INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600
Modified Newton’s method for supervised training of dynamical neural networks for applications in associative memory and nonlinear identification problems

Bhalala, Smita Ashesh, M.S.
The University of Arizona, 1991
MODIFIED NEWTON'S METHOD FOR SUPERVISED TRAINING OF
DYNAMICAL NEURAL NETWORKS FOR APPLICATIONS IN ASSOCIATIVE
MEMORY AND NONLINEAR IDENTIFICATION PROBLEMS

by

Smita Ashesh Bhalala

A Thesis Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE
WITH A MAJOR IN ELECTRICAL ENGINEERING
In the Graduate College
THE UNIVERSITY OF ARIZONA

1991
STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotations from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgement the proposed use of the material is in the interest of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: [Signature]

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

[Signature] [Date]

Malur K. Sindareshan
Professor of Electrical Engineering
ACKNOWLEDGEMENTS

I wish to express my appreciation to Dr. Malur K. Sundareshan for his helpful guidance and encouragement during the course of this study. I would also like to thank Dr. Hal S. Tharp and Dr. Robert A. Schowengerdt for donating their time as members of the final exam committee. I am especially grateful to Dr. Ahmet Karakasoglu for sharing his knowledge of neural networks.

I wish to thank the Hughes Aircraft Company Fellowship Program for providing the financial support and time to pursue this degree. My management and friends, Lou Giattino, Stan Wong, Brian Cline, and Jim Gottlieb deserve special recognition for providing the working environment which made this study possible.

I would like to thank my husband, Ashesh, for his understanding and encouragement during the course of this study. The arrival of my wonderful son, Samit, provided the boost necessary to complete this research. Finally, I must thank my parents, Madhu and Babu Bhalala and Jaya and Parshottam Tala, without whom this would have never been possible.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>6</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>8</td>
</tr>
<tr>
<td>CHAPTER 1 - INTRODUCTION</td>
<td>9</td>
</tr>
<tr>
<td>1.1 A Brief Overview of Neural Networks</td>
<td>10</td>
</tr>
<tr>
<td>1.2 Application of Neural Networks in Information Processing</td>
<td>14</td>
</tr>
<tr>
<td>1.3 Outline of Thesis</td>
<td>16</td>
</tr>
<tr>
<td>CHAPTER 2 - NEURAL NETWORK MODELS AND LEARNING ALGORITHMS</td>
<td>17</td>
</tr>
<tr>
<td>2.1 Static Network vs. Dynamical Network</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Classification of Learning Schemes</td>
<td>24</td>
</tr>
<tr>
<td>2.3 Learning Rules for Supervised Training</td>
<td>26</td>
</tr>
<tr>
<td>CHAPTER 3 - LEARNING RULES FOR ASSOCIATIVE MEMORY APPLICATIONS</td>
<td>30</td>
</tr>
<tr>
<td>3.1 Basic Properties of Dynamical Networks</td>
<td>31</td>
</tr>
<tr>
<td>3.1.1 Characterization of Equilibrium Points</td>
<td>33</td>
</tr>
<tr>
<td>3.1.2 Selection of Gain of the Nonlinear Function</td>
<td>36</td>
</tr>
<tr>
<td>3.2 Learning Rule Using Modified Newton’s Method</td>
<td>38</td>
</tr>
<tr>
<td>3.3 Convergence Criteria for Modified Newton’s Method</td>
<td>45</td>
</tr>
<tr>
<td>3.4 Comparison of Learning Rules</td>
<td>48</td>
</tr>
<tr>
<td>3.4.1 Training Rule using LMS Method</td>
<td>49</td>
</tr>
<tr>
<td>3.4.2 Discussion with Examples</td>
<td>50</td>
</tr>
<tr>
<td>3.5 Conclusions</td>
<td>61</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 4 - LEARNING RULES FOR NONLINEAR IDENTIFICATION APPLICATIONS</td>
<td></td>
</tr>
<tr>
<td>4.1 Description of Neural Network</td>
<td>67</td>
</tr>
<tr>
<td>4.2 Learning Rule using Modified Newton’s Method</td>
<td>69</td>
</tr>
<tr>
<td>4.3 Comparison of Learning Rules</td>
<td>76</td>
</tr>
<tr>
<td>4.3.1 Training Rules using LMS Method</td>
<td>77</td>
</tr>
<tr>
<td>4.3.2 Discussion with Examples</td>
<td>78</td>
</tr>
<tr>
<td>4.4 Conclusions</td>
<td>89</td>
</tr>
<tr>
<td>CHAPTER 5 - CONCLUSIONS</td>
<td>93</td>
</tr>
<tr>
<td>LIST OF REFERENCES</td>
<td>97</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Biological Neuron Model</td>
<td>12</td>
</tr>
<tr>
<td>1-2</td>
<td>Artificial Neuron Model</td>
<td>13</td>
</tr>
<tr>
<td>2-1</td>
<td>Static Feedforward Node</td>
<td>19</td>
</tr>
<tr>
<td>2-2</td>
<td>A Sigmoidal Function</td>
<td>21</td>
</tr>
<tr>
<td>2-3</td>
<td>Diagram of Multilayer Feedforward Static Network</td>
<td>22</td>
</tr>
<tr>
<td>2-4</td>
<td>Diagram of a Layer of Dynamical Nodes with Feedback</td>
<td>23</td>
</tr>
<tr>
<td>2-5</td>
<td>Architecture of Dynamical Neural Network</td>
<td>25</td>
</tr>
<tr>
<td>3-1</td>
<td>Flow Diagram of Associative Memory Learning Algorithm</td>
<td>51</td>
</tr>
<tr>
<td>3-2</td>
<td>Learning Curve for Example 3.1 with Sigmoidal Gain ( \lambda = 2\pi )</td>
<td>53</td>
</tr>
<tr>
<td>3-3</td>
<td>Learning Curve for Example 3.1 with Sigmoidal Gain ( \lambda = 30\pi )</td>
<td>55</td>
</tr>
<tr>
<td>3-4</td>
<td>Learning Curve for Example 3.1 with Sigmoidal Gain ( \lambda = 0.01\pi )</td>
<td>56</td>
</tr>
<tr>
<td>3-5</td>
<td>Learning Curve for Example 3.2</td>
<td>58</td>
</tr>
<tr>
<td>3-6</td>
<td>Learning Curve for Example 3.3 with ( \mu = 0.09 )</td>
<td>59</td>
</tr>
<tr>
<td>3-7</td>
<td>Learning Curve for Example 3.3 with ( \mu = 0.05 )</td>
<td>60</td>
</tr>
<tr>
<td>3-8</td>
<td>Learning Curve for Example 3.3 with ( \mu = 0.07 )</td>
<td>62</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3-9</td>
<td>Learning Curve for Example 3.3 with $\mu = 0.18$</td>
<td>63</td>
</tr>
<tr>
<td>4-1</td>
<td>Architecture for 3-layer Dynamical Network</td>
<td>70</td>
</tr>
<tr>
<td>4-2</td>
<td>Flow Diagram of Nonlinear Identification Algorithm</td>
<td>79</td>
</tr>
<tr>
<td>4-3</td>
<td>Identification Architecture</td>
<td>80</td>
</tr>
<tr>
<td>4-4</td>
<td>Learning Curve for Example 4.1 with Sigmoidal Gain $\lambda = 20\pi$</td>
<td>82</td>
</tr>
<tr>
<td>4-5</td>
<td>Output of Plant and Output of Trained Network for Example 4.1 with Sigmoidal Gain $\lambda = 20\pi$</td>
<td>84</td>
</tr>
<tr>
<td>4-6</td>
<td>Learning Curve for Example 4.1 with Sigmoidal Gain $\lambda = 3.5\pi$</td>
<td>85</td>
</tr>
<tr>
<td>4-7</td>
<td>Output of Plant and Output of Trained Network for Example 4.1 with Sigmoidal Gain $\lambda = 3.5\pi$</td>
<td>86</td>
</tr>
<tr>
<td>4-8</td>
<td>Difference Between Desired Output and Trained Neural Network Output for Example 4.1 for Modified Newton's Method</td>
<td>87</td>
</tr>
<tr>
<td>4-9</td>
<td>Learning Curve for Example 4.1 with Sigmoidal Gain $\lambda = 0.01\pi$</td>
<td>88</td>
</tr>
<tr>
<td>4-10</td>
<td>Learning Curve for Example 4.2</td>
<td>90</td>
</tr>
<tr>
<td>4-11</td>
<td>Output of Plant and Output of Trained Network for Example 4.2</td>
<td>91</td>
</tr>
</tbody>
</table>
ABSTRACT

There have been several innovative approaches towards realizing an intelligent architecture that utilizes artificial neural networks for applications in information processing. The development of supervised training rules for updating the adjustable parameters of neural networks has received extensive attention in the recent past. In this study, specific learning algorithms utilizing modified Newton's method for the optimization of the adjustable parameters of a dynamical neural network are developed. Computer simulation results show that the convergence performance of the proposed learning schemes match very closely that of the LMS learning algorithm for applications in the design of associative memories and nonlinear mapping problems. However, the implementation of the modified Newton's method is complex due to the computation of the slope of the nonlinear sigmoidal function, whereas, the LMS algorithm approximates the slope to be zero.
CHAPTER 1

INTRODUCTION

In recent years, one of the ultimate challenges in science has been to develop a computer that learns with a sense of perception and recognition. This natural intelligence in a machine would display behavior analogous to biological systems, rather than executing sequential stepwise processes as is done by digital computers. Presently, all automated information processing is based on digital computers. The development of digital computers and the recent trend towards utilizing parallel processors, has provided more insight in solving technical problems [1].

Digital computers are purely information processing units. They have limited abilities in solving self-adaptive, decision-making problems. Prior to processing information of some function, a person has to understand that function, write a step-by-step algorithm for its implementation, and then program the computer to carry out the desired task. Due to these limitations of the digital computer, researchers are inspired to build an architecture with decision-making abilities. The motivation in developing such an architecture is based on current studies of the mammalian brain, particularly the cerebral cortex.

The human brain is one of the most complex architectures known to mankind with an appealing ability relating to thinking, remembering, problem solving, and
observation. These capabilities have inspired many scientists to attempt computer modeling of the brain's operation and structure in order to devise an alternate scheme of processing information.

1.1 A Brief Overview of Neural Networks

Currently, researchers are trying to create and modify a computer model of the brain. The model must match the functionality of the brain in a very simplified manner. This is the foundation for the study of neurocomputing. A simple framework for a neurocomputing system is a brain-like structure called a neural network. A neural network is a model that consists of a large number of simple, mutually interconnected neuron-like elements which process information by developing associations between objects in response to their environments [2].

Neurocomputing is generally based on functions and structures of the brain. There need not however exist a complete correspondence between the detailed functioning of the physiological-neural system and the artificial-neural system. Due to this lack of correspondence between these two systems, it is important to outline the analogy between them for the purpose of processing information [2].

The human brain primarily consists of neurons, axons, dendrites, and synapses [3]. Neurons are a large number of nerve cells with simple nonlinear characteristics. Axons are attached to neurons and transmit output signals through the host neuron. The synapses are the weighted connections of neurons. Finally, the function of dendrites is to receive information from other neurons through synapses which occur
where the dendrites of two different neurons meet.

Physiological information processing in the neuron is shown in Figure 1-1. First, input signals enter the neuron through synapses, which regulate the amount of information that passes from the neuron. The signals pass through the activated area of the neuron which adds the various input signals. After summing the signals, the resultant signal passes through a nonlinear threshold function which determines the energy level above which the neuron will generate an output. If the sum is greater than the threshold value, then the neuron sends energy through the axon where it is transmitted to other synapses or it may also be fed back into the original synapses. If the sum is less than the threshold, the neuron will not process the information.

