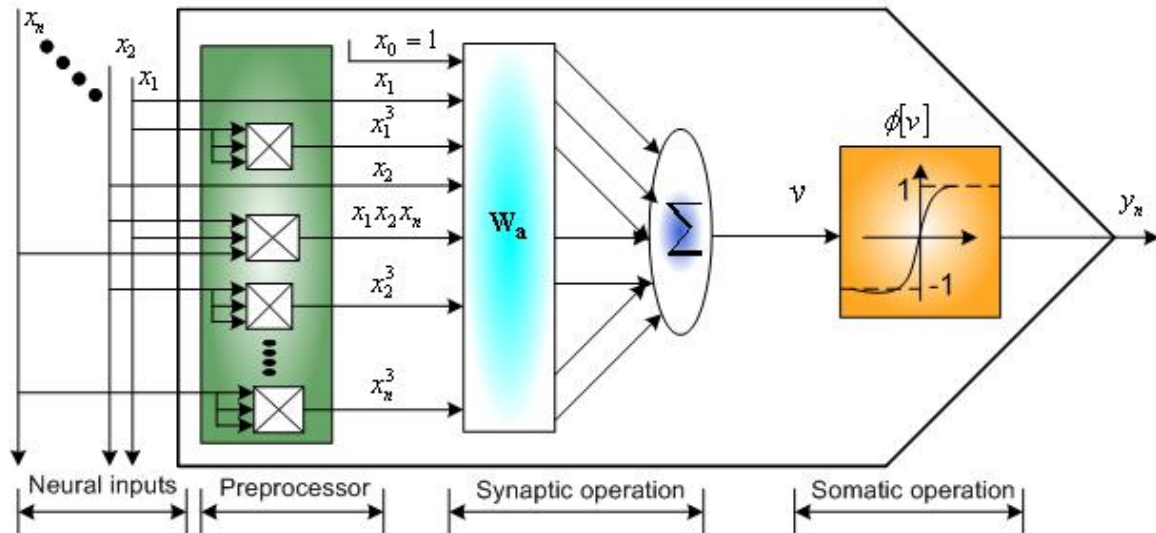


Figure 3.5 Schematic representation of the neural unit with CSO with synaptic and somatic operations.

$x_0 = 1$ is the threshold and $\mathbf{x}_a = [x_0 \ x_1 \ x_2 \ \cdots \ x_{n-1} \ x_n]^T$ is the vector of augmented neural inputs augmented with the threshold (bias). The output, $y_n(k)$ of the somatic operation is defined as a nonlinear mapping, $y_n(k) = \phi[v(k)]$ which provides a bounded output for the neural inputs.

To accomplish an approximation task for an input-output $\{x(k), y_n(k)\}$, the learning algorithm for the neural unit with CSO can be developed on the basis of the gradient descent method. The error function is formulated as

$$J[e(k)] = E\left\{\frac{1}{2}e^2(k)\right\} = E\left\{\frac{1}{2}[y_d(k) - y_n(k)]^2\right\} \quad (3.19)$$

where $e(k) = y_d(k) - y_n(k)$, $y_d(k)$ is the desired output and $y_n(k)$ is the output of the neural network. The error function is minimized using the steepest-descent technique and the weights are updated in discrete time (k) as given by the following equations

$$w_{ijl}(k+1) = w_{ijl}(k) + \Delta w_{ijl}(k) \quad (3.20)$$

$$\Delta w_{ijl}(k) = -\mu \frac{\partial J[e(k)]}{\partial w_{ijl}(k)} = -\mu \frac{1}{2} \frac{\partial e^2(k)}{\partial w_{ijl}(k)} \quad (3.21)$$

$$\begin{aligned} &= -\mu \frac{\partial [y_d(k) - y_n(k)]^2 / 2}{\partial w_{ijl}(k)} \\ &= -\mu [y_d(k) - y_n(k)] \frac{\partial [y_d(k) - y_n(k)]}{\partial w_{ijl}(k)} \\ &= -\mu e(k) \frac{\partial [y_d(k) - y_n(k)]}{\partial w_{ijl}(k)} \\ &= \mu e(k) \frac{\partial y_n(k)}{\partial w_{ijl}(k)} \end{aligned} \quad (3.22)$$

The changes in the weights are given by the cross-correlation between the error and the corresponding neural input terms. Even though the neural unit with CSO is a static neural unit, it can be implemented as a neuro-controller for complex control systems such as satellite control etc. A detailed description of the neural unit with CSO as neuro-controller will be given in Chapter 5.

3.4 General Methodology to Develop HONUs with Higher-Order Synaptic Operation

In this section, a general method for developing the HONUs is presented. In control systems, any higher-order systems can be expressed as a combination of first and second-order systems. Likewise, any HONU can be expressed as a combination of the neural unit with LSO and the neural unit with QSO. They can be considered as basic neural mathematical models with which any HONU can be represented. The basic principles and concepts of these neural models are utilized in the formulation of the HONUs with higher-order synaptic operations.

Let N be the order of the neuron and n be the number of inputs to the neuron. Consider $\mathbf{x}_a = [x_0, x_1, x_2, x_3, \dots, x_n]^T \in R^{n+1}$, $x_0 = 1$ to be the augmented vector of neural inputs, where $x_0 = 1$ accounts for the threshold (bias). Now consider the synaptic operation of the neuron when the order of the neuron is 1; that is, for $N=1$, the synaptic operation for the neural unit with LSO is given by the following expression as

$$\begin{aligned} (v)_{N=1} &= \mathbf{W}_a^T \mathbf{x}_a = \mathbf{x}_a^T \mathbf{W}_a \\ &= \langle \mathbf{W}_a, \mathbf{x}_a \rangle \text{ (inner product of two vectors } \mathbf{W}_a \text{ and } \mathbf{x}_a \text{)} \\ &= \sum_{i=0}^n w_i x_i \end{aligned} \quad (3.23)$$

For $N=2$, the synaptic operation for the neural unit with QSO is given by the following expression as

$$\begin{aligned} (v)_{N=2} &= \mathbf{x}_a^T \mathbf{W}_a \mathbf{x}_a \\ &= \sum_{i=0}^n \sum_{j=i}^n w_{ij} x_i x_j \end{aligned} \quad (3.24)$$

For $N=3$, the synaptic operation for the neural unit with CSO is given by the following expression as

$$(v)_{N=3} = \sum_{i=0}^n \sum_{j=i}^n \sum_{l=j}^n w_{ijk} x_i x_j x_l \quad (3.25)$$

Similarly, for the higher-orders the synaptic operation is given as

$$(v)_{N=4} = \sum_{i=0}^n \sum_{j=i}^n \sum_{l=j}^n \sum_{m=l}^n w_{ijlm} x_i x_j x_l x_m \quad (3.26)$$

$$(v)_{N=5} = \sum_{i=0}^n \sum_{j=i}^n \sum_{l=j}^n \sum_{m=l}^n \sum_{o=m}^n w_{ijlmo} x_i x_j x_l x_m x_o \quad (3.27)$$

$$\begin{array}{ccccccc} \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \end{array}$$

$$(v)_{N-1} = \sum_{i=0}^n \sum_{j=i}^n \sum_{l=j}^n \sum_{m=l}^n \cdots \cdots \cdots \sum_{y=u}^n w_{ijkl\cdots} \cdots \cdots x_i x_j x_l x_m x \cdots \cdots x_y$$

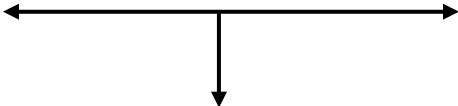
$$(v)_N = \sum_{i=0}^n \sum_{j=i}^n \sum_{l=j}^n \sum_{m=l}^n \cdots \cdots \cdots \cdots \sum_{z=y}^n w_{ijlm\cdots} \cdots \cdots x_i x_j x_l x_m x \cdots \cdots x_z \quad (3.28)$$

Equation (3.28) is the general expression of the synaptic operation for the N^{th} order of the neuron. From Eqns. (3.23 to 3.28), it is clear that each synaptic expression of any order consists of mainly two terms; that is,

- Sigma's (\sum); and
- An ensemble of weights and the neural inputs.

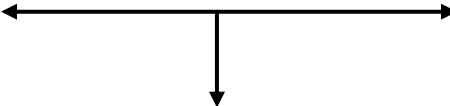
Equations (3.23 and 3.24) can be represented in a nice matrix notation but it is difficult to represent the synaptic equation in a matrix notation for the HONUs (N=3, 4, 5...so on. Hyper matrices are beyond the knowledge of the author). In order to simplify the representation given by the Eqn. (3.28), the above two operators would be useful. Consider the Eqn. (3.28) which can be broken down into two operations

$$(v)_N = \sum_{i=0}^n \sum_{j=i}^n \sum_{l=j}^n \sum_{m=l}^n \cdots \cdots \cdots \cdots \sum_{z=y}^n w_{ijlm\cdots} \cdots \cdots x_i x_j x_l x_m x \cdots \cdots x_z$$



$$\left[\sum_{\lambda^N}^n \right]^N$$

Sigma Tuner



$$\langle w_{\lambda^N}, [x]_{\lambda^N}^N \rangle$$

Correlation operator

where sigma tuner precludes the exponential increase of the parameters for the HONUs and the correlation operator provides the correlation strength between the input signals and the accumulated knowledge stored at the synapses in the form of weights. The sigma tuner contains three variables; that is, n, N, λ^N which are defined in Eqn. (3.29)

$$\begin{array}{ccc}
 \text{Number of the inputs to} & \left[\sum_{\lambda^N}^n \right] & \text{Order of the neuron} \\
 \text{the neuron} & \leftarrow & \rightarrow \\
 & & \text{Pruner}
 \end{array}
 \quad (3.29)$$

where

$$n = [2, 3, 4, \dots, n-1, n] \in R$$

$$N = [1, 2, 3, 4, \dots, N-1, N] \in R$$

$\lambda^N = [i, j, k, l, \dots, y, z, \dots, N]$ and the variables take any one value of the set $[0, 1, 2, 3, 4, \dots, N-1, N] \in R^{N+1}$. The superscript of the pruner indicates the number of variables to consider in the sigma tuner.

$$\left[\sum_{\lambda^N}^n \right]^N = \left[\sum_{i, j, \dots, z, \dots, \lambda^N}^n \right]^N = \sum_{i=0}^n \sum_{j=i}^n \sum_{k=j}^n \sum_{l=k}^n \dots \dots \dots \sum_{\lambda^N = \lambda^{N-1}}^n$$

$$\langle w_{\lambda^N}, [x]_{\lambda^N}^N \rangle = \langle w_{ij \dots z \dots \lambda^N}, [x]_{i, j, \dots, z \dots \lambda^N}^N \rangle$$

$$= w_{ijk \dots z \dots \lambda^N} x_i x_j x_l x_m x \dots \dots x_{\lambda^N}$$

Now, the general synaptic expression for any neural unit is given as

$$\begin{aligned}
 (v)_N &= \left[\sum_{i, j, \dots, z, \dots, \lambda^N}^n \right]^N \langle w_{ij \dots z \dots \lambda^N}, [x]_{i, j, \dots, z \dots \lambda^N}^N \rangle \\
 (v)_N &= \left[\sum_{\lambda^N}^n \right]^N \langle w_{\lambda^N}, [x]_{\lambda^N}^N \rangle
 \end{aligned}
 \quad (3.30)$$

Sigma tuner Correlation operator

When the network includes Pi-operator, number of weights increase exponentially which decreases the network efficiency during the training process. On other hand, the Sigma tuner decreases the number of weights without sacrificing the HONNs performance.

For better understanding of the Eqns. (3.29) and (3.30), the expansion of the expression is as follows:

$$\begin{aligned}
 & \left\{ \begin{array}{l} N = 1 \\ \left[\sum_{\lambda^1}^n \right]^1 \langle w_{\lambda^1}, [x]_{\lambda^1}^1 \rangle = \sum_{i=0}^n w_i x_i \end{array} \right. \\
 & \left\{ \begin{array}{l} N = 2 \\ \left[\sum_{\lambda^2}^n \right]^2 \langle w_{\lambda^2}, [x]_{\lambda^2}^2 \rangle = \left[\sum_{i,j}^n \right]^2 \langle w_{ij}, [x]_{i,j}^2 \rangle = \sum_{i=0}^n \sum_{j=i}^n w_{ij} x_i x_j \end{array} \right. \\
 & \left\{ \begin{array}{l} N = 3 \\ \left[\sum_{\lambda^3}^n \right]^3 \langle w_{\lambda^3}, [x]_{\lambda^3}^3 \rangle = \left[\sum_{i,j,l}^n \right]^3 \langle w_{ijk}, [x]_{i,j,l}^3 \rangle = \sum_{i=0}^n \sum_{j=i}^n \sum_{l=j}^n w_{ijl} x_i x_j x_l \end{array} \right. \\
 & \begin{array}{cccccccccc} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array} \\
 & \left\{ \begin{array}{l} N = N \\ \left[\sum_{\lambda^N}^n \right]^N \langle w_{\lambda^N}, [x]_{\lambda^N}^N \rangle = \left[\sum_{i,j,\dots,z,\dots,\lambda^N}^n \right]^N \langle w_{ij\dots z\dots\lambda^N}, [x]_{i,j,\dots,z,\dots,\lambda^N}^N \rangle \\ = \sum_{i=0}^n \sum_{j=i}^n \sum_{l=j}^n \sum_{m=l}^n \dots \dots \sum_{\lambda^N = \lambda^{N-1}}^n w_{ijk\dots z\dots\lambda^N} x_i x_j x_l x_m x \dots \dots x_{\lambda^N} \end{array} \right. \quad (3.31)
 \end{aligned}$$

The somatic operation provides a nonlinear mapping of the aggregated signal ‘ v ’ yielding an output signal ‘ y_n ’. Mathematically, the neural output y_n can be represented as follows $y_n = \phi[v]$. Equation (3.31) is the generalised representation of the higher-order synaptic operation for the higher-order neural units. The neuron fires a signal only when the aggregated signal exceeds certain threshold associated with the neuron. However for mathematical representation, the threshold can be shifted to the aggregation operation. Now, it can be shown that many existing neural structures can be derived from this generalized representation of the higher-order synaptic operation. This is only possible when the bias is associated with the synaptic operation.

$$(v)_N = \sum_{i=0}^N \sum_{j=1}^N \sum_{l=1}^N \sum_{m=1}^N \dots \dots \dots \sum_{p=N}^N \sum_{q=N}^N w_{ijlm} \dots \dots \dots x_i x_j x_l x_m x \dots \dots \dots x_p x_q$$

3.5 Summary

Neural networks have undoubtedly been biologically inspired, but the close correspondence between them and the real neural systems is still rather weak. Vast discrepancies exist between both the architectures and capabilities of artificial and natural neural networks. Despite the loose analogy between artificial and natural neural systems, a new structure of a computational neuron called the quadratic neural unit (neural unit with QSO) was developed. The architecture and mathematical model of the neural unit with QSO has been presented. Neural unit with QSO incorporates linear as well as nonlinear combinations of neural inputs generated by the preprocessor. The neural unit with LSO is a subset of the neural unit with QSO and the higher-order neural units can be expressed using different combinations of the neural unit with QSO. The efficiency of the neural unit with QSO was greatly enhanced as the size of the weight matrix is reduced from $[n \times n]$ to $\left[\frac{n \times (n+1)}{2} \right]$, and $n = (N+1)$. An algorithm for updating the weights of the neural unit with QSO and an implementation scheme for the proposed learning algorithm have been presented. The concept of neural unit with QSO can be extended to develop any other higher-order neural units such as the neural unit with CSO and so on. The structure and mathematical details of the neural unit with CSO has been outlined. The learning and adaptation algorithm for the neural unit with CSO was presented. Due to their higher-order combination of the neural inputs, either the neural unit with QSO or

the neural unit with CSO can be trained to learn and control the unknown dynamic systems. A general methodology for developing the HONUs with higher-order synaptic operations is presented using sigma tuner and correlation operator.

CHAPTER 4

Applications of the Higher-Order Neural Units to Static Problems: Pattern Classification and Function Approximation

4.1 Introduction: Biological Motivation

Biological systems employ the principles and concepts of pattern recognition to perform complex tasks such as recognising different visual, temporal and logical patterns. Using a broad enough interpretation, it is easy to find diverse applications of pattern recognition in every intelligent activity. In psychology, pattern recognition is defined as the process in which the external signals are converted into meaningful perceptual experiences by the sense organs (Pavlidis 1977). However, in engineering, it is viewed as a classification problem, where an object is assigned to one of the many classes. A basic task for the majority of biological systems (human beings, animals) is to decide if a particular pattern is the same or different from another pattern. In fact, a significant proportion of the information that is absorbed by biological systems is presented in the form of patterns. The human visual system, for example, has to differentiate if a certain image represents a friend's face or that of a stranger. From an information-processing point of view, this task is achieved inherently by the neurons which are considered as the basic building blocks of the central nervous system (CNS). The human brain, the carbon based cognitive computing faculty, is based upon a different class of logic similar to fuzzy logic and soft computing whereas the silicon based digital computing machines are based on binary logic. In the late 1960's, the stimulation-response of the primitive neuron was modeled using the threshold logic (McCulloch and Pitts 1943). Since then, it has become a practice to implement the logic circuits using neural structures. This methodology helped better in understanding the basic concepts of neural networks applied to pattern recognition (Gupta and Rao 1994 and Gupta et al. 2003).

The classification problem needs *a priori* input data which may be generated by different mechanisms and the goal is to separate the data into various possible classes. The desired response is a set of arbitrary labels (a different integer is normally assigned

to each one of the classes), so every element of a class will share the same label. Since class assignments are mutually exclusive, a classifier needs a nonlinear mechanism such as an all-or-nothing switch to classify the patterns. At a higher level of abstraction, both the classification and the regression problems seek systems that transform inputs into desired responses. It is long old tradition/customary to utilize the Adaline and the LMS rule as pieces to build pattern classifiers (Lau 1992 and Principe et al. 2000). The Adeline can be applied for classification when the system topology is extended with a threshold as a decision device. However, there is no guarantee of good performance because the coefficients are being adapted to fit (in the least square sense) the data to the labels 1 and -1, and not to minimize the classification error. This is a simple example with only two classes. For the multiple-class case the results become even more fragile. The conclusion is that there is a need to develop a new methodology to study and design accurate classifiers.

In this chapter, neural implementation of basic logic circuits such as **OR**, **AND**, and **Exclusive-OR (XOR)** are described using the neural unit with quadratic synaptic operation (QSO). The structure and the learning algorithm of neural unit with QSO for realizing the logic circuits are proposed in Sections 4.2 and 4.3. Computer simulation studies for realizing the different logic circuits are presented in Section 4.4. A statistical explanation is provided for the critical analysis of the neural unit with QSO in Section 4.5. The concept of Mahalanobis distance is introduced and a modified form of it is presented in Sections 4.6 and 4.7. The simulation results of the logic circuits are analysed in Section 4.8 followed by a brief introduction to function approximation in Section 4.9.

4.2 Structure of the Neural Unit with QSO for Realizing the Logic Circuits

The structure of the neural unit with QSO for realizing the logic circuits is shown in Fig. 4.1. The synaptic connections of the proposed neural unit are weighted summation of linear and quadratic combinations of the neural inputs. The preprocessor of the neural unit with QSO generates the higher-order combination of the inputs inside the neuron. The preprocessed inputs (x_1^2, x_1x_2, x_2^2) determine how well the neural unit with QSO can generalize the patterns outside the training set. The structure of the neural unit with QSO

is not associated with any sort of dynamic features (lateral recurrence, self recurrence). However, the higher-order inputs of the neural unit with QSO when employed in a network describe the dynamic characteristics of the network. These inputs take advantage of predefined relationships between the input nodes of the neural unit. As explained in Chapter 3, Section 3.2.1, the neural unit with QSO requires less number of training passes to generalize the patterns behind transformations.

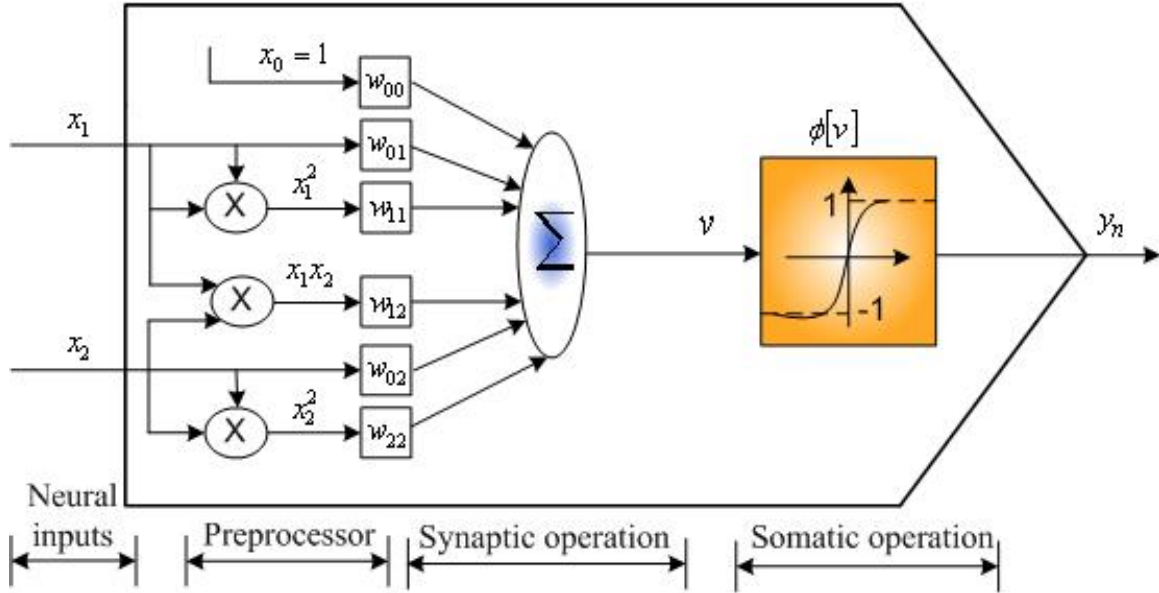


Figure 4.1 Schematic representation of the neural unit with QSO for realizing the logic circuits such as **OR**, **AND**, and Exclusive-OR (**XOR**).

From Chapter 3 Section 3.4, the mathematical model of the neural unit with QSO for realizing the logic circuits can be expressed as

$$v = \mathbf{x}_a^T \mathbf{W}_a \mathbf{x}_a \quad (4.1)$$

$$y_n = \phi[v] \quad (4.2)$$

where $\mathbf{x}_a = [x_0 \ x_1 \ x_2] \in R^3$, is the augmented vector of neural inputs including bias,

$x_0 = 1$, threshold (bias) of the neuron, and

$$\mathbf{W}_a = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ 0 & w_{11} & w_{12} \\ 0 & 0 & w_{22} \end{bmatrix} \text{ is the augmented synaptic weight matrix of the neural}$$

unit with QSO and $w_{00} = 1$ (dc gain).

Before proceeding further, it is important to formulate the problem which evaluates the developed concept of the neural unit in a simple way. Any measurement can be thought of as a point in space called the pattern or the input space. The natural distribution of points, any deviation of these points from normalancy, leads to the definition of classes or category regions in the pattern space. The goal of the pattern recognition is to build machines called classifiers which will automatically assign measurements to the respective classes. A natural way to make the class assignment is to define the boundary called the decision surface. The decision surface is not trivially determined for many real world problems. It varies from time to time, from pattern to pattern and changes for the same pattern depending on the hour of the day, the subject's state and so on. Thus the central problem in pattern-recognition is to define the shape and placement of the decision boundary so that the class-assignment errors are minimized. A learning and adaptive algorithm minimizes the error to give the optimal solution. Here the word optimal doesn't necessarily mean good performance. It simply provides the best possible performance with the available data. The next section focuses on developing the learning and adaptive algorithm for realizing the logic circuits.

4.3 Learning Algorithm for Realizing the Logic Circuits

Many concepts and algorithms have been developed to mimic the learning process of the biological neural networks (Rosenblatt 1959, Minsky and Papert 1969, Fukushima 1983, and Grossberg 1988). Consider the neural system as depicted in Fig. 4.2. The system has both learning and adaptation algorithm. The output of the logic function (desired model) is a scalar-desired output which continuously varies with the changes in the input. The objective of this scheme is to find a neural output $y_n(k)$ which is almost equivalent to the desired logic function $y_d(k)$. This can be defined in terms of the error $e(k)$, as the difference between the desired response $y_d(k)$ and the neural response $y_n(k)$. Thus, the error is defined as

$$e(k) = y_d(k) - y_n(k) \quad (4.3)$$

Ideally, the error should approach zero with increasing time (learning iteration). However, in practical situations, it is only possible to minimize the error which is expressed in

terms of performance index. For this purpose, an error function $J[e(k)]$ is defined as expected value of some even function of the error. One of the methods is to minimize the performance index with respect to the weights of the neural network. Based on this principle, a learning and adaptive algorithm was developed in Chapter 3 to modify the neural unit with QSO parameters. From Equations (3.8 and 3.13), the parameters of neural unit with QSO are updated based on the following algorithm; that is, the gradient vector associated with the augmented weight matrix is given as

$$\Delta \mathbf{W}_a(k) = \mu e(k) \phi'[v(k)] \mathbf{x}_a^T(k) \mathbf{x}_a(k)$$

and the adaptation algorithm is given as

$$\mathbf{W}_a(k+1) = \mathbf{W}_a(k) + \mu e(k) \phi'[v(k)] \mathbf{x}_a^T(k) \mathbf{x}_a(k) \quad (4.4)$$

where $\mu > 0$ is the adaptability rate in the learning scheme. For realization of logic circuits, the slope of the nonlinear activation function $\phi'[v(k)]$ is considered as a constant value. The implementation of the learning and adaptation algorithm of the neural unit with QSO is shown in Fig. 4.2.

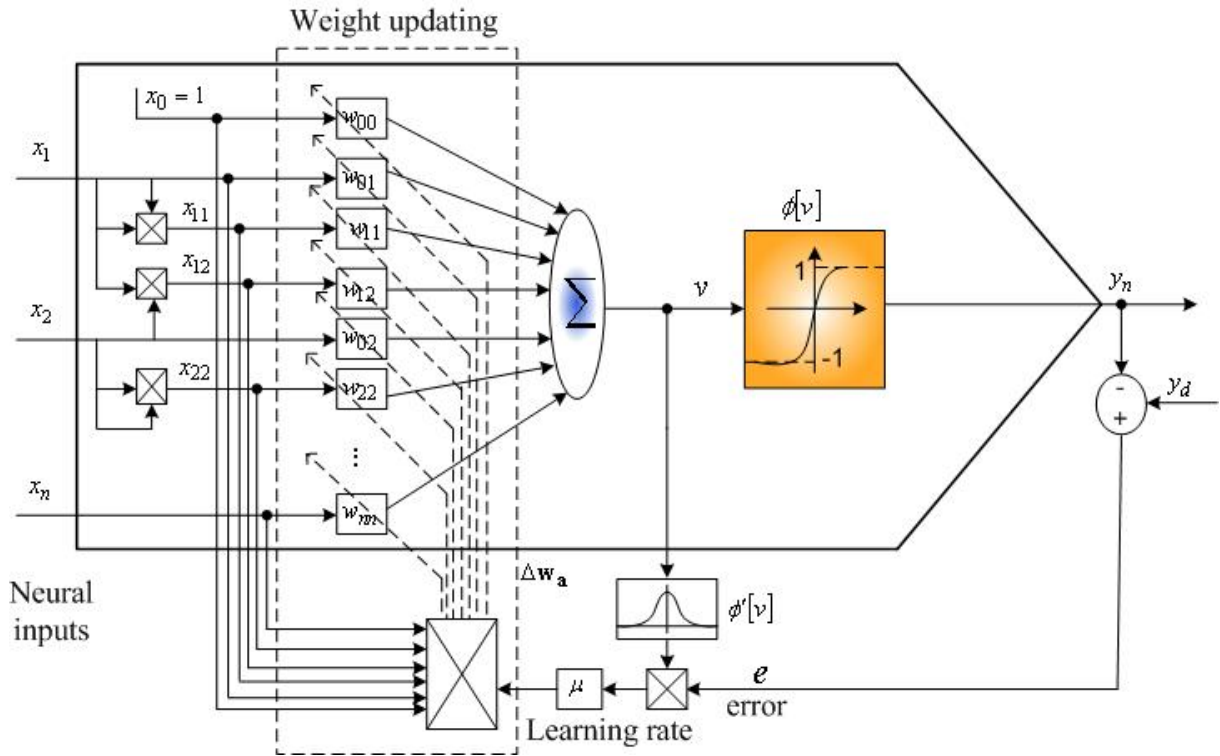


Figure 4.2 Learning and adaptation scheme for the realization of logic circuits.

