NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

BACK-PROPAGATION NEURAL NETWORKS
IN ADAPTIVE CONTROL OF
UNKNOWN NONLINEAR SYSTEMS

by

Alpay Cakarcan

June 1994

Thesis Advisor:	Roberto Cristi
Co-Advisor:	Ralph Hippenstiel

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.		
1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE BACK-PROPAGATION NEURAL NETWORKS IN ADAPTIVE CONTROL OF UNKNOWN NONLINEAR SYSTEMS		5. FUNDING NUMBERS
6. AUTHOR(S) Alpay Cakarcan		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE A

13. ABSTRACT *(maximum 200 words)*

The objective of this thesis research is to develop a Back-Propagation Neural Network (BNN) to control certain classes of unknown nonlinear systems and explore the network's capabilities. The structure of the Direct Model Reference Adaptive Controller (DMRAC) for Linear Time Invariant (LTI) systems with unknown parameters is first analyzed and then is extended to nonlinear systems by using BNN. Nonminimum phase systems, both linear and nonlinear, have also been considered.

The analysis of the experiments shows that the BNN DMRAC gives satisfactory results for the representative nonlinear systems considered, while the conventional least-squares estimator DMRAC fails. Based on the analysis and experimental findings, some general conditions are shown to be required to ensure that this technique is satisfactory. These conditions are presented and discussed. It has been found that further research needs to be done for the nonminimum phase case in order to guarantee stability and tracking. Also, to establish this as a more general and significant control technique, further research is required to develop more specific rules and guidelines for the BNN design and training.

14. SUBJECT TERMS Back-Propagation Neural Network, Direct Model Adaptive Control, Nonlinear Systems			15.NUMBER OF PAGES 84
			16.PRICE CODE
17.SECURITY CLASSIFICATION OF REPORT	18.SECURITY CLASSIFICATION OF THIS PAGE	19.SECURITY CLASSIFICATION OF ABSTRACT	20.LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Back-Propagation Neural Networks In Adaptive Control of Unknown Nonlinear Systems

by

Alpay Çakarcan
Lieutenant Junior Grade, Turkish Navy
B.S.O.S., Turkish Naval Academy, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE (EE)

from the

NAVAL POSTGRADUATE SCHOOL
June 1994

Author: [redacted]
Alpay Cakarcan

Approved by: [redacted]
Roberto Cristi, Thesis Advisor

[redacted]
Ralph Hippenstiel, Co-Advisor

[redacted]
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

The objective of this thesis research is to develop a Back-Propagation Neural Network (BNN) to control certain classes of unknown nonlinear systems and explore the network's capabilities. The structure of the Direct Model Reference Adaptive Controller (DMRAC) for Linear Time Invariant (LTI) systems with unknown parameters is first analyzed and then is extended to nonlinear systems by using BNN. Nonminimum phase systems, both linear and nonlinear, have also been considered.

The analysis of the experiments shows that the BNN DMRAC gives satisfactory results for the representative nonlinear systems considered, while the conventional least-squares estimator DMRAC fails. Based on the analysis and experimental findings, some general conditions are shown to be required to ensure that this technique is satisfactory. These conditions are presented and discussed. It has been found that further research needs to be done for the nonminimum phase case in order to guarantee stability and tracking.

Also, to establish this as a more general and significant control technique, further research is required to develop more specific rules and guidelines for the BNN design and training.

Thesis
C18555
C.2

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

In appreciation for their time, effort, and patience, many thanks go to my instructors, advisors, and the staff and faculty of the Electrical and Computer Engineering Department and Weapons System Engineering Department, NPS.

I would like to offer special thanks to Professor Cristi, my thesis advisor, to Professor Ralph Hippenstiel, and to Chat, my friend, for their patience, support and encouragement.

I. INTRODUCTION

In the past three decades, major advances have been made in adaptive identification and control for identifying and controlling Linear Time Invariant LTI systems with unknown parameters. The choice of the identifier and controller structures is based on well established results in systems theory. The adaptive control theory is then improved and applied to nonlinear dynamic plants employing neural networks with the right choice of the identifier and controller structures.

The objective of this thesis research is to develop a Back-Propagation Neural Network (BNN) to control certain classes of unknown nonlinear dynamical systems. Initial analysis is directed towards a Direct Model Reference Adaptive Controller (DMRAC) for an unknown LTI system. Results of the simulation are displayed for this system. The same analysis is then performed for an unknown nonminimum phase system. The adaptive control theory is then applied to nonlinear unknown systems by employing neural networks. Basically, four nonlinear models have been analyzed and simulated. Lastly, the adaptive control algorithm is tried with nonlinear unknown nonminimum phase systems again by employing neural networks.

In what follows, Chapter II presents the analysis of a DMRAC for an unknown LTI system and for an unknown nonminimum phase LTI system. Chapter III concerns neural networks in general and adaptive control of nonlinear plants employing neural networks.

Chapter IV is composed of the simulations that have been carried out. The results, conclusions and discussions are given in Chapter V.