Similarly, each neuron in an artificial neural network becomes a processing element. The axons and dendrites both become wires and the synapses represent the sums of weights of other processing elements [2].

In a neurocomputing system, the information processing in the neuron is illustrated in Figure 1-2. The input signals enter the processing element through weighted connections where information is stored. The input signals are added and are fed into a nonlinear threshold function, which produces an output that is a function of the input signals. This output is created if the threshold conditions are met. To the best of our knowledge, this is a fair comparison between the behaviors of biological and artificial neurons [2].

Neural networks are comprised of a number of processing elements organized into layers. Some networks may have one or more hidden layers with many hidden
Figure 1-1. Biological Neuron Model
Figure 1-2. Artificial Neuron Model
nodes. The power of the network lies in the interconnection strengths of the nodes within and between layers. Usually within a network, the processing elements are interconnected in a way that best suits a specific application.

One important attribute of a neurocomputing system is its adaptive and training capability. In the training procedure, the weighted connections between the processing elements in the network are adjusted to achieve a desired output. The specific characteristics of neural networks in regard to training and adaptability to store information will be discussed later in this thesis.

1.2 Application of Neural Networks in Information Processing

Several research groups are trying to apply neurocomputing systems in information processing for solving specific problems. Some newly emerged capabilities of neural networks have successfully encouraged their application in areas of pattern recognition, signal processing, and controls. In this section, we will briefly discuss the application of neural networks to associative memory and nonlinear identification.

Generally, the conventional digital computer will retrieve and store information in a sequence of deterministic operations. There is a need for specified memory locations or addresses to perform these operations. In contrast, an important trait of the neurocomputing system is the way it retrieves and stores information. This system has no specific memory locations for storing data. The memory in neural networks is both distributed and associative [4]. The information
is distributed throughout the network in the form of weighted connections between processing elements. When a network accepts an input, the associative memory compares the input data with all possible matched data in the memory. Then it uses the weighted connections distributed throughout the network to retrieve appropriate information.

Content addressable memory or associative memory models are based on storing and retrieving binary patterns which make them suitable for hardware implementation [5,6]. These models are adapted by adjusting the interconnection strength between the processing elements. The application of neural networks for associative memory problems will be examined in more detail in the later sections of this thesis.

In the past, major advances have been made in the area of adaptive identification. More specifically, these methods are used in controller design by identifying an unknown plant model. A major advantage of utilizing neural networks for this application comes from their ability to adapt and learn the nonlinear input/output characteristics for the so-called "black box" system. Thus, a detailed specification of performance for the plant to be controlled, is not necessary. A nonlinear function approximation obtained by adjusting the parameters of the neural network model can be used in a variety of information processing applications. The use of learning rules for application to nonlinear identification will be discussed in detail later in this thesis.
1.3 Outline of Thesis

The objective of the thesis is to evaluate the learning capabilities of neural networks using modified Newton's method for applications in associative memory and nonlinear identification problems and to compare the relative performance with the LMS algorithm. The organization of the remaining portion of the thesis is as follows.

In Chapter 2, a brief description of multilayer static networks and dynamical recurrent networks is provided in addition to an overview of their learning rules.

In Chapter 3, the detailed derivation of the learning rule using the modified Newton's method for application in associative memory design is shown. The performance of this learning rule is compared with the performance of the updating scheme utilizing the LMS method. Also provided are the stability and convergence properties for the modified Newton's method. Simulation results for several examples are shown to illustrate the convergence properties of the various learning schemes.

Chapter 4 provides the detailed derivation of learning rules utilizing the modified Newton's algorithm for solving nonlinear identification problems. A comparison of the performance characteristics of this learning scheme with the LMS algorithm is provided through the execution of several detailed examples.

Lastly, Chapter 5 summarizes the contributions of this work and provides some directions for further investigation.
In order to understand the concepts and the methodologies of neural networks more clearly, we must briefly review some of the more popular models used in information processing. It should be noted that this discussion will not cover all aspects of neural network models available in the literature, but will focus only on selected continuous-time neural network models and their training characteristics.

The two primary elements in a neural network are the processing elements and the network's interconnection strength. The processing elements or nodes are simple nonlinear devices that receive and process input signals and generate an output signal. This output signal is sent to the other nodes and possibly back to itself as an input signal. The structure of the neural network, as previously mentioned, is highly dependent on the strength of connections between the nodes. The connection can be made between layers of the network. A layer in the network is formed when many nodes are layered in a parallel arrangement. Overall, the structure of neural networks varies with the nature of processing that occurs within the elements in a static or dynamic sense.
2.1 Static Network vs. Dynamical Network

In recent years, there has been an enormous amount of research performed on the various structures of neural network models. Two classifications of neural networks have received considerable attention. These classes are multilayer static networks and dynamical recurrent networks [7].

The architecture of a multilayer static network consists of processing elements connected together in a cascaded, feedforward manner. Generally, a feedforward node has several input signals and a single output signal as shown in Figure 2-1. Each input signal \( x_j \) is multiplied by a weight \( w_{ij} \) such that the effective input \( s_i \) to the node is a weighted sum of the incoming signals. The output \( y_i \) is formed by passing \( s_i \) through a nonlinear threshold function. Thus, a node in a common static feedforward network is described by

\[
\begin{align*}
    s_i(x) & = \sum_{j=1}^{n} w_{ij} x_j \\
    y_i(x) & = f_i(s_i(x))
\end{align*}
\]

where \( x_j(\cdot) \in \mathbb{R} \rightarrow \mathbb{R}, j=1, 2, \ldots, n, \) are the inputs, \( y_i(\cdot) \in \mathbb{R} \rightarrow \mathbb{R}, i=1, 2, \ldots, n \) are the outputs, \( W \in \mathbb{R}^{nxn} \) \( W=[w_{ij}]_{j=1,2,...,n} \) is the weight matrix and \( f_i(\cdot) \in \mathbb{R} \rightarrow \mathbb{R} \) is the threshold function. There have been several linear and nonlinear threshold functions proposed. For our discussion, this threshold function is a continuous, monotonically nondecreasing, and bounded nonlinear function such as a *sigmoidal function*. For our application, the typical input-output function will vary between -1 and +1 as shown...
Figure 2-1. Static Feedforward Node
in Figure 2-2.

Usually, the static feedforward architecture consists of one input layer, several hidden layers, and a single output layer as illustrated in Figure 2-3. A layer of nodes is created by connecting a number of nodes to the same input vector. Many layers may be cascaded with the outputs of one layer connected to the inputs of the next layer in a feedforward direction forming a multilayer network.

The feedforward static network has proven to be very useful in such applications as speech processing, nonlinear identification, signal processing, pattern recognition and control. In spite of the demonstrated success of multilayered feedforward networks, researchers are leaning towards utilizing processing elements with dynamic structure and feedback connections in the architecture of neural networks, to exploit the additional range of capabilities these networks provide.

The architecture of a recurrent network consists of layered dynamic processing elements. The general mathematical model for the nodes can be described by a differential equation of the form

\[ \dot{x}_i = -a_i x_i + f(x_i, W, b), \quad i = 1, 2, \ldots, n, \]  

(2-3)

where \( x_i \) is the output of node \( i \), \( a_i \) represents the gain of the direct feedback to itself, \( W \) is the weight matrix, \( b \) is an external bias input signal vector and \( f(\cdot) \) is a nonlinear function. This nonlinear function has characteristics similar to the one used for the static feedforward network. A schematic of a layer of dynamical nodes illustrated in Figure 2-4.
Figure 2-2. A Sigmoidal Function
Figure 2-3. Diagram of Multilayer Feedforward Static Network
Figure 2-4. Diagram of a Layer of Dynamical Nodes with Feedback
Researchers have conducted extensive studies on the computational properties of the generic dynamic model (2-3) with recurrent connections. These studies have resulted in a distinct dynamic model due to Hopfield [4,8] which is described by

$$\dot{x} = -Ax + Wf(x) + b$$

(2-4)

where $x \in \mathbb{R}^n \ni x = [x_1, x_2, ..., x_n]^T$, $A \in \mathbb{R}^{n \times n}$ $\ni A = \text{diag}[a_1, a_2, ..., a_n]$, $W \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, and $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is the nonlinear vector function with sigmoidal elements. The dynamical model described by (2-4) is often called the Hopfield model. An architectural representation of the Hopfield model in a block diagram format is illustrated in Figure 2-5. This model can be electronically implemented in hardware simply by using amplifiers and analog active RC circuits [9]. The computation and learning capabilities of this model have been very effective in storing vectors for associative memory applications.

2.2 Classification of Learning Schemes

The most enticing characteristic of neural networks is their learning capabilities. Neural networks are self-adaptive dynamic systems because of the possibility of providing some or all of the nodes with the ability to self-adjust. Thus, they are able to modify the outputs under certain unknown conditions. Primarily, the algorithms used to train a neural network are either supervised or unsupervised. These techniques allow the network to change the characteristics of the output with time, depending on the nature of the input signal received from the environment [6].

In unsupervised learning, the network is trained without a detailed knowledge
Figure 2-5. Architecture of Dynamical Neural Network
of the desired output data. Thus, the network requires minimal knowledge of the error signal to form an internal classification of the given input data. After each learning iteration, error signals are fed back to produce the desired response by the network.

Unlike unsupervised learning techniques, the supervised learning procedures for neural networks are more popularly discussed in the literature. This learning technique uses the specific information of the error generated by the network output and the desired output. After each iteration, the network compares its output with the desired response. It then trains the network by updating the weight until the error reaches an acceptable level. The supervised learning techniques have proven to be quite successful in many applications of neural networks and will be the focus of this thesis.

2.3 Learning Rules for Supervised Training

Research in developing learning algorithms for neural networks dates back to the mid 1960s. The single or multilayered Perceptron models proposed by Rosenblatt [10] can be trained using the Perceptron Convergence Theorem. A Perceptron model is a simple type of feedforward network. The weights are optimized using supervised learning techniques. Another optimization technique called the least mean square (LMS) rule was proposed by Widrow and Hoff [11]. Due to the success of the LMS algorithm in adaptive signal processing applications, this algorithm has also attained wide recognition in the training of neural networks.
Currently, an equally popular supervised learning algorithm called backpropagation [12] has received extensive attention. Although originally these algorithms were developed for use in static feedforward networks, they are also being applied to dynamical recurrent networks. In this section, we shall briefly discuss the backpropagation algorithm and extensions to this approach in an attempt to enhance learning performance.

A great amount of effort is invested in improving the learning properties and performance capabilities of the backpropagation algorithm. This algorithm passes the error information in a backward propagated fashion from the output nodes through the processing layers by adjusting the interconnection weights of the network. The weights are optimized by using a steepest descent gradient search for the minimization of the error function. To gain a better understanding of the learning rules for the optimization of adjustable parameters, it is necessary to present a network architecture with adjustable parameters and input-output characteristics. For illustration purposes, an example of an input-output network with adjustable weight parameter \( w_{iu} \) is

\[
y_k = \sum_{i=1}^{n} w_{ki} x_i \quad (2-5)
\]

where \( y_k \in \mathbb{R} \) is the network output and \( x_i \in \mathbb{R} \) is the network input. In general, the weight matrix \( W \) is optimized by minimizing the error function

\[
E = \frac{1}{2} (y_d - y_k)^2 \quad (2-6)
\]
There exist a variety of optimization techniques for the minimization of the error described by (2-6). In this thesis, a modified Newton's algorithm utilizing second derivatives of the error function is developed for the optimization of the dynamical neural network parameters. Motivations behind the selection of this algorithm for training neural networks resulted from the improved accuracy this algorithm provides in general and also a number of recent publications that have recommended the use of this approach for possible future investigations.