4.4 Realization of Logic Circuits using Neural Units with Quadratic Synaptic Operation (QSO)

In this section, neural implementations of binary logic circuits are discussed. Conventionally, binary logic operations are given for unipolar values 0's and 1's. Without loss of generality, these operations can be extended for bipolar inputs $\{-1, 1\}$ as well. In many important practical applications, the signals obtained from the external world through measurements, are bipolar. The variables such as temperature and voltage are considered as bipolar signals. There are two simple transformation equations that convert unipolar signals to bipolar signals and vice versa. The equations are given as

- (i) Unipolar to bipolar: $\beta(k) = 2\alpha(k) - \gamma$, $\beta(k) \in [-\gamma, \gamma]$ and
- (ii) Bipolar to unipolar: $\frac{1}{2}[\beta(k) + \alpha]$, $\alpha(k) \in [0, \gamma]$

where $\alpha(k)$, and $\beta(k)$ are two classes of signals. Thus, for two neural inputs, $x_1(k)$ and $x_2(k)$ an output, $y(k)$ is defined as a logical combination of $x_1(k)$ and $x_2(k)$ as

$$y(k) = f[x_1(k), x_2(k)] \quad (4.5)$$

where $f[\cdot]$ is a logical function usually defined as a combination of various logical operations such as **OR**, **AND**, **NOT** etc. These are well developed logic circuits which are extensively used in digital computers and control mechanisms. It is of interest to show that solving these logic circuits is equivalent to finding the decision surfaces in the pattern space such that the given data patterns are located on the decision surfaces (Gupta et al. 2003). Since the concept of higher-order neural synaptic operation is developed mainly for the nonlinear problems, the logic circuits are realized as follows: Exclusive-OR (**XOR**), **OR**, and **AND**.

4.4.1 Realization of XOR (Exclusive-OR) using a Neural Unit with Quadratic Synaptic Operation (QSO)

The **XOR** problem demonstrates the efficiency of a neural unit with QSO in providing the solution to a nonlinear classification problem. Since the **XOR** logic is not linearly separable, multilayered neural networks consisting of neural unit with LSOs are

required to classify the patterns. On the other hand, a single neural unit with QSO can solve the **XOR** problem using the quadratic function given by Eqn. (4.1). Consider a two variable **XOR** function defined as

$$f(x_1, x_2) = x_1 \oplus x_2 \quad (4.6)$$

where $x_1, x_2 \in \{-1, 1\}$ are the bipolar binary inputs. Four bipolar learning patterns and the corresponding desired outputs are used to implement the **XOR** logic. The patterns are given in the Table 4.1.

Table 4.1 Truth Table for XOR Logic

Neural Inputs		Desired Outputs
x_1	x_2	$y_d = x_1 \oplus x_2$
-1	-1	-1: Class B1
-1	1	1: Class A1
1	-1	1: Class A2
1	1	-1: Class B2
Class A = Class A1 \cup Class A2 Class B = Class B1 \cup Class B2		

Now, consider a higher-order neural unit which is used to realize this logic function as

$$y = w_{00}x_0 + w_{01}x_1 + w_{02}x_2 + w_{12}x_1x_2 + w_{11}x_1^2 + w_{22}x_2^2 = 0 \quad (4.7)$$

and the desired response $y_d(k)$ is given by

$$y_d(k) = \text{sgn}[y] \quad (4.8)$$

The following sets of equations are obtained by substituting the neural input values from the Table.4.1 in Eqn. (4.7):

$$\begin{cases} w_{00} - w_{01} - w_{02} + w_{12} + w_{11} + w_{22} < 0 \\ w_{00} - w_{01} + w_{02} - w_{12} + w_{11} + w_{22} > 0 \\ w_{00} + w_{01} - w_{02} - w_{12} + w_{11} + w_{22} > 0 \\ w_{00} + w_{01} + w_{02} + w_{12} + w_{11} + w_{22} < 0 \end{cases} \quad (4.9)$$

The set of equations given by the Eqn. (4.9) are indeterminate. Therefore there is no unique solution; that is, there is more than one solution that can satisfy the desired logic function. It is easy to implement the Eqns. (4.7 and 4.8) through simulation studies but finding the analytical solution is a difficult task. However, the above equations can be solved if Eqn. (4.9) is expressed in a matrix form by taking $\text{sgn}(\text{Eqn. 4.9})$ as

$$\begin{bmatrix} \begin{bmatrix} +1 & -1 & -1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & +1 \\ +1 & +1 & -1 & -1 & +1 & +1 \\ +1 & +1 & +1 & +1 & +1 & +1 \end{bmatrix}_{(4 \times 6)} & \begin{bmatrix} w_{00} \\ w_{01} \\ w_{02} \\ w_{12} \\ w_{11} \\ w_{22} \end{bmatrix}_{(6 \times 1)} \end{bmatrix} = \begin{bmatrix} -1 \\ +1 \\ +1 \\ -1 \end{bmatrix}_{(4 \times 1)} \quad (4.10)$$

The above equations have the solution which is given as

$$w_{00} + w_{11} + w_{22} = 0, \quad w_{01} = 0, \quad w_{02} = 0, \quad w_{12} = -1,$$

and

$$y = -x_1 x_2$$

Hence the logic function can be implemented by a simple second-order polynomial. For simulation studies, the desired bipolar outputs are generated by means of a second-order polynomial function which is modified as

$$y_d(k) = -\text{sgn}(x_1 \cdot x_2 - 0.5) \quad (4.11)$$

The bias is added to the desired function in order to avoid the simulation glitch; that is, if the neural inputs x_1 and x_2 take value '0' during the simulation, the sgn of the function cannot be determined without bias. To overcome this simulation glitch, the bias is added to the logic function. The range of bias is between -1 and 1 as the maximum value of $x_1 x_2 = \pm 1$. The structure of the neural unit with QSO for implementing the **XOR** logic is shown in Fig. 4.1. The neural output is a function of weighted summation of linear and quadratic combination of the inputs. Since the logic function is a second-order polynomial, several nonlinear decision boundaries (hyperbolic, elliptical or parabolic) exist that can separate the patterns of the different logic circuits. The orientation, placement and geometry of the decision boundaries depend on the initial values of the synaptic weights (Specht 1967, Widrow and Michael 1992). This realization ability of the

neural unit with QSO is superior to the ability of a neural unit with linear synaptic operation (LSO) as the latter cannot achieve the nonlinear classification.

4.4.1.1 Simulation Results for the XOR Logic

The learning scheme proposed for training the neural unit with QSO is shown in Fig. 4.2. The adaptation algorithm was implemented using the SIMULINK toolbox in MATLAB 6.5. The initial values of the synaptic weights were generated using a random function (*randn*) in MATLAB 6.5. The convergence speed of a learning process depends on the choice of the learning rate as well as on the choice of the initial weight values (Guptal et al. 2003). Since the **XOR** logic has more than one solution, classification of patterns as class A or class B depends on the orientation of the nonlinear decision boundary. In this section, two non-linear boundaries were described classifying the patterns for the **XOR** logic.

The learning algorithm is developed based on the backpropagation theorem. Backpropagation training with too small a learning rate will make slow progress while too large a learning rate will proceed much faster. This may simply produce oscillations which results in relatively poor solutions. Both of these conditions are generally detectable through experimentation and sampling of results after a fixed number of training epochs. Typical values for the learning rate parameter are numbers between 0 and 1; that is, $0.05 < \mu < 0.75$. A learning rate $\mu = 0.03$ was chosen for the simulation studies using the trial and error method. There is one more factor that needs to be taken care of during backpropagation; that is, initialization of weights in the neuron. *Random initial state* - unlike many other learning systems, the neural network begins in a random state. The network weights are initialized to some choice of random numbers with a range typically between -1 and 1.

From the simulation studies, it is observed that the error converged to zero after 200 iterations. Table 4.2 shows the synaptic weights for the zeroth and the 200th iteration. Figures 4.3 and 4.5 show the convergence of the error $e(k)$ with each learning iteration k . This indicates that the neural output $y_n(k)$ closely followed the desired output $y_d(k)$. The nonlinear decision boundaries (hyperbolic, elliptical) separating the patterns were shown

in Figs. 4.4, 4.6, and 4.8. It is presumed that the pattern space of class A and B is clustered around the points {Class A: $[(-1, 1) (1, -1)]$ and Class B: $[(-1, -1) (1, 1)]$ }.

Table 4.2 Initial and Final values of the Synaptic Weights for the XOR Logic

Iterations			Synaptic weights						Boundary equations and type of boundaries	
Case	k		w_{00}	w_{01}	w_{02}	w_{11}	w_{12}	w_{22}		
I	Initial	0	-.773	.818	.748	.793	-.525	.369	$v = -0.902x_0 + -7.8e-5x_1 + 1.1e-4x_2 + .6632x_1^2 + -.996x_1x_2 + .2392x_2^2$	Ellipse
	Final	170	-.902	-7.8e-5	1.1e-4	.6632	-.996	.2392		
II	Initial	0	.847	.397	.779	-.996	-.961	-.803	$v = 1.164x_0 - 6.8e-5x_1 + 6.1e-5x_2 - .678x_1^2 + -.999x_1x_2 + -.485x_2^2$	Hyperbola
	Final	135	1.164	-6.8e-5	6.1e-5	-.678	-.999	-.485		
III	Initial	0	.323	-.87	-.153	.977	.031	-.332	$v = 3.05e-4x_0 - 2.45e-4x_1 + 3.18e-5x_2 + .6543x_1^2 - .9997x_1x_2 - .6547x_2^2$	Hyperbola
	Final	230	3.05e-4	-2.4e-4	3.18e-5	.6543	-.999	-.657		

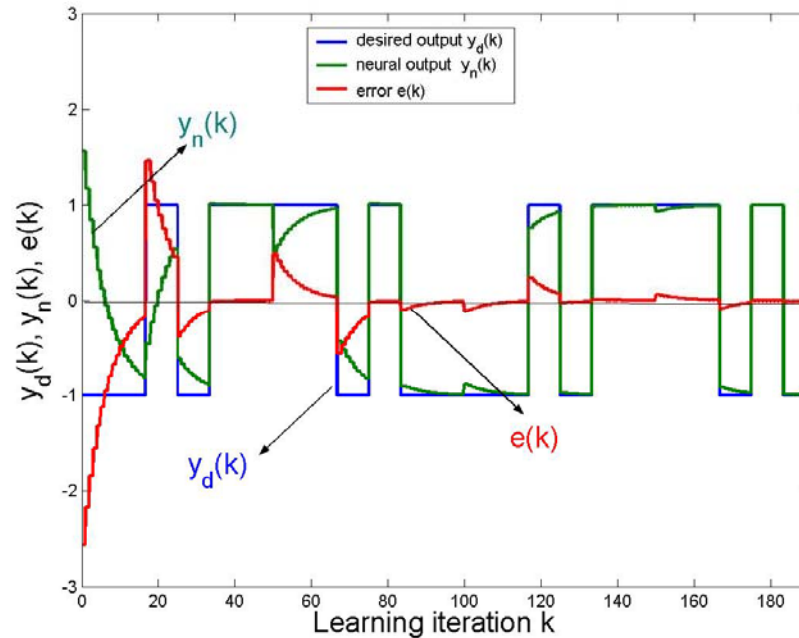
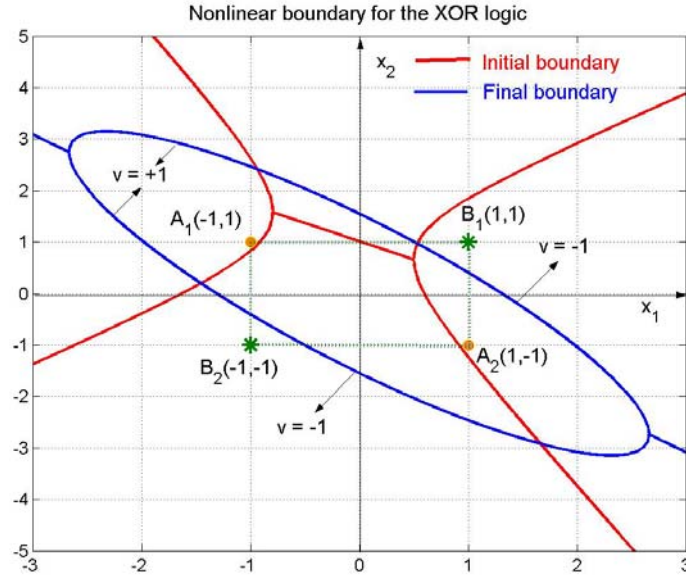


Figure 4.3 Desired output, $y_d(k)$ neural output, $y_n(k)$ and error $e(k)$ with the learning iteration k .



$$v = -0.902x_0 - 7.8e-5x_1 + 1.1e-4x_2 + .6632x_1^2 + -.996x_1x_2 + .2392x_2^2$$

Figure 4.4 Hyperbolic boundary separating the patterns belonging to Class A and Class B for **XOR** logic with a single neural unit with QSO.

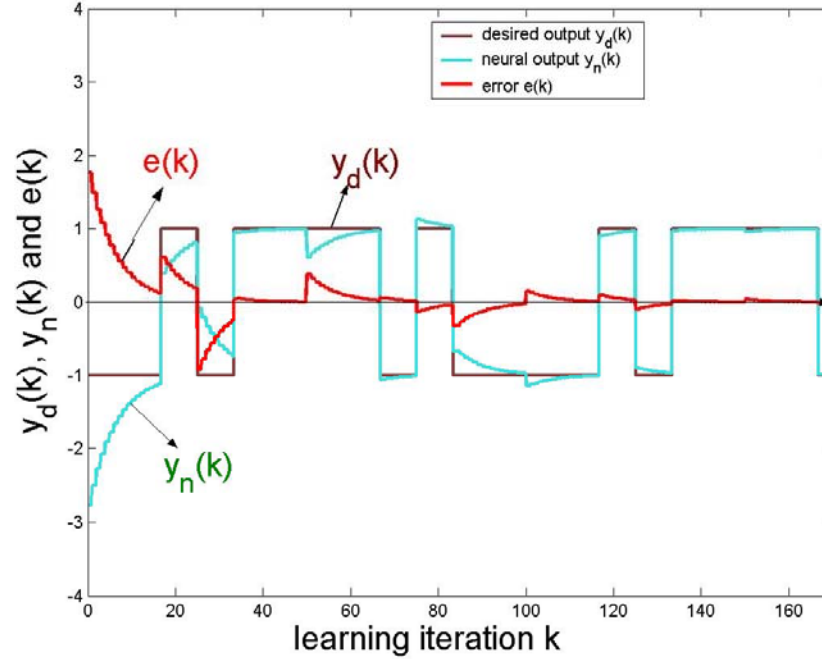
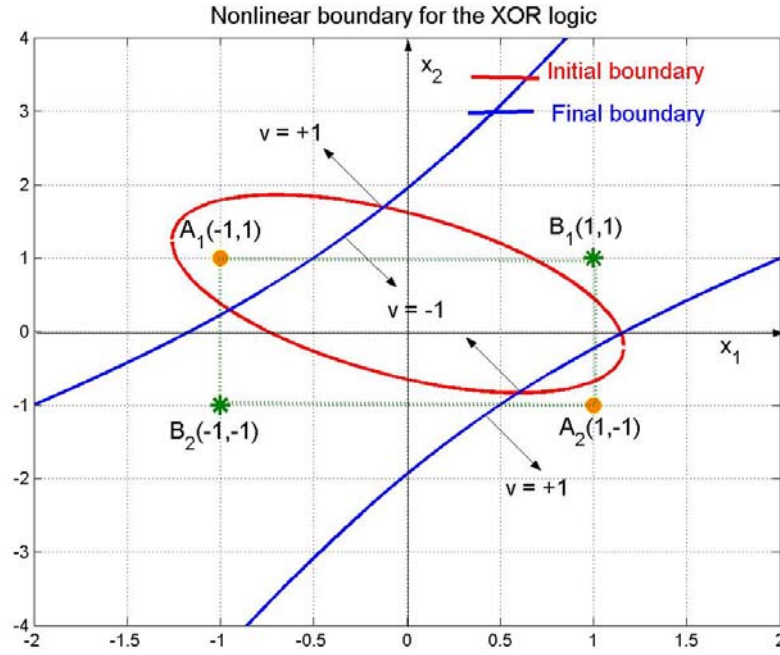


Figure 4.5 Desired output, $y_d(k)$ neural output, $y_n(k)$ and error $e(k)$ with the learning iteration k .



$$v = 1.164x_0 - 6.8e-5x_1 + 6.1e-5x_2 - 6.78x_1^2 + -.999x_1x_2 + -.485x_2^2$$

Figure 4.6 Elliptical boundary separating the patterns belonging to Class A and Class B for the **XOR** logic with a single neural unit with QSO.

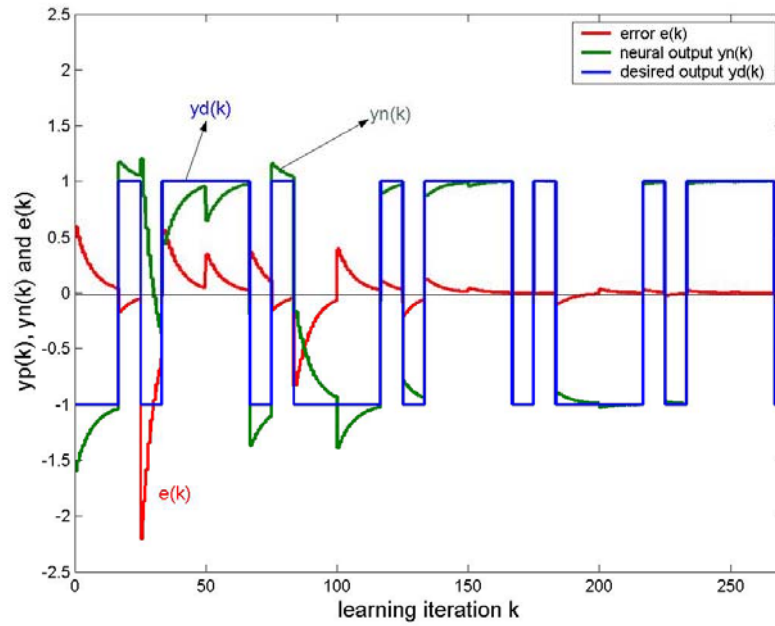
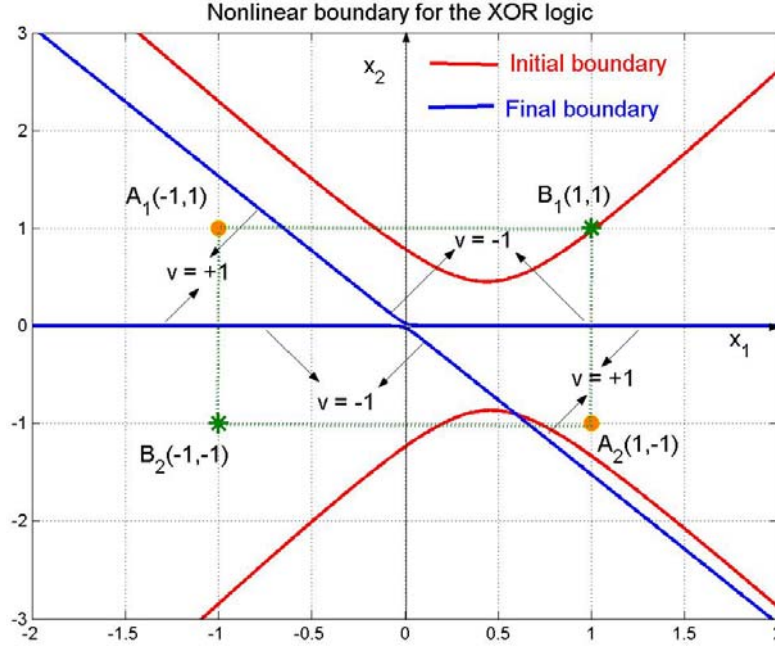


Figure 4.7 Desired output, $y_d(k)$ neural output, $y_n(k)$ and error $e(k)$ with the learning iteration k .



$$v = 3.05e-4x_0 - 2.45e-4x_1 + 3.18e-5x_2 + .6543x_1^2 - .9997x_1x_2 - .6547x_2^2$$

Figure 4.8 Hyperbolic boundary separating the patterns belonging to Class A and Class B for the **XOR** logic with a single neural unit with QSO.

4.4.2 Realization of OR and AND Logic Circuits using a Neural Unit with QSO

The neural unit with QSO is the most general neural unit that can generate higher and lower order neural units with different combinations. Therefore, neural unit with QSO can be used for linear as well as nonlinear separable forms of the pattern classification problems. In this section, the neural unit with QSO was implemented to realize the linearly separable **OR** and **AND** logic circuits. The truth tables for both the logic circuits are given in Table 4.3. The desired bipolar outputs for **OR** and **AND** are generated by mathematical logic functions which are given as

$$\mathbf{OR} : y_d(k) = \text{sgn}(x_1 + x_2 + 1) \quad (4.12)$$

$$\mathbf{AND} : y_d(k) = \text{sgn}(x_1 + x_2 - 1) \quad (4.13)$$

The learning algorithm developed in Chapter 3, Section 3.7 was implemented to separate the patterns for the **OR** and **AND** logic circuits. The neural unit with QSO generated nonlinear discriminant surfaces, shown in Figs. 4.9, 4.10 and 4.11 to separate the patterns

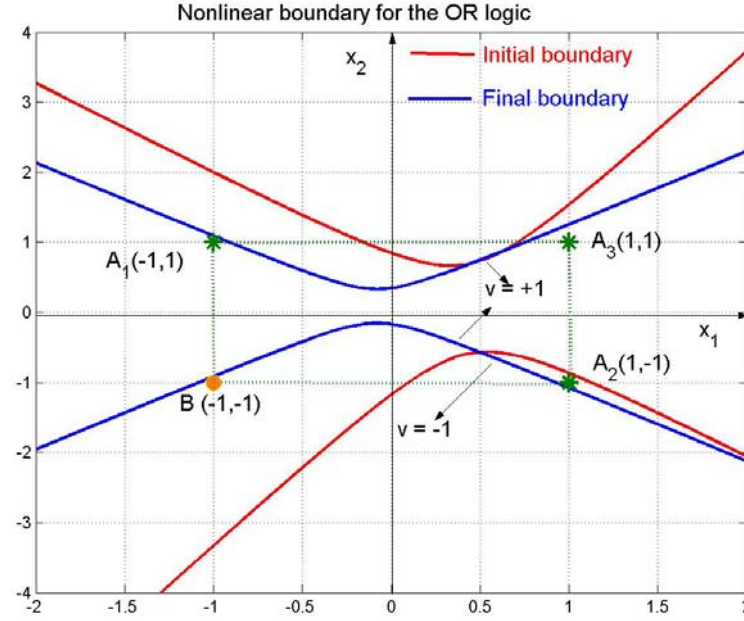
of the **OR** and **AND** logic. Table 4.4 shows the initial and final values of the synaptic weights for learning iterations $k=0$ and $k=200$.

Table 4.3 Truth Table for OR and AND Logic

Neural inputs		Desired outputs	
x_1	x_2	$y_d = x_1 \text{ OR } x_2$	$y_d = x_1 \text{ AND } x_2$
-1	-1	-1: Class B	-1: Class B2
-1	1	1: Class A1	-1: Class B1
1	-1	1: Class A2	-1: Class B3
1	1	1: Class A3	1: Class A

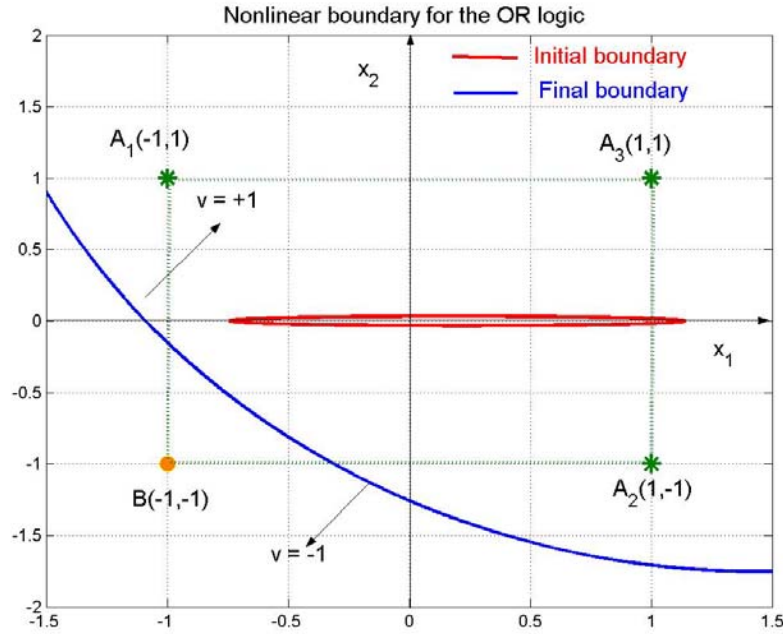
Table 4.4 Initial and Final values of the synaptic weights for the OR and AND Logic

OR Logic										
Iterations			Synaptic weights						Boundary equations and type of boundaries	
Case	k		w_{00}	w_{01}	w_{02}	w_{11}	w_{12}	w_{22}		
I	Initial	0	3	-8	-1	9	3	-3	$v = 0.333x_0 + 1.000x_1 + .999x_2 + 6.336x_1^2 - 8.6e-4x_1x_2 - 5.664x_2^2$	Hyperbola
	Final	150	0.333	1.00	0.999	6.336	-8.6e-4	-5.664		
II	Initial	0	.847	.397	.779	-.996	-.961	-.803	$v = 1.497x_0 + 1.000x_1 + .999x_2 - .3453x_1^2 - 2.3e-4x_1x_2 - .1523x_2^2$	Parabola
	Final	140	1.497	1.00	0.999	-.3453	-2.3e-4	-.1523		
AND Logic										
I	Initial	0	3	-8	-1	9	3	-3	$v = -0.330x_0 + .999x_1 + .999x_2 + 5.667x_1^2 - 0.001x_1x_2 - 6.332x_2^2$	Hyperbola
	Final	150	-0.33	0.999	0.999	5.667	-0.001	-6.332		



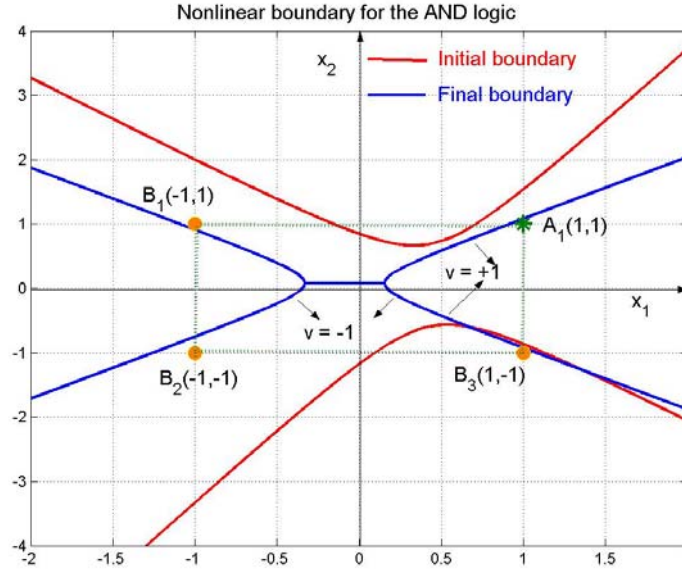
$$v = 0.333x_0 + 1.000x_1 + .999x_2 + 6.336x_1^2 - 8.6e-4x_1x_2 - 5.664x_2^2$$

Figure 4.9 Inverted hyperbolic boundary separating the patterns belonging to Class A and Class B for OR logic with a single neural unit with QSO.



$$v = 1.497x_0 + 1.000x_1 + .999x_2 + -.3453x_1^2 - 2.3e-4x_1x_2 - .1523x_2^2$$

Figure 4.10 Parabolic boundary (nonlinear) separating the patterns belonging to Class A and Class B for OR logic with a single neural unit with QSO.



$$v = -0.330x_0 + .999x_1 + .999x_2 + 5.667x_1^2 - 0.001x_1x_2 - 6.332x_2^2$$

Figure 4.11 Hyperbolic boundary separating the patterns belonging to Class A and Class B for AND logic with a single neural unit with QSO.