II . ADAPTIVE CONTROL OF UNKNOWN LTI SYSTEMS

A. ADAPTIVE CONTROL IN GENERAL

The search for design techniques to control systems with unknown parameters has drawn much attention in recent years. Adaptive control is currently one of the most commonly used methods in the control of systems with uncertain dynamics. Several applications of adaptive control such as ship steering, aircraft control, robot manipulation, chemical process control and bio-medical engineering have been performed in the past years [Ref.1]. The general structure of an adaptive control system is shown in Figure 1.

During the development of adaptive control, two major classes have emerged: learning systems, which lead to the introduction of learning automatons in the control literature, and; adaptive systems using a model reference, known as model reference adaptive control systems (DMRAC). The design of the adaptation algorithm in DMRAC is based on stability theory since the stability of the closed loop system is a fundamental requirement in the design of control systems.

In adaptive control of an unknown linear time invariant (LTI) system, unknown parameters are estimated by an on-line estimator. Based on the estimated parameters, an adequate design can be achieved to implement the chosen control law. This process is commonly referred to as *indirect adaptive control*. Figure 2 shows the structure of an indirect adaptive control system.

On the other hand, it is possible to parametrize the unknown system in terms of the control parameters (e.g. the state-feedback gains) to implement the chosen control law. In this case, the estimation and the control processes are carried on together. This alternative approach is called *direct adaptive control*. Figure 3 shows the structure of a direct adaptive control system.

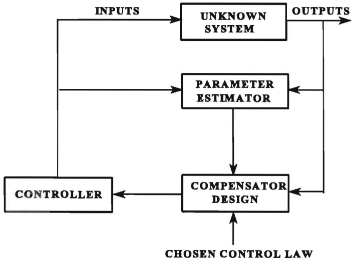

Figure 1. General Adaptive Control Structure

The main idea in all models is the parameter estimation. When estimating the parameters, the uncertainties must be expressed linearly in terms of a set of unknown parameters.

In linear systems, the regression can be adequately used to obtain the state measurements or observations from the systems, with the unknown parameters as coefficients. However, in nonlinear systems, nonlinear functions of the measurements or observations are generally required. With unknown nonlinear systems, these nonlinear functions cannot be specified most of the time. So, the use of neural networks as generic parametric models is highly recommended in such cases.

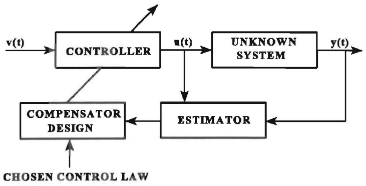

Figure 2. Indirect Adaptive Control Algorithm

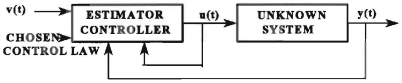

Figure 3. Direct Adaptive Control Algorithm

Before developing a neural network based direct model reference adaptive controller for unknown nonlinear systems, the design of a DMRAC for unknown LTI systems will be presented.

B. ANALYSIS OF A DMRAC FOR UNKNOWN LTI SYSTEMS

Consider an LTI system

$$\boldsymbol{A(q)y(t) = B(q)u(t)}, \tag{2-1}$$

with $\boldsymbol{A(q)}$ and $\boldsymbol{B(q)}$ being polynomial operators[1] with unknown coefficients. For DMRAC design, it is assumed that:

1. $\boldsymbol{A(q)}$ is monic and degree[$\boldsymbol{A(q)}$]=n is known.

2. The transfer function $\boldsymbol{B(q)/A(q)}$ is strictly proper and the relative degree is known.

The goal of the controller is to move the closed loop system's dynamics such that

$$\boldsymbol{D(q)\,y(t)=v(t)}, \tag{2-2}$$

where $\boldsymbol{D(q)}$ is the arbitrary monic stable characteristic polynomial operator of the desired system, and $\boldsymbol{v(t)}$ is an external input. The general structure of DMRAC is shown in Figure 4.

By employing a steady-state Kalman filter, observer-state-feedback system yields the following structure for the feedback controller,

[1]The argument q of the polynomial operators is the forward time-shift operator in discrete time modelling.

$$u(t) = \frac{h(q)}{\alpha(q)} u(t) + \frac{k(q)}{\alpha(q)} y(t) + v(t), \tag{2-3}$$

where $\alpha(q)$ is the monic characteristic polynomial operator of the observer. It can be chosen arbitrarily, provided that the roots are in the stable region. The polynomial operators $h(q)$ and $k(q)$ are the feedback polynomial operators of the unknown system.

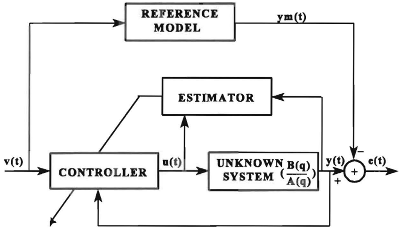

Figure 4. General Structure of the DMRAC

By using partial state representation, equation (2-1) can be reconstructed as

$$A(q)\, z(t) = u(t),$$
$$y(t) = B(q)\, z(t). \tag{2-4}$$

Combining equations (2-3) and (2-4) yields the following:

$$[\alpha(q)A(q) - h(q)A(q) - k(q)B(q)]z(t) = \alpha(q)v(t), \qquad (2\text{-}5)$$
$$y(t) = B(q)z(t)$$

To obtain the desired closed loop behavior, the Diophantine equation

$$\alpha(q)A(q) - h(q)A(q) - k(q)B(q) = \frac{1}{b_1}\alpha(q)D(q)B(q), \qquad (2\text{-}6)$$

should be satisfied so that the closed loop system poles coincide either with those of the reference model or with the system's zeros. The factor $1/b_1$ is needed to ensure that the right side of the equation is monic and the roots of $B(q)$ are assumed to be inside the unit circle (i.e. minimum phase) [Ref.2].