A simple learning algorithm was recently introduced in a PhD dissertation [6] written in 1991 by Sudharsanan for training of the adjustable parameters of the dynamical neural network by utilizing the LMS optimization technique for minimization of the error function. This work compared the backpropagation algorithm and the LMS algorithm for the learning of network parameters and conclusively demonstrated the superb performance results of the LMS algorithm. Thus, it is not necessary to compare the performance of the learning algorithms using modified Newton's algorithm and the backpropagation algorithm. Among the learning algorithms used for applications discussed in this thesis, only the LMS algorithm and the modified Newton's algorithm are considered for the training of dynamical neural networks.

The weight updating that employs the modified Newton's method for the model of (2-5) is

\[
\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \mu \frac{\partial E/\partial \mathbf{w}}{\partial^2 E/\partial \mathbf{w}^2}. \tag{2-7}
\]
Evidently, the rate of convergence of this method is dependent on the nature of the gradient function. For most applications, the second derivative of the error must be determined.

Previously mentioned optimization schemes use an estimate of the gradient for the minimization of the error, whereas in the LMS algorithm, the error itself is used as an estimate for the gradient. The LMS algorithm takes the form

$$w^{\text{new}} = w^{\text{old}} + \mu (y_d - y_x)x$$ \hspace{1cm} (2-8)

The learning rules for neural networks using optimization techniques such as the LMS algorithm and the modified Newton's algorithm will be discussed in more detail in the later chapters of this thesis for application in associative memory and nonlinear identification problems.
CHAPTER 3

LEARNING RULES FOR ASSOCIATIVE MEMORY APPLICATIONS

In the recent past, there has been significant progress in the development of efficient procedures for associative memory designs utilizing dynamical neural networks. Of primary interest in the design of associative memory problems is to store the desired set of memory vectors as stable equilibrium points of the network. The dynamical behavior of a network to function as an associative memory can be addressed through the qualitative properties of its equilibrium points.

The qualitative analysis of the network is concerned primarily with ensuring the stability properties of the equilibrium points and the asymptotic behavior of the solution trajectories. The local stability of each equilibrium point ensures the convergence of the solution trajectory of the network within the region of attraction of the equilibrium point. A large region of attraction for an equilibrium point prevents the convergence of a network trajectory starting in a neighborhood of that equilibrium point from approaching other unwanted (spurious) stable equilibrium points.

A proper selection of the network parameters is realized through a synthesis procedure such that storage of a specified memory vector as a stable equilibrium point is ensured in a specified quadrant of the state space. An efficient synthesis
procedure for handling such a problem, involves the selection of the sigmoidal nonlinear functions of the dynamical network by tailoring their slopes appropriately.

The qualitative properties of the equilibrium points of the network and the convergence properties of learning algorithms form the basis for the network to function as an associative memory. The network parameters are adjusted by developing a learning rule for the minimization of the deviation between the desired memory vectors to be stored and the stable equilibrium points of the network.

In this chapter, we shall review some of the basic qualitative properties of dynamical neural networks necessary for a proper selection of the adjustable parameters of the network and the nonlinear sigmoidal function to serve as an associative memory. This will be followed by the derivation of the learning algorithm using modified Newton's method to optimize the adjustable parameters of the neural network model. Convergence criteria for this learning scheme will then be derived. Finally, the performance of this learning algorithm will be compared with the performance of the LMS algorithm employed in this problem in [17].

3.1 Basic Properties of Dynamical Networks

The main emphasis in this chapter is in utilizing a continuous-time dynamical neural network to function as an associative memory. The mathematical model of such a network with \( n \) processing elements can be described by

\[
\dot{x} = -Ax + Wg(x) + b
\]

(3-1)

where \( x \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}, A = \text{diag}[a_1, a_2, \ldots, a_n], W \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n, \) and \( g: \mathbb{R}^n \rightarrow \mathbb{R}^n \)
is a vector-valued threshold function with sigmoidal characteristics. The selection of
the nonlinear functions $g_i, i = 1, 2, ..., n$ for this network is often guided by the
following conditions:

(i) $x g_i(x_i) > 0 \ \forall \ x_i \in \mathbb{R}$

(ii) $\lim_{x_i \to -\infty} g_i(x_i) = \text{Sgn}(x_i)$

(iii) $g_i(x_i)/x_i \geq g_i(v_i)/v_i \ \forall \ |x_i| \leq |v_i|$

(iv) $g_i'(x_i) = \frac{dg_i(x_i)}{dx_i} > 0 \ \forall \ x_i \in \mathbb{R} .$

Condition (i) states that the nonlinear function must be in the first and third
quadrants while condition (ii) states that the function is a saturating function
bounded by +1 and -1. Conditions (iii) and (iv) imply that the function is a
monotonically nondecreasing one. The above sigmoidal characteristics are satisfied
by nonlinear functions such as $g_i(x_i) = \tan^{-1}(\lambda x_i)$ and $g_i(x_i) = \tanh(\lambda x_i)$ [8,13,16,17]
where $\lambda$ is the gain of the sigmoidal function. A typical graph of such sigmoidal
functions has previously been shown in Figure 2-2. For our application, it is desired
to have a network with equilibrium points stored near the saturation region of $g(\cdot)$.
Furthermore, a nonlinear sigmoidal function with a large gain is often selected in
order to quickly attain saturation. Such a tailoring of the function for dynamical
neural networks can result in simple and more efficient learning algorithms [6].

The dynamical neural network model described by $n$ scalar differential equations is
equivalent to the model specified in (3-1). These scalar equations take the form

\[
\frac{dx_i}{dr} = -ax_i + \sum_{j=1}^{n} w_{ij} g_j(x_j) + b_i, \quad i = 1, 2, ..., n
\]

(3-2)

where \( x_i \in \mathbb{R} \) is the state of the \( i \)-th processing element, \( W = [w_{ij}] \in \mathbb{R}^{n \times n} \) is the weight matrix, \( b_i \in \mathbb{R} \) is the \( i \)-th element of the bias input vector, and \( n \) is the number of processing elements.

The efficiency of the dynamical neural network model described by (3-1) for information processing applications can be characterized by its equilibrium points. A qualitative analysis of the equilibrium points of the network is an essential task for realizing appropriate performance and the required computations for desired applications. The equilibrium points of (3-1) in the state space are given by \( x^* \in \mathbb{R}^n \)

\[ x^* = [x_1^*, x_2^*, ..., x_n^*]^T \]

where \( x_i^* \) is defined by

\[
\frac{dx_i^*}{dr} = 0, \quad i = 1, 2, ..., n
\]

This will result in the set of \( n \) nonlinear algebraic equations of the form

\[
a_x x_i^* = \sum_{j=1}^{n} w_{ij} g_j(x_j^*) + b_i
\]

3.1.1 Characterization of Equilibrium Points

A qualitative analysis of the network concerns evaluation of characteristics of the equilibrium points and their stability properties. These properties of the network equilibrium points are of fundamental importance in the proper selection of the
values of the adjustable parameters of the network. Also, the properties of the equilibrium points of the network can be used as a basis in selecting an appropriate network structure for the desired applications. For instance, in optimization applications, one is interested in designing a network with a unique equilibrium point which is a globally stable equilibrium point. The globally stable attractor prevents the convergence of network trajectories to local minima or other unwanted states. On the other hand, for associative memory applications, one is interested in constructing a network architecture with multiple equilibrium points which correspond to the various memory vectors to be stored.

Consequently, the adjustable parameters of the network and the nonlinear functions should satisfy appropriate conditions to guarantee the local stability of each equilibrium point. The local stability of an equilibrium point can be ensured by the rapid convergence of network trajectories from an initial point within the region of attraction of the equilibrium point to this equilibrium state.

For a reliable retrieval of the corresponding stored memories in associative memory designs, one is interested in selecting the network parameters such that the equilibrium points are confined to specific regions of the state space. Such a design, if possible, enables the network equilibrium points to have large basins of attraction, which in turn detracts the convergence of network trajectories to unwanted stable states. When the network is designed to function as an associative memory, it is not guaranteed that the network will have only the stored vectors as the stable equilibrium points, but many other unwanted (spurious) stable vectors may also be
introduced. This will result in smaller regions of attraction for the stored vectors which may result in only a partial recall of memory. To aid in the discussion, the conditions for confinement of an equilibrium point obtained in [6,26] will be outlined with the following definition.

**Definition 1:** A quadrant containing a set of points denoted by $\mathbf{I}(\xi, z) \in \mathbb{R}^n$ is defined as $\mathbf{I}(\xi, z) = \{ \xi : \xi z_i > 0; \forall i = 1, 2, ..., n \}$ where $z = [z_1, z_2, ..., z_n]^T \in \mathbb{R}^n \ni z_i \neq 0 \forall i = 1, 2, ..., n$ is a fixed point and $\xi = [\xi_1, \xi_2, ..., \xi_n]^T$.

For the sake of clarity, a specific equilibrium point $x^*$ described by (3-1) shall be considered in the quadrant $\mathbf{I}(\xi, x^*)$. A sufficient condition for the confinement of the equilibrium point $x^*$ in the distinct quadrant $\mathbf{I}(\xi, x^*)$ of the state space is given by [6,26] all the principal minors of matrix $A-WF$ being positive, where $F \in \mathbb{R}^{n \times n}$ is defined as $F = \text{diag}(x^*_1, x^*_2, ..., x^*_n)$ where $x^*_i, i = 1, 2, ..., n$, are the elements of the equilibrium point $x^*$.

The condition for stability of an equilibrium point $x^*$ of (3-1) can be ensured by designing a network such that the network trajectories starting at an initial point in the vicinity of this equilibrium point, converges rapidly to this steady-state. The local stability of an equilibrium point $x^*$ is guaranteed when either of the following hypotheses [6,26] are satisfied.

(i) $\text{Spec}(WG-I) \subset \text{LHP}$ where $I \in \mathbb{R}^{n \times n}$ (Identity matrix) and $G \in \mathbb{R}^{n \times n}$ where $G = \text{diag}[g_1', g_2', ..., g_n']$ such that $g_i' = \frac{\partial g_i(x_i)}{\partial x_i} |_{x_i = x^*_i}$, for $i = 1, 2, ..., n$.

(ii) $M = \frac{\partial}{\partial z} [(I-WG)^T + (I-WG)]$ is a positive definite matrix.
A detailed proof of this result is presented in [6,26]. Also, it should be mentioned that several of the earlier studies [8,13,14,15] have extensively analyzed the stability properties of neural networks described by (3-1). Next, we shall discuss the effects of the gain of the nonlinear sigmoidal function on the stability conditions for the network equilibria.

3.1.2 Selection of Gain of the Nonlinear Function

It must be emphasized that in a majority of neural network applications, either in associative memory designs or in nonlinear input-output mapping problems, it is desirable to design the network with equilibrium points assigned in the saturation region of the nonlinear sigmoidal functions. The selection of very large gains for the sigmoidal functions is of importance in order to ensure the locations of equilibria in the saturation region. However, one should be careful in using excessively large gains for the nonlinear functions, because it may then become difficult to ensure the uniqueness of these equilibrium points in their respective quadrants. This is evident by showing that $F \in \mathbb{R}^{m \times n} \ni F = \text{diag} \left[ \frac{g_1(x_1^*)}{x_1^*}, \frac{g_2(x_2^*)}{x_2^*}, ..., \frac{g_n(x_n^*)}{x_n^*} \right]$ and the selection of an extremely large gain for the sigmoidal function implies a large norm for the $F$ matrix. Therefore, it may become difficult to satisfy the condition for all principal minors of the matrix $A-WF$ to be positive. The selection of a very small value for the gain of the sigmoidal function, on the other hand, may result in unstable conditions in which stable equilibrium points of the network do not correspond to the desired vector to be stored. Another interest for locating the equilibrium points in the
saturation region arises from having a desirable stability behavior of the network trajectory by ensuring the condition $\text{Spec}(WG-I) \subset \text{LHP}$. This stability condition can be guaranteed by specifying the elements of the diagonal matrix $G$ to be arbitrarily small.