4.5 How does the Neural Unit with QSO provide a Better Solution than the Neural Unit with LSO?

Statistical techniques and neural networks are widely used for classification in various pattern recognition problems. Statistical classifiers include linear discriminant function (LDF), quadratic discriminant function (QDF) etc. for classifying the patterns (Liu et al. 2004). The neural unit with QSO belongs to the class of optimal classifiers which are developed based on the statistical models of data. Statistical decision theory proposes very general principles to construct optimal classifiers. The basic function of the classifier is to make a decision. According to statistical pattern recognition, what matters for classification are the *a posteriori* probabilities $P(A_i/x)$, but they are generally unknown. The *a posteriori* probabilities are given by Eqn. 4.7

$$P(A_i/x) = \frac{p(x/A_i)P(A_i)}{P(x)} \quad (4.14)$$

Bayes' rule provides a way to estimate the *a posteriori* probabilities. Equation 4.14 provides a way to compute the *a posteriori* probability by multiplying the *a priori*

probability for the class, $P(A_i)$ with the likelihood, $p(x/A_i)$ that the data x was produced by the class A_i (Principe et al. 2000). The likelihood can be estimated from the data by assuming a probability density function (pdf). Normally the pdf is the Gaussian distribution which is given by Eqn. (4.15).

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{(x-\mu)^2}{\sigma^2}\right)\right) \quad (4.15)$$

where μ = mean and is given as $\frac{1}{\Gamma} \sum_{i=1}^{\Gamma} x_i$

σ^2 = variance and is given as $\frac{1}{\Gamma} \sum_{i=1}^{\Gamma} (x_i - \mu)^2$

Γ = Number of samples.

Using Baye's rule, it is easy to estimate *a posteriori* probabilities from the data as mean and variance are known. Fisher showed that the optimal classifier chooses the class A_i that maximizes the *a posteriori* probability $P(A_i/x)$ that the given sample x belongs to the class; that is, x belongs to class A_i if

$$P(A_i/x) > P(A_j/x) \text{ for all } j \neq i \quad (4.16)$$

The problem is that the *a posteriori* probability cannot be measured directly but Baye's rule provides the means to calculate it. The separation boundary is the point where the two *a posteriori* probabilities are identical. A general method to compute the Bayesian threshold is to substitute the likelihoods and find the value of x that gives us equal *a posteriori* probabilities. Figure 4.12 explicitly shows that the class assignment is not error free. In fact, the tail of the "Class A1 (say Orange)" likelihood extends to the right of the intercept point, and the tail of the "Class A2 (say Apple)" likelihood extends to the left of the decision line S. The error in the classification is given by the sum of the areas under these tails, so the smaller the overlap the better is the classification accuracy. The maximum *a posteriori* probability assignment minimizes the probability of the error and is therefore optimal. Intuitively, one can conclude that the classification error depends on three factors

- The distance between the cluster (class) centers (mean μ) for a given cluster variance σ^2
- The variance σ^2 of each cluster distribution
- The value of the Bayesian threshold

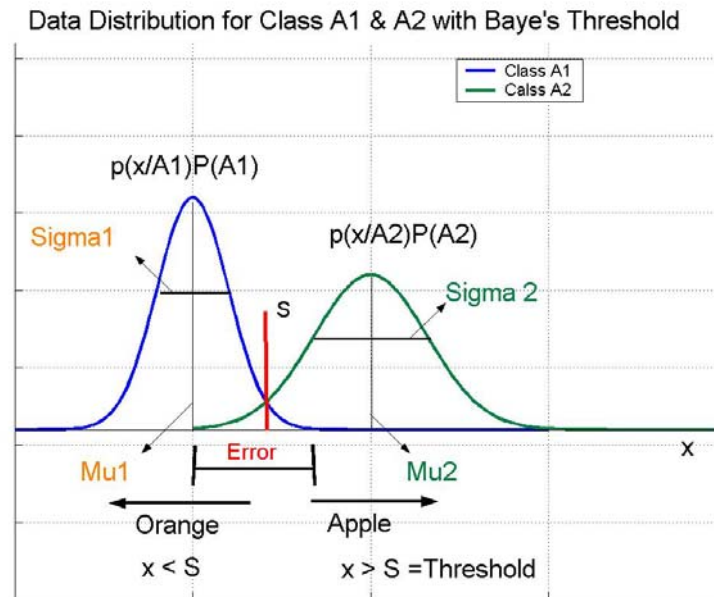


Figure 4.12 The probability density function (pdf) of two Classes A1 and A2 with Bayesian Threshold S.

If the distance between the cluster centers is larger for a given variance, the smaller the misclassification (minimum overlap) and the overall classification error. Likewise, for the same distance between the cluster means, the error is smaller if the variance of each cluster distribution is smaller. Since the decision region is a function of the threshold chosen, the error depends on the threshold also. It is clearly evident from the Fig. 4.12 that the error is associated with the area under the tails of the class distributions. The Bayesian threshold quantifies the maximum error that can occur during the misclassification. Thus it is concluded that the probability of the error depends on the cluster mean difference, the cluster variance and the Bayesian threshold. Ideally for pattern classification problems, it is desirable to have a minimum misclassification error (even zero if possible). The error can be minimized in two ways:

- The distance can be increased between the cluster centers and
- The second possibility is to vary the variance around the cluster centers

There is no assurance that increasing the distance between the class centers will minimize the error. The error can be minimized only when the tails of the pdf's (Gaussians) are controlled in Fig. 4.12; that is, the class variance around the class mean should be as small as possible. This implies that one can minimize the error with the Bayes rule by selecting the threshold in such a way that the *a posteriori* probability is maximized. Hence, the metric for the classification is a function of both the mean and the variance of each class. The placement of the decision surface is determined by the class distance normalized by the class variance. This distance is called the Mahalanobis distance (Principe et al. 2000). The Following section briefly discusses the Mahalanobis distance and a modified Mahalanobis distance that is formulated to support the concept of the neural unit with QSO.

4.6 Mahalanobis Distance

The Mahalanobis distance is the exponent of the multivariate Gaussian distribution (Principe et al. 2000), which is given by

$$p(x) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Omega|^{\frac{1}{2}}} \exp\left(-\frac{(x - \boldsymbol{\mu})^T \Omega^{-1} (x - \boldsymbol{\mu})}{2}\right) \quad (4.17)$$

where T indicates the transpose, $|\Omega|$ is the determinant of Ω , and Ω^{-1} the inverse of Ω . Note that in the equation $\boldsymbol{\mu}$ is a vector containing the data means in each dimension, i.e. the vector has dimension equal to D.

$$\boldsymbol{\mu} = \begin{bmatrix} \mu 1 \\ \mu 2 \\ \vdots \\ \mu D \end{bmatrix}$$

The covariance is a matrix of dimension D x D where D is the dimension of the input space. The matrix Ω is

$$\Omega = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1D} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{D1} & \sigma_{D2} & \cdots & \sigma_{DD} \end{bmatrix} \quad (4.18)$$

and its elements are the product of dispersions among sample pairs in the i^{th} and j^{th} coordinates (Γ is the number of samples in the data set):

$$\sigma_{ij} = \frac{1}{\Gamma - 1} \sum_{k=1}^{\Gamma} \sum_{m=1}^{\Gamma} (x_{i,k} - \mu_i)(x_{j,k} - \mu_j) \quad (4.19)$$

The covariance measures the variance among pairs of dimensions. Notice the difference in number of elements between the column vector m (D components) and the matrix Σ (D^2 components). The Mahalanobis distance is a normalized distance from the cluster center. The Mahalanobis distance is called M-distance for simplicity. The M-distance is shown in Fig. 4.14. It is obvious from Fig. 4.14 that, for classification, the dispersion of the samples around the cluster mean also affects the placement of thresholds for optimal classification. It is therefore reasonable to normalize the Euclidean distance (distance between cluster centers) by the sample dispersion around the mean, which is measured by the covariance matrix. The covariance matrix for each class is formed by the sample variance along pairs of directions in the input space. The covariance matrix measures the density of samples of the data cluster in the radial direction from the cluster center in each dimension of the input space. So, it quantifies the shape of the data cluster (Principe et al. 2000).

4.7 Modified Mahalanobis Distance (MM-Distance)

The objective of this thesis is to reduce the number of adaptable weights without sacrificing the neural performance. The number of parameters (weights) in the covariance matrix increases with the increase in dimensions of the input space. In order to reduce the number of parameters, there is a need to modify the M-distance equation without changing the concept of Mahalanobis. The structure of MM-distance is similar to the distance formula proposed by Mahalanobis except the numbers of elements in the formula are reduced significantly. Consider the elements of the covariance matrix Ω given by Eqn. (4.18)

$$\Omega = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1D} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{D1} & \sigma_{D2} & \cdots & \sigma_{DD} \end{bmatrix}$$

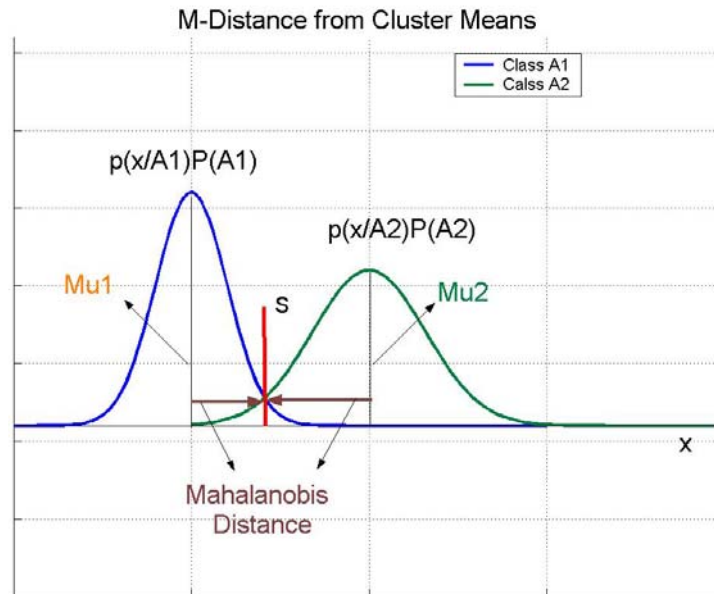
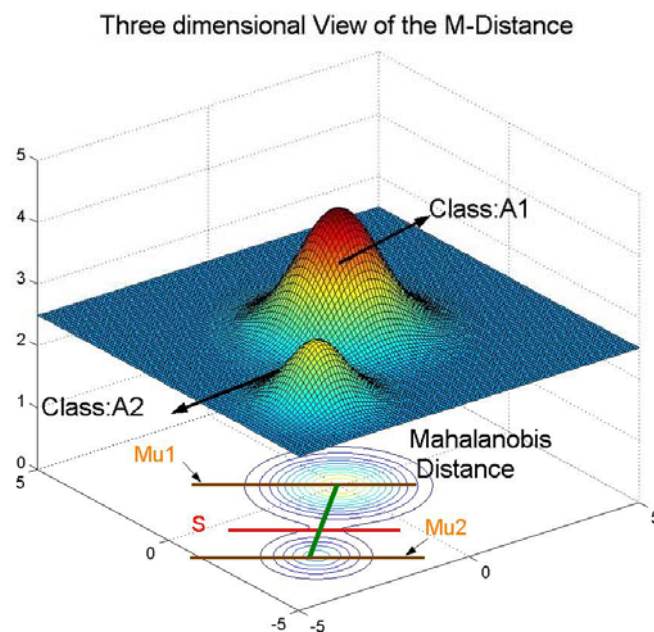


Figure 4.13 Mahalanobis distance (M-distance) from Class A1 and A2.



Mu1, Mu2 : Means of the two classes A1 & A2

S : Decision surface

Distance between S & Mu1, S & Mu2 : M- Distance

Figure 4.14 Three dimensional view of the M-distance.

The covariance matrix for each class is formed by the sample variance along pairs of directions in the input space. For two dimensional problems; that is, Class A1 and Class A2, the covariance matrix for each class are

$$\Omega_{A1} = \Omega_{A2} = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}_{2 \times 2}$$

The covariance matrix measures the density of samples of the data cluster in the radial direction from the cluster center in each dimension of the input space. So, it quantifies the shape of the data cluster. A careful observation to the covariance matrix reveals that the element σ_{12} is same as the element σ_{21} . This holds good even for the D dimensions of the elements in the input space. So, the modified covariance matrix for the two dimension problem is given as

$$\Omega = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ 0 & \sigma_{22} \end{bmatrix}_{2 \times 2}$$

and for D-dimensions

$$\Omega = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1D} \\ 0 & \sigma_{22} & \cdots & \sigma_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \sigma_{DD} \end{bmatrix}_{D \times D} \quad (4.19)$$

The covariance matrix is always symmetric and positive definite (because of quadratic form and inverse always exist). It is positive definite, that is, the determinant is always greater than zero. The diagonal elements are the variance of the input data along each dimension. The off-diagonal terms are the covariance along pairs of dimensions. It is stated earlier that the placement of the decision region depends on three factors; that is, the distance between the class centers, the variance of each class centers and the threshold. The covariance matrix encapsulates the affect of covariance beautifully, but ignores other two factors completely. Hence, it is reasonable to incorporate the threshold term (bias) and the cluster mean to precisely determine the placement of the decision surface. Then the MM-distance is given by the same Eqn. (4.17)

$$p(x) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Omega|^{\frac{1}{2}}} \exp \left(-\frac{(x - \mu)^T \Omega^{-1} (x - \mu)}{2} \right)$$

where T indicates the transpose, $|\Omega|$ is the determinant of Ω , and Ω^{-1} the inverse of Ω which is given by the Eqn. (4.20)

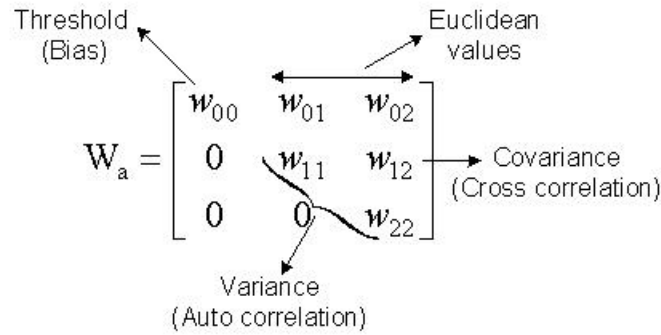
$$\Omega = \begin{bmatrix} \sigma_{00} & \sigma_{01} & \sigma_{02} & \dots & \sigma_{0D} \\ 0 & \sigma_{11} & \sigma_{12} & \dots & \sigma_{1D} \\ 0 & 0 & \sigma_{22} & \dots & \sigma_{2D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_{DD} \end{bmatrix}_{[(D+1) \times (D+1)]} \quad (4.20)$$

The first row of the modified covariance matrix encapsulates the affect of the bias and cluster mean. The elements of the covariance matrix are the product of dispersions among sample pairs in the i^{th} and j^{th} coordinates:

$$\sigma_{ij} = \frac{1}{\Gamma - 1} \sum_{k=0}^{\Gamma} \sum_{m=k}^{\Gamma} (x_{i,k} - \mu_i)(x_{j,k} - \mu_j) \quad (4.21)$$

The covariance matrix Ω is an upper or lower triangle matrix that provides the sufficient condition for the placement of decision surface. The modified covariance matrix incorporates the affect of the Euclidean distance between the cluster centers, the threshold and the dispersion of the samples around the cluster mean which affects the placement of the thresholds for optimal classification. Hence, the structure of the covariance matrix is critical for the placement and shape of the discriminant functions in pattern space. Since the distance metric for classification is normalized by the covariance, if the class means stay the same but the covariance changes, the placement and shape of the discriminant function will change. Finally, it is concluded that the modified covariance matrix (weight matrix of the neural unit with QSO) is associated with the following terms

- Threshold (bias);
- Distance of the cluster means from the decision boundary (Euclidian distance);
- Covariance's (cross-correlation) among pairs of dimensions above the diagonal elements; and
- Variances (auto-correlation) of the input data of each dimension along the diagonal.



4.8 Analysis of the Simulation Results

4.8.1 Exclusive-OR (XOR) Logic

The simulation results shown in Section 4.4 are for a particular value of an input signal whereas the following results shown are for different values of the input signals. The separation surface is given by the Eqn. 4.1, which yields the following equations for two different initial conditions

$$(a) : v = 1.098 + 0.242x_1 + 0.26x_2 - 0.745x_1^2 - 1.890x_1x_2 - 0.552x_2^2 = 0 \quad (4.22)$$

$$(b) : v = .947 + 0.497x_1 + 0.679x_2 - 0.896x_1^2 + 1.061x_1x_2 - 0.703x_2^2 = 0 \quad (4.23)$$

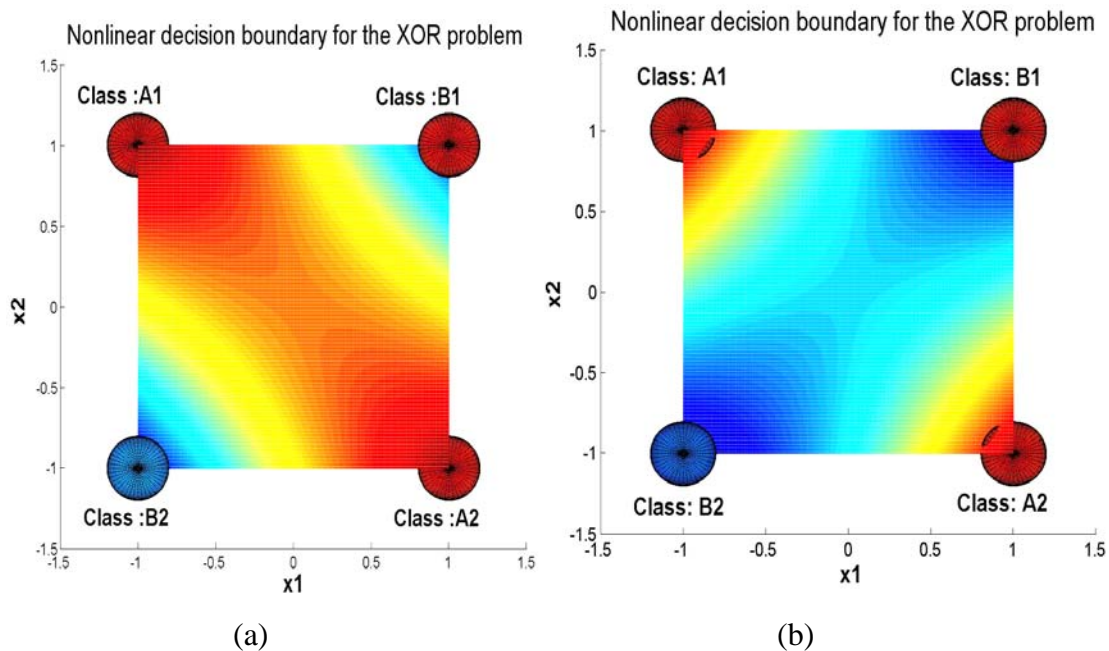


Figure 4.15 Nonlinear decision boundaries separating the patterns belonging to Class A and Class B for the **XOR** logic with a single neural unit with QSO.

Figure 4.15 show nonlinear decision boundaries separating the patterns belonging to Class A and Class B for the **XOR** logic with a single neural unit with QSO. From the above discussion, it was clear that the weight matrix must be symmetric and positive definite. The final weight matrices obtained for the **XOR** logic with different initial conditions are presented in Table. 4.5.

Table 4.5 XOR Data Analysis

Exclusive-OR (XOR)		
	Figure 4.15 : (a)	Figure 4.15 : (b)
Category	Nonlinear	Nonlinear
Inputs range	$-1 \leq x_1, x_2 \leq 1$	$-1 \leq x_1, x_2 \leq 1$
Output range	$x_1, x_2 \subseteq [-1, +1]$	$x_1, x_2 \subseteq [-1, +1]$
Initial Weight matrix	$\mathbf{W}_a = \begin{bmatrix} -0.847 & 0.397 & 0.779 \\ 0 & -0.996 & -0.961 \\ 0 & 0 & -0.803 \end{bmatrix}$	$\mathbf{W}_a = \begin{bmatrix} -0.773 & 0.818 & 0.748 \\ 0 & 0.793 & -0.525 \\ 0 & 0 & 0.369 \end{bmatrix}$
Final Weight matrix	$\mathbf{W}_a = \begin{bmatrix} 1.098 & 0.242 & 0.260 \\ 0 & -0.745 & -1.890 \\ 0 & 0 & -0.552 \end{bmatrix}$	$\mathbf{W}_a = \begin{bmatrix} 0.947 & 0.497 & 0.679 \\ 0 & -0.896 & 1.061 \\ 0 & 0 & -0.703 \end{bmatrix}$
Type	Symmetric	Symmetric
Determinant of the \mathbf{W}_a	0.4515	0.5965
Characteristic of the \mathbf{W}_a	Positive definite	Positive definite

The two equations are quadratic in 2-D space as shown in Fig. 4.15. In these cases the decision surface yields the smallest classification error for this problem. This may not necessarily mean good performance but it simply means the best possible performance with the chosen initial condition.

4.8.2 OR Logic

Figure 4.16 shows nonlinear decision boundaries separating the patterns belonging to Class A and Class B for the **OR** logic with a single neural unit with QSO. The separation surface is given by the Eqn. 4.1, which yields the following equations for two different initial conditions. The equations are

$$(a) : v = 0.333 + 1x_1 + 0.999x_2 + 6.336x_1^2 - 8.6e - 4x_1x_2 - 5.664x_2^2 = 0 \quad (4.24)$$

and

$$(b) : v = 1.497 + 1x_1 + 0.999x_2 - 0.3453x_1^2 - 2.3e - 4x_1x_2 - 0.1523x_2^2 = 0 \quad (4.25)$$

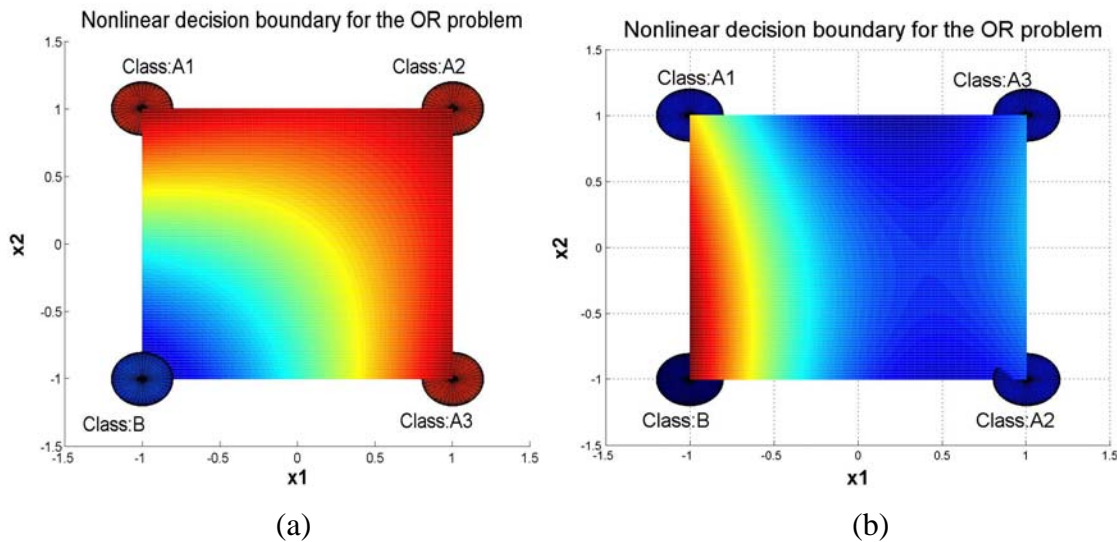


Figure 4.16 Nonlinear decision boundaries separating the patterns belonging to class A and class B for the **OR** logic with a single neural unit with QSO.

For the **OR** logic, the linear and nonlinear decision surfaces can provide the classification as it belongs to the class of linearly separable problems. It is important to stress that the linear discriminant is less powerful than the quadratic discriminant. A linear discriminant primarily utilizes differences in means for classification. If the two classes have the same mean, the linear classifier will always produce poor results. However, the quadratic

discriminant does a much better job because it can utilize the differences in the covariance. The following table summarizes the **OR** logic simulation using a single neural unit with QSO.

Table 4.6 OR Data Analysis

OR Logic Circuit		
	Figure 4.16 : (a)	Figure 4.16 : (b)
Category	Linear	Linear
Inputs range	$-1 \leq x_1, x_2 \leq 1$	$-1 \leq x_1, x_2 \leq 1$
Output range	$x_1, x_2 \subseteq [-1, +1]$	$x_1, x_2 \subseteq [-1, +1]$
Initial Weight matrix	$\mathbf{W}_a = \begin{bmatrix} -0.773 & 0.818 & 0.748 \\ 0 & 0.793 & -0.525 \\ 0 & 0 & 0.369 \end{bmatrix}$	$\mathbf{W}_a = \begin{bmatrix} 3 & -8 & -1 \\ 0 & 9 & 3 \\ 0 & 0 & -3 \end{bmatrix}$
Final Weight matrix	$\mathbf{W}_a = \begin{bmatrix} 1.497 & 1 & 0.999 \\ 0 & -0.3453 & -2.3e-4 \\ 0 & 0 & -0.1523 \end{bmatrix}$	$\mathbf{W}_a = \begin{bmatrix} 0.333 & 1 & 0.999 \\ 0 & 6.336 & -8.6e-4 \\ 0 & 0 & -5.664 \end{bmatrix}$
Type	Symmetric	Symmetric
Determinant of the \mathbf{W}_a	0.0787	-11.9504
Characteristic	Positive definite	Negative definite

4.8.3 AND Logic

Figure 4.17 shows nonlinear decision boundaries separating the patterns belonging to Class A and Class B for the **AND** logic with a single neural unit with QSO. The separation surface is given by the Eqn. 4.1, which yields the following equation

$$v = 0.7965 + 1.0248x_1 + 1.0248x_2 - 1.0465x_1^2 + .9860x_1x_2 - 0.8535x_2^2 = 0 \quad (4.26)$$

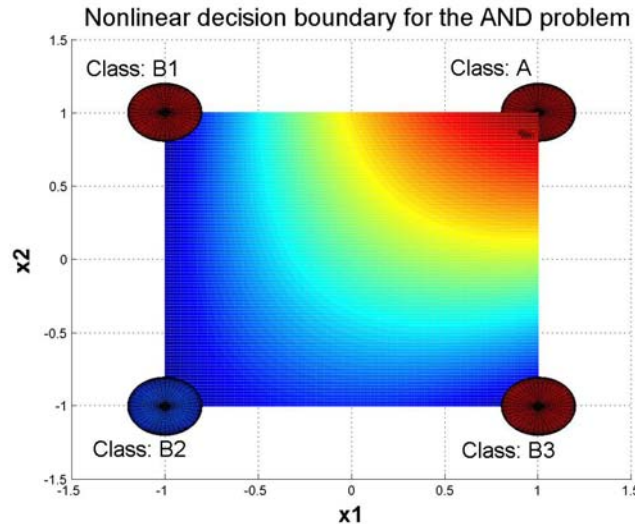


Figure 4.17 Nonlinear decision boundaries separating the patterns belonging to Class A and Class B for the **AND** logic with a single neural unit with QSO.