In equation (2-6), the polynomial operators $A(q)$ and $B(q)$ are assumed to be relatively co-prime (i.e. there is no pole-zero cancellation) which guarantees a unique solution for $h(q)$ and $k(q)$.

Since $A(q)$ and $B(q)$ are unknown polynomial operators, an estimator is required to estimate the system parameters on-line, based on the generic recursive regression analysis. The regression equation,

$$\alpha(q)u(t) = h(q)u(t) + k(q)y(t) + \frac{1}{b_1}\alpha(q)D(q)y(t), \qquad (2\text{-}7)$$

is obtained by using partial state transformation.

Using q as the forward time-shift operator, the filtered input and output signals can be defined as

$$\begin{aligned} q^{-n}\alpha(q)\, y^{F}(t) &= y(t), \\ q^{-n}\alpha(q)\, u^{F}(t) &= u(t). \end{aligned} \tag{2-8}$$

Equation (2-7) can be expressed in a more convenient form as follows:

$$q^{-r}u(t) = q^{-(n+r)}h(q)\, u^{F}(t) + q^{-(n+r)}k(q)\, y^{F}(t) + \frac{1}{b_1}q^{-r}D(q)\, y(t), \tag{2-9}$$

where n is the order of the unknown system and r is the number of the closed loop system poles which must be placed to match those of the reference model. Equation (2-9) can be represented in a matrix formation as

$$q^{-r}u(t) = \Phi^{T}(t)\Theta_0,$$

where

$$\Phi(t) = \begin{bmatrix} u^F(t-r-1) \\ u^F(t-r-2) \\ \cdots \\ u^F(t-r-n+1) \\ y^F(t-r-1) \\ y^F(t-r-2) \\ \cdots \\ y^F(t-r-n+1) \\ q^{-r}D(q)y(t) \end{bmatrix} \qquad \Theta_0 = \begin{bmatrix} h_1 \\ h_2 \\ \cdot\cdot \\ h_{(n-1)} \\ k_1 \\ k_2 \\ \cdot\cdot \\ k_{(n-1)} \\ \dfrac{1}{b_1} \end{bmatrix} \tag{2-10}$$

By using the linear regressor $\Phi(t)$ and recursive estimation of Θ_0 as

$$\Theta(t+1) = \Theta(t) + \frac{P(t)\Phi(t)\,[u(t-r) - \Phi^T(t)\Theta(t)]}{1 + \Phi^T(t)P(t)\Phi(t)}, \tag{2-11}$$

and,

$$P(t+1) = P(t) + \frac{P(t)\Phi(t)\Phi^T(t)P(t)}{1 + \Phi^T(t)P(t)\Phi(t)}, \tag{2-12}$$

the unknown parameters of the system are estimated. Hence, the control equation (2-3) can be rewritten as

$$u(t) = q^{-n}h(q)u^F(t) + q^{-n}k(q)y^F(t) + \frac{1}{b_1}v(t). \tag{2-13}$$

Equation (2-13) can be represented as

$$u(t) = \Phi^T_c(t)\Theta_0,$$

$$\Phi_c(t) = \begin{bmatrix} u^F(t-1) \\ u^F(t-2) \\ \dots \\ u^F(t-n+1) \\ y^F(t-1) \\ y^F(t-2) \\ \dots \\ y^F(t-n+1) \\ v(t) \end{bmatrix} \tag{2-14}$$

It can be noticed that all the equations in both estimation and control phases are the same except for the last element in the vector $\boldsymbol{\Phi}(t)$ and $\boldsymbol{\Phi}_c(t)$. Hence, this identical structure of the estimator and the controller prevents the unnecessary intermediate control design calculations and speeds up the control process.

Figure 5 illustrates the estimation and the control algorithm of the DMRAC where $\hat{\boldsymbol{\Theta}}(t)$ is a linear associative memory with recursive estimation to minimize the mean square errors between $u(t)$ and $\hat{u}(t) = \boldsymbol{\Phi}^T(t)\hat{\boldsymbol{\Theta}}(t)$.

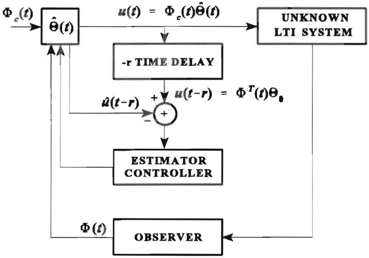

Figure 5. Estimation and Control Algorithm of the DMRAC

As an illustration, Appendix A contains a numerical example of the design of a DMRAC for an unknown minimum phase LTI system.

C. ANALYSIS OF A DMRAC FOR UNKNOWN NONMINIMUM PHASE LTI SYSTEM

Consider the LTI system in section B, with polynomial operators ***A(q)*** and ***B(q)*** which cause the system to be nonminimum phase (i.e. at least one zero outside the unit circle in z-domain).