In the determination of the gain of the nonlinear sigmoidal function for network design, one should initially start with a selection of small gain sigmoidal function and increase the value of the gain until stability conditions are satisfied. Furthermore, it is observed that when the gain of the sigmoidal function is increased, saturation is attained more quickly which results in placing the network equilibria in the saturation region. The effects of varying the gain of the nonlinear sigmoidal function on the location of the equilibria of the network will be demonstrated later via results of numerical simulations of specific examples.

Although outside the scope of this thesis, it should be noted that there are several analytical design procedures to determine the adjustable parameters for the neural network to function as an efficient associative memory. A design procedure proposed by Li et al [13] requires the interconnection matrix $W$ to be symmetric. This requirement imposes a restriction on the capability of the network to store specified sets of memory vectors. The synthesis procedure proposed by Sudharsanan [6, 26] does not impose any restrictions on the symmetry of the weight matrix. Through an efficient synthesis procedure, one can select the network parameters such that the vectors to be stored are the stable equilibrium points of the network with large regions of attraction and are the unique equilibria in their respective quadrants.
of the state space.

In this thesis, the adjustment of the network parameters and appropriate selection of the nonlinear sigmoidal functions is demonstrated by training the network such that the network trajectories converge to the stable states of the network. The implementation of the learning algorithms for adjustment of the network parameters for associative memory problems is the topic of the following sections.

3.2 Learning Rule Using Modified Newton’s Method

One of the most interesting computational features of neural networks in processing information is its learning capability. Learning algorithms allow the network to adaptively adjust its parameters by comparing the network response with the desired response. A popular learning algorithm called backpropagation [12] has received an extensive amount of attention in the past. This method uses a steepest descent gradient search to optimize the adjustable parameters of the network by minimizing the error between the actual network response and the desired response. Recently, it has been found that the performance of standard backpropagation has some drawbacks in terms of its convergence properties. The rate of convergence of this algorithm is extremely slow for networks of large sizes and the algorithm often converges to a local minimum [18]. Presently, one of the major interests of research is in the area of developing optimization techniques which have better performance than the backpropagation algorithm [27].
The development of learning algorithms for the adjustable parameters of the

dynamical neural network model of (3-1) can be based on various optimization
methods for the minimization of the error function. In the literature on adaptive
signal processing, Widrow et al [19] have made a general observation that
minimization performed by Newton’s method seems to converge more rapidly than
steepest descent algorithms. Bhat et al [20] and Sudharsanan [6] have also suggested
the use of Newton’s method to perform minimization for future research because it
may lead to better convergence properties than the backpropagation algorithm.
These observations were the incentive for selecting modified Newton’s method to
optimize the neural network parameters by minimization of the error function in this
thesis. The learning algorithms presently developed are customized for networks
with equilibrium points in the saturation region of the sigmoidal function $g(\cdot)$.

For associative memory applications, a learning rule is developed for updating
the elements of the weight matrix $W$ and the bias vector $b$ while minimizing the error
between the stable equilibrium points of the network and the desired memory vectors
to be stored. A mathematical derivation of the learning rule using the modified
Newton’s method for optimization of these network parameters is presented here.

For a simplified presentation of the learning procedure, we shall first consider
a single desired vector $y = [y_1, y_2, \ldots, y_n]^T$ to be stored as a stable equilibrium point
of the network model described by (3-1). A stable equilibrium point of this network
is $x^* = [x_1^*, x_2^*, ..., x_n^*]$ where $x_i^*$ are defined as

$$x_i^* = \sum_{j=1}^{n} w_{ij} g_j(x_j^*) + b_i, \quad i=1,2,...,n.$$  \hspace{1cm} (3-3)

The learning algorithm is tailored to minimize the error function which can be defined as

$$E = \frac{1}{2} \sum_{i=1}^{n} (y_i - x_i^*)^2.$$  \hspace{1cm} (3-4)

An iterative supervised learning algorithm can be developed by employing the modified Newton's method for optimization of the network parameters for minimization of the error function $E$. This approach involves computation of the first and second derivatives of $E$ with respect to the elements of the weight matrix $W$. Therefore, the updating procedure for the weight matrix $W$ at any iteration can be stated as

$$w_{lk}^{new} = w_{lk}^{old} - \mu \frac{\partial E/\partial w_{lk}}{\partial^2 E/\partial w_{lk}^2}, \quad l,k=1,2,...,n$$  \hspace{1cm} (3-5)

where $\mu$ is the adaptive updating parameter.

For computing $\partial E/\partial w_{lk}$ from (3-4),

$$\frac{\partial E}{\partial w_{lk}} = -\sum_{i=1}^{n} (y_i - x_i^*) \frac{\partial x_i^*}{\partial w_{lk}}$$  \hspace{1cm} (3-6)

where the weight matrix $W$ is a nonlinear function of $x$. 
Using (3-3), one can evaluate $\frac{\partial x_i^*}{\partial w_{ik}}$ as

$$\frac{\partial x_i^*}{\partial w_{ik}} = \sum_{j=1}^{N} \left[ w_j g_j'(x_j^*) \frac{\partial x_j^*}{\partial w_{ik}} + g_j(x_j^*) \delta_{jk} \right]$$

where $\delta_{ij}$ is the Kronecker delta. Assuming that $\frac{\partial x_i^*}{\partial w_{ik}} = 0$ when $i \neq k$, results in

$$\frac{\partial x_i^*}{\partial w_{ik}} = \frac{g_k(x_k^*)}{1-w_kg_k'(x_k^*)}$$ \hspace{1cm} (3-7)

Substituting in (3-6) gives

$$\frac{\partial E}{\partial w_{ik}} = \frac{-(y_i-x_i^*)g_k(x_k^*)}{1-w_kg_k'(x_k^*)}$$ \hspace{1cm} (3-8)

Similarly, differentiating (3-6)

$$\frac{\partial^2 E}{\partial w_{ik}^2} = \left( \frac{\partial x_i^*}{\partial w_{ik}} \right)^2 - (y_i-x_i^*) \left( \frac{\partial^2 x_i^*}{\partial w_{ik}^2} \right)$$ \hspace{1cm} (3-9)
and from (3-7)

\[
\frac{\partial^2 x_i}{\partial w_k^2} = \frac{(1-w_k g_i(x_i^*))^2 g_k'(x_k^*) g_k(x_k^*)}{D} + \frac{[g_k(x_k^*)]^2 w_i g_i''(x_i^*)(1-w_k g_k'(x_k^*))}{D} \]

\[
+ \frac{g_k(x_k^*) g_i(x_i^*) \delta_{ik}(1-w_k g_k'(x_k^*))}{D} \delta(x_i^*) \delta_j \delta_i \quad (3-10)
\]

where

\[
D = (1-w_i g_i'(x_i^*))^3(1-w_k g_k'(x_k^*)) \quad (3-11)
\]

Substituting (3-7) and (3-10) into (3-9) gives

\[
\frac{\partial^2 E}{\partial w_k^2} = \frac{g_k(x_k^*)^2(1-w_k g_k'(x_k^*))^2 g_k'(x_k^*)}{D} \]

\[
- \frac{g_k(x_k^*) (y_i-x_i^*)(1-w_k g_k'(x_k^*))^2 g_k'(x_k^*)}{D} \]

\[
- \frac{w_i g_i''(x_i^*) g_k(x_k^*) (y_i-x_i^*)(1-w_k g_k'(x_k^*))}{D} \]

\[
- \frac{g_k(x_k^*) g_i(x_i^*) \delta_{ik}(1-w_k g_k'(x_k^*))}{D} \delta(x_i^*) \delta_j \delta_i \quad (3-12)
\]

where \( D \) is defined in (3-11).
Therefore, inserting (3-8) and (3-12) into the updating rule of (3-5) results in

\[ w_{ik}^{\text{new}} = w_{ik}^{\text{old}} + \mu \frac{N}{D_1 - D_2 - D_3 - D_4} \]

where

\[ N = (y_i - x_i^*) (1 - w_{ik} g_i'(x_i^*))^2 (1 - w_{ik} g_k'(x_k^*)) \]

\[ D_1 = g_k(x_k^*) (1 - w_{ik} g_i'(x_i^*)) (1 - w_{ik} g_k'(x_k^*)) \]

\[ D_2 = (y_i - x_i^*) (1 - w_{ik} g_i'(x_i^*))^2 g_k(x_k^*) \]

\[ D_3 = (y_i - x_i^*) w_{ik} g_i''(x_i^*) g_k(x_k^*) (1 - w_{ik} g_k'(x_k^*)) \]

\[ D_4 = (y_i - x_i^*) g_i'(x_i^*) \delta_{ik} (1 - w_{ik} g_k'(x_k^*)) (1 - w_{ik} g_i'(x_i^*)) \]

Similarly, the updating scheme for the input bias vector can be obtained as

\[ b_i^{\text{new}} = b_i^{\text{old}} + \mu \frac{(y_i - x_i^*) (1 - w_{ik} g_i'(x_i^*))^2}{(1 - w_{ik} g_i'(x_i^*)) - (y_i - x_i^*) w_{ik} g_i''(x_i^*)} \]

Again, these derivations of learning rules only consider a single desired vector to be stored as a stable equilibrium point. For multiple vectors \( \{y^1, y^2, \ldots, y^m\} \) to be stored as the stable equilibria \( x^{*1}, x^{*2}, \ldots, x^{*m} \) of the network model of (3-1), requires some modification of updating rules.

The learning algorithm for the minimization of error between the memory vectors to be stored and the stable equilibrium points of the network can be
established with the error function

\[ E = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{m} (y_j^i - x_j^i)^2 \]  

(3-13)

where \( m \) denotes the number of vectors to be stored, \( n \) is the number of processing elements in the network, \( x_j^i \) is the \( i \)-th element of the \( j \)-th vector \( \mathbf{x}^j \) and \( y_j^i \) is the corresponding element of the desired vector \( \mathbf{y}^i \) to be stored. The learning algorithm for the storage of multiple memory vectors can then be obtained as

\[ w_{ik}^{new} = w_{ik}^{old} + \mu \sum_{j=1}^{m} \frac{N'}{D_1' - D_2' - D_3' - D_4'} \]

where

\[ N' = (y_j^i - x_j^i)(1 - w_{ik}g_k'(x_j^i))(1 - w_{ik}g_k'(x_k^j)) \]

\[ D_1' = g_k(x_k^j)(1 - w_{ik}g_k'(x_j^i))(1 - w_{ik}g_k'(x_k^j)) \]

\[ D_2' = (y_j^i - x_j^i)(1 - w_{ik}g_k'(x_j^i))^2g_k(x_k^j) \]

\[ D_3' = (y_j^i - x_j^i)w_{ik}g_k''(x_k^j)(1 - w_{ik}g_k'(x_k^j)) \]

\[ D_4' = (y_j^i - x_j^i)g_k'(x_k^j)\delta_{ik}(1 - w_{ik}g_k'(x_k^j))(1 - w_{ik}g_k'(x_k^j)) \]

and

\[ b_{i}^{new} = b_{i}^{old} + \mu \sum_{j=1}^{m} \frac{(y_j^i - x_j^i)(1 - w_{ik}g_k'(x_j^i))^2}{(1 - w_{ik}g_k'(x_k^j)) - (y_j^i - x_j^i)w_{ik}g_k''(x_k^j)} \]

The simulation results of numerical examples using this learning algorithm will be illustrated later in this chapter. In the next section, we shall evaluate the convergence properties of the modified Newton's algorithm.
3.3 Convergence Criteria for Modified Newton's Method

An item of particular interest concerning the performance of the learning algorithm is the evaluation of its ability to converge and the speed of convergence. The convergence of the updating algorithm is achieved when the objective function reaches some prespecified minimum value. The minimum of the error function (3-4) occurs when the gradient approaches zero as the iteration step approaches infinity implying an asymptotic convergence property. An appropriate selection of the convergence rate parameter \( \mu \) is required for a rapid convergence of the learning algorithm. It will be shown that a bound on \( \mu \) can be established for the modified Newton's method which is directly related to the number of processing elements in the network.