This is quadratic in 2-D space as shown in Fig.4.17. Both linear and nonlinear classifiers can provide the correct pattern classification but nonlinear classifier provides the best possible solution. The following table summarizes the **AND** logic simulation using a single neural unit with QSO.

Table 4.7 AND Data Analysis

AND Logic Circuit	
	Figure 4.17
Category	Linear
Inputs range	$-1 \leq x_1, x_2 \leq 1$

Output range	$x_1, x_2 \subseteq [-1, +1]$
Initial Weight matrix	$\mathbf{W}_a = \begin{bmatrix} -0.773 & 0.818 & 0.748 \\ 0 & 0.793 & -0.525 \\ 0 & 0 & 0.369 \end{bmatrix}$
Final Weight matrix	$\mathbf{W}_a = \begin{bmatrix} 0.7965 & 1.0248 & 1.0248 \\ 0 & -1.0465 & 0.9860 \\ 0 & 0 & -0.8535 \end{bmatrix}$
Type	Symmetric
Determinant of the \mathbf{W}_a	0.7114
Characteristic	Positive definite

The analysis of the logic circuits, **XOR**, **OR** and **AND**, strengthened the importance of the covariance matrix (weight matrix) as it decides the placement of the decision boundary for the classification of patterns. The sign of the determinant of the weight matrix determines the type of classification: good or poor classification. In the above simulation studies, the sign of the determinant of the weight matrix was always positive definite (PD) when the patterns were clearly separated and the determinant was negative definite (ND) when the patterns were misclassified. This was clearly evident in the simulation study of the **OR** logic circuit. In Figure 4.16 (a), the patterns belonging to Class A and Class B are clearly separated by the decision boundary whereas in Fig. 4.16 (b), one of the patterns belonging to Class A lies on the decision boundary. Intuitively, the next question that arises is that what happens when the determinant of the weight matrix is zero? There are two interesting possibilities for this question

- All the elements of the weight matrix can be zero
- All the elements of the weight matrix should be zero except the first row

The first possibility is beyond discussion and the second possibility is an interesting one. The weight matrix is a row vector; that is,

$$\mathbf{W}_a = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ or } \mathbf{W}_a = [w_{00} \quad w_{01} \quad w_{02}] \quad (4.27)$$

This implies that a neural unit with LSO is required for that specific pattern classification. The other interesting aspect of the above matrix is that the two classes are mutually exclusive as there is no cross and auto correlation terms associated with the inputs in the weight matrix. This boils down to another important conclusion that the choice of neural unit for a particular problem depends on the amount and type of data that is being processed by the neuron. Depending on these factors, one can decide before hand the type of neuron to be employed for a particular problem. This is technically termed as discriminant sensitivity to the size and type of data. Hence when the data is complex (mutually dependent and large data), it is always advisable to employ HONUs (neural unit with higher-order synaptic operations) for a better solution. This does not necessarily mean that the neural unit with LSOs can not provide a better solution but the difficulty is in deciding the number of neural units with LSOs that are required for obtaining the same solution.

Important Observations Regarding Decision Surfaces for Logic Circuits

- (i) The discriminant surfaces could be elliptical, parabolic and hyperbolic for separating the patterns of the **XOR** logic. If the decision surface is hyperbolic, the bias cannot exceed the numerical value $2\sqrt{2}$ because this is the critical condition for which the patterns lie on the surface as shown below. It may not be true for other logic circuits such as **OR** and **AND**.
- (ii) There can not be a circular discriminant surface of any radius for separating the patterns of the **XOR** logic. If a neural unit with quadratic synaptic operation provides a circular discriminant surface as one of the solution for the **XOR** logic then the elements other than the diagonal would be zero; that is, the weight matrix would have the elements as shown below

$$\mathbf{W}_a = \begin{bmatrix} w_{00} & 0 & 0 \\ 0 & w_{11} & 0 \\ 0 & 0 & w_{22} \end{bmatrix} \quad (4.28)$$

This implies that the weight matrix is a diagonal matrix and the diagonal elements are the eigen values of the quadratic synaptic operation but it was stated previous that the circle cannot be a solution. Hence an eigen value does not provide the solution for separating the patterns of the **XOR** logic.

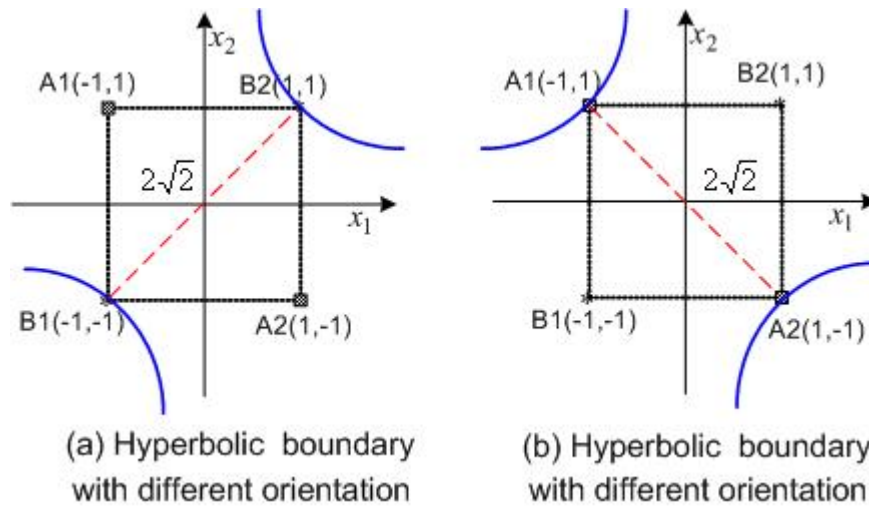


Figure 4.18 Critical bias for a hyperbolic decision boundary separating the patterns belonging to Class A and Class B

- (iii) The neural unit with quadratic synaptic operation always gives better performance as it finds the minimum in less number of iterations.

Based on these observations, it is prude to conclude the following discriminate surfaces for separating the patterns belonging to different basic logic circuits

Table 4.8 Discriminant solutions for classifying the patterns of basic logic circuits

S. No	Different basic logic circuits	Discriminant surfaces for separating the patterns belonging to the basic logic circuits
1	OR	Linear, Parabolic, Circular, Elliptical, and Hyperbolic
2	AND	Linear, Parabolic, Circular, Elliptical, and Hyperbolic
3	XOR	Parabolic, Elliptical, and Hyperbolic

4.9 Function Approximation

The problem of identification is a typical example of function approximation. One of the most significant characteristics of the neural networks is their ability to approximate any arbitrary nonlinear function to desired degree of accuracy. Neural networks potentially offer a general framework for modeling and control of nonlinear systems. The problem of learning a mapping from input and output space using neural networks is equivalent to the problem of estimating the system that transforms inputs and outputs given a set of examples of input-output pairs. Training a neural network using input-output data from a nonlinear dynamic system is considered as a problem of functional approximation.

Function approximation seeks to describe the behaviour of very complicated functions by ensembles of simpler functions. Recently, a number of researchers have shown that multilayer static (feedforward) neural networks (MFNNs) can approximate any continuous function to desired degree of accuracy. Either Stone-Weirstrass theorem or Kolmogorov theorem has been employed for the theoretical development of functional approximation capabilities of neural networks. Other important analytic tools such as *Series expansion*, *Trigonometric polynomials (Fourier expansion)* are also widely used as function approximators, but their computation is bit more involved. In this section, the universal approximation capabilities of the HONNs are studied using a single neural unit with QSO. Computer simulations are presented to demonstrate the functional approximation capabilities of the neural unit with QSO.

4.9.1 Simulation Studies

It is demonstrated in this section, through computer simulation studies, that the neural unit with quadratic synaptic operation as the basic computing node, can approximate any arbitrary nonlinear function. Let $f(\mathbf{x})$ be a real function of a real valued vector $\mathbf{x} = [x_1 x_2 \cdots x_n]^T$ that is *square integrable* over the real numbers R^n . The goal of the function approximation using the neural unit with QSO is to describe the behaviour of the function $f(\mathbf{x})$, in a compact area S of the input space, by a combination of simpler functions such that

$$\left| f(\mathbf{x}(k)) - \hat{f}(\mathbf{w}_a, \mathbf{x}(k)) \right| < \varepsilon \quad (4.29)$$

where ε can be made arbitrarily small. The function $\hat{f}(\mathbf{w}_a, \mathbf{x}(k))$ is called an approximant to $f(\mathbf{x}(k))$. The learning scheme employed for this task is shown in Fig. 4.19.

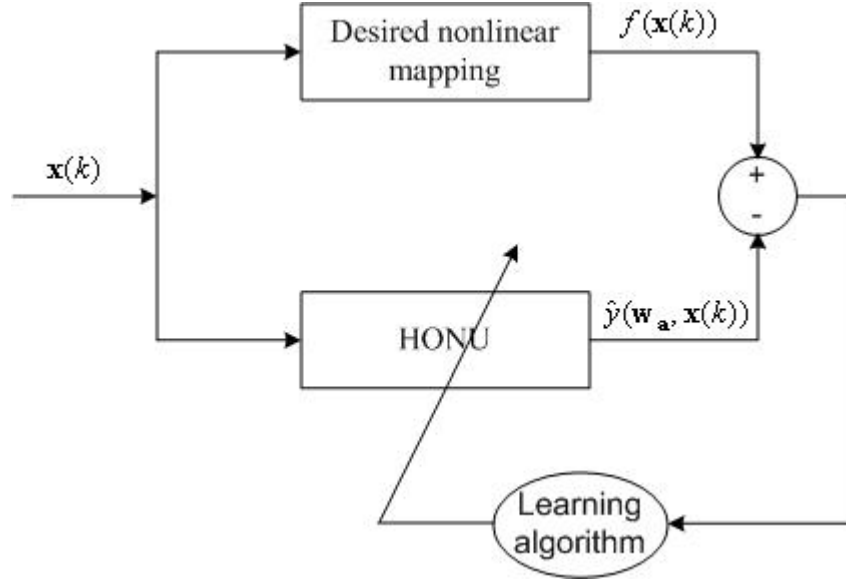


Figure 4.19 The learning scheme for functional approximation using a HONU.

For evaluating function approximation ability of the neural unit with QSOs, four simulation examples are discussed in this section. The task consists of learning a representation for an unknown one variable nonlinear function $f(\mathbf{x}(k))$. Examples 1 and 2 demonstrate the neural unit with QSOs ability to approximate arbitrary nonlinear functions as shown in Figs. 4.20 and 4.21. The simulation studies are carried out with different type of inputs such as sinusoidal, square, sawtooth and random signals.

The nonlinear functions used in these examples were as follows:

Example 1: $f(x(k)) = x(k)$, and

Example 2: $f(x(k)) = 0.5x(k) + 2x^2(k)$

The mapping function obtained by a single neural unit with QSO for different type of inputs is shown in Fig. 4.20. The error reduced to desired limit (0.001) with in 50 learning iterations. The approximation accuracy shown in Fig. 4.20 is extremely high by

the neural unit with QSO. This is an evidence for the high approximation ability of the proposed neural structure.

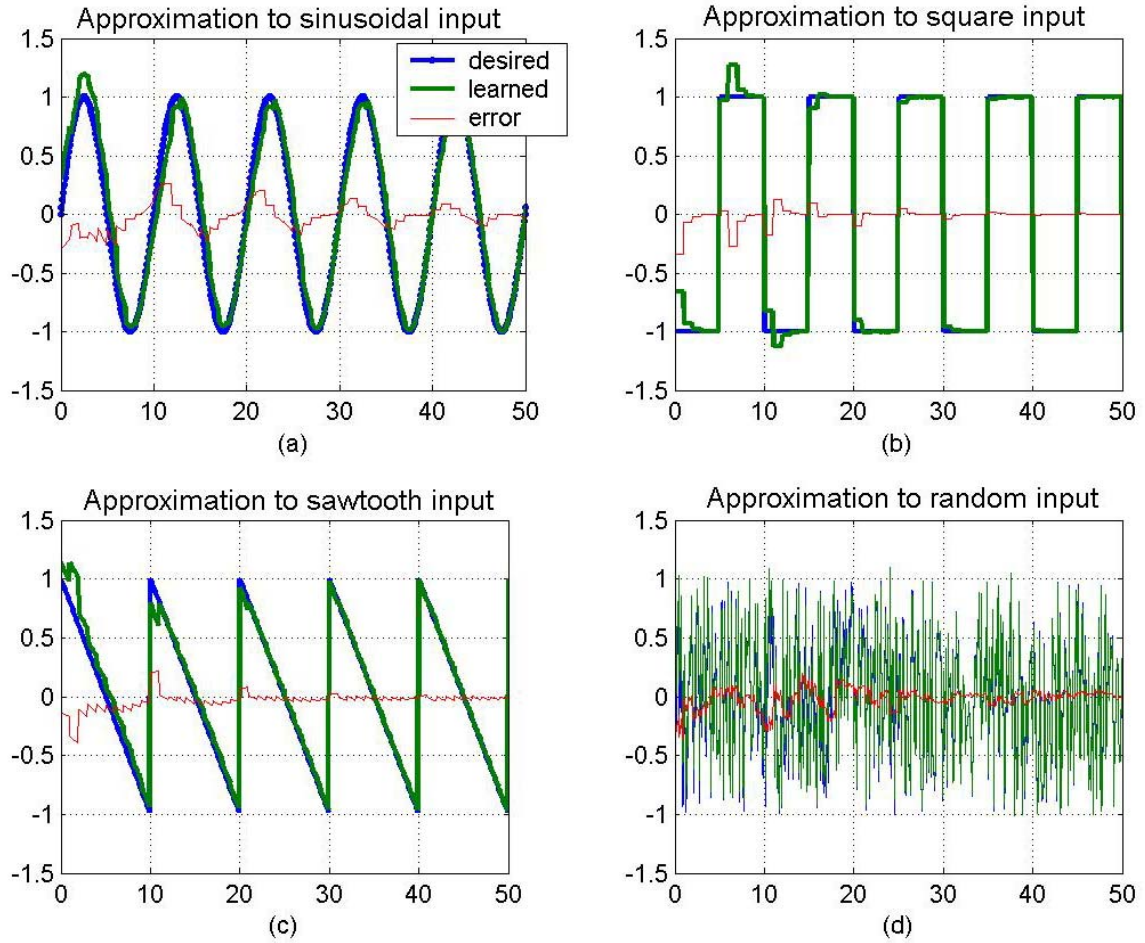


Figure 4.20 Function approximation of a nonlinear function ($f(x(k)) = x(k)$,) using a neural unit with QSO for Example 1 with different inputs such as sinusoidal, square, sawtooth and random signals.

In example 2, trigonometric functions were added to the desired nonlinear function to inspect the adaptability of the proposed neural unit. The simulation studies were shown in Fig. 4.21 for different inputs. In Examples 1 and 2, the neural unit with QSO took more learning iterations to approximate the desired nonlinear function when the input signal is random.

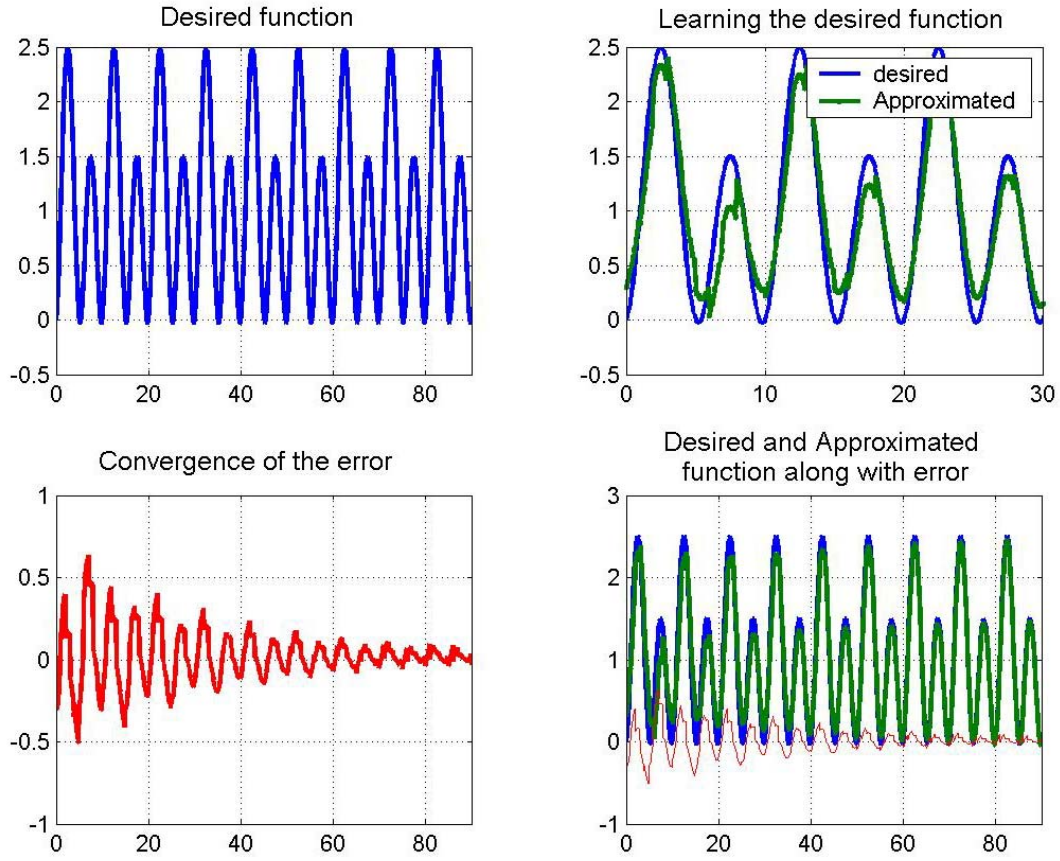


Figure 4.21 Function approximation of a nonlinear function using a neural unit with QSO for Example 2 ($f(x(k)) = 0.5x(k) + 2x^2(k)$) with a sinusoidal signal.

In Example 3, the desired nonlinear function was changed during the learning process to study the adaptiveness of the neural unit with QSO. The nonlinear functions used in this example were as follows:

$$f(x(k)) = \sin(2\pi k / 250) \quad \text{for } 0 \leq k \leq 500 \text{ and}$$

$$= \frac{0.5\mathbf{x}(k)}{\sqrt{1 + \mathbf{x}^2(k)}} \quad \text{for } 500 \leq k \leq 1000.$$

The simulation studies for this example were shown in Fig. 4.22. The desired nonlinear function is changed at 500th learning iteration. Figure 4.22 show the different phases of the simulation studies.

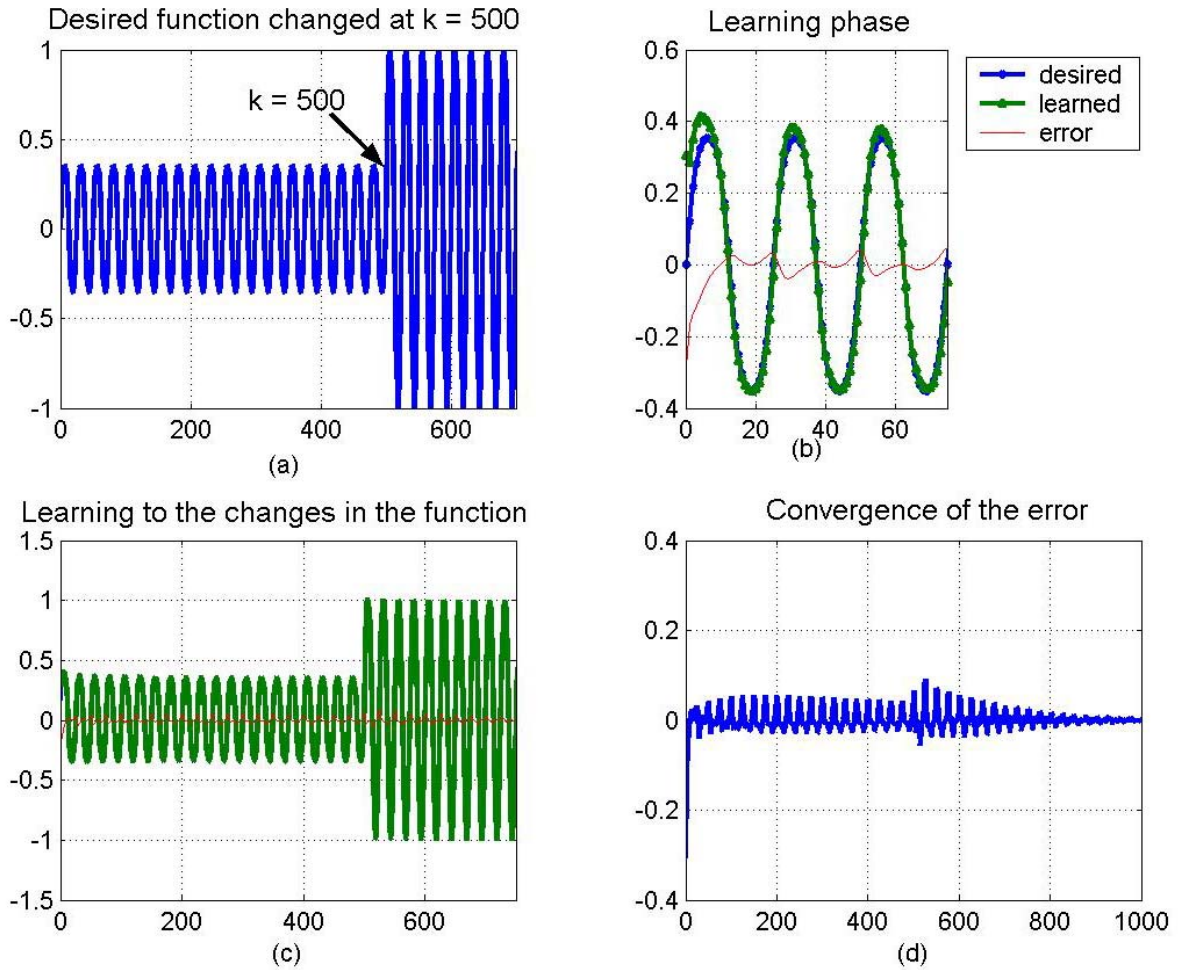


Figure 4.22 Function approximation of a nonlinear function using a neural unit with QSO for Example 3 with change in desired function during the simulation at $k = 500$

In Example 4, another peculiar nonlinear desired function was chosen as follows:

$$f(x(k)) = x^3(k) + 0.3\sin(2\pi x(k)) + 0.1\sin(5\pi x(k))$$

where

$$x(k) = \sin(2\pi k / 250).$$

The simulation studies were shown in Fig.4.23 along with learning phase during the simulation study. It is observed from the Fig. 4.23 that the neural unit with QSO required more number of iterations to approximate the function. This is due to the fact that the function is associated with more nonlinearity. The above results do indicate that the

neural unit with QSO can approximate arbitrary nonlinear functions to desired degree of accuracy. It was observed during the simulation studies that the neural unit with QSO adapted to the changes in nonlinear function during the approximation process. One thing should be noted that the high approximation accuracy is achieved only using a single neural unit with QSO. Better results can be achieved if a network of neural unit with QSOs is used for function approximation.

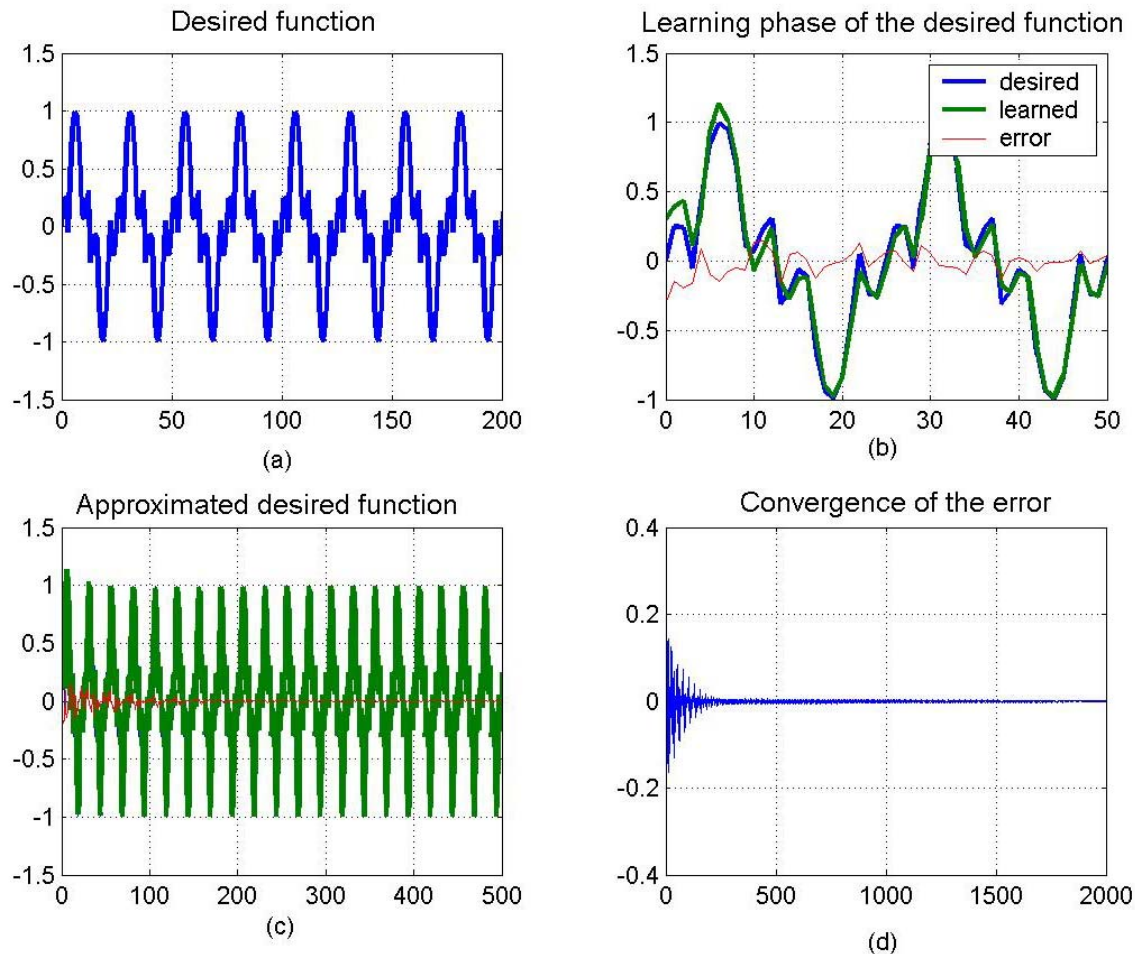


Figure 4.23 Function approximation of a nonlinear function using a neural unit with QSO for Example 4 with an arbitrary desired function.

4.10 Conclusions

The concept of the neural unit with QSO appears to be promising as it can process lower and higher-order inputs similar to the processing function of the biological neuron. Properties such as learning and adaptation associated with neural unit with QSO were

examined in great detail with examples. Different logic circuits such as **OR**, **AND**, and **Exclusive-OR (XOR)** are realized using a single neural unit with QSO. The mathematical model of the neural unit with QSO is greatly examined. The weight matrix beautifully encapsulates the concept of the Euclidian distance, the Mahalanobis distance and the effect of the threshold (bias) on the shape and the placement of the discriminant surface. The approximation capabilities of the neural unit with QSO were discussed in this chapter. The accuracy of the approximation does depend on the structure of the neurons employed in a network. The simulation studies of the neural unit with QSO provide enough evidence that it is a better computational node for the function approximation problems. It is well known fact that the MFNNs were considered as universal approximators for continuous functions. However, in author's view that a network of neural unit with QSOs **may** provide a better approximation results than the approximation achieved by the MFNNs. Apart from this, The HONUs with higher-order synaptic operations can be expressed using different combinations of the neural unit with QSO. Hence, it is the most general neural unit which can deal with both linearties and nonlinearities of the real world problems. The learning and adaptation algorithm for using the neural unit with QSO as a state feedback controller for control problems is being investigated in next chapter.