Defining the polynomial operators ***A(q)*** and ***B(q)***

$$\begin{aligned} A(q) &= q^{n} + a_1 q^{(n-1)} + \ldots\ldots + a_n, \\ B(q) &= q^{n} + b_1 q^{(n-1)} + \ldots\ldots + b_n, \end{aligned} \tag{2-15}$$

and using the same structure for the feedback controller in equation (1-3), the following Diophantine equation

$$h(q)A(q) + k(q)B(q) = \alpha(q)[A(q) - P^{*}(q)], \tag{2-16}$$

is obtained. $\alpha(q)$ is the monic characteristic polynomial operator of the observer. It is also arbitrary and stable. $P^{*}(q)$ is the stable arbitrary characteristic polynomial of the desired closed loop system.

In nonminimum phase systems, for estimating ***h(q)*** and ***k(q),*** the filtered partial state $z^{F}(t)$ must be replaced by an estimate in terms of the available signals $u^{F}(t)$ and $y^{F}(t)$. This can be done using the Bezout identity as follows:

$$b(q)B(q) + c(q)A(q) = 1, \tag{2-17}$$

which holds provided $A(q)$ and $B(q)$ are mutually co-prime polynomials. Hence, defining $b(q)$ and $c(q)$ as

$$\begin{aligned} b(q) &= b_1 q^{n} + b_2 q^{(n-1)} + \ldots\ldots + b_n, \\ c(q) &= q^{n} + c_1 q^{(n-1)} + \ldots\ldots + c_n, \end{aligned} \tag{2-18}$$

Equation (2-10) can be rewritten as

$$\alpha(q)u(t) = \Phi^T(t)\Theta_0,$$

$$\Phi(t) = \begin{bmatrix} u^F(t-r-1) \\ u^F(t-r-2) \\ \ldots \\ u^F(t-r-n+1) \\ y^F(t-r-1) \\ y^F(t-r-2) \\ \ldots \\ y^F(t-r-n+1) \\ \bar{u}^F(t-r-1) \\ \bar{u}^F(t-r-2) \\ \ldots \\ \bar{u}^F(t-r-n+1) \\ \bar{y}^F(t-r-1) \\ \bar{y}^F(t-r-2) \\ \ldots \\ \bar{y}^F(t-r-n+1) \end{bmatrix} \qquad \Theta_0 = \begin{bmatrix} h_1 \\ h_2 \\ \ldots \\ h_{(n-1)} \\ k_1 \\ k_2 \\ \ldots \\ k_{(n-1)} \\ b_1 \\ b_2 \\ \ldots \\ b_n \\ c_1 \\ c_2 \\ \ldots \\ c_n \end{bmatrix} \tag{2-19}$$

where

$$\bar{u}^F(t) = P^*(q)\alpha(q)u^F(t),$$

$$\bar{y}^F(t) = P^*(q)\alpha(q)y^F(t).$$

So, the algorithm in order to obtain a DMRAC for unknown LTI system is developed by using an adaptive pole placement algorithm for unknown nonminimum phase LTI systems.

Using equation (2-17), the state $z(t)$ can be estimated as

$$z(t) = c(q)\,u(t) + b(q)\,y(t)\ , \tag{2-20}$$

which implies that $b(q)$ and $c(q)$ are the parameters of the observer.

Appendix B contains a numerical example of the design of a DMRAC for an unknown nonminimum phase LTI system.

III. NEURAL NETWORKS IN ADAPTIVE CONTROL OF UNKNOWN NONLINEAR SYSTEMS

A. NEURAL NETWORKS IN GENERAL

Neural networks are composed of many simple elements operating in parallel. These elements are inspired by biological nervous systems. The network function is determined largely by the connections between the elements [Ref.3].

Neural networks have been trained to perform complex functions in various fields of applications including pattern recognition, identification, classification, speech, vision and control systems. Neural networks have been studied for many years in the hope of achieving human-like performance in the fields of speech and image recognition [Ref.4].Today neural networks can be trained to solve problems that are unsuitable to conventional computers [Ref.3].

A neural network is usually a layered network consisting of an input layer, an output layer and at least one layer of nonlinear processing elements. The nonlinear processing elements, which sum incoming signals and generate output signals according to some predefined function are called *neurons*. The neurons are connected by terms with variable weights. The output of one neuron multiplied by a weight becomes the input of an adjacent neuron of the next layer.