A detailed analytical derivation of the bound on \( \mu \) for the LMS algorithm is presented in [17]. The upper bound on \( \mu \) for the modified Newton's algorithm can similarly be derived by using the Lipschitz condition on the error function

\[
|\nabla(E(w_1)) - \nabla(E(w_2))| \leq K|w_1 - w_2|
\]

where \( ||\cdot|| \) denotes the \( L_2 \) norm, \( K > 0 \) is a Lipschitz constant and \( w_1, w_2 \in \mathbb{R}^n \). The estimation of the Lipschitz constant can be done by using the Mean Value Theorem [17,21] for a point \( w \in \mathbb{R}^n \) \( \exists \tilde{w} = [w_{11}, w_{12}, \ldots, w_{1n}, w_{21}, \ldots, w_{2n}, \ldots, w_{nl}, \ldots, w_{nm}]^T \). The theorem states that

\[
\nabla(E(w_1)) - \nabla(E(w_2)) = (\nabla^2(E(\tilde{w}))(w_1 - w_2)
\]
Then

\[ |\nabla (E(w_1)) - \nabla (E(w_2))| \leq |\nabla^2 (E(w))| ||w_1 - w_2|| \]

where \( |\nabla^2 (E(w))| = K \) is estimated from the gradient,

\[ \frac{\partial E}{\partial w_k} = -\frac{(y_i - x_i^*) g_k(x_k^*)}{1 - w_l g_l'(x_l^*)} . \]

The second derivative with respect to \( w_{rs} \) is obtained by

\[
\frac{\partial E}{\partial w_{rs}} \left( \frac{\partial E}{\partial w_k} \right) = \frac{(1 - w_l g_l'(x_l^*)) \left[ g_k(x_k^*) \frac{\partial x_k^*}{\partial w_{rs}} + g_k(x_k^*) \frac{\partial w_k}{\partial w_{rs}} \right]}{(1 - w_l g_l'(x_l^*))^2} + \frac{(y_i - x_i^*) g_k(x_k^*) \frac{\partial w_{rs}}{\partial w_k}}{(1 - w_l g_l'(x_l^*))^2}
\]

where

\[ \frac{\partial x_k^*}{\partial w_{rs}} = \frac{g_k(x_k^*)}{1 - w_l g_l'(x_l^*)} . \]

Therefore,

\[ \frac{\partial E}{\partial w_{rs}} \left( \frac{\partial E}{\partial w_k} \right) = \frac{N_1 - N_2 - N_3 - N_4}{D} \]
where

\[ N_1 = \delta_n g_k(x_k^*) g_s(x_s^*)(1 - w_k g_k'(x_k^*)) \]
\[ N_2 = (y_i - x_i^*) g_k(x_k^*) g_s(x_s^*)(1 - w_k g_k'(x_k^*)) \]
\[ N_3 = (y_i - x_i^*) g_k(x_k^*) w_k g_k''(x_k^*) g_s(x_s^*)(1 - w_k g_k'(x_k^*)) \]
\[ N_4 = \delta_n \delta_s (y_i - x_i^*) g_k(x_k^*) g'(x_i^*)(1 - w_k g_k'(x_k^*)) (1 - w_k g_k'(x_k^*)) \]
\[ D = (1 - w_k g_k'(x_k^*))^3 (1 - w_k g_k'(x_k^*)) \].

Since the learning rule is tailored for the equilibrium points in the saturation region of the sigmoidal function \( g(\cdot) \), the derivatives \( g'(\cdot) \) and \( g''(\cdot) \) can be approximated to be zero. Thus, \( \frac{\partial^2 E}{\partial w_{rs} \partial w_{lk}} \) can be approximated by

\[
\frac{\partial}{\partial w_{rs}} \left( \frac{\partial E}{\partial w_{lk}} \right) = \delta_n g_k(x_k^*) g_s(x_s^*)
\]

which is identical to the approximation obtained in [15] for the LMS algorithm.

Therefore,

\[ \nabla^2(E(w)) = \text{diag}[H, H, ..., H] \]

where

\[
H = \begin{bmatrix}
    g_1^2(x_1^*) & g_1(x_1^*)g_2(x_2^*) & \cdots & g_1(x_1^*)g_n(x_n^*) \\
    g_2(x_2^*)g_1(x_1^*) & g_2^2(x_2^*) & \cdots & g_2(x_2^*)g_n(x_n^*) \\
    \vdots & \vdots & \ddots & \vdots \\
    g_n(x_n^*)g_1(x_1^*) & g_n(x_n^*)g_2(x_2^*) & \cdots & g_n^2(x_n^*)
\end{bmatrix}
\]

Since the maximum value of \( |g(\cdot)| \) is 1, the Lipschitz constant \( K \) can be estimated as \( \|\nabla^2(E(w))\| \leq \eta. \) Under the Lipschitz condition, the updating rule for
the weight matrix (3-5) is convergent for $0 < \mu < 2c/K$ where $c = \frac{1}{2} \lambda_\text{min}(P) > 0$. $\lambda_\text{min}(P)$ is defined as the minimum eigenvalue of matrix $P \in \mathbb{R}^{n \times n}$, where $P = [(I-WG)^{-1} + (I-WG)^{-1}]$. The value for constant $c$ can be approximated by assuming that the equilibria are almost in the saturation region where the slope is negligible ($G=0$). This result is an estimation for $c=1$ and hence the upper bound on $\mu$ can be written as $\mu < 2/n$. For learning multiple memories $m$, the number of memory vectors should be used to estimate the upper bound as $\mu < 2/mn$. It can be seen that the bounds on $\mu$ for the learning algorithms is dependent on the size of the network to ensure satisfactory convergence. A detailed analytical proof of the convergence of the LMS algorithm is presented in [17].

3.4 Comparison of Learning Rules

The performance of the learning algorithm using the modified Newton's method derived above will be compared to that of the LMS [17] method by evaluating the convergence properties by considering several numerical examples. Sudharsananan [17] has noted the superb performance of the LMS algorithm when compared to the standard backpropagation [12]. Therefore, it is not necessary to compare the performance of the modified Newton's method with that of the backpropagation algorithm as part of this thesis and only a comparison with the LMS algorithm will be given. For facilitating this, the updating rule for the LMS algorithm will be briefly outlined, followed by detailed examples to illustrate the performance of each method through simulation.
3.4.1 Training Rule using LMS Method

The adjustable parameters of the dynamical network given by (3-1) can be updated by using the LMS approach originally proposed by Widrow and Hoff [11]. The detailed derivation of the LMS algorithm for training the dynamical neural network model with recurrent connections to function as associative memory has been given by Sudharsanan [6]. The development of this algorithm was based on the gradient descent approach. The LMS algorithm was noted to have superior performance when compared to the standard backpropagation algorithm [12] in terms of faster convergence, ease in implementation, and a better accuracy in computation (when solving identical numerical examples). This algorithm was customized for a network with equilibrium points in the saturation region of the sigmoidal function $g(\cdot)$. An appropriate selection of the sigmoidal function with a large gain was considered in order to store the equilibrium points in the saturation region of $g(\cdot)$.

The LMS learning rule developed by Sudharsanan [17] for adjusting the network parameters $W$ and $b$ for problems of storing multiple memories by a minimization of the error function (3-13) is

$$w_{ik}^{new} = w_{ik}^{old} + \mu \sum_{j=1}^{m} (y_i^j - x_i^j) g_k(x_i^j)$$

and

$$b_{i}^{new} = b_{i}^{old} + \mu \sum_{j=1}^{m} (y_i^j - x_i^j)$$
where $\mu$ is the convergence rate parameter.

Note that with this updating algorithm, the adjustable parameters of the network are optimized without the need for computing any time derivative of the sigmoidal function $g(\cdot)$ and hence, the implementation does not require an estimation of the gradient for minimization of the error function. This simplifies the algorithm considerably over the modified Newton's method.

### 3.4.2 Discussion with Examples

In this section, we shall present the procedure for implementation of the above learning rules for associative memory applications. The performance of the learning algorithms will be illustrated through results obtained by simulation of specific numerical examples.

The implementation of the above learning rules for the network in (3-1) to function as an associative memory is shown in the flow diagram of Figure 3-1. As stated earlier, the primary objective of this learning procedure is to update the network parameters such that the error is minimized between the stable equilibrium points of the network and the desired memory vectors to be stored. After the error has been minimized, the resultant values of $W$ and $b$ yield network equilibrium points $x^*_i, \ i = 1, 2, \ldots, m$, which closely approximate the vectors $y^i, \ i = 1, 2, \ldots, m$, to be stored.

The design procedure outlined earlier is followed to illustrate the performance of the modified Newton's method and the LMS method by constructing several
Figure 3-1. Flow Diagram of Associative Memory Learning Algorithm
numerical examples. The results of the examples will illustrate the performance of these learning algorithms.

**Example 3.1:** The vectors \( \{v_1, v_2, v_3, v_4, v_5\} \) are to be stored as the stable equilibrium points of the network described by (3-1) where \( v_1 = [2.09, 2.01, -4.19, -5.09, -3.1]^T \), \( v_2 = [2.01, 2.79, 4.59, 2.89, 3.39]^T \), \( v_3 = [-2.29, -3.01, 2.79, 1.9, 4.76]^T \), \( v_4 = [3.99, 3.87, -2.38, 2.69, 4.01]^T \), and \( v_5 = [-3.09, -4.19, -1.91, 4.19, -2.10]^T \). The nonlinear sigmoidal function
\[
g(x_i^*) = (2/\pi)\tan^{-1}(2\pi x_i^*)
\]
is selected. The initial values for the adjustable parameters of this five neuron network are selected as \( W = 5[I] \) and \( b = [0.5, 0.5, 0.5, 0.5, 0.5]^T \). For an appropriate selection of \( \mu \), the bound \( \mu < 2/mn = 0.08 \) can be used. Therefore, \( \mu = 0.07 \) is selected for this example.

The convergence performance resulting from the use of the modified Newton's method and the LMS method are shown graphically in Figure 3-2. The converged values of the parameters in \( W \) and \( b \) are

\[
W = \begin{bmatrix}
1.6692 & 1.3489 & -0.9533 & -0.3598 & 1.3168 \\
1.1847 & 1.5496 & -0.2816 & -0.1889 & 0.8612 \\
0.4261 & 0.4355 & 3.5822 & 1.9570 & -1.0688 \\
0.2164 & 0.2266 & 0.0826 & 5.1762 & -1.1769 \\
-0.3437 & -0.3444 & -0.3163 & -0.1912 & 3.8631
\end{bmatrix}
\]

\[
b = [-0.1441, -0.4926, -0.6159, -1.4863, 0.7947]^T
\]

Figure 3-2 clearly illustrates that each algorithm converges quite rapidly. In fact, the rate of convergence is effectively identical for both algorithms. Therefore, one algorithm has not proven to be more attractive than the other for this example.
Figure 3-2. Learning Curve for Example 3.1 with Sigmoidal Gain $\lambda = 2\pi$
The location of the equilibrium points can be changed by varying the gain of the nonlinear sigmoidal function. By selecting a larger gain for the nonlinear sigmoidal function \( g_t(x_t^*) = \frac{2}{\pi} \tan^{-1}(30\pi x_t^*) \), the convergence of the learning algorithm is quickly attained as shown in the learning curve of Figure 3-3. A small gain value for the sigmoidal function \( g_t(x_t^*) = \frac{2}{\pi} \tan^{-1}(0.01\pi x_t^*) \) results in a divergence of the learning algorithm as illustrated in Figure 3-4. Therefore, a proper selection of the nonlinear sigmoidal function can result in acceptable stability behavior by the network trajectories. It can be noted, however, that the implementation of the learning rule using the modified Newton's method is more complex due to the size of the updating rules.