CHAPTER 5

Applications of Higher-Order Neural Units with Higher-Order Neural Synaptic Operations for Control Problems

5.1 Introduction: A Brief Review

Neural networks are able to implement many nonlinear functions of control systems with higher degree of autonomy. The most commonly used neural network is the multilayer feedforward neural network (MFNNs) where no information is fed back during the operation. However, there is feedback information available during the training process. Supervised learning methods, where the neural network is trained to learn input/output patterns presented to it, are typically used. More often different versions of the backpropagation algorithm are used to adjust the neural network weights during the training process. This is generally a slow and very time consuming process as the algorithm usually takes longer time to converge. The convergence of the training process depends on the activation functions in the neural units; that is, sigmoidal, signum and gaussian functions are typically used depending on the requirement of the problem.

One of the important applications of the MFNNs is for the identification and the control of nonlinear dynamic systems. Such networks can generate input/output maps which can approximate any function with a desired accuracy. One may have to use a large number of neurons, but any desired approximation can be accomplished with a multilayer network with one or two hidden layer of neurons. For a given task, this network architecture is associated with large number of neurons. On the other hand, higher-order neural networks (HONNs) require less number of neurons for achieving the same task but they are associated with large number of parameters (weights). However, for certain problems these weights are greatly reduced by constraining the architecture of the network; that is, for the problems that need to be classified regardless of some transformation groups such as translation, scaling and rotation. HONNs are also used for identification and control of nonlinear dynamic systems, and their performance is compared with the performance of conventional multilayer neural networks with linear

synaptic operation. Computer simulation studies reveal that the HONN models are more effective for the control of nonlinear dynamic systems (Gupta and Rao 1994).

In this chapter, applications of HONUs with higher-order synaptic operations to control problems are discussed. Section 5.2 deals with the implementation of HONUs for control of linear systems where the neural unit with QSO and the neural unit with CSO are used as neuro-controllers for the satellite attitude control. Simulation studies of the HONUs as neuro-controllers to control the linear systems are discussed in Section 5.3. Similarly, applications of the HONUs as neuro-controllers to control the nonlinear systems are discussed in Section 5.4. The stability analysis of nonlinear systems using the energy approach and the lyapunov method are presented in Section 5.5. A new damping function is developed in Section 5.6. In Section 5.7, neural unit with CSO is implemented as a nonlinear state feedback controller for control of unknown, varying parameter and structure, nonlinear dynamic system. The simulation results are presented in Section 5.8 followed by the conclusion in Section 5.9.

5.2 HONUs for Control of the Linear Systems

The proposed HONUs utilize the linear and nonlinear correlation terms, and avoid the problem of combinatorial explosion of higher-order terms. In HONUs, there is no requirement to select the number of hidden units as in multilayer feedforward networks. HONUs are capable of learning linear and nonlinear functions extremely quickly due to the predefined relationships between the input nodes (Gupta and Knopf 1994). It was shown that these neural units can learn and process both functions and their derivatives with ease. This motivates system scientists to develop novel hierarchical and multilayer HONUs network architectures that can be trained using standard error backpropagation learning techniques. The following sections describe the application and implementation of HONUs such as the neural unit with QSO and the neural unit with CSO as neuro-controllers for control problems. They are implemented as neuro-controllers for satellite attitude control problem.

5.2.1 Neural unit with QSO as a Neuro-Controller for Satellite Control Attitude Problem

Satellites, as shown in Fig. 5.1, usually require attitude control equipment such as antennas, sensors, and solar panels should be properly oriented. Antennas are pointed towards a particular location on the earth, while solar panels need to be oriented towards the sun for maximum power generation. Consider the motion of satellite in its pitch plane. The angle Θ that describes the satellite orientation must be measured with respect to an inertial reference as depicted in Fig. 5.1. The control force comes from the reaction jets that produce a moment of $F_c d$ about the mass center. Applying the basic Newton's law to one dimensional rotational system, the equation of motion is written as

$$F_c d = u = I\ddot{\Theta} \quad (5.1)$$

The output of this system, Θ results from integrating the input torque twice and this type of system often referred to as the double integrator plant. Since, there are two states associated with the system; neural unit with QSO can be implemented as a neuro-controller for satellite attitude control. The block diagram representation of the neural unit with QSO as a controller was shown in Fig 5.2.

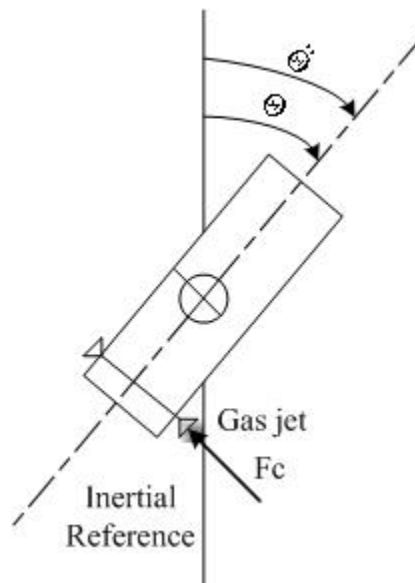


Figure 5.1 Schematic representation of satellite control in its pitch plane (one dimensional view)

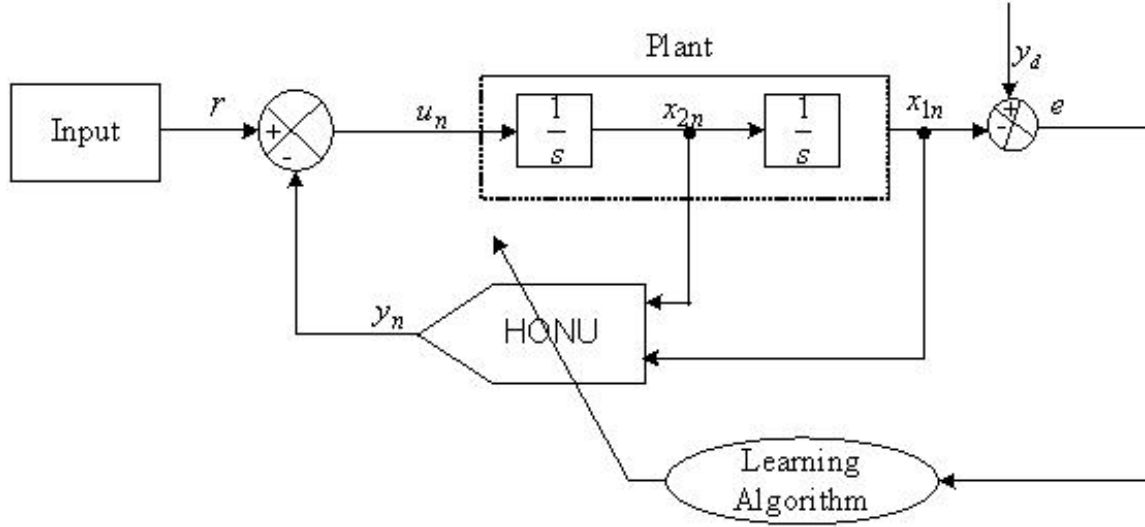


Figure 5.2 Schematic representation of the HONU's (the neural unit with QSO and the neural unit with CSO) as neural-controllers for the satellite attitude control

Defining the augmented vectors of neural inputs and neural weights, the synaptic operation for neural unit with QSO is given as

$$v = \sum_{i=0}^2 \sum_{j=i}^2 w_{ij} x_i x_j = \mathbf{x}_a^T \mathbf{W}_a \mathbf{x}_a \quad (5.2)$$

$$= w_{00}x_0x_0 + w_{01}x_0x_1 + w_{02}x_0x_2 + w_{11}x_1x_1 + w_{12}x_1x_2 + w_{22}x_2x_2$$

where

$$\mathbf{x}_a = [x_0 \quad x_1 \quad x_2]^T \in \mathbf{R}^{2+1}, \quad x_0 = 1$$

$$\mathbf{W}_a = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ 0 & w_{11} & w_{12} \\ 0 & 0 & w_{22} \end{bmatrix} \in \mathbf{R}^{(2+1) \times (2+1)}$$

The neural unit with QSO can be implemented as a nonlinear controller for complex problems such as satellite control problem and for problems in identification and inverse dynamic adaptive control. From Figure 5.2, the governing equations for the satellite attitude control model are given as

$$\dot{x}_{1m} = x_{2m} \quad (5.3)$$

$$\dot{x}_{2m} = u_m = -k_{pm}x_{1m} - k_{vm}(1 - x_{1m}^2)x_{2m} + r \quad (5.4)$$

where k_{pm}, k_{vm} are position and velocity gains of the model respectively. Comparing Eqns. (5.2, 5.4), it is clear that only a subset of neural unit with QSO is required for control but a careful observation at the governing equation of the satellite control model indicates that a cubic term is associated with the control signal. Since the choice of neural controller depends on the structure and order of the system, the objective is to synthesize a control signal, u_n (as function of x_{1n}^2 and x_{2n}) in an optimal fashion (Gupta, 1970). Keeping this in mind, a HONU called the neural unit with CSO is described in detail as a neuro-controller for linear and nonlinear systems in the following sections.

5.2.2 Neural unit with CSO as a Neuro-Controller for Satellite Control Attitude Problem

The neural unit with CSO can be considered as a combination of three neural units with LSO or a neural unit with LSO and a neural unit with QSO. The structure of the neural unit with CSO for satellite control attitude problem is shown in Fig. 5.3.

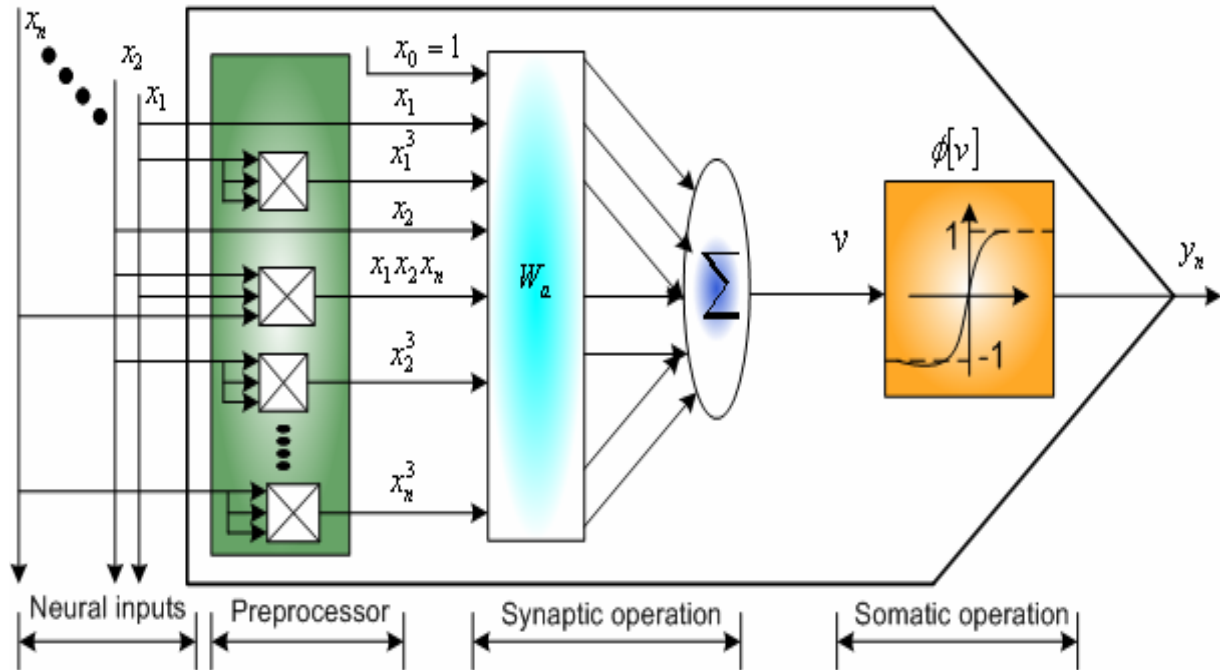


Figure 5.3 Structure of the neural unit with CSO for control applications.

The synaptic operation of the neural unit with CSO is defined as follows:

$$\begin{aligned}
 v &= \sum_{i=0}^2 \sum_{j=i}^2 \sum_{k=j}^2 w_{ijk} x_i x_j x_k \\
 &= w_{000} x_0 x_0 x_0 + w_{001} x_0 x_0 x_1 + w_{002} x_0 x_0 x_2 \\
 &\quad + w_{011} x_0 x_1 x_1 + w_{012} x_0 x_1 x_2 + w_{022} x_0 x_2 x_2 \\
 &\quad + w_{111} x_1 x_1 x_1 + w_{112} x_1 x_1 x_2 + w_{122} x_1 x_2 x_2 \\
 &\quad + w_{222} x_2 x_2 x_2
 \end{aligned} \tag{5.5}$$

where $x_0 = 1$ and the somatic operation is given as

$$\begin{aligned}
 y_n &= \phi(v) \\
 &= \phi \left[\sum_{i=0}^2 \sum_{j=i}^2 \sum_{k=j}^2 w_{ijk} x_i x_j x_k \right]
 \end{aligned} \tag{5.6}$$

where y_n is the output of the neural unit with CSO and $\phi(\cdot)$ is the activation function. In Figure 5.4, the neural unit with CSO is used as an identifier of a plant. The typical identification process consists of three components: the system to be identified, a postulated model (in this case either the neural unit with CSO or the neural unit with QSO) and an adaptation algorithm which updates the model based upon an error criterion. Since the adaptation algorithm shown in Fig. 5.4 processes the difference between the outputs of the neural unit with CSO and the system, the structure is called as parallel identifier. The importance of this type of identification is that it completely describes the external behaviour of the system. Normally, the injected signal to the plant is white noise or a signal of broad spectrum to excite all modes of the plant. If the input to the plant (and the model-neural unit with CSO) is sufficiently general and the weights of the neural network are adjusted for a sufficiently long time which ensures the minimum error for any input. These types of inputs are highly desirable which assures the convergence of the identification model to the desired set.

The objective of the neural unit with CSO is to emulate the plant by forcing the error to zero (within the tolerance limit) and to identify the synaptic weights equal to the plant parameter values. The learning and adaptation algorithm was already developed in

Chapter 3. In this identification process, the neural unit with CSO is considered as a static neural unit because it does not have any feedback properties; that is, the output of the neural unit with CSO depends only on its current inputs and the synaptic weights.

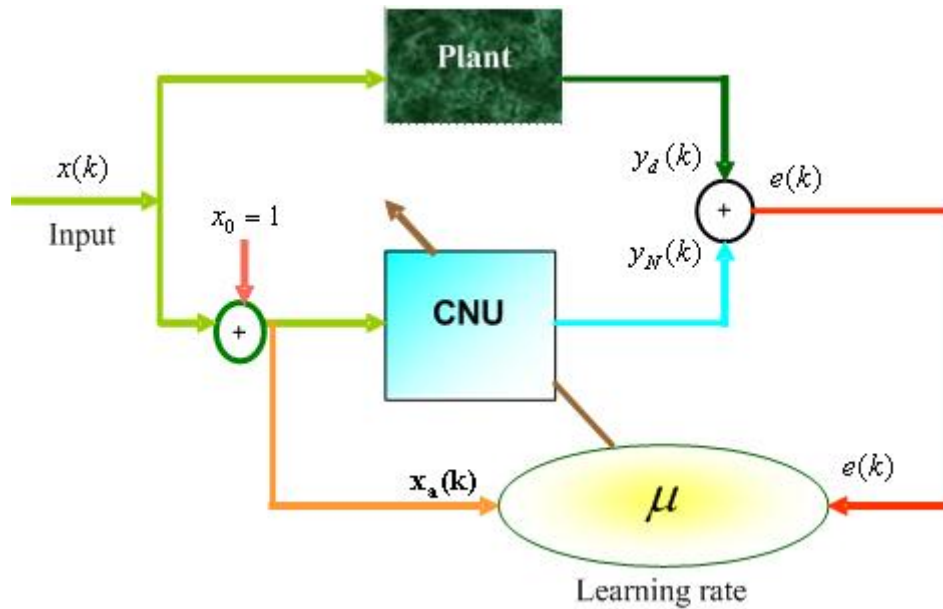


Figure 5.4 Identification of a Plant using a neural unit with CSO

Even though the neural unit with CSO is a static, it can be implemented as a neuro-controller for complex control systems such as satellite control etc. The following section describes the implementation of the neural unit with CSO as a dynamic neural unit.

5.2.2.1 Dynamic Cubic Neural Unit with CSO for Control Applications

In this section, neural unit with CSO is implemented as a dynamic neural unit for control of complex systems (Satellite Attitude Control). The structure of the neural unit with CSO reveals that it is not associated with any dynamics but the introduction of states as feedback makes it a dynamic with several stable points. Hence, feeding back the states of the plant to the neural unit with CSO makes the whole structure a dynamic neuro-controller. The simplified model of the satellite system is shown in Fig. 5.5. The purpose of this control system is to control the attitude of the space vehicle in one dimension. The block diagram of the space vehicle system shows that two states, the position x_1 and the

velocity x_2 , are feedback by the position and rate sensors to form the closed loop control. The governing equations for the block diagram are given by Eqns. (5.3, 5.4).

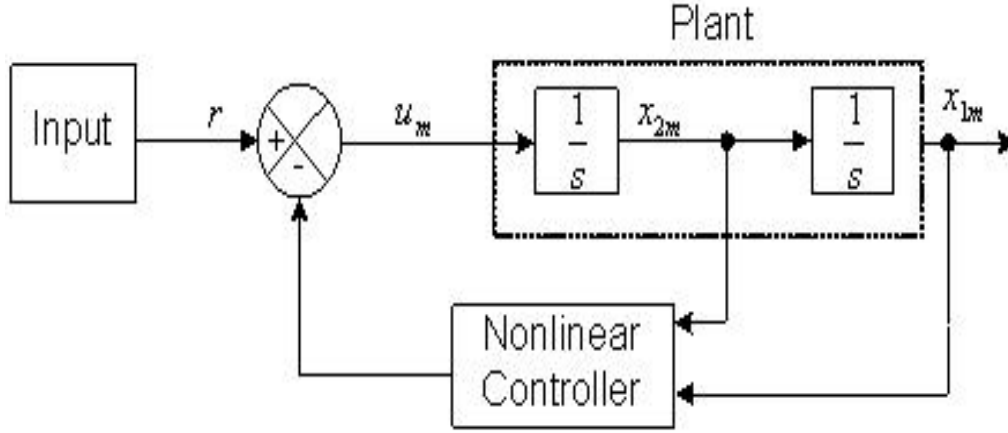


Figure 5.5 Block diagram of space vehicle (satellite control) system with nonlinear controller.

The control signal u_m is associated with the cubic parameter i.e. $x_{1m}^2 x_{2m}$. It was clearly evident that the neural unit with LSO and the neural unit with QSO don't have a cubic term in their structures as discussed in previous sections. This implies that they require more training passes or iteration time to control the satellite system. However, the neural unit with CSO is inherently associated with the cubic form in its structure as $x_1 x_1 x_2$. This inherent structure helps the neural unit with CSO to adapt the plant more effectively and efficiently.

According to the Eqns. (5.3, 5.4), n is equal to two because there are two states as inputs to the neural unit with CSO such as the velocity (x_2) and the position (x_1). Thus, the synaptic operation of the neural unit with CSO is given as

$$\begin{aligned}
 &= w_{000}x_0x_0x_0 + w_{001}x_0x_0x_1 + w_{002}x_0x_0x_2 \\
 &\quad + w_{011}x_0x_1x_1 + w_{012}x_0x_1x_2 + w_{022}x_0x_2x_2 \\
 &\quad + w_{111}x_1x_1x_1 + w_{112}x_1x_1x_2 + w_{122}x_1x_2x_2 \\
 &\quad + w_{222}x_2x_2x_2
 \end{aligned} \tag{5.7}$$

Only a subset of the neural unit with CSO parameters are required to control the satellite system i.e. w_{001} , w_{002} and w_{112} , each of which represent $-k_p$, $-k_v$ and k_v in sequence.

The equation for the dynamic neural unit with CSO can be rewritten as

$$\dot{x}_{1n} = x_{2n} \quad (5.8)$$

$$\dot{x}_{2n} = u_n \quad (5.9)$$

$$u_n = w_{001}x_0x_0x_{1n} + w_{002}x_0x_0x_{2n} + w_{112}x_{1n}x_{1n}x_{2n} + r \quad (5.10)$$

$$y_p = x_{1n} \quad (5.11)$$

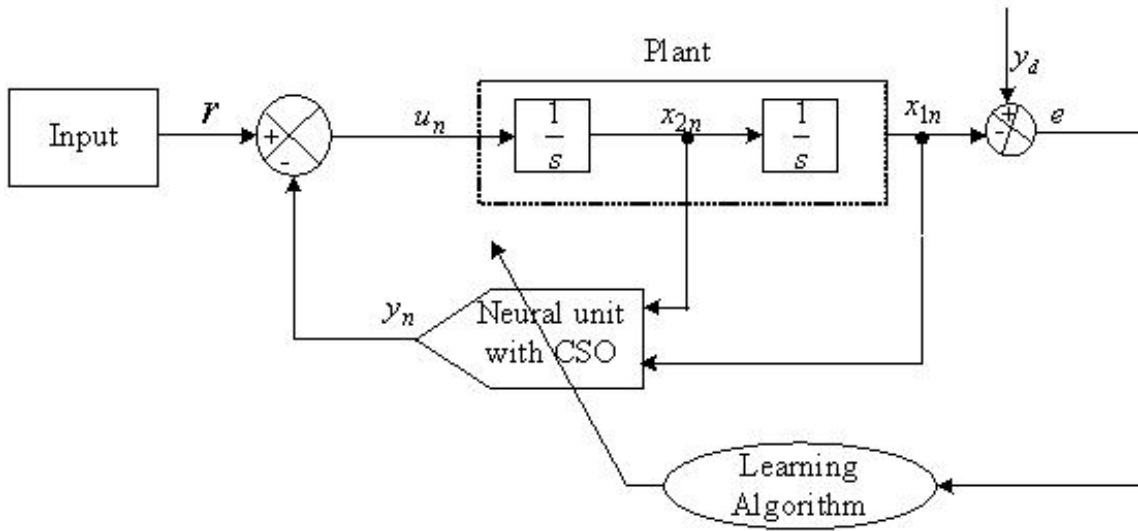
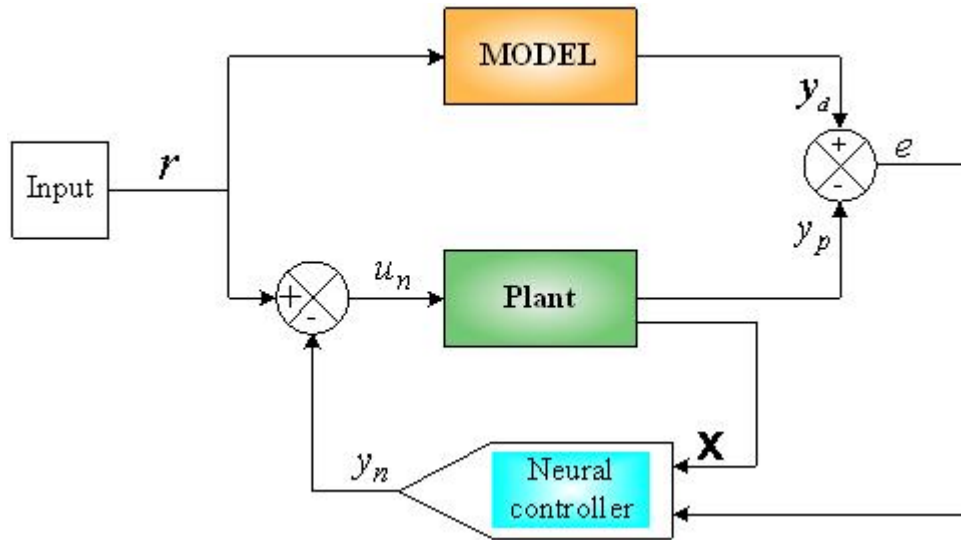


Figure 5.6 Adaptation of the neural unit with CSO as a nonlinear neuro-controller for satellite attitude control system.

5.3 Simulation Results

In this section, the response of the satellite control with different neuro-controllers such as neural unit with LSO, neural unit with QSO and the neural unit with CSO are compared. A block diagram depicting the different neural structures as state controllers is shown in Fig. 5.7. The control methodology shown in Fig. 5.7 is called as Model Reference Adaptive Control (MRAC). This is the one of the most popular modern control methods that attracted many system scientists in control design. Selection of the model is referred to as structure estimation, where the model input-output signals and the internal components of the model are determined. In general, the model structure is derived using prior knowledge. In dynamic systems, the choice of the order of the model is always a

nontrivial problem. It is a compromise between reducing the unmodelled dynamics and increasing the complexity of the model which can lead to model stabilizability difficulties.



r : Input signal [Step or Square Inputs]

y_n : Neural unit [neural unit with LSO, neural unit with QSO, neural unit with CSO]

u_n : Control signal for the plant

y_d : Desired output

y_p : Output of the plant when

$\mathbf{x}_a : [x_0 \ x_1 \ x_2]^T$ are the neural inputs

e : Error between the desired output and the output of the plant

Figure 5.7 Block diagram depicting different neural structures as neuro-controllers for a complex control system (satellite attitude control)

In many practical cases, it is tacitly assumed that the reference model is linear. This is because linear systems theory is well developed and methods of choosing linear models that have desired properties are well established. This does not necessarily mean that the reference model cannot be a nonlinear model but an important practical consideration is that the dynamic mapping represented by the system, containing the controller and the plant, should approximate the reference model as k tends to infinity (Miller et al. 1990).

In the model based approaches, the controller can be seen as an algorithm operating on a model of process and optimized in order to reach the given control design objectives (Ikonen and Najim 2002). For this study, a linear and a nonlinear model are considered for satellite attitude control. The objective of the attitude control is to point an antenna towards certain stellar object in its pitch plane. Therefore, inputs like step and square wave signals are considered for simulation studies. The simulation result shows that the neural unit with CSO can control a plant very effectively when compared with other neural structures as neuro-controllers.

Most model validation tests are based on simply the difference between the simulated output and the measured output. The primary concern in satellite attitude control is the transient response of the plant. The transient response should be as fast as possible but without the overshoot. Simulations studies show that the neural controllers require 10 seconds to place the satellite to a desired state. Since it is a closed loop control, the response of the satellite can be improved by increasing the gain of the neural controllers but there is always an overshoot associated with it. This severely hinders the flexibility of the neural controllers despite its adaptive capabilities.