A single neuron with n inputs is shown in Figure 6. The individual input $x(j)$, weighted by the element $w(1,j)$ of the matrix w, are summed to form the weighted inputs to the transfer function Γ. The neuron has a bias b and an output y given by

$$y = \Gamma \left[\sum_{j=1}^{n} w(1,j)\ x(j) + b \right] . \tag{3-1}$$

As in equation (3-1), the transfer function net input is the sum of the weighted inputs and the bias b. This sum is the argument of the transfer function. The weight vector w, and the input vector x can be represented as

$$w = [\ w(1,1)\ w(1,2)\ .....\ w(1,n)\ ] , \qquad x = \begin{bmatrix} x(1) \\ x(2) \\ \dots \\ x(n) \end{bmatrix} . \tag{3-2}$$

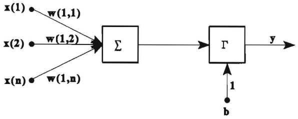

Figure 6. A Single Neuron Model

Two or more of these neurons may be combined in a layer and a particular network might contain one or more such layers. First consider a single layer of neurons where each element of the input vector $\boldsymbol{x}$ is connected to each neuron input through a weight matrix $\boldsymbol{w}$. In this case, there are $\boldsymbol{m}$ neurons and each of them has a summer and the summer outputs form an $\boldsymbol{m}$ element vector z. The transfer function net input is the sum of its appropriately weighted inputs and bias $\boldsymbol{b}$. At the end, all neuron outputs form an $\boldsymbol{m}$ element output vector y. A one layer neural network with $\boldsymbol{n}$ inputs and $\boldsymbol{m}$ neurons is shown in Figure 7. So, equation (3-1) can be represented for this case as

$$y = \Gamma \ [\sum_{i=1}^{m}\sum_{j=1}^{n} w(i,j) \ x(j) + b(i) \ ] \ . \tag{3-3}$$

All weights can be represented in matrix format as

$$w = \begin{bmatrix} w(1,1) & w(1,2) & ..... & w(1,n) \\ w(2,1) & w(2,2) & ..... & w(2,n) \\ ... & ... & ..... & ... \\ w(m,1) & w(m,2) & ..... & w(m,n) \end{bmatrix} , \tag{3-4}$$

where each row represents the weights of one layer.

A network can have several layers where each layer has a weight matrix $\boldsymbol{w}$, a bias vector $\boldsymbol{b}$, a weighted input z to the transfer function, and an output vector $\boldsymbol{y}$. Layers whose outputs are the network's outputs are called *output layers*. All other layers are called *hidden layers*. Commonly an input vector is presented to a network, the outputs are calculated and an algorithm is applied to determine the weight element changes. However, one may want to apply more than one input vector simultaneously and get the network's response to each one of them. This operation is called batching and this will not be part of this thesis.

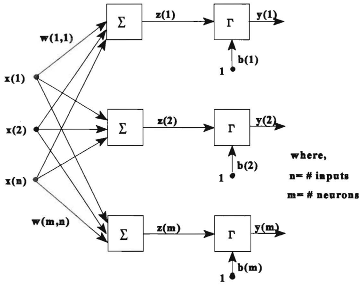

Figure 7. A one Layer Neural Network

Initialization of the weights and bias elements to small positive and negative values provides enough variation in the weights and biases so that neurons in the network start out with a range of behaviors that can be taken advantage of by the learning rule.

There are several learning rules such as Hebbian, Instar, Kohonen, Outstar learning rules [Ref.3]. Among the most common learning rules is *backpropagation* which adjusts the weights and biases of the network in order to minimize the mean squared error criterion. This is a gradient algorithm and it is done by continually adjusting the values of the weights and biases in the direction of the gradient of an appropriate cost function.

The most distinctive and appealing feature of many neural networks is that they learn by examples.

Currently, the most popular and commonly used neural networks for control system design is the *Back-Propagation Neural Networks* (BNN) which is discussed in the following section.

B. ANALYSIS OF BACK-PROPAGATION NEURAL NETWORKS

A Back-Propagation Neural Network is a multilayer, feed-forward network which has an input layer, an output layer and at least one hidden layer. Neurons are found in the output and hidden layer(s), while the input layer has only input connections feeding the neurons in the first hidden layer. There exists no feedback or even interconnection between neurons in the same layer. There is generally a bias input for each neuron with an associated non-zero weight. A three layer network is shown in Figure 8. It has one output layer and two hidden layers.

In Figure 8, ***wi(j,k)*** is defined as the connection weight for the path from the j^{th} neuron in the $(i-1)^{th}$ layer to the k^{th} neuron in the i^{th} layer. It is assumed that the weight vector ***wi(j,k)*** is a constant so that a partial derivative can be rigorously defined [Ref.5]. Then the BNN in Figure 8 can be represented mathematically as

$$y3 = \Gamma 1(w3 * \Gamma 2(w2 * \Gamma 3(w1 * x + b1) + b2) + b3), \tag{3-5}$$

where $x(j)$ is the input vector to the BNN.

In the learning process, the BNN adjusts weights wi for all i, to minimize a suitable function of the error between the output y and a desired output y_d where $y_d(t)$ is the signal to be approximated given the input $x(t)$, $t = 1,....,N$.

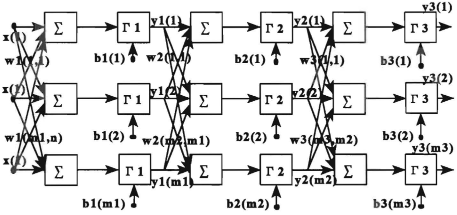

Figure 8. A Multilayer Neural Network

The most common error function used is

$$J = \frac{1}{2}\sum_{t=1}^{N} \|y(t|\Theta) - y_d(t)\|^2, \tag{3-6}$$

where $y_d(t)$ is the signal to be approximated given the input $x(t)$, $t = 1,....,N$ and $y(t|\boldsymbol{\Theta})$ the output of the neural network with parameter $\boldsymbol{\Theta}$. A general picture of this mechanism is given in Figure 9.