Example 3.2: The vectors \( \{v_1, v_2, v_3, v_4\} \) to be stored as the stable equilibria of the three neuron network are \( v_1 = [3.0, 3.0, 3.0]^T \), \( v_2 = [-5.0, 4.0, 2.0]^T \), \( v_3 = [3.0, -3.8, 4.0]^T \), and \( v_4 = [-3.5, -3.0, -5.5]^T \). The nonlinear sigmoidal function \( g_t(x_t^*) = \frac{2}{\pi} \tan^{-1}(5\pi x_t^*) \) and the adaptive updating gain parameter \( \mu = 0.15 \) were selected. The initial values for the adjustable parameters of this network were selected as \( W = 4[I] \) and \( b = [0.1, 0.1, 0.1]^T \).

The converged values of the parameters \( W \) and \( b \) are

\[
W = \begin{bmatrix}
4.0420 & -0.0039 & -0.7459 \\
-0.4960 & 3.4454 & 0.0929 \\
0.2526 & -0.4915 & 4.5346
\end{bmatrix}
\]

\[
b = [-0.2510, 0.0034, -1.2290]^T
\]
Figure 3-3. Learning Curve for Example 3.1 with Sigmoidal Gain $\lambda=30\pi$
Figure 3-4. Learning Curve for Example 3.1 with Sigmoidal Gain $\lambda = 0.01\pi$
The learning curves demonstrating the performance of the modified Newton's method and the LMS learning algorithm are depicted in figure 3-5. It is easily seen that the error approaches fairly small values within a few iterations.

**Example 3.3:** The vectors \( \{v_1, v_2, v_3, v_4\} \) to be stored as the stable equilibrium points of the network are \( v_1=[3.1, 3.0, -3.2, -3.1, -3.1]^T \), \( v_2=[2.0, 2.0, 2.6, 2.9, 2.4]^T \), \( v_3=[-2.0, -2.0, 2.0, 1.9, 2.05]^T \), and \( v_4=[-0.9, -1.0, -3.75, -4.0, -3.4]^T \). The dynamical network consists of five neurons with the selected nonlinear sigmoidal function given by \( g(x_i^*) = \frac{2}{\pi} \tan^{-1}(5\pi x_i^*) \). The initial values for the adjustable parameters of this network were selected as \( W=0.5[I] \) and \( b=[0.1, 0.1, 0.1, 0.1, 0.1]^T \) and the value of \( \mu = 0.09 \) was selected by using the upper bound \( \mu < 2/mn \).

The convergence of the modified Newton's method and the LMS method are shown graphically in Figure 3-6. The converged values of the parameters \( W \) and \( b \) are

\[
W = \begin{bmatrix}
1.2867 & 0.7640 & -0.0033 & -0.2705 & -0.2689 \\
0.7655 & 1.2846 & 0.0149 & -0.2547 & 0.2528 \\
0.1523 & 0.1372 & 1.4617 & 0.7444 & 0.7273 \\
0.2420 & 0.2389 & 0.6117 & 1.4556 & 0.9563 \\
0.0822 & 0.0797 & 0.5129 & 0.8754 & 1.3923 \\
\end{bmatrix}
\]

\[
b = [-0.5329, 0.4837, -0.5808, -0.5695, -0.5055]^T
\]

Again, Figure 3-6 shows that the error has decreased to substantially low levels within a few iterations. The convergence performance for the modified Newton's method when varying the value of \( \mu \) for Example 3.3 is depicted in Figures 3-7 and
Figure 3-5. Learning Curve for Example 3.2
Figure 3.6. Learning Curve for Example 3.3 with ω = 0.009
Figure 3-7. Learning Curve for Example 3.3 with $\mu = 0.05$
3-8 for $\mu = 0.05$ and 0.07, respectively. A smaller value of $\mu$ results in a slower convergence of the error function whereas a larger value of $\mu$ results in a faster convergence. The learning curves shown in Figure 3-9 for $\mu = 0.18$ indicates a divergence of both learning algorithms. It should be noted that this value of $\mu$ exceeds the upper bound of 0.1 as determined by the bounding relationship and hence the convergence of the learning scheme cannot be guaranteed in this case. Evidently, a small value of $\mu$ needs to be selected for a network with a large number of processing elements.

From the simulation results, it can be seen that the bound on $\mu$ is dependent on the size of the network to guarantee a satisfactory convergence. However, the graphs do indicate that the LMS algorithm results in a slightly faster convergence than that resulting from the modified Newton’s method. This is due to the simplicity of the LMS algorithm relative to the modified Newton’s method.

3.5 Conclusions

A dynamical neural network with feedback and recurrent connections was applied to the associative memory problem. The qualitative properties of network equilibria were briefly discussed in the course of designing a network to function as an associative memory. A learning algorithm was derived by considering the modified Newton’s method to optimize the network parameters. The convergence
Figure 3-8. Learning Curve for Example 3.3 with $\mu = 0.07$
Figure 3.9. Learning Curve for Example 3.3 with $\mu = 0.18$
properties of this algorithm were analyzed and a development of appropriate bounds on the adaptive step size parameter $\mu$ was given. This learning algorithm was compared to the LMS [17] algorithm by considering several numerical examples and observing their performance characteristics.

The results from numerical simulation of the modified Newton's algorithm and the LMS algorithm for application in associative memory problems demonstrate some significant features. The learning rules obtained using the modified Newton's algorithm display slightly poorer convergence properties in terms of speed. They use greater accuracy in the computation of the needed updates for the adjustable parameters which results in longer computation time per iteration. This is due to the more accurate approximation for the slope of the nonlinear sigmoidal function, whereas the LMS algorithm approximates the slope to be zero. Through numerical simulation, it is evident that the selection of the nonlinear function with a small gain sigmoid can result in divergence of the learning algorithms.

It is concluded that the LMS algorithm affords a simple implementation of the training procedures for adjustment of the network parameters, whereas the implementation of the modified Newton's method is generally more complex due to the size of the updating equations. For a more detailed study of the learning rule using modified Newton's method, we shall apply it to the nonlinear identification problems in the next chapter.
There has been a significant interest of late in the development of efficient learning procedures for the identification of nonlinear systems using neural networks. The input-output mapping capabilities of networks with a multilayer architecture can be used in various applications in digital signal processing, image processing, and controls. The identification of an unknown nonlinear system can be performed by approximating the input-output characteristics of the system using an appropriate neural network architecture.

The dynamical neural network can be utilized as an input-output mapper function \( f(x, \xi) \) that can approximate the desired continuous function \( f : \mathbb{R}^m \rightarrow \mathbb{R}^p \), where \( \xi \) is the vector with all adjustable parameters of the network, \( m \) is the dimension of the input vector and \( p \) is the dimension of the output vector. In this application, one generates an input sequence of training numbers \( z^1, z^2, \ldots \) by selecting different \( z^k \in \mathbb{R}^m \) and obtains corresponding output sequences \( y^k = f(z^k) \in \mathbb{R}^p \). The adjustable parameter \( \xi \) is then optimized by minimizing the error function

\[
E = \sum_k |y^k - f(z^k, \xi)|^2.
\]
An efficient approach for nonlinear mapping using an \( n \)-dimensional dynamical neural network was proposed by Pineda [16]. In this approach, some selected processing elements of the network with dimension \( n \geq \{m,p\} \) are used as the input and output nodes. The unselected processing elements can be assigned as hidden nodes of the architecture. The exact selection of the input, output, and hidden nodes for input-output mapping problems was not stated in [16]. Static 3-layer neural networks with one input layer, one hidden layer, and one output layer have commonly been used to solve function approximation problems by Hornik et al [22], Cybenko [23] and Funahashi [24]. The standard backpropagation algorithm [12] has been used to train static networks by propagating the error backwards from the output nodes through the hidden nodes to accomplish the optimization of the weights by using a gradient descent search method. A three layer architecture of a dynamical neural network with assigned input-output layers recently proposed by Sudharsanan [25] will be used in this thesis for application of function approximation.

In this chapter, we shall provide a brief introduction explaining the solution of identification problems using neural networks for the control of unknown nonlinear dynamical systems. This will be followed by the derivation of learning rules using the modified Newton's method to optimize the dynamic parameters of the network. Then the performance of this learning algorithm will be compared with the performance resulting from the LMS method [25] through the evaluation of numerical examples.
4.1 Description of Neural Network

The problem of interest here is to train a multilayer dynamical neural network with a selected architecture to solve the nonlinear identification problem by approximating the desired function. To aid in the discussion of the identification problem, we shall consider simple mapping problems with a scalar output. The objective of the exercise is to train a network with an input-output mapping function to approximate the desired mapping function $f: \mathbb{R}^n \rightarrow \mathbb{R}$.

The input vector can be denoted as $z(k) = [z_1(k), z_2(k), ..., z_m(k)]^T$ which corresponds to the neural network output $y_n(k)$ at the $k$-th updating instant of time. The desired output to be approximated is denoted by $y_d(k)$. The dynamical neural network with recurrent connections utilized in nonlinear identification applications can be described by

$$\dot{x} = -x + Wg(x) + Bz(k)$$
$$y_n(k) = h^T x^*$$

(4-2)

where $x \in \mathbb{R}^n$, $W \in \mathbb{R}^{nn}$, $B \in \mathbb{R}^{nxm}$, $h \in \mathbb{R}^m$, and $g(\cdot): \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a vector-valued function with sigmoidal characteristics. This network is appropriately tailored for the equilibrium points to lie in the saturation region of the nonlinear functions. The vector $x^*$ represents a stable equilibrium of (4-2) corresponding to the application of the input vector $z(k)$ at the $k$-th updating step. The equilibrium point $x^*$ of (4-2) can be obtained by

$$x^* = Wg(x^*) + Bz(k)$$
$$y_n(k) = h^T x^*$$

(4-3)
or equivalently,

\[ x_i^* = \sum_{j=1}^{n} w_j g_j(x_j) + \sum_{l=1}^{m} b_l z_l(k), \quad i = 1, 2, ..., n \]  

(4-4)

\[ y_n(k) = \sum_{i=1}^{n} h_i x_i^* \]

where \( x_i^* \), \( w_j \), \( g_j \), \( b_l \), \( z_l \), \( h_i \) are elements of \( x^* \in \mathbb{R}^n \), \( W \in \mathbb{R}^{nxn} \), \( g \in \mathbb{R}^n \), \( B \in \mathbb{R}^{nxm} \), \( z \in \mathbb{R}^m \), and \( h \in \mathbb{R}^n \), respectively.

A basic necessity in the design of a network for application to nonlinear input-output mapping problems is the satisfaction of appropriate stability properties by the network equilibria. The stability of network equilibria ensures exponential convergence of the network trajectories to steady-state conditions. The dynamics of the network (4-2) in terms of its relaxation time are assumed to be faster than the updating dynamics of the learning algorithm to allow proper results of the steady-state output \( y_n(k) \) to be obtained. This can be achieved by satisfying the stability conditions for the steady-state solution \( x^* \). The stability conditions are guaranteed if the following are satisfied:

(i) \( \text{Spec}(WG-I) \subset \text{LHP} \) where \( G \in \mathbb{R}^{nxn} \Rightarrow G = \text{diag}[g_1', g_2', ..., g_n'] \)

(ii) \( M = \frac{1}{2}[(I-WG)^T + (I-WG)] \) is a positive definite matrix at each time instant \( k \).