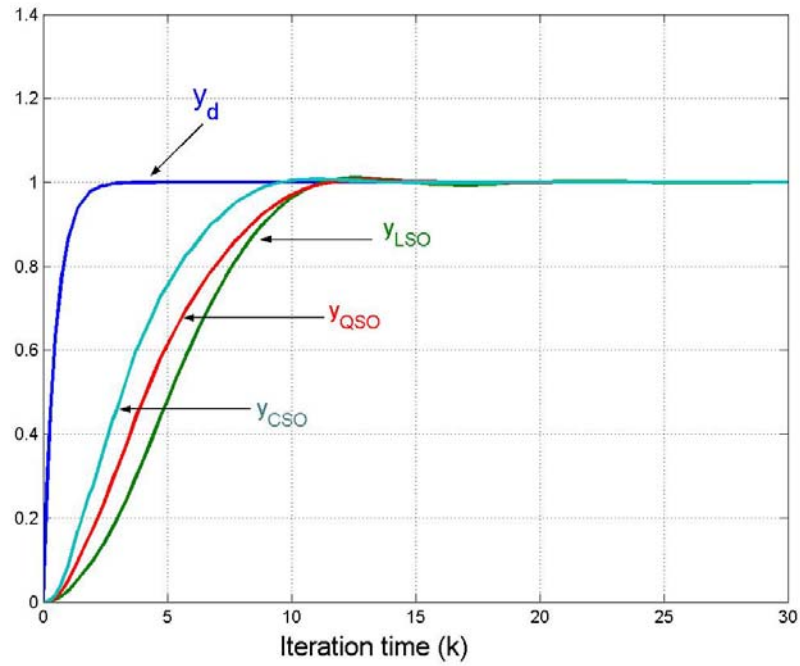
5.3.1 Simulation Results for Satellite Control (Linear Model)

Consider the attitude control of satellite in its pitch plane. The governing equations for the linear model are given as

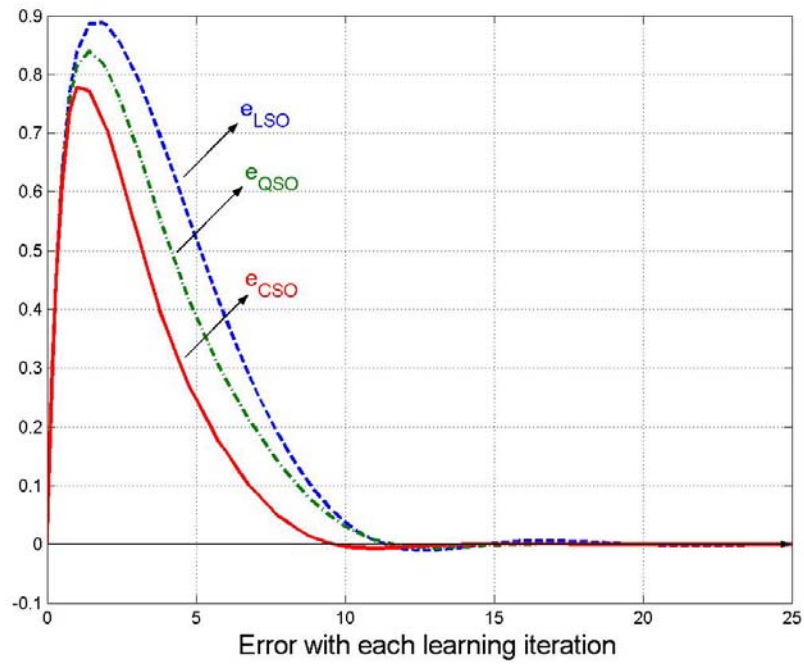
$$\dot{x}_1 = x_2 \quad (5.12)$$

$$\dot{x}_2 = -k_p x_1 - k_v x_2 + r \quad (5.13)$$

where $k_p = 1$, $k_v = 2$ are position and velocity gains respectively. The simulations of the satellite attitude control with these parameters using above differential equations are carried out. Figures 5.8 and 5.10 illustrate the comparison between the step and square responses of the satellite when three different neuro-controllers (neural unit with LSO, neural unit with QSO, and neural unit with CSO) are used. It can be noticed that the time required for the neural unit with CSO controller to drive the satellite to desired position is shorter than that of the other two controllers. Figures 5.8 (b), 5.9 (b), and 5.10 (b) show the convergence of the error with square and step inputs with different neural controllers.

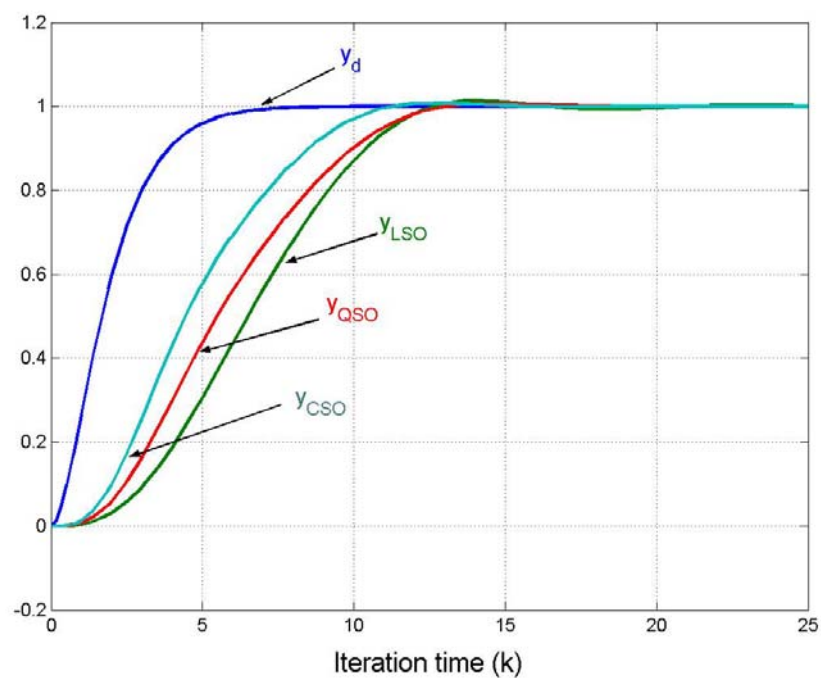


(a). Step response of the plant with different neural controllers.

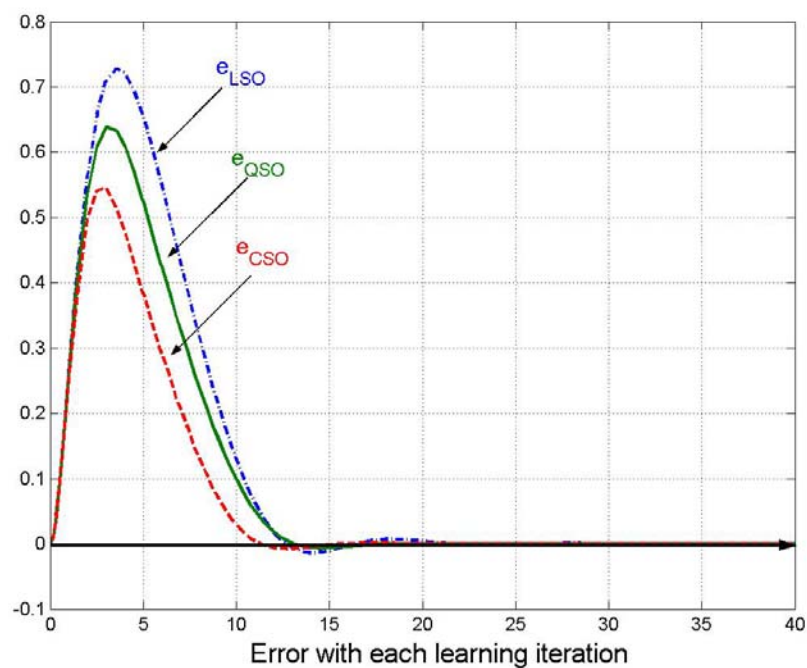


(b). The Convergence of the error with each learning iteration.

Figure 5.8 Step and error response of the satellite control with the three different neural controllers when the model is a first order system.

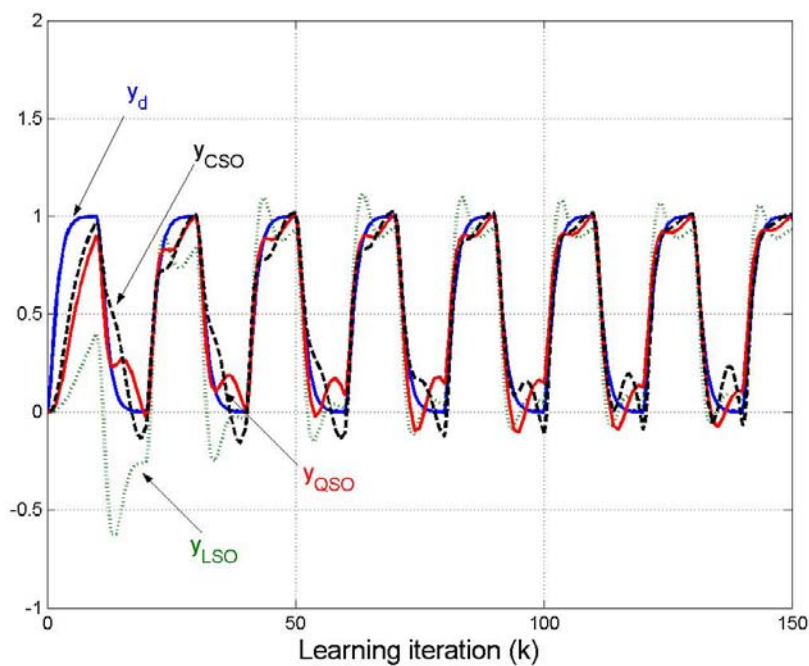


(a). Step response of the plant with different neural controllers.

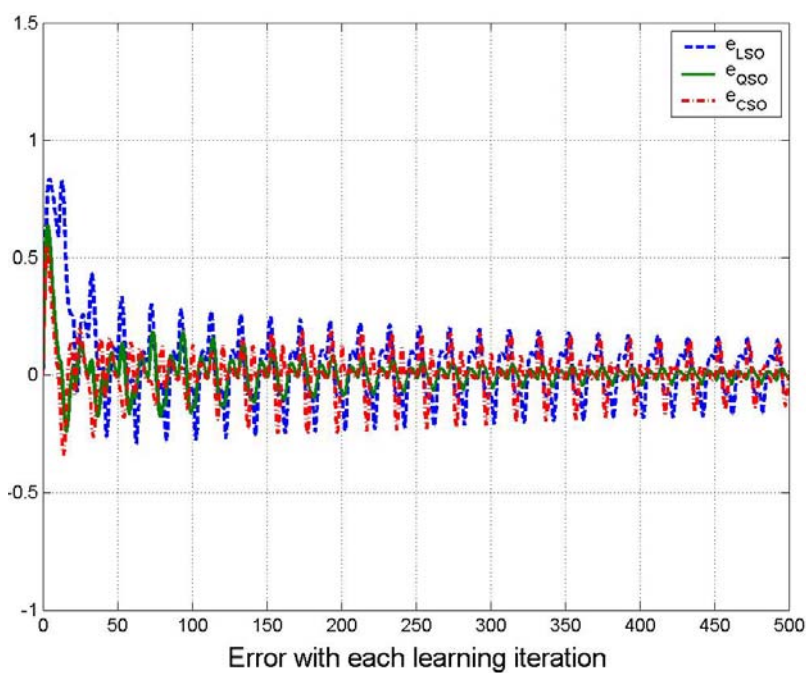


(b). The Convergence of the error with each learning iteration.

Figure 5.9 Step and error response of the satellite control with the three different neural controllers when the linear model is used.



(a). Square input response of the plant with different neural controllers.



(b). The Convergence of the error with each learning iteration.

Figure 5.10 Square input and the error response of the satellite control with the three different neural controllers when the linear model is used as MRAC.

5.3.2 Simulation Results for Satellite Control (Nonlinear Model)

The transient response of the system can be improved by employing a nonlinear or time varying linear controller. The governing equations for the satellite attitude control with a nonlinear model are given as

$$\dot{x}_1 = x_2 \quad (5.14)$$

$$\dot{x}_2 = -k_p x_1 - k_v(1 - x_1^2)x_2 + r \quad (5.15)$$

where $k_p=1.05$, $k_v=1.8$ are position and velocity gains respectively. The velocity gain k_v is a function of both position and velocity. The controller implements a nonlinear damping which varies with position as shown in Fig. 5.11. For Large x_1 , the damping is negative and for small x_1 , the damping is positive. The advantage of using this kind of nonlinear damping is to improve systems transient response by making damping small for large x_1 (or possibly negative) and large for small x_1 . The damping limits can be increased or decreased depending on the design requirements of the system.

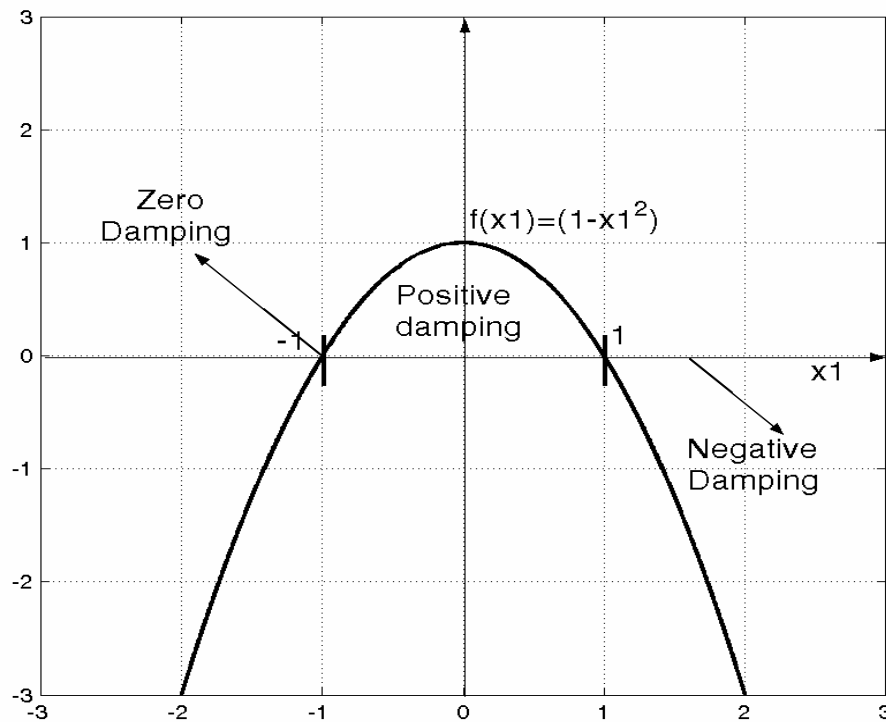
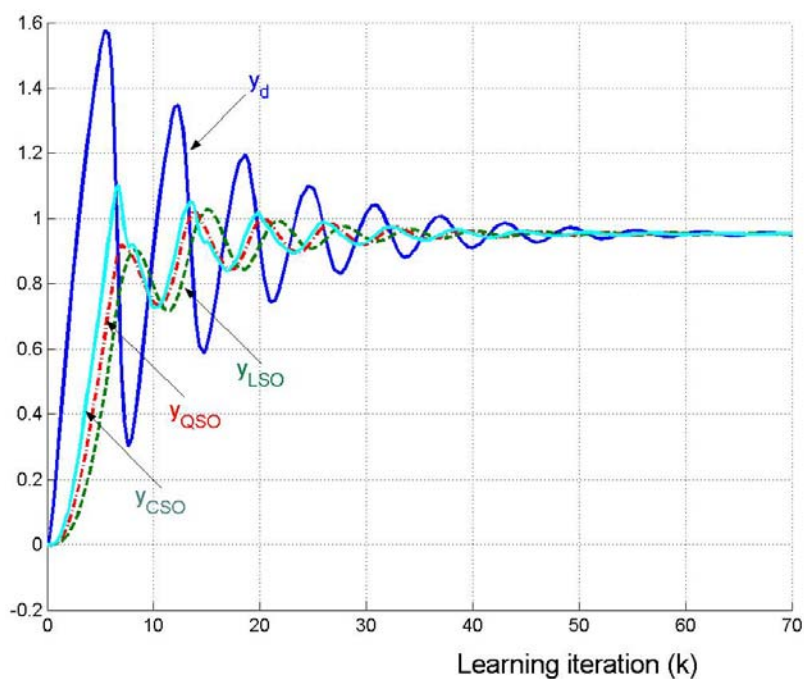
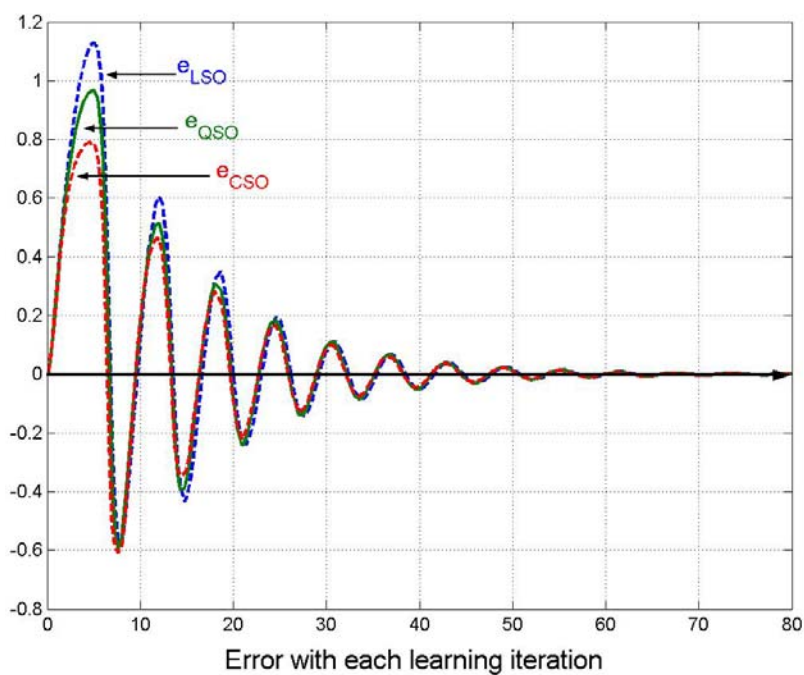


Figure 5.11 Nonlinear damping function



(a). Step input response of the plant with the three different neural controllers.



(b). The convergence of the error with each learning iteration.

Figure 5.12 Step input and the error response of the plant (satellite control) with different controllers when the nonlinear model is used as MRAC.

Figure 5.12 depicts the step response of the satellite with three different neuro-controllers (neural unit with LSO, neural unit with QSO, and neural unit with CSO). It can be noticed that the time required for the neural unit with CSO controller to drive the satellite to desired position is shorter than that of the other two neuro-controllers. The transient response is associated with overshoot and almost required sixty iterations to settle which is not acceptable in the design of the control systems.

From the simulation studies, it is concluded that the plant response is too sensitive to the changes in the parameters of the neural units (neural unit with LSO, neural unit with QSO, and neural unit with CSO). This was clearly evident when the plant was subjected to square inputs. The response of the plant was affected by the dynamics of the model notably in the transient phase. The transient response of the plant is faster with neural unit with CSO as a neuro-controller and an effort to improve the transient response resulted either in overshoot or instability of the plant forcing the response to infinity.

A novel neural structure, called the neural unit with CSO, was presented as a neuro-controller for the control of a complex system such as satellite attitude control. The neural unit with CSO, as a dynamic neural unit with CSO, was able to control the satellite system better than the other neural controllers. The error converged to zero in less number of iterations with the dynamic neural unit with CSO. However, the response of the plant was associated with overshoot and the speed of the response was damped when the model was associated with the higher-order dynamics. This could be accounted to the large number of parameters that are associated with the neural unit with CSO. The solution to this problem is to find a method which reduces the overshoot without sacrificing the speed of the transient response. The following section describes a method to develop such type of controllers for linear and nonlinear systems. In this case, it is mainly focused on the development of new controllers which are subset of HONUs for control of nonlinear systems.

5.4 HONUs for Control of Nonlinear Systems

To model the input/output behaviour of a dynamical system, the neural network is trained using input/output data and the weights of the neural network are adjusted most often using the backpropagation algorithm. Because the typical application involves

nonlinear systems, the neural network is trained for classes of inputs and initial conditions. The underlying assumption is that the nonlinear static map generated by the neural network can adequately represent the system's behaviour in the ranges of interest for a particular application. There is of course the question of how accurately a neural network, which realizes a static map, can represent the input/output behaviour of a dynamical system. One must provide the neural network with the information about the history of the system – typically delayed inputs and outputs. The amount of information needed depends on the desired accuracy but there is a trade-off between the accuracy and the computational complexity of the training process. This is because the number of inputs used affects the number of weights in the neural network and subsequently the training time. One sometimes starts with as many delayed input signals as the order of the system, and then modifies the neural network accordingly. The number of neurons in the hidden layer(s) is typically chosen based on empirical criteria and one may iterate over a number of networks to determine the neural network that has a reasonable number of neurons and accomplishes the desired degree of approximation.

As seen previously, MFNN has only a linear correlation between the input vector and the synaptic vector. Extensive research attempts have been made by Rumelhart et al. [1986], Giles and Maxwell [1987], Softky and Kammen [1991], Xu et al [1992], Taylor and Commbes [1993], and Homma and Gupta [2002b] to develop HONUs for different applications. HONUs have been proved to have good computational, storage, pattern recognition, and learning properties and are realizable in hardware (Taylor and Commbes 1993). These networks satisfy the Stone-Weierstrass theorem and hence considered to improve the approximation and generalization capabilities of the network. In recent years, adaptive neural control schemes have been found to be particularly useful for the control of nonlinear systems with unknown nonlinear functions. In the literature of adaptive neural control, neural networks (NNs) are primarily used as on-line approximators for the unknown nonlinearities due to their inherent approximation capabilities. However, the conventional models of neurons cannot deal with the discontinuities in the input training data. In an effort to overcome such limitations of conventional neural unit with LSOs, some researchers have turned their attention to HONN models (Gupta et al. 2003).

In control systems theory, it is always difficult to prove typical control system properties such as stability when HONNs are used as neuro-controllers. The main reason is the mathematical complexity that is associated with the nonlinear systems. In the earlier NN control schemes, optimization techniques were mainly used to derive parameter adaptation laws, which lack for analytical results about stability and performance. To overcome these problems, several elegant adaptive NN control approaches have been proposed based on Lyapunov's stability theory (Gupta Control notes 1970's, and Ogata 1984). One main advantage of these schemes is that the parameter adaptation laws are derived based on Lyapunov synthesis and therefore stability of the closed-loop system is guaranteed. However, one limitation of these schemes is that they can only applied to nonlinear systems where certain types of matching conditions are required to be satisfied. Recently, some progress has been made in this area and certain important theoretical results have begun to emerge, but clearly the overall area is still at its infancy. The encouraging news is that there are successful applications of neural networks in control systems that work, and this certainly provides clues and guidelines for the corresponding theoretical development. This was the motivation to develop HONNs for nonlinear systems.

In the following section stability analysis of nonlinear systems using Energy and Lyapunov methods are discussed. A new damping function called Universal damping function is introduced. A neuro-controller is designed using the universal damping function for control of unknown, varying parameter and structure, nonlinear dynamic system.

5.5 Stability Analysis of Nonlinear Systems

Most closed loop systems become unstable as gains are increased in an attempt to achieve higher performance. It is therefore correct to regard stability considerations as forming a rather general upper limit to control system performance (Leigh 2004). Stability is the most important subject in the control performance but there is no general stability analysis technique that will always determine stability for a given nonlinear system. A nonlinear system can have many stable points. When possible, it is always desirable to know the location of the stable points and which initial conditions would

converge to a given stable point. If the system is linear and time invariant, many stability criteria are available. Among them are the Nyquist stability criterion, Routh's stability criterion, etc. If the system is nonlinear, or linear but time varying, however, then such stability criteria do not apply. Nevertheless, there are two popular methods for analyzing the stability of nonlinear systems (Alligood et al. 1996). They are

- Motion in a Potential Field
- Lyapunov Method

5.5.1 Energy Method: Motion in a Potential Field

Taking a cue from mechanics, consider the principle of conservation energy: In the absence of damping or any external forces, the system neither gains nor loses energy. Given an initial condition (x_0, y_0) , the energy function E remains constant on the level curve $F(t, (x_0, y_0))$, for all time t :

$$\frac{dE}{dt}(F(t, (x_0, y_0))) = 0$$

where $F(t, (x_0, y_0))$ is the flow of the solution. There is no need to know the explicit solution $(x(t), y(t))$ of the governing equations. The governing motion in potential field for simple pendulum (assuming no damping and no external forces) is given as (Alligood et al. 1996)

$$\ddot{x} + \frac{\partial P}{\partial x} = 0 \tag{5.16}$$

This is another way of viewing Newton's second law of motion; that is, acceleration is proportional to the force, which is the negative of the gradient of the potential field. Multiplying the Eqn. (5.16) with \dot{x} and integrating both sides:

$$\begin{aligned} \ddot{x}\dot{x} + \frac{\partial P}{\partial x} \frac{\partial x}{\partial t} &= 0 \\ \frac{1}{2} \dot{x}^2 + P(x) &= E_1 \end{aligned} \tag{5.17}$$

where E_1 is the constant of integration (Alligood et al. 1996). This leads to a simple technique for drawing the phase plane solutions of motion in a potential field. For example, the phase plane solutions of the pendulum problem are shown in Fig. 5.13. The

solutions move along the level curves and the equilibrium points are denoted by star symbol. Each trajectory of the system is trapped in a potential energy well and the total energy is constant for different trajectories.

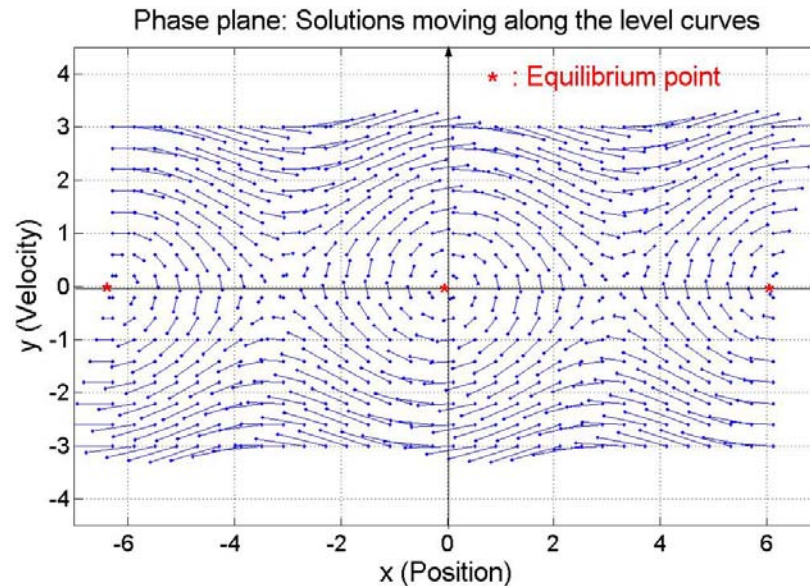


Figure 5.13 Level curves of the energy function and the solutions move along them. The equilibrium points are denoted by stars.

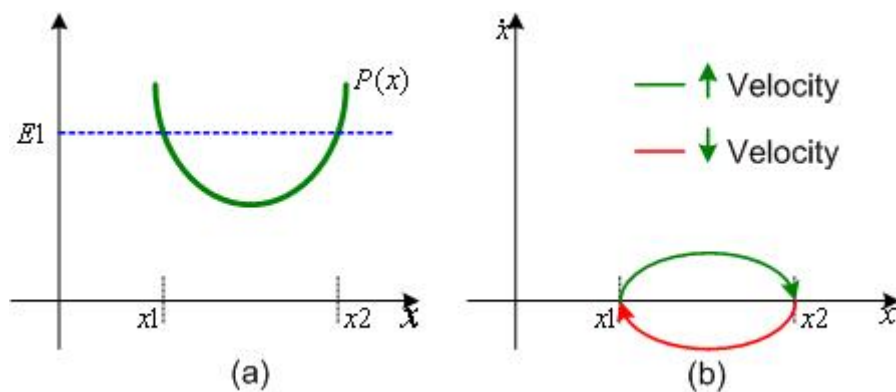


Figure 5.14 Phase plane curves from the potential. (a) Graph of a potential energy function $P(x)$. (b) A periodic orbit indicating absence of damping.

In Figure 5.14, a trajectory with a fixed total energy E_1 tries to climb out near x_1 or x_2 , the kinetic energy goes to zero, as the energy E converts completely into potential energy. The function E provides a useful partition of the points (x, y) in the phase plane into individual solution trajectories. For a conservative system, the energy is conserved

but for a non conservative system, Eqn. (5.17) is no longer valid as it is associated with the damping and periodic forces. The basic idea of using energy-like functions is to investigate the dynamics of solutions that can be applied to more general equations rather than Eqn. (5.16). The theory of Lyapunov functions, a generalization of potential energy functions, gives us a global approach toward determining asymptotic behavior of solutions. By using any one of the methods, it is possible to determine the stability of a system without solving the state equations. There is a relationship between the Energy method and the Lyapunov function which will be explained in the next section.

5.5.2 Lyapunov Function

Late in the nineteenth century the Russian mathematician A.M. Lyapunov developed an approach to stability analysis, now known as direct method of Lyapunov. It was widely used for stability analysis of linear and nonlinear, both time-invariant and time varying systems. Lyapunov's theorem is limited in determining the stability in small regions about equilibrium points. It determines *stability in the small*. Ideally, the objective is to determine the stability in larger regions of the state space so that the stability about the equilibrium point is said to be *globally asymptotically stable or asymptotically stable in large* (Ogata 1984, Gupta). In many cases when Lyapunov method was applied, it was difficult to find suitable Lyapunov functions and when using linear approximation, it can not be decided whether nonlinear systems were stable or not on the outside of the region where linear stability theory can be applied. Therefore, a new stability analysis method is required to develop, which can be easily applied to nonlinear systems. In this Section, the stability analysis based on the higher-order (second order) derivatives of the neural unit with CSO and its application to nonlinear systems is discussed.