The BNN implements a modification of the gradient descent algorithm to update each weight at time $t+1$ as follows:

$$\boldsymbol{\Theta}(t+1) = \boldsymbol{\Theta}(t) - \mu \frac{1}{2} \frac{\partial \| e(t) \|^2}{\partial \boldsymbol{\Theta}} \bigg]_{\boldsymbol{\Theta}(t)}, \tag{3-7}$$

where $\mu > 0$ is the learning rate and $\boldsymbol{\Theta}$ is the weight vector and $e(t) = y(t|\boldsymbol{\Theta}) - y_d(t)$.

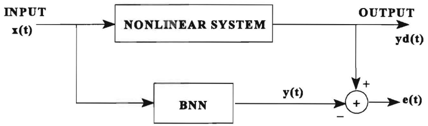

Figure 9. Obtaining the Error in BNN

A general flow diagram for a BNN is given in Figure 10. The algorithm that is used to simulate the BNN is presented below. The MATLAB codes to accomplish this task are in Appendix C. For simulation purposes, a two layer neural network is chosen which is quite effective compared with three or more layered networks. A sigmoid is used as the nonlinear transfer function.

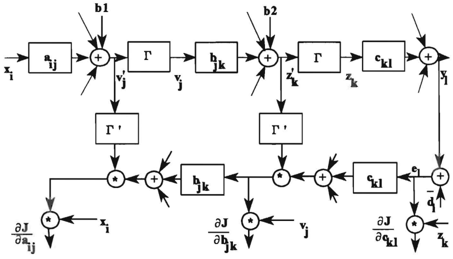
b1
b2
x_i
a_{ij}
+
v'_j
Γ
v_j
b_{jk}
z'_k
z_k
c_{kl}
y_l
Γ'
*
e_l
d_l
−
$\frac{\partial J}{\partial a_{ij}}$
$\frac{\partial J}{\partial b_{jk}}$
$\frac{\partial J}{\partial c_{kl}}$

Figure 10. A General Flow Diagram for BNN

In order to derive the back-propagation algorithm, the weights, biases, input and output are defined as

$$w1 = \begin{bmatrix} a_{11} \\ a_{12} \end{bmatrix} \quad w2 = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \quad w3 = [f_{11} \quad f_{12}] \,,$$

$$b1 = \begin{bmatrix} b_{11} \\ b_{12} \end{bmatrix} \quad b2 = \begin{bmatrix} b_{21} \\ b_{22} \\ \dots \\ b_{2m_1} \end{bmatrix} \quad u = [u] \quad y = [y] \,. \tag{3-8}$$

The input to the first layer is given as

$$v' = w1 * u + b1$$

$$v' = \begin{bmatrix} a_{11} \\ a_{12} \end{bmatrix} u + \begin{bmatrix} b_{11} \\ b_{12} \end{bmatrix} = \begin{bmatrix} u a_{11} + b_{11} \\ u a_{12} + b_{12} \end{bmatrix} = \begin{bmatrix} v_1' \\ v_2' \end{bmatrix} . \tag{3-9}$$

Hence, the output of the first layer, v can be represented as

$$v = sigmoid\ (v') = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} . \tag{3-10}$$

Likewise, the input to the second layer is given as

$$z' = w2 * v + b2$$

$$z' = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} v_1 + c_{12} v_2 + b_{21} \\ c_{21} v_1 + c_{22} v_2 + b_{22} \end{bmatrix} = \begin{bmatrix} z_1' \\ z_2' \end{bmatrix} . \tag{3-11}$$

And the output of the second layer can be given as

$$z = sigmoid\ (z') = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} . \tag{3-12}$$

So, the output of the network can be written as

$$y = w3 * z = [f_{11} \quad f_{12}] \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = f_{11} * z_1 + f_{12} * z_2 . \tag{3-13}$$

The error can be calculated as

$$e = y - y_d = d3 \ . \tag{3-14}$$

Back-Propagating the error through the network, the followings are obtained:

$$\begin{aligned} d2 &= diag\ (w3' * d3) * dsig\ (z) \\ d1 &= diag\ (w2' * d2) * dsig\ (v'). \end{aligned} \tag{3-15}$$

Partial differentiations of an appropriate cost function with respect to all weight vectors multiplied by a proper learning rate are subtracted from the weight vectors of the previous iteration to obtain the new weight vectors as follows:

$$\begin{aligned} w3 &= w3 - \mu * \frac{\partial J}{\partial w3} = w3 - \mu\,(d3 * z') \\ w2 &= w2 - \mu * \frac{\partial J}{\partial w2} = w2 - \mu\,(d2 * v') \\ w1 &= w1 - \mu * \frac{\partial J}{\partial w1} = w1 - \mu\,(d1 * u') \ . \end{aligned} \tag{3-16}$$

In a similar way, we can show that for the bias terms,

$$b2 = b2 - \mu * \frac{\partial J}{\partial b2} = b2 - \mu * d2 \ ,$$

$$b1 = b1 - \mu * \frac{\partial J}{\partial b1} = b1 - \mu * d1 \ . \qquad (3\text{-}17)$$

IV. APPLICATIONS AND SIMULATIONS

A. EXPERIMENTING WITH THE BNN DMRAC FOR UNKNOWN MINIMUM PHASE SYSTEMS

In this chapter, the results of the experiments using the BNN DMRAC on various nonlinear SISO systems, are presented. Four experiments are conducted using software simulations for the four classes of unknown nonlinear systems considered in Chapter III. The main purpose of these experiments is to see under what conditions the proposed BNN DMRAC works. The software simulation programs used in these experiments are listed in Appendix C.