A detailed proof of the above stability conditions can be found in [6,26]. It is clear that for a specified equilibrium point of the network one can easily satisfy the stability condition (i) when the elements of \( G \) have small values. Therefore, it is
desired to select the locations of the equilibrium points in the saturation region of the nonlinear sigmoidal functions. One can control the location of any equilibrium by selecting an appropriate gain for the nonlinear sigmoidal function $g(\cdot)$. An exponential convergence of network trajectories can be attained if a proper selection of the slope of the nonlinear function is made. The effect of the slope of the nonlinear function on the stability of network trajectories will be demonstrated later through numerical simulation.

The three layer architecture of the dynamical neural network described by (4-2) consists of the outer layers with input and output processing elements and the internal dynamic hidden layer. This hidden layer is itself a dynamical neural network described by (4-2) since it is structured with nonlinear processing elements with sigmoidal characteristics. The output of the network for information processing is obtained from a linear combination of the outputs of the hidden nodes. The 3-layer architecture proposed by Sudharsanan [25] for application in nonlinear mapping is shown in Figure 4-1. The primary reason for considering the three layer architecture with feedback and recurrent connections are better approximation abilities and reduction in number of hidden nodes [16]. The implementation of learning algorithms for adjustment of the network parameters for nonlinear input-output mapping problems will be discussed in the following section.

4.2 Learning Rule using Modified Newton's Method

The development of supervised training algorithms to solve function
Figure 4-1. Architecture for 3-layer Dynamical Network
approximation problems has received extensive attention in the recent past. The training algorithms allow the network to adaptively adjust the parameters while producing the desired response. The updating rules will be developed for the adjustable parameters of the network described by (4-2). The convergence rate of the learning algorithm depends on the stability of the network equilibria and the appropriate selection of the gain of the nonlinear sigmoidal function $g(\cdot)$. The required convergence of the trajectories can be ensured by an appropriate selection of the gain for the nonlinear sigmoidal functions such that the stability conditions are satisfied.

The learning rules are provided for updating the elements of the weight matrix $W$, the bias input matrix $B$ and the output gain vector $h$ by minimizing the error between the desired output and the output of the neural network described by (4-1). An analytical derivation of the learning rule using modified Newton’s method for optimization of the network parameters is presented here.

For a simplified presentation of the learning algorithm, we shall consider a scalar output such that the resultant mapping problem is $f: \mathbb{R}^m \rightarrow \mathbb{R}$. The neural network output is denoted $y_h(k)$ at the $k$-th updating instant and the desired output to be followed is denoted $y_d(k)$. The standard error propagation will be used to acquire information since there exist only feedforward connections for processing between the hidden layer and the output layer. The learning rules are developed by
minimizing the error function

\[ E(k) = \frac{1}{2} |y_d(k) - y_N(k)|^2 = \frac{1}{2} |y_d(k) - h^T x^*|^2 \]  \hspace{1cm} (4-5) 

where \( x^* \) is a stable equilibrium of the network and is defined by

\[ x^* = Wg(x^*) + Bz(k) \]  \hspace{1cm} (4-6) 

or equivalently,

\[ x_i^* = \sum_{j=1}^{n} w_{ij} g_j(x_j^*) + \sum_{j=1}^{n} b_{ij} z_j(k), \quad i = 1, 2, \ldots, n \]  \hspace{1cm} (4-7) 

An iterative supervised learning algorithm can be developed for the optimization of the network parameters by employing the modified Newton's method for the minimization of the error function \( E \) with respect to the network parameters. Hence, the updating procedure for the elements of the weight matrix \( W \) at any iteration can be stated as

\[ w_{ik}^{new} = w_{ik}^{old} - \mu_1 \frac{\partial E/\partial w_{ik}}{\partial^2 E/\partial^2 w_{ik}^2}, \quad l,k = 1, 2, \ldots, n \]  \hspace{1cm} (4-8) 

where \( \mu_1 \) is the adaptive updating parameter.

Now computing \( \partial E/\partial w_{ik} \) from (4-5),

\[ \frac{\partial E}{\partial w_{ik}} = -h_f(y_d - y_N) \frac{\partial x_i^*}{\partial w_{ik}} \]  \hspace{1cm} (4-9) 

where the weight matrix \( W \) is a nonlinear function of \( x^* \).
Using (4-7), one can evaluate \( \frac{\partial x^*_i}{\partial w_{lk}} \) as

\[
\frac{\partial x^*_i}{\partial w_{lk}} = \sum_{j=1}^{N} \left[ w_{ij}g_j(z_j^*) \frac{\partial x^*_j}{\partial w_{lk}} + g_j(z_j^*) \delta_{jk} \right]
\]

where \( \delta_{ij} \) is the Kronecker delta. Assuming that \( \frac{\partial x^*_i}{\partial w_{lk}} = 0 \) when \( i \neq l \), results in

\[
\frac{\partial x^*_i}{\partial w_{lk}} = \frac{g_k(x_k^*)}{1-w_i g_i'(x_i^*)}.
\]  

(4-10)

Substituting in (4-9) gives

\[
\frac{\partial E}{\partial w_{lk}} = \frac{-h_f(y_d-y_N) g_k(x_k^*)}{1-w_i g_i'(x_i^*)}.
\]  

(4-11)

Similarly, differentiating (4-9)

\[
\frac{\partial^2 E}{\partial w_{lk}^2} = h_f \left( \left( \frac{\partial x^*_i}{\partial w_{lk}} \right)^2 - (y_d-y_N) \left( \frac{\partial^2 x^*_i}{\partial w_{lk}^2} \right) \right)
\]

(4-12)
and differentiating (4-10), one obtains

$$\frac{\partial^2 x_i^*}{\partial r_{ik}^2} = \frac{(1-w_kg_k'(x_i^*))^2 g_k'(x_i^*) g_k(x_i^*)}{D}$$

$$+ \frac{[g_k'(x_i^*)]^2 w_k g_k''(x_i^*) (1-w_kg_k'(x_i^*))}{D}$$

$$+ \frac{g_k(x_k^*) g_k'(x_i^*) \delta_{ik} (1-w_kg_k'(x_i^*)) (1-w_kg_k'(x_i^*))}{D} \quad (4-13)$$

where

$$D = (1-w_kg_k'(x_i^*))^3 (1-w_kg_k'(x_i^*)) .$$

Substituting (4-10) and (4-13) into (4-12) gives

$$\frac{\partial^2 E}{\partial r_{ik}^2} = \frac{h_k^2 g_k(x_k^*)^2 (1-w_kg_k'(x_i^*)) (1-w_kg_k'(x_i^*))}{D}$$

$$- \frac{h_k g_k(x_k^*) (y_d-y_d)(1-w_kg_k'(x_i^*))^2 g_k'(x_k^*)}{D}$$

$$- \frac{h_k w_k g_k''(x_i^*) g_k(x_k^*)^2 (y_d-y_d)(1-w_kg_k'(x_i^*))}{D}$$

$$- \frac{h_k g_k(x_k^*) g_k'(x_i^*) \delta_{ik} (y_d-y_d)(1-w_kg_k'(x_i^*)) (1-w_kg_k'(x_i^*))}{D} \quad (4-14)$$
Finally, inserting (4-11) and (4-14) into the updating rule of (4-8) results in

\[ w_{lk}^{\text{new}} = w_{lk}^{\text{old}} + \mu_1 \frac{N}{D_1 - D_2 - D_3 - D_4} \quad l,k = 1,2,\ldots,n \]

where

\[ N = (y_d - y_N)(1 - w_l g'_l(x_i^*))^2(1 - w_k g'_k(x_k^*)) \]
\[ D_1 = g_k(x_k^*)h_l(1 - w_l g'_l(x_i^*))(1 - w_k g'_k(x_k^*)) \]
\[ D_2 = (y_d - y_N)(1 - w_l g'_l(x_i^*))^2 g'_l(x_i^*) \]
\[ D_3 = (y_d - y_N)w_l g''_l(x_i^*)g_k(x_k^*)(1 - w_k g'_k(x_k^*)) \]
\[ D_4 = (y_d - y_N)g_l(x_i^*)g_k(x_k^*)(1 - w_k g'_k(x_k^*)) \]  

Similarly, the updating scheme for the elements of the input matrix \( B \) can be obtained as

\[ b_{lk}^{\text{new}} = b_{lk}^{\text{old}} + \mu_2 \frac{(y_d - y_N)(1 - w_l g'_l(x_i^*))^2}{h_l(1 - w_l g'_l(x_i^*)) - (y_d - y_N)x_i^*w_l g''_l(x_i^*)} \quad l = 1,2,\ldots,n, \quad k = 1,2,\ldots,m \]

and for the elements of the output gain vector \( h \) can be developed as

\[ h_i^{\text{new}} = h_i^{\text{old}} + \mu_3 \frac{(y_d - y_N)}{x_i^*} \quad l = 1,2,\ldots,n \]

where \( \mu_2 \) and \( \mu_3 \) are the updating gains.

These learning rules for the nonlinear mapper are similar in form but distinct from the rules developed for associative memory problems in Chapter 3. For identification applications, an input vector is presented at each iteration and the
network parameters are optimized at that iteration instead of summing all error values for a given set of vectors. The learning rules are tailored for networks with equilibrium points in the saturation region of the nonlinear sigmoidal function \( g(\cdot) \). This can be attained by selecting a large value of the gain for the sigmoidal function in order to have small values for the elements of the diagonal matrix \( G \) to guarantee stability conditions. The updating gain parameters \( \mu_1, \mu_2, \) and \( \mu_3 \) have to be appropriately selected such that desired convergence properties result. For instance, an estimate for the upper bound on the adaptive stepsize parameter \( \mu_1 \) can be obtained as \( 2/n \) thus showing the relation between the convergence properties of the algorithm and the size of the network [6]. With a similar type of analysis, a bound for the updating gain \( \mu_2 \) has been derived by Sudharsanan [6] as \( \mu_2 < 2/\xi(k) \forall k \) where \( \xi(k) = \max_i [z_i^2(k) + \sum_{j=1}^{m} |z_i(k)z_j(k)|] \). The convergence properties for nonlinear mapping applications are similar to the ones developed in Chapter 3. A more detailed derivation of the convergence criteria and the bounds for the updating parameters can be found in [6].

4.3 Comparison of Learning Rules

A comparison of the performance of the learning algorithm with the modified Newton's method derived above and the LMS algorithm [25] will be given by evaluating the convergence properties from considering some numerical examples. The superb performance of the LMS algorithm when compared to standard backpropagation in this application has been noted by Sudharsanan [25]. Therefore
it is not necessary to compare the performance characteristics of the modified Newton's method to the backpropagation algorithm as part of this thesis and only a comparison with the LMS method will be given.

4.3.1 Training Rules using LMS Method

The adjustable parameters of the dynamical neural network described by (4-2) with feedback and recurrent connections can also be optimized by using the LMS algorithm proposed by Sudharsanan [25]. The updating rules were tailored for the design of the network to function as a nonlinear mapper. This algorithm was customized for a network with the equilibrium points in the saturation region of the sigmoidal function $g(-)$. This can be accomplished by selecting an appropriate gain for the sigmoidal function $g(-)$, such that the exponential stability of the network trajectories is ensured.

A brief summary of the LMS learning algorithm proposed by Sudharsanan [25] for adjusting the network parameters $W, B, \text{ and } h$ for nonlinear input-output mapping problems by minimizing the error function of (4-5) can be given as

$$w_i(k+1) = w_i(k) + \mu_1 h_i(k)(y_d(k) - y_N(k))g'(x_j) \quad i,j = 1, 2, ..., \ n$$

$$b_{ij}(k+1) = b_{ij}(k) + \mu_2 h_i(k)(y_d(k) - y_N(k))z_j(k) \quad i = 1, 2, ..., \ n, \ j = 1, 2, ..., \ m$$

and

$$h_i(k+1) = h_i(k) + \mu_3(y_d(k) - y_N(k))x_i^* \quad i = 1, 2, ..., \ n$$
where $k$ denotes the updating instant, $h_t$, $w_y$, and $b_y$ are elements of $h \in \mathbb{R}^n$, $W \in \mathbb{R}^{m \times n}$, and $B \in \mathbb{R}^{m \times m}$, respectively, and $\mu_1$, $\mu_2$, and $\mu_3$ are the updating gain parameters.