Before dealing with stability of nonlinear control systems, it should be recalled that the stability of linear systems does not depend on initial conditions, but it depends only on the real values of the poles (Ogata 1984, Leigh 2004). In case of second order nonlinear dynamic systems, the motion in the potential field approach can be used to assess the stability of the nonlinear dynamic systems. Stability analysis of these systems often involves solving the differential equations of the form,

$\ddot{x}(t) + k_v(1 - x^2)\dot{x}(t) + k_p x(t) = r(t)$ where the damping is not only a variable but also a function of position. The differential equations describing the dynamics of this type of nonlinear systems can be expressed in a more general form as

$$\ddot{x}(t) + f(\dot{x}(t), x(t)) + \frac{\partial P(x(t))}{\partial(x(t))} = r(t) \quad (5.18)$$

where $\frac{\partial P(x(t))}{\partial(x(t))}$ is the partial derivative of potential function and $\dot{x}(t), x(t)$ are the states

of a nonlinear system. The unity static gain in Eqn. (5.18) assures zero steady state error and the damping assures the stability. It should be mentioned that the unity gain can be reached either by neural unit with CSO or neural unit with QSO controller once the plant has been identified. For dissipative second order systems, the derivative of the energy function E (Alligood et al. 1996) is

$$\begin{aligned} \dot{E} &= \ddot{x}(t)\dot{x}(t) + \frac{\partial P(x(t))}{\partial x(t)} \dot{x}(t) \\ &= -f(\dot{x}(t)x(t))\dot{x}(t) \end{aligned} \quad (5.19)$$

The negative sign indicates that the total energy is decreasing along the orbits. Contrary to the potential field approach, the concept of the Lyapunov function can be applied for stability analysis of any dynamical system of any order.

Theorem [Alligood et al. 1996]:

If \bar{x} is an equilibrium point of the system of state differential equations $\dot{x} = f(x)$ and there exists a Lyapunov function for \bar{x} , then \bar{x} is stable. If a strict Lyapunov function exists, then the equilibrium \bar{x} is asymptotically stable.

In case of nonlinear second order systems, the task simplifies as the nonlinear differential equations describing the system can be expressed in state space representation as

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -x_1(t) - f[x_2(t), x_1(t)] + r(t) \end{aligned} \quad (5.20)$$

then, the Lyapunov function can be chosen as

$$V(x_1, x_2) = \sum_{i=1}^n \frac{1}{2} x_i^2 = \frac{1}{2} x_1^2 + \frac{1}{2} x_2^2, n = 2 \quad (5.21)$$

where its derivative is given as

$$\begin{aligned}
 \dot{V}(x_1, x_2) &= x_1 \cdot \dot{x}_1 + x_2 \cdot \dot{x}_2 \\
 &= x_1 \cdot x_2 + x_2 \cdot (-x_1 - f(x_2, x_1)) \\
 &= -x_2 \cdot f(x_1, x_2) = \dot{E}
 \end{aligned} \tag{5.22}$$

From Equations (5.19) and (5.21), one concludes that the derivative of Lyapunov function (5.22) is equal to the derivative of the energy function for a second order nonlinear system (5.18). Now, the task is to develop a new controller which satisfies the following criteria:

- **The stability of a nonlinear control system is assured if the state space region in which the system to be controlled lies entirely within the basin of attraction and the solution of nonlinear system should converge to the equilibrium;**
- **The morphological properties of the basin of attraction to the equilibrium are related to the properties of the strict Lyapunov function;**
- **The larger area of existence of the strict Lyapunov function assures the larger basin of attraction to the equilibrium; and**
- **Therefore, the existence of the strict Lyapunov function over large area, as large as possible, should be achieved to assure stability of a non-linear control loop.**

5.5.3 Concept of Nullclines

The method of nullclines is a technique for determining the global behaviour of solutions of competing species models. This method provides an effective means of finding some trapping regions for some differential equations. The nullclines provide a bulk picture of how things change at different points in the phase plane. The concept of nullclines is used to identify the stability region and provides a clear picture of the variations of the slope field. Figure 5.15 shows the flow of one stable and one unstable solution of an autonomous system given by Eqn. (5.23)

$$\ddot{x}(t) + k_v \cdot (1 - x_1^2(t)) \cdot \dot{x}(t) + k_p \cdot x(t) = r(t) \tag{5.23}$$

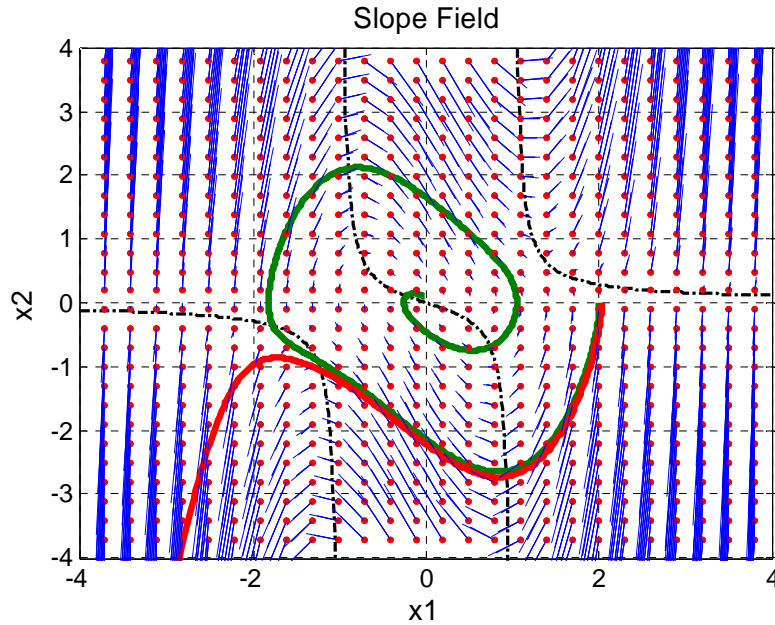


Figure 5.15 The slope field, flow of two solutions, and nullclines of closed loop control (5.18) in the phase plane.

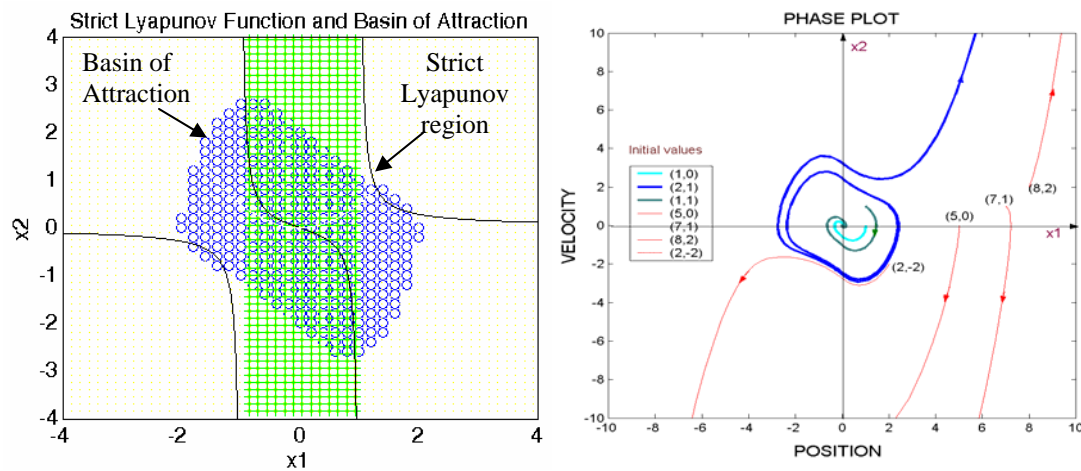


Figure 5.16 The stability region for a nonlinear PD control system (5.23), area of decreasing energy and the nullclines.

where $k_p = 1$, $k_v = 2.34$ and the boundary of basin of attraction is partly denoted by the stable trajectory converging to equilibrium $[0,0]$. Figure 5.16 shows the basin of attraction denoted by 'o' and the area of existence of strict Lyapunov function (with vertical '+' stripes) in state space. The curves are nullclines denoting $\dot{x}(t) = 0$. In following sections, it will be shown that the appropriately modified subset of neural unit with CSO can

extend not only the range of the stability of a nonlinear controller over $[x, \dot{x}] \in R^2$ (assuming correctly identified plant) but also it can make the nonlinear controller to have optimal performance from any initial condition for any desired value.

5.6 Neural Nonlinear Controller

This study presents a methodology for specifying a neural controller for a system about which no *a priori* model information is available. The possibility of realizing feedback controllers for nonlinear dynamical systems is explored using neural designs. This topic has been addressed by many other researchers. Many prior studies have used the capability of neural networks to synthesize complex nonlinear functions. One such use is to implement state space control laws for partially known nonlinear dynamical systems. Most of these designs use iterative off-line techniques to develop the neural controller. These controllers are then inserted in a conventional feedback structure. The iterative adjustment of parameters can produce effective designs. However, such methods are computationally slow, and come equipped with no assurance of convergence. For on-line adjustment of controller parameters, the iterative methods are too slow and, in addition, introduce the danger of coupling with process dynamics. The design procedure of the present study presumes no model information about the plant. However, the design format uses minimal *a priori* information that is broadly applicable to control applications. The focus here is in tracking the output trajectory. To evaluate the neural controller in this context, it is implemented in a closed loop feedback fashion in the system. The inputs to the neural controller are the system states. The neuro-controller accepts a job description in the form of a goal trajectory that the system outputs are to be driven along. It then causally determines the input signals which stimulate the system to track the desired output trajectory. The design uses no *a priori* information about the reference trajectory. While most neural network researchers draw tacit inspiration from neurological phenomena, very few neural network designs actually reflect such origins. However, the neural architecture has no sigmoidal activation function. Because of this, I make no claims concerning resemblance to natural neural calculations. The use of higher-order derivatives provides a capability to store and distinguish information from the cross correlations and auto correlation terms of the HONU structure. Thus, from the most

recent I/O values and the target trajectory values, the HONU (neural unit with CSO), can interpolate (if necessary extrapolates) to determine the choice of the input value. Intuitively, one might view the neural unit with CSO controller as a very fast look-up and extrapolation device, with a subset of synaptic interconnections. The neural unit with CSO controller consists of a new damping function called universal damping function which provides robust tracking of the input. The following section explains the development of new damping function.

5.6.1 Development of New Damping Function

A nonlinear damping function which improves the system transient response was discussed in Section 5.3.2. Even though it improves the system transient response by making damping small for large x_1 and large for small x_1 but the operating region of x_1 remains in the range of $[-1,1]$. However, the required damping magnitude can be achieved by varying the damping gain. Now, the objective is to develop a method that determines the stability in larger regions of state space for achieving the stability in the large. The new damping function is shown in Fig. 5.17. Figure 5.18 shows the different phases in the development of new damping function that enlarges the operating region. In Figure 5.18 (b), the absolute value of the function ensures positive damping. Initially, the damping is zero at the points 'B', 'C' and gradually increased to the desired value as the target approached the point 'A'. However, the operating region still remains in the same range i.e. $[-1, 1]$. Now, in order to increase the operating region, the damping function needs to be modified. Consider the damping function with the absolute value

$$F(x) = k_v \text{abs}(1 - x^2) \quad (5.24)$$

The above equation assures positive damping in the entire range of x but the desired damping value lies in the range of 'zero' and 'one'. The damping function is again reformulated to achieve the desired characteristics.

$$F(x) = k_v \text{abs}(1 - (x - x_d)^2) \quad (5.25)$$

where x_d is the desired location. Figure 5.18 (c) shows the damping function has moved to desired target but the operating region still lies in the range of $[-1,1]$. Now, a new gain

term called the shift gain associated with the position is introduced in the damping function

$$F(x) = k_v abs(1 - (x - x_d)^2 k_{v_1}) \quad (5.26)$$

where k_{v_1} is the gain associated in expanding the operating region. In order to achieve faster transient response, the damping should be zero or even negative. This condition evaluates the shift gain as

$$(1 - (x - x_d)^2 k_{v_1}) \leq 0$$

$$k_{v_1} = \frac{1}{(x_0 - x_d)^2} \quad (5.27)$$

The damping function with desired characteristics is shown in Fig. 5.18(d) and is given as

$$F(x) = k_v abs \left[1 - \frac{(x - x_d)^2}{(x_0 - x_d)^2} \right] \quad (5.28)$$

where k_v = Damping gain

x_0 = Initial condition (Starting point)

x_d = desired location (Target to reach)

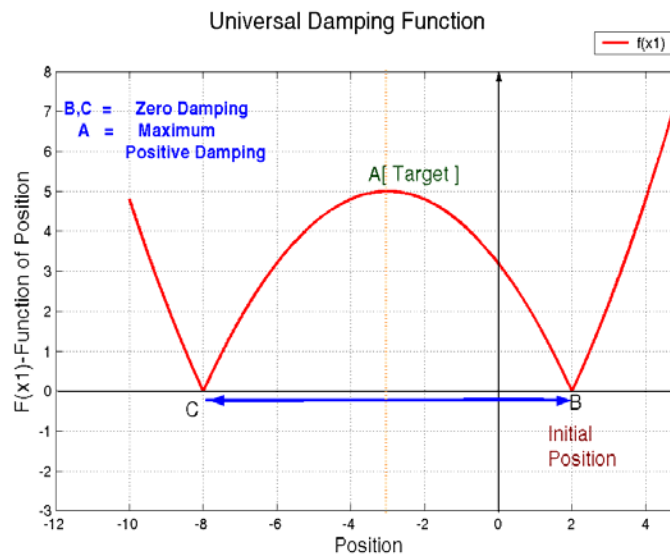


Figure 5.17 Universal damping function

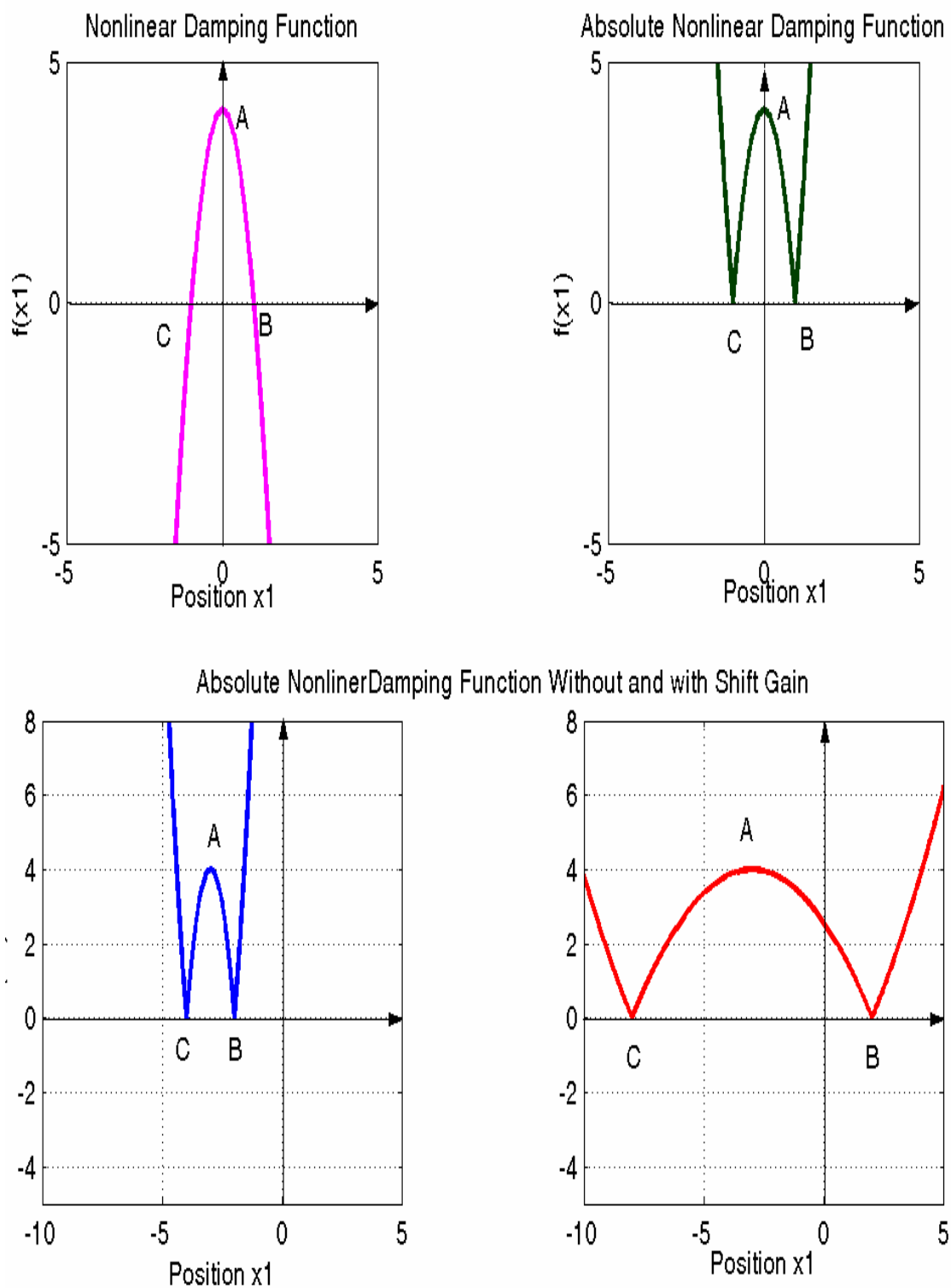


Figure 5.18 Different phases in the development of new damping function - universal damping function

5.7 Non-Linear State Feedback Neuro-Controller for Control of unknown, varying Parameter and Structure, Nonlinear Dynamic System

In this section, a structure of nonlinear state feedback controller for any second order systems with a variable damping function is introduced. A linear state feedback controller is applied to the linear second order system which is given as

$$\ddot{x}(t) = u(t) \quad (5.29)$$

then, the closed loop control equation ideally yields to a linear differential equation which is given as

$$u(t) = -k_p x(t) - k_v \dot{x}(t) + r(t) \quad (5.30)$$

with k_p , k_v as position and velocity feedback gains. Since the gains k_v as well as k_p are constant, the system is constrained in velocity by which it can reach the equilibrium point. In case of linear systems, real parts of the poles determine the stability of the closed loop control system. However, in case of nonlinear systems, there is no general method to determine the stability of the system. For example, a nonlinear equation given as

$$\dot{x}_2 = u(t) = f_1(x_1, x_2) + f_2(x_1) = 0 \quad (5.31)$$

where $x = x_1$, $\dot{x} = x_2$, has only an approximate solution. In Equation (5.31), damping is a variable and can be a function of the states (x_1, x_2) , the constants (weights), and the input value. This implies that the stability of a nonlinear system depends on many factors such as order of the nonlinearity, damping, nature of the input etc. For nonlinear systems, there are more degrees of freedom to modify the quality of the response (faster and with no overshoot) contrary to linear systems, where the damping is constant during the whole control process. A new damping function developed in Section 5.6.1 is introduced in the design of the controller as

$$f_{damping} = k_v \cdot \left| 1 - (x_1(t) - r(t))^2 k_{v1} \right| \cdot x_2(t) \quad (5.32)$$

where k_v, k_{v1} are the gains of the damping function. The damping gain k_v achieves the required magnitude of the damping and k_{v1} provides the variation of the damping from the initial position to the final position (target). The absolute value in the Eqn. (5.32)

assures the stability of the nonlinear controller for $[x_1, x_2] \in R^2$. The purpose of the nonlinear controller is to achieve a faster response by reducing the damping to a small value (initially zero) and gradually increasing it to a positive optimum value such that the system will not overshoot as soon as the desired position is reached. Assuming any square-like input function, the magnitude of k_{v1} is determined by the initial position x_0 and the desired position x_d ; that is,

$$1 - (x_1(t) - r(t))^2 \cdot k_{v1} = 1 - (x_1(0) - x_d)^2 \cdot k_{v1} = 0 \quad (5.33)$$

where $x_1(0)$ is the initial position. Therefore, the damping function is given as

$$f_{damping} = k_v \cdot \left| 1 - \left(\frac{x_1(t) - r(t)}{x_1(0) - x_d} \right)^2 \right| \cdot x_2(t) \quad (5.34)$$

where $x_1(0)$ is the initial value and x_d is the desired value. Thus the proposed structure of nonlinear stable state feedback controller is given by Eqn. (5.35).

$$\boxed{r(t) = k_v \cdot \left| 1 - \left(\frac{x_1(t) - r(t)}{x_1(0) - x_d} \right)^2 \right| \cdot x_2(t) + kp \cdot x_1(t)} \quad (5.35)$$

Assuming the control input is represented by any step-like (or square-like) function, Figure 5.19 depict the characteristic features of the nonlinear damping which is applied to the modified subset of neural unit with CSO as a state controller to an unstable, second order plant (Eqn. 5.29). The proposed state controller (Eqn. 5.35) gives almost three times faster response without any overshoot compared with any other linear controllers. The proposed structure of neuro-controller which uses the neural unit with CSO for nonlinear plant identification is shown in Fig. 5.20. A modified subset of the neural unit with CSO is depicted as a state feedback controller (i.e. controller with universal variable damping function) in the same block diagram.

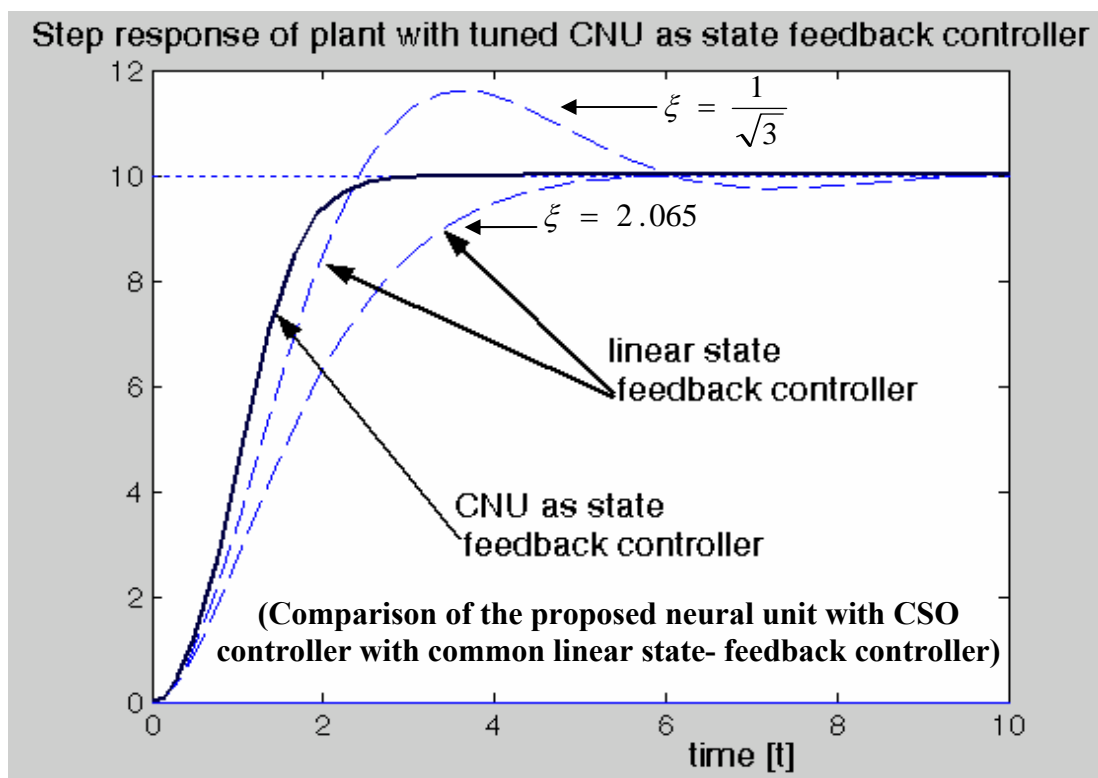


Figure 5.19(a) Step response of a satellite attitude control with different controllers.

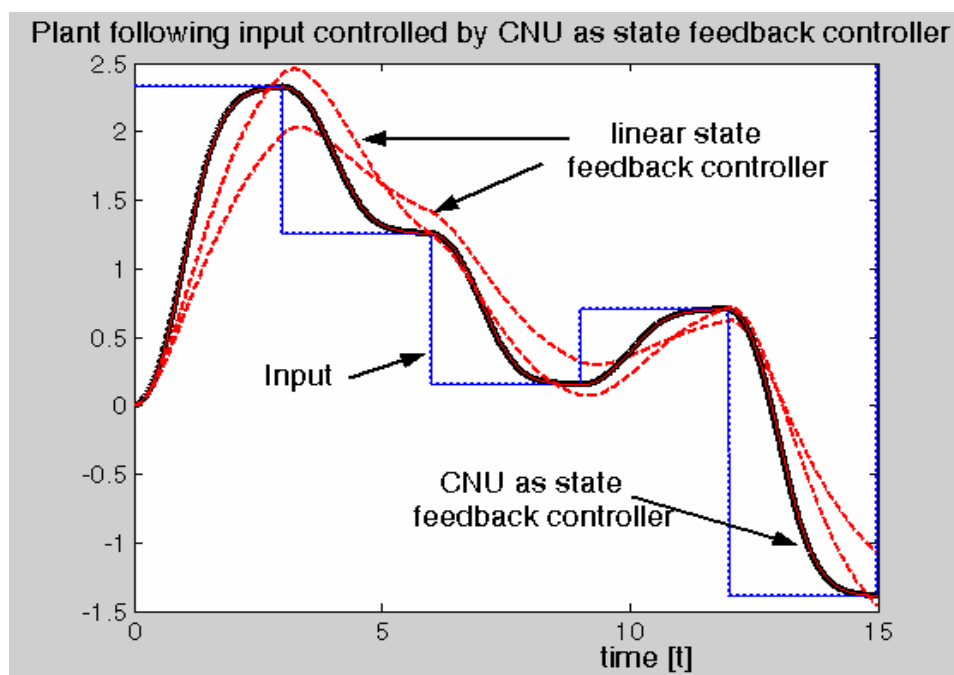


Figure 5.19(b) Response to square-like input function.

Figure 5.19 Step response of a nonlinear system (Eqn. 5.29) with proposed neural controller.

First, the neural unit with CSO identifies the plant by back propagation (e.g., in case of an unstable system, data can be acquired by measuring the position, velocity and acceleration with different plant initial conditions). Now, the neural unit with CSO acts as an identifier of the plant (shown in Fig. 5.20) and the identified parameters are passed to the function $f(\mathbf{x}, \mathbf{w})$ which corresponds to the state space representation of the dynamic system (Eqn. 5.20). The plant parameters that were identified by the neural unit with CSO are fed to the neuro-controller defined by the function as $f_{controller}$ (modified subset of the neural unit with CSO shown in Fig. 5.20). Now, the system switches into control mode.

In this mode, the neural unit with CSO can perform the control and identification of varying plant parameters with varying plant structure.