Lastly, an experiment for an unknown nonlinear nonminimum phase system is conducted based on the system Model 2.

As stated in [Ref.1], four important classes of unknown nonlinear SISO systems are considered for direct adaptive control using the BNN. They are modeled in discrete-time for analysis and simulation. These are the system models used in [Ref.6] for which BNN indirect adaptive control has been successfully demonstrated. The four models are the following with Γ, a continuous smooth function:

(1) Model 1 :

$$y(t+1) = \sum_{k=0}^{n-1} a_k \, y(t-k) + \Gamma \, [(u(t), u(t-1), \ldots, u(t-m+1)] \tag{4-1}$$

Large mechanical systems, hard nonlinearities such as input saturation, dead zones or backlash are readily described by this model [Ref.1].

(2) Model 2 :

$$y(t+1) = \Gamma\ [y(t), y(t-1), \ldots, y(t-n+1)] + \sum_{k=0}^{m-1} b_k\ u(t-k) \tag{4-2}$$

The action of viscous drag on an underwater vehicle can be modelled by this model [Ref.1].

(3) Model 3 :

$$y(t+1) = \Gamma\ [y(t), y(t-1), \ldots, y(t-n+1)] + \Gamma\ [u(t), u(t-1), \ldots, u(t-m+1)] \tag{4-3}$$

Underwater vehicles subjected to input saturation and viscous drag can be formulated by this model [Ref.1].

(4) Model 4 :

$$y(t+1) = \Gamma\ [y(t), y(t-1), \ldots, y(t-n+1), u(t), u(t-1), \ldots, u(t-m+1)] \tag{4-4}$$

Bilinear systems are part of this model [Ref.1].

1. Experiment 1: System Model 1

In the first experiment, a nonlinear system described by Model 1 was controlled by a BNN DMRAC. The chosen nonlinear system is given in equation (4-1) as

$$y(t+1) = 0.3\, y(t) + 0.6\, y(t-1) + u(t)^3 + 0.3\, u(t)^2 - 0.4\, u(t)\ . \tag{4-5}$$

According to the suggested training procedures, the BNN was first trained off-line. The training set made of a randomly generated training input $u(t)$ and the resulting output $y(t)$ of the neural network and also the external input $v(t)$ and the desired response of the system, $y_m(t)$ are shown in Figure 11. The Training set consists of 10 000 data points each for the input and the output measurements. The external input to the system, $v(t)$ is a sum of sinusoids with different magnitudes and frequencies. The BNN was next placed on-line to control the system. During the on-line control phase, the BNN recursively learns to adapt to the required control structure. The output of the controlled system compared to that of the reference model is displayed in Figure 12. The reference model used in the experiment is chosen as

$$y_m(t) = 0.8\, y_m(t-1) + v(t-1)\ , \tag{4-6}$$

where $v(t)$ is the external input to the system.

2. Experiment 2: System Model 2

In the second experiment, a nonlinear system described by Model 2 was used. It is governed by

$$y(t+1) = \frac{y(t)\, y(t-1)\, [y(t) + 2.5]}{1 + y(t)^2 + y(t-1)^2} + u(t)\ . \tag{4-7}$$

The input to the BNN estimator will be the regressor vector $\Phi(t)$ of equation (2-10). $\Phi_c(t)$ in equation (2-14) is the input vector during the control phase. The same reference model and the same procedures for off-line training and on-line control-plus-learning were employed. The BNN estimator was first trained off-line with a 10 000 point training set. The training input ***u(t)*** and the resulting output ***y(t)*** of the neural network and also the external input, ***v(t)*** and the desired response, $\boldsymbol{y_m(t)}$ of the system are shown in Figure 13.

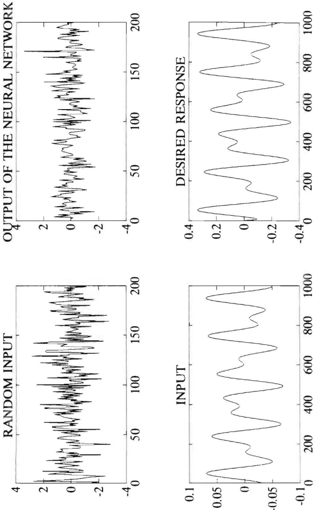

Figure 11. The Training Input, the Resulting Output, the External Input and the Desired Response for the System Model 1

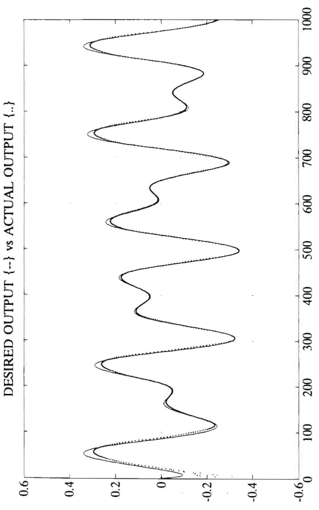

Figure 12. The Output of the Controlled System Compared To That of the Reference Model for System Model 1