Note that in this updating algorithm, the adjustable parameters of the network are optimized without using any time derivative of the nonlinear sigmoidal function $g(\cdot)$. Therefore, the algorithm is considerably simplified by not requiring any estimation of the gradient for the minimization of the error function.

4.3.2 Discussion with Examples

In this section, we shall present the procedure for implementation of the nonlinear identification scheme described above. The performance of the learning algorithms will be illustrated through simulation results of a numerical example.

The procedure for implementation of the above learning rule for the network described in (4-2) for nonlinear input-output identification is shown by the flow diagram of Figure 4-2. The structure of the identification problem is graphically depicted in Figure 4-3. The primary objective of the identification procedure is to optimize the network parameters by minimizing the error between the outputs of actual system and the neural network.

The following example will demonstrate the performance of the modified Newton's method and the LMS algorithm [25] for nonlinear mapping problems using the 3-layer dynamical network.
Figure 4-2. Flow Diagram of Nonlinear Identification Algorithm
Figure 4-3. Identification Architecture
Example 4.1: The problem is to identify a dynamical system described by the difference equations

\[ y(k+1) = f(y(k), z(k)) \]

where \( z(\cdot) \) is the input to the system, \( y(\cdot) \) is the output of the system and \( k \) represents the discrete time step. The input-output mapping data patterns are generated by using a function of the form,

\[ f(y(k), z(k)) = 1.1\sin(\cos(y(k))) + 1.5z(k) \]

Several exercises using the three layer dynamical network were performed to identify the plant output \( y(k+1) \) using \( y(k) \) and \( z(k) \) as the inputs to the network. The input \( z(k) \) is selected as a sequence of random numbers uniformly distributed between -1 and 1. A 3-layer network with two hidden nodes in the hidden layer is used.

The initial values for the adjustable parameters \( W, B, \) and \( h \) were selected as \( W = [I], B = [I], \) and \( h = [0.2, 0.2]^T, \) respectively. The learning rules were implemented with the selection of the updating gain parameters \( \mu_1 = 0.9, \mu_2 = 0.25, \) and \( \mu_3 = 0.01. \) A nonlinear sigmoidal function with a large gain value was selected as \( g_i(x_i^*) = (2/\pi)\tan^{-1}(20\pi x_i^*). \) The training was conducted over each cycle of 100 data points. The desired output value for a given input value was compared to the predicted output. The mean squared error at the end of each cycle was then calculated. The algorithm was simulated for several cycles using the same sequence of inputs. The learning curves that illustrate the performance of these algorithms are given in Figure 4-4. These curves indicate that the LMS algorithm
Figure 4-4. Learning Curve for Example 4.1 with Sigmoidal Gain $\lambda = 20\pi$
has a slightly faster convergence than that resulting from the modified Newton's method. After the completion of training, the output of the plant and the output of the trained network are shown in Figure 4-5. It can be observed that the two output curves are nearly identical with a negligible amount of error in the identification of the unknown system dynamics.

The effects of the gain of the nonlinear sigmoidal function on the stability of network equilibria were also observed. A nonlinear sigmoidal function with smaller gain values was selected as $g_i(x_i^*) = (2/\pi)\tan^{-1}(3.5\pi x_i^*)$. The resulting performance of the learning algorithm is depicted in Figure 4-6. It is evident that the steady-state error is larger with a smaller sigmoidal gain which results in a poorer identification. This is shown in Figure 4-7 where the output of the neural network does not correspond directly to the plant model. More clearly, Figure 4-8 illustrates the difference between the desired output and the neural network output $(y_d - y_N^*)$ for each time step $k$. Using excessively small values for the gain of the sigmoidal function such as $g_i(x_i^*) = (2/\pi)\tan^{-1}(0.01\pi x_i^*)$, results in divergence of the learning algorithm as illustrated in Figure 4-9. From these studies, it can be concluded that the selection of a large gain value for the sigmoidal function results in a steeper convergence with relatively smaller steady-state error. On the other hand, decreasing the gain of the sigmoidal function results in slower convergence and a larger steady-state error.
Figure 4-5. Output of Plant and Output of Trained Network for Example 4.1 with Sigmoidal Gain $\lambda = 20\pi$
Figure 4-6. Learning Curve for Example 4.1 with Sigmoidal Gain $\lambda = 3.5\pi$
Figure 4-7. Output of Plant and Output of Trained Network for Example 4.1 with Sigmoidal Gain $\lambda = 3.5\pi$
Figure 4-8. Difference Between Desired Output and Trained Neural Network Output for Example 4.1 for Modified Newton's Method
Figure 4-9. Learning Curve for Example 4.1 with Sigmoidal Gain $\lambda = 0.01\pi$
Example 4.2: The dynamical plant to be identified is described by

\[ y_d(k+1) = 1.2 \cos(\sin(y_d(k))) + 0.8 \sin(2y_d(k)) + 1.3(z(k) + 0.1) \]

Input-output mapping data was generated using an input signal

\[ z(k) = \frac{1}{6} \left[ \sin(0.5r(k)) + \cos(10r(k)) + \cos^2(4r(k)) + \cos(5r(k)) + \sin(20r(k)) \right] \]

where \( r(k) = k\pi / 10 \). The initial values for the adjustable parameters \( W, B, \) and \( h \) were selected as \( W = [I], B = [I], \) and \( h = [-0.5, -0.5]^T \), respectively. The updating gain parameters were chosen to be \( \mu_1 = \mu_2 = \mu_3 = 0.01 \). The nonlinear sigmoidal function was selected as \( g(x^*_i) = \frac{2}{\pi} \tan^{-1}(10\pi x^*_i) \). The training was conducted over each cycle of 100 data pairs and the mean squared error at the end of each cycle was computed. The algorithm was executed for several cycles using the same 100 pairs of input-output data. The performance of the identification is illustrated in Figure 4-10. The figure indicates that the error has approached steady-state by the 17-th cycle implying fast convergence rate for both algorithms. The output of the nonlinear plant and the output of the trained neural network were compared after the mean squared error had approached a steady-state value. Both outputs are illustrated in Figure 4-11 which exhibits the output curves being almost identical.

4.4 Conclusions

A 3-layer dynamical neural network with feedback and recurrent connections was utilized for solving nonlinear system identification. A learning algorithm was developed using the modified Newton's method to optimize the network parameters
Figure 4-10. Learning Curve for Example 4.2
Figure 4-11. Output of Plant and Output of Trained Network for Example 4.2
by minimizing the error between the output of the nonlinear dynamical system and
the output of the neural network. This learning technique was compared to the LMS
algorithm [25] through numerical examples.

The result of the numerical simulation of the learning algorithm by using the
modified Newton's method and the LMS algorithm for application in nonlinear
mapping, demonstrates some superior features of the LMS algorithm. The learning
rules that use the modified Newton's algorithm have slightly poorer convergence
properties in terms of speed. This algorithm uses greater accuracy in the
computation. This is due to the accurate computation for the slope of the nonlinear
sigmoidal function, whereas the LMS algorithm approximates the slope to be zero.
The computation time per iteration for the modified Newton's method is longer than
the LMS algorithm. The LMS algorithm is simple to implement, whereas the
implementation of the modified Newton's method may be complex due to the size
of the updating equations.
CHAPTER 5

CONCLUSIONS

In this thesis, the design of continuous-time dynamical neural network models described by (3-1) was discussed with application in associative memory and nonlinear identification problems. In Chapter 1, a brief overview of neural networks for applications in information processing was provided. The relations that exist between the functions of biological neural systems of the human brain and artificial neural networks was shown. The importance of neural networks for application in information processing was also outlined.

A detailed overview of the architectures and learning algorithms for neural networks was provided in Chapter 2. The two important classes of static multilayer networks and dynamical neural networks with feedback and recurrent connections were discussed. The classifications of supervised and unsupervised learning schemes were also outlined.

A supervised learning algorithm for the optimization of dynamical neural network parameters employing modified Newton's method for minimization of the error function was developed. This learning algorithm was then compared to the LMS algorithm proposed by Sudharsanan [6,17,25]. Since the learning capabilities of the LMS algorithm in providing a superior performance compared to standard
backpropagation rule [12] has been demonstrated, it was not necessary to compare this algorithm to the modified Newton's method. The algorithms developed here were tailored for neural networks with equilibrium points in the saturation region of the nonlinear sigmoidal function.

The major contributions of this thesis can be outlined from Chapters 3 and 4. The proposed learning rules utilizing the modified Newton's method were derived in Chapter 3 to optimize the adjustable parameters of the dynamical neural network designed for application in associative memory problems. The performance of this learning algorithm was compared to the LMS method. A qualitative analysis of the dynamical neural network in terms of the location of the network equilibria and its stability properties was briefly mentioned. The proper tailoring of the nonlinear sigmoidal function to control the location of the equilibrium points in the saturation region was observed via simulation as a necessary condition for the stability of solution trajectories.

A learning algorithm was developed for optimization of network parameters using the modified Newton's method for minimization of the error function. The convergence criteria for this algorithm were developed which include an analytical derivation for the upper bound on the adaptive updating parameter. The relation between the size of the network and the convergence of the learning algorithm was acknowledged.

The simulation results obtained by considering several numerical examples demonstrated that the LMS learning algorithm generally has a faster convergence
rate than that of the modified Newton's method. Furthermore, the learning rules using the modified Newton's method are considerably more complex to implement due to the size of the updating equations. This complexity of the learning rules is a result of the computation required for the first and second derivatives of the nonlinear sigmoidal function.

The learning rules using the modified Newton's method were developed in Chapter 4 for the adjustment of the parameters for the neural network to function as a nonlinear input-output mapper. A brief description of the structure of a 3-layer network and its design in approximating the input-output characteristics of an unknown nonlinear system were provided. It was determined that an efficient selection of the gain of the nonlinear sigmoidal function should be made to assign the equilibrium points in the saturation region in order to achieve proper convergence of the network trajectories for successful identification performance. The performance of this learning algorithm was compared to the LMS algorithm through simulation of a numerical example. Once again, the LMS algorithm resulted in a slightly faster rate of convergence, since the slope of the nonlinear sigmoidal function is assumed to be zero, whereas the modified Newton's method required the accurate computation of the slope.

Future investigation in the area of neural networks pertaining to the problems considered in this thesis can be conducted in many directions. The primary course may be focused on an analytical approach or an application development. One of the most interesting challenges lie in the development of a simple and efficient
learning algorithm which demonstrates acceptably fast convergence with an accurate computation of the slope of the nonlinear sigmoidal function. For associative memory applications, it is desired to determine an analytical relationship between the number of stable equilibria of the network and the selection of the gain for the nonlinear sigmoidal functions.

For the application of nonlinear mapping, it is desirable to have analytical studies towards establishing an exact number of hidden nodes needed in the dynamic hidden layer for optimum identification. After identification is completed, this network can be employed to function as a controller for a system with nonlinear dynamics. For example, a dynamical neural network can be designed to identify a potential target by implementing it in the seeker of a tactical missile. Also, the design of the neural network controller for the seeker dynamics would be an optimum approach for the control of the missile, since the dynamics of the seeker may become uncontrollable due to the high acceleration maneuvers by the missile. The use of a neural network architecture for application in nonlinear mapping can be extended to the design of a real-time, fault tolerant controller for nonlinear systems.
LIST OF REFERENCES