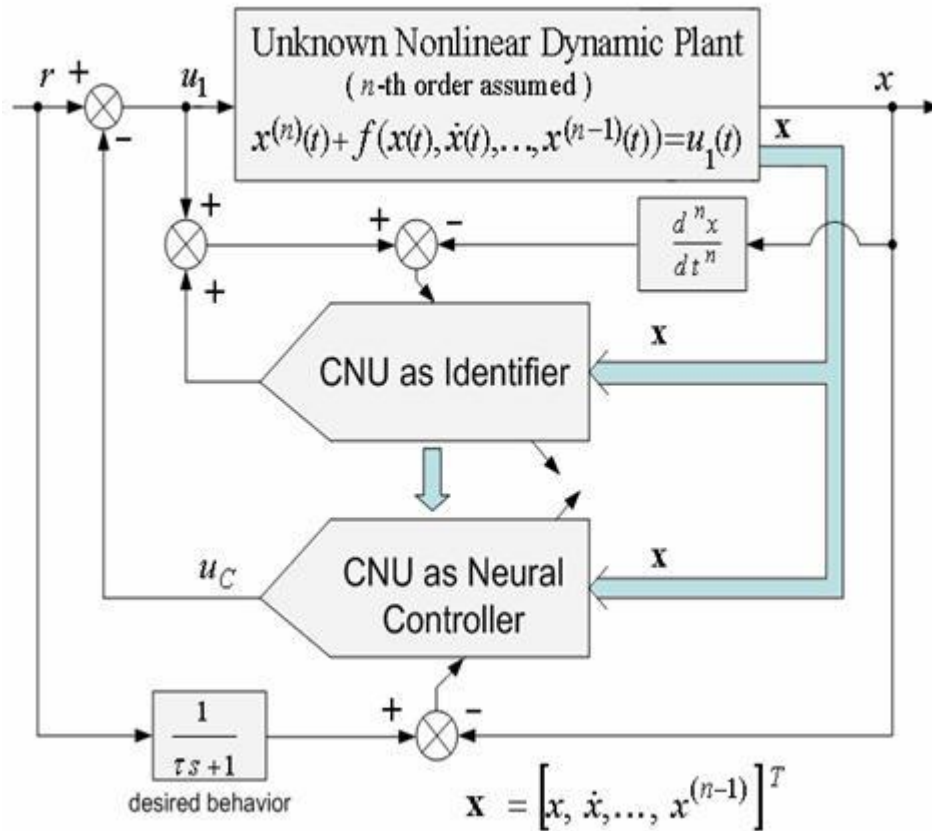


Figure 5.20 Neural unit with CSO in a dual mode, as a nonlinear neuro-controller and as an identifier, performs state feedback control for optimal performance.

However, it should be observed that convergence of the nonlinear neuro-controller tuned by back propagation (BP) in the above simulation is very sensitive (especially in case of

higher-order nonlinear systems), thus if the neural unit with CSO is to be operated in a dual mode, increased attention has to be paid on the accuracy of initially identified system parameters, size of learning rate, and the quality of data for identification.

5.8 Simulation Results

The unknown plant was represented as $\ddot{x}(t) + 0.1\dot{x}(t) + 2x(t) + 1.5x^3(t) = u(t)$. Initial weights were set as $[-10, -10, -10]$, learning rate for identification was chosen as $\mu=0.01$. The learning rate for neural control mode was set much lower than the learning rate for identification ($\mu=0.00033$) because the BP method was very sensitive. The results shown below come from simulations conducted in two distinct operating modes. In the first mode, the neural unit with CSO identified the plant and the structure switched into control mode using the identified weights as constants in the subset of neural unit with CSO. Figure. 5.21 shows that the weights were not exactly identified and the response deviated with increase in the step size of the input

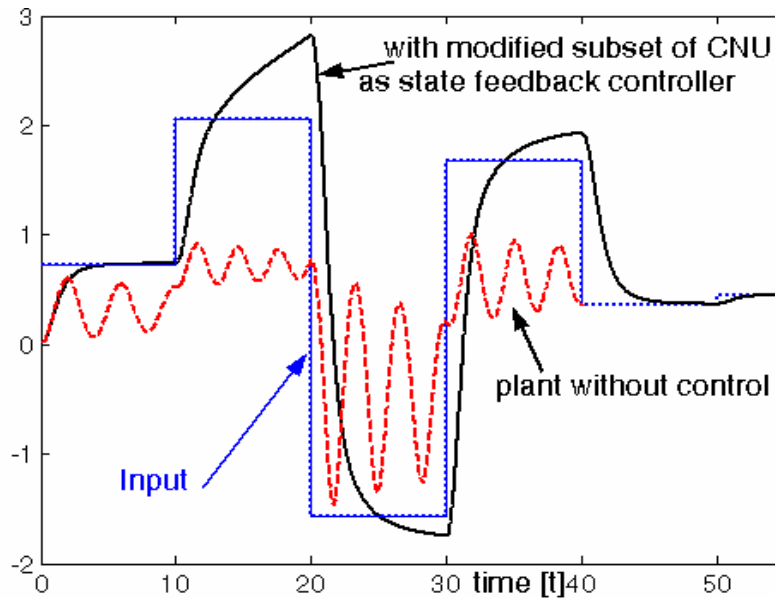


Figure 5.21 Nonlinear plant controlled by neural unit with CSO as state feedback controller with constant parameters identified by the neural unit with CSO.

In the second mode, the identified weights were used for further adaptation of the proposed neural state controller (Figs. 5.22(a), 5.22(b)) while the controller was simultaneously performing control and continuous identification of the plant parameter

values. Figure 5.22(b) manifests the capability of the controller to handle changes in parameter values of the plant where the parameter of damping changed from $a = 0.1$ to $a = -0.5$ at $t \approx 140s$, which made the plant unstable.

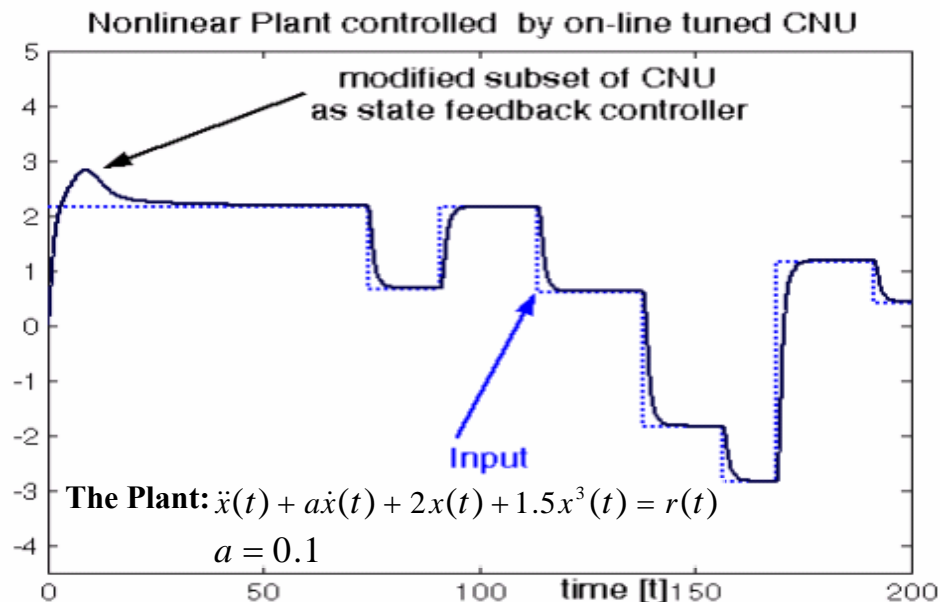


Figure 5.22(a) Nonlinear plant controlled by on-line tuned neural unit with CSO (as state feed back controller) by BP method.

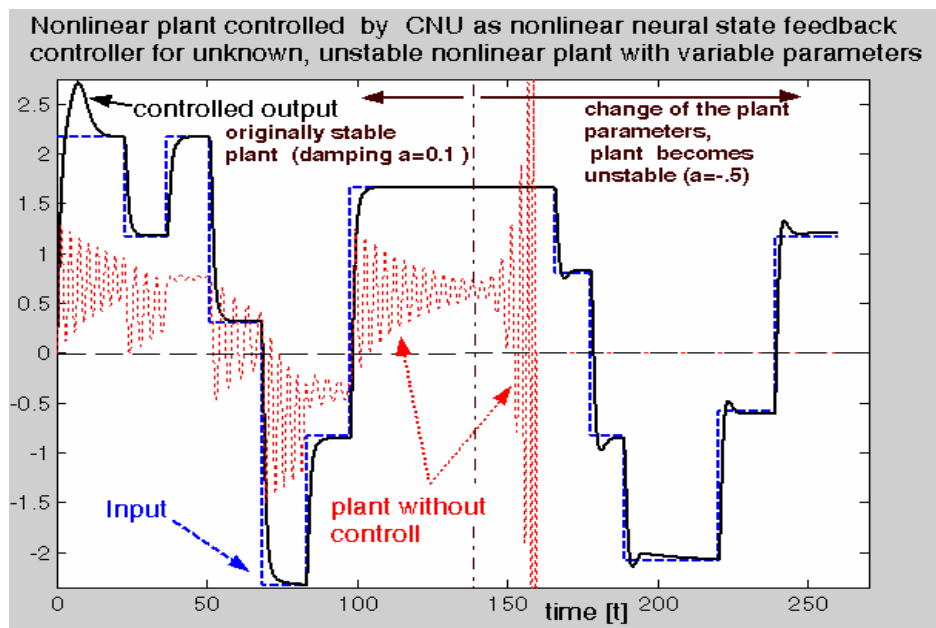


Figure 5.22 (b) Nonlinear plant controlled by neural unit with CSO as nonlinear neural state feedback controller for unknown, unstable nonlinear plant with variable parameters

5.9 Conclusions

A novel structure of a nonlinear state feedback controller which can perform almost three times faster than common linear controllers without overshoot for any square-like input function has been proposed. In combination with cubic neural unit, this controller can be applied to control any unknown, linear or nonlinear, stable or unstable second order system. The fast performance of the nonlinear controller structure has been demonstrated on a simplified satellite control problem. The capability of the neural unit with CSO (or the neural unit with QSO) to identify and control an unknown non-linear second order system with varying parameters has been demonstrated on a forced Duffing oscillator working with varying parameters including transition from stable to unstable modes. The development of neural structures such as quadratic and cubic neural units, or a subset of any higher-order neural unit and they are the potential tools for identification and control of an unknown nonlinear system with varying parameters and structure of higher-order thereby providing the scope for further research.

Moreover, the experiments with these neural structures contributes to deeper understanding to real nonlinear dynamic systems and promises to reveal fairly novel ways of handling complex dynamic systems such as a human cardiovascular system.

CHAPTER 6

Conclusions

6.1 Concluding Remarks

Neurons are the basic building block of the central nervous system (CNS) which is a central feature of the life. The CNS governs how we grow, respond to stress and challenge, and regulate factors such as body temperature, blood pressure, and cholesterol levels. The mechanisms operate at every level, from the interaction of proteins in cells to the interaction of organisms in complex ecologies. Like wise, the mathematical models of neurons do operate at different levels, from the interaction of robots at molecular level to the interaction of modern machines at macro level like “Spirit Rover” operating in a complex unpredictable environment.

Neural networks have undoubtedly been biologically inspired, but the close correspondence between them and the real neural systems is still rather weak. Despite the loose analogy between the mathematical models and the natural neural systems, a new structure of the neuron called the neural unit with quadratic synaptic operation (QSO) was developed. The architecture and mathematical model of the neural unit with QSO has been presented. The neural unit with QSO incorporates linear as well as nonlinear combinations of weighted neural inputs generated by the preprocessor. The performance of the neural unit with QSO was greatly enhanced as the size of the weight matrix was reduced from

$$[n \times n] \text{ to } \left[\frac{n \times (n+1)}{2} \right], \text{ and } n = (N + 1)$$

where n is the number of inputs and N is the order of the neuron. This was a very significant contribution as it improves the performance of the neuron; that is, the training time would be reduced greatly as the number of parameters in the weight matrix was reduced. The concept of neural unit with quadratic synaptic operation (QSO) can be extended to develop any other higher-order neural units such as the neural unit with cubic synaptic operation (CSO) and so on. Due to their higher-order combination of the neural inputs, either the neural unit with QSO or the neural unit with CSO can be trained to learn

and control the unknown nonlinear dynamic systems. A general methodology for developing the HONUs with higher-order synaptic operations was presented using sigma tuner and correlation operator. The proposed neural model closely resembles the structure of the biological neuron. However, it is not claimed that the neural model in this thesis incorporates all properties of the biological neuron.

The Human ability to find patterns in the external world is ubiquitous. It is at the core of our ability to respond in a more systematic and reliable manner to external stimuli. Humans do it effortlessly, but the mathematics underlying the analysis and the design of pattern-recognition machines are still in their infancy. The neural unit with QSO is a basic step towards the development of such efficient machines to deal with the real world problems which are complex and unpredictable.

In Chapter 4, different logic circuits such as **OR**, **AND**, and **Exclusive-OR (XOR)** were realized using a single neural unit with QSO. The mathematical model of the neural unit with QSO was closely examined. The weight matrix beautifully encapsulates the concept of the Euclidian distance, the Mahalanobis distance and the affect of the threshold (bias) on the shape and the placement of the discriminant surface. The approximation capabilities of the neural unit with QSO were discussed in this chapter. The accuracy of the approximation does depend on the structure of the neurons employed in a network. The simulation studies of the neural unit with QSO provide enough evidence that it is a better computational node for the function approximation problems. It was well known fact that the MFNNs were considered as universal approximators for continuous functions. However in authors view, a network of neural units with QSOs would provide better approximation results than the results achieved by the MFNNs. Apart from this, the HONUs with higher-order synaptic operations can be expressed using different combinations of the neural unit with QSO. Hence, it is the most general neural unit which can deal with both linearties and nonlinearities of the real world problems.

A novel structure of a nonlinear state feedback controller with universal damping function was proposed. This controller provided almost three times faster transient response than the common linear controllers without overshoot for any square-like input function. In combination with cubic neural unit, this controller can be applied to control

any unknown, linear or nonlinear, stable or unstable second-order system. The fast performance of the nonlinear controller structure was demonstrated on a simplified satellite control problem. The neural unit with CSO (or the neural unit with QSO) to identify and control an unknown nonlinear second order system with varying parameters was implemented on forced Duffing oscillator with varying parameters including transition from stable to unstable modes. The development of neural structures as neuro controllers with quadratic and cubic synaptic operations, or a subset of any higher-order synaptic operation would provide a better performance than any other linear or nonlinear controllers. These neural controllers are very sensitive to the learning rate and utmost care must be taken during the training process. They can be considered as the potential tools for identification and control of an unknown nonlinear system with varying parameters and structure of higher-order systems thereby providing the scope for further research. In most control systems, disturbances of one type or another exist. In this research, only a simplified model of satellite attitude control was considered for simulation studies. The system response to disturbance inputs, noise and parameter sensitivity were not considered in the simulation studies. However, the simulation studies and the experiments with these neural structures could contribute to the deeper understanding of nonlinear dynamic systems and promise to reveal fairly novel ways of handling complex dynamic systems such as a human cardiovascular system, robot path planning and weather forecasting.

6.2 Contributions of the Thesis

The main contribution of this thesis was the development of higher-order neural units with higher-order synaptic operation. Based on this concept, two neural units one with quadratic synaptic operation and the other with cubic synaptic operation were proposed. The mathematical model closely resembles the topology of the biological neuron in the central nervous system (CNS). Of course, the mathematical complexity restricts one to incorporate all features of the biological neuron. The learning and adaptation algorithms and their implementation scheme were outlined for the proposed neural units.

In conventional control systems, first and second-order systems can adequately represent any higher-order system and fairly reveal the characteristics of the system. Likewise, it is presumed that the neural units with quadratic and cubic synaptic operation can express any higher-order neural unit with higher-order synaptic operation. The major contribution of this thesis was the reduction and representation of the parameters in the novel weight matrix without losing the important information associated with the neural inputs. For the neural unit with quadratic synaptic operation (QSO), the parameters are represented in an upper triangular matrix in a quadratic form. In advanced mathematics, quadratic representation has significance because there are many applications in which the quadratic function appears and many functions can be approximated by them in small neighborhoods, especially near local minimum points.

A general expression for the higher-order neural unit with higher-order synaptic operation was given in Chapter 3. Any $N-1^{\text{th}}$ higher-order neuron is a subset of N^{th} higher-order neuron. This assumption is valid only when the bias of the neuron is associated with the augmented weight matrix. It is always possible to find $(N-1)$ higher-order neurons in any N^{th} higher-order neuron; that is, neural units with quadratic and linear synaptic operations are a subset of the neural unit with cubic synaptic operations. A close observation of the general higher-order synaptic operation reveals that the neural units with different synaptic operations are systematically arranged from top to bottom at different levels of the pyramid known as the neural unit with N^{th} order synaptic operation. This type of structure and representation of synaptic operation may lead to different direction of research in the field of neural networks. The structure is completely different from the conventional representation of neural units in the following way

- i. Cross and self correlations of the neural inputs are considered. These inputs incorporate pretrained data there by reduce the training time during the learning process.
- ii. The parameters are reduced significantly without losing the important information associated with the neural units.

The performance of the neural units with quadratic and cubic synaptic operation was compared, through simulation studies, with the conventional neural units and the existing higher-order neural structures. In particular, the neural unit with quadratic synaptic operation was applied to static problems such as pattern recognition and function approximation problems. For both problems, the performance of the neural unit with QSO was found to be better than the conventional neural units especially in realization of the logic circuits such as **XOR**, **OR**, and **AND**. This is due to the fact that the mathematical model of the neural unit encapsulates the effect of the threshold (bias), the mean (Euclidian distance) and the auto and self correlation terms (radial distance). The concept of Mahalanobis distance interprets the neural unit with QSO as an optimal classifier.

The analysis of the logic circuits, **XOR**, **OR** and **AND**, strengthened the importance of the weight matrix (covariance matrix) as it decides the placement of the decision boundary for the classification problems. The sign of the determinant of the weight matrix determines the type of classification: good or poor classification. During the simulation studies, it was observed that the neural unit with QSO approximated the nonlinear functions to the desired degree of accuracy. The author believes that the better results could be achieved if a network of neural units with QSO is used for function approximation problems.

Well known adaptive methods such as model reference adaptive control (MRAC) methods were used to study the neural units with higher-order synaptic operation as neuro controllers for the control of complex problems such as satellite attitude control. A new damping function called universal damping function was implemented in the neuro controllers which increased the speed of the transient response three times faster than the transient response achieved by the conventional controllers. The main advantage of using this type of damping function was that it provides the robust tracking of the input without any overshoot in the transient response. It was found that the neuro controllers were too sensitive to the gain of the learning process. The author highly recommends utmost care to be taken while choosing the learning rate for the neural units with higher-order synaptic operation as neural controllers.

The development of the neural units with higher-order synaptic operations like quadratic synaptic operation (QSO) and cubic synaptic operation (CSO) is a basic step towards the development of intelligent machines to deal with the real world problems which are complex and unpredictable. Though researchers still have a long way to provide significant breakthroughs into an understanding of the human intelligentsia-Cognition and Perception, this work hints at the possibilities of developing the useful biological mathematical models for engineering applications.

6.3 Future Scope of the Research

This thesis has presented the basic concept of higher-order neural units with higher-order synaptic operation and their topology based on the structure of the biological neuron for control and pattern recognition problems. The author does not make any claims that the proposed structure of the neural unit has all features of the biological neuron.

The performance of the neural units can be improved by adapting the slope of the activation function and including the dynamic elements such as delays in the mathematical model of the neuron but the inclusion comes at the cost of the mathematical complexity; that is, the problem arises in developing the learning and adaptation algorithms. Inter and Intra feedback can be associated with the structure of neuron. These neurons are called P-N type (Positive and Negative neurons). This type of neural structures have potential applications in the analysis of complex problems such as image processing, human cardiovascular system, robot path planning and weather forecasting.

From technical point of view important questions regarding the overall performance, speed and stability of the higher-order neural units need to be addressed. It is very difficult to find a general stability rule for nonlinear systems that would readily address the stability of the system. Basic concepts like energy method and Lyapunov function should be redefined and refined for achieving the desired stability. The real challenge would be the incorporation of these learning and adaptive algorithms for general higher-order neuron into hardware circuitry.

Recent advances in the field of fuzzy logic and the new emerging field, fuzzy neural networks which is a marriage between the fuzzy logic and neural networks, should

provide significant breakthroughs for the implementation and the realization of the subjective phenomena such as cognition and perception for the creation of intelligent machines. It would be interesting and challenging to integrate the principles of the higher-order neural units and the fuzzy logic to develop completely new area of research.

References

1. Alligood, K. T., Sauer, T. D., and Yorke, J. A. [1996], "Chapter 7: Differential Equations," in *Chaos: An Introduction to Dynamical Systems*, New York, Springer-Verlog.
2. Antsaklis, P. J. [1994], "Defining Intelligent Control," Report of the Task Force on Intelligent Control, Chair, *IEEE Control Systems Magazine*, pp. 4-5 & 58-66.
3. Eric R. Kandel, James H. Schwartz, Thomas M. Jessel [2000], *Principles of Neural Science*, McGraw-Hill Companies, Inc. U.S.A (Previous edition copyright 1991 –Appleton & Lange).
4. Fu, K. S. [1970], "Learning Control Systems? Review and Outlook," *IEEE Transactions on Automatic Control*, vol. AC- 15, pp. 210-221, April.
5. Fukushima, K., Miyake, S. and Ito, T. [1983], "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Trans. Systems, Man and Cybernatics*, Vol. 13, No. 5, pp. 826-834, Sept/Oct.
6. Giles, C. L. and Maxwell, T. [1987], "Learning Invariance and Generalization in higher-order networks," *Appl. Optics*, Vol.26, pp.4972-4978.
7. Grossberg, S. [1988], "Nonlinear Neural Networks: Principles, Mechanisms and Architectures," *Neural Networks*, Vol. 1, pp. 17-61
8. Gupta, M. M. [1970's], "Notes on Some Advanced Topics in Feedback Control System Design," *ME 441.3: Advanced topics in Linear Control System*, Intelligent Systems Research Laboratory, University of Saskatchewan, Saskatoon, Canda.
9. Gupta, M. M. [1986] Ed., "*Adaptive Methods for Control System Design*," IEEE Press, New York.
10. Gupta, M. M. and Rao, D. H. [1994], "Neuro Control Systems: A Tutorial," in *Neuro Control Systems: Theory and Applications*, Gupta, M. M. and Rao, D. H. (eds.), IEEE Press, New York, pp. 1-46.
11. Gupta, M. M., Liang, J. and Homma, N. [2003], "*Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory*," IEEE Press and Wiley-Interscience, published by John Wiley & Sons, Inc.

12. Hagan, M. T., Demuth, H. B., and Beale, M. [1999], "Neural Network Design," PWS Publishing Company, Boston, MA.
13. Heywood, M. and Noakes, P. [1995], "A Framework for Improved Training of Sigma-Pi Networks," *IEEE Transaction on Neural Networks*, vol.6, N: 4, pp. 893-902.
14. Homma, N. and Gupta, M. M. [2002 b], "A general Second-Order Neural Unit," *Bull. Coll. Med. Sci. Tohoku Univ.*, Vol. 11, No. 1, pp. 1-6.
15. Hopfield, J.J. [1990], "Artificial Neural Networks are Coming," *IEEE Expert*, An Interview by W. Myers, pp. 3-6.
16. Ikonen, E., and Najim, K. [2002], "Chapter 1: *Introduction to Identification*," Advanced Process Identification and Control, Marcel Decker, Inc., New York, pp.3.
17. Kuroe, Y., Ikeda, H. and Mori, T. [1997], "Identification of Nonlinear Dynamical Systems by Recurrent High-Order Neural Networks," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 70-75.
18. Lau, C. [1992], "Neural Networks: *Theoretical Foundations and Analysis*" IEEE Press, New York.
19. Leda, V. and Francis, L. M. [1995], "Learning Capability Assessment and Feature Space Optimization for higher-order Neural Networks," *IEEE Trans. on Neural Networks*, vol. 6, no. 1, pp. 267-272.
20. Leigh, J. R. [2004], "Chapter 7: *Limits to Performance*," in Control Theory, Second edition, *IEE Control Series 64*, London, UK.
21. Lilly, S. and Max B. Reid [1993], "Coarse-Coded higher-order Neural Networks for PSRI Object Recognition," *IEEE Trans. on Neural Networks*, Vol. 4, No. 2, pp. 276-283
22. McCulloch, W. S. and Pitts, W. [1943], "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133
23. Miller, W. Thomas, Sutton, Richard S., and Werbos, Paul J. [1990], *Neural Networks for Control*, The MIT Press, Massachusetts.
24. Minsky, M. L., and Papert, S. A. [1969], *Perceptrons*, MIT Press, Cambridge, MA

25. Narendra, K. S. [1986], "Adaptive and Learning Systems-Theory and Applications," Plenum Press, New York.
26. Ogata, K. [1984], Modern Control Engineering, Easter Economy Edition, Prentice-Hall Electrical Engineering Series, New Delhi.
27. Ortega, R. and Tang, T. [1989] "*Robustness of Adaptive Controller-A Survey*," Automatica, vol.25, no.5, pp. 651-677.
28. Panel discussion on "Machine Learning in a Dynamic World," Proc. of the 3rd IEEE Intern. Symposium on Intelligent Control, Arlington, VA, August 24-26, 1988.
29. Pavlidis, T. [1977], "Structural Pattern Recognition," *Springer Series in Electrophysics*, Vol. 1, pp. 1-10
30. Peek, M.D. and Antsaklis, P.J. [1990], "Parameter Learning for Performance," IEEE Control Systems Magazine, December, pp. 3- 11.
31. Principe, J. C., Euliano, N. R., and Lefebvre, W. C. [2000], "Neural and Adaptive Systems: Fundamentals through Simulations" John Wiley & Sons, Inc.
32. Rao, D. H. [1994], "*Development of Dynamic Neural Structures with Control Applications*," Ph.D thesis, University of Saskatchewan, Saskatoon, SK, Canada. S7N 5A9.
33. Roger L. K, Aric B. L, Samuel H. R, and Donna S. R., "The Biological Basis of the Immune System as a Model for Intelligent Agents," MSU/NSF Engineering Research Center for Computational Field Simulation, Mississippi State, MS 39762-9627, U.S.A.
34. Rosenblatt, F. [1959], "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, Vol. 65, pp. 386-408
35. Rumelhart, D. E. and McClelland, J. L. [1986], "Parallel Distributed Processing: Explorations in the Microstructure of Cognition," The MIT Press, Cambridge, MA.
36. Rumelhart, D. E., Hinton, G. E. and Williams R. J. [1986], "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, vol. 1,

- D.E. Rumelhart and J.L. McClelland, Eds. Cambridge, MA: MIT Press, pp. 318-362.
37. Russ B. Altman [2001], "Perspectives- Challenges for Intelligent Systems in Biology," *IEEE Intelligent Systems*.
 38. Shin, Y. and Ghosh, J. [1991], "The Pi-Sigma Network: An Efficient higher-order Network for Pattern Classification and Function Approximation," *Proc. Int.Joint Conference on Neural Networks IJCNN*, Seattle, vol. I, pp. 13-18.
 39. Sklansky, J. [1996] "Learning Systems for Automatic Control," *IEEE Transactions on Automatic Control*, vol. AC-11, pp. 6-19.
 40. Softky, R. W. and Kammnen, D. M. [1991], "Correlations in High Dimensional or Asymmetrical data Sets: Hebbian Neuronal Processing," *Neural Networks*, Vol. 4, No.3, pp. 337-347.
 41. Song, Y [2001], "*Development of Dynamic Neural Units with Control Applications*," Masters Thesis, University of Saskatchewan, Saskatoon, SK, Canada. S7N 5A9.
 42. Specht, D. F. [1967], "Vectorcardiographic Diagnosis using the Polynomial Discriminant Method of Pattern Recognition," *IEEE Trans. Biomed. Eng.*, Vol. BME-14, pp. 90-95.
 43. Stevens, C. F. [1968], "Synaptic Physiology," *proc. IEEE*, vol.79, no.9, pp916-930.
 44. Taylor, J. G. and Commbes, S. [1993], "Learning higher-order Correlations," *Neural Networks*, Vol. 6, No. 3, pp. 423-428.
 45. Tsypkin, Y. [1968], "Self-Learning: What Is It?," *IEEE Transactions on Automatic Control*, vol. AC-13, pp. 608-612, December.
 46. Widrow, B. and Michael, A. Lehr [1992], "30 years of Adaptive Neural Networks: Perceptron, Madaline, and Back-propagation," in *Neural Networks: Theoretical Foundations and Analysis*, edited by Clifford Lau, IEEE Press, pp. 27
 47. Xu, X., Oja, E., and Suen, C. Y. [1992], "Modified Hebbian Learning for Curve and Surface Fitting," *Neural Networks*, Vol. 5, No. 3, pp. 441-457.

48. Zhengquan, H. and Siyal, M. Y. [1998], "Modification on higher-order Neural Networks," *Proceedings of the Artificial Networks in Engineering Conference*, Vol. 8, pp. 31-36.