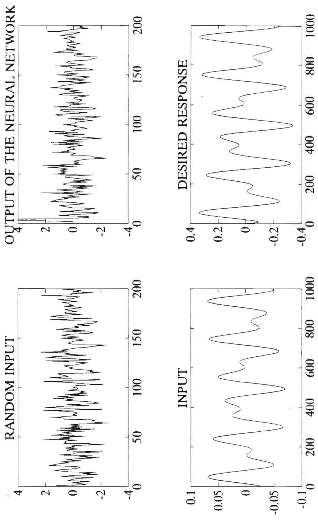

Figure 13. The Training Input, the Resulting Output, the External Input and the Desired Response for the System Model 2

Finally, the BNN was placed on-line to control, and learn the control structure and estimate the parameters simultaneously. As shown in Figure 14, the system with the BNN DMRAC successfully tracks the model reference system closely. To optimize the performance of the control system, many different learning parameters and training data were tried. However, once a trained BNN works, it tracks the reference model reasonably well for the inputs with similar characteristics.

3. Experiment 3: System Model 3

In this experiment, a nonlinear system described by Model 3 is chosen as

$$y(t+1) = \frac{y(t)}{1 + y(t)^2} + u(t)^3 \; . \tag{4-8}$$

The same procedures, used for the previous experiments were conducted. The training input $u(t)$ and the resulting output $y(t)$ of the neural network, and also the actual input $v(t)$ and the desired response $y_m(t)$ of the system are shown in Figure 15. The comparison between the actual system output and the reference model output are displayed in Figure 16.

4. Experiment 4: System Model 4

In the final experiment, the nonlinear system is governed by:

$$y(t+1) = \frac{y(t)\, y(t-1)\, y(t-2)\, u(t-1)\, [y(t-2) + 1] + u(t)}{1 \quad y(t \quad 1)^2 \quad y(t \quad 2)^2} \; . \tag{4-9}$$

Other than the order of the system, there is not much change in the input vectors. Once again, the same procedures were applied to this nonlinear model. The training input ***u(t)*** and

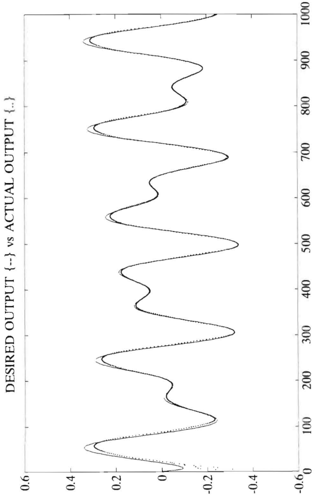

Figure 14. The Output of the Controlled System Compared To That of the Reference Model for System Model 2

the resulting output $y(t)$ of the neural network, and also the actual input $v(t)$ and the desired response $y_m(t)$ are displayed in Figure 17. The output of the controlled system compared to that of the reference model is displayed in Figure 18.

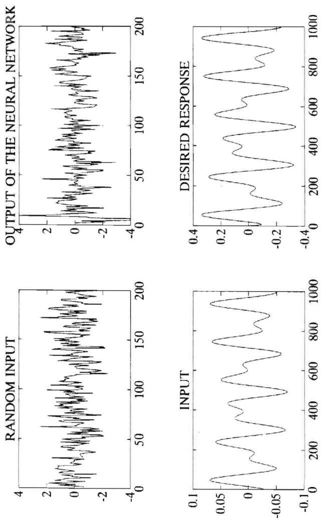

Figure 15. The Training Input, the Resulting Output, the External Input and the Desired Response for the System Model 3

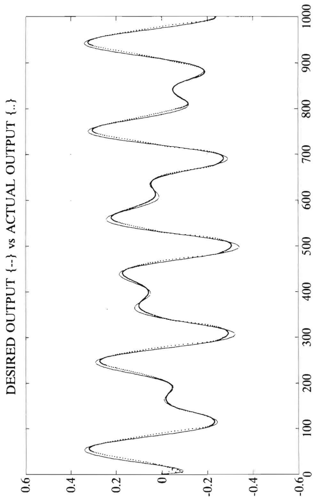

Figure 16. The Output of the Controlled System Compared To That of the Reference Model for System Model 3

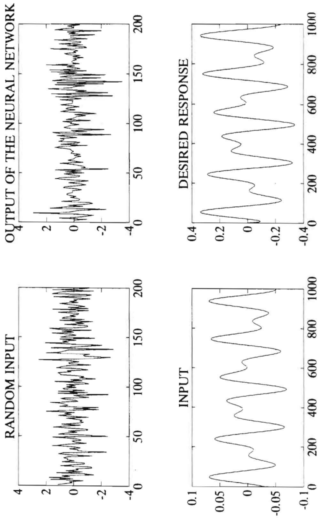

Figure 17. The Training Input, the Resulting Output, the External Input and the Desired Response for the System Model 4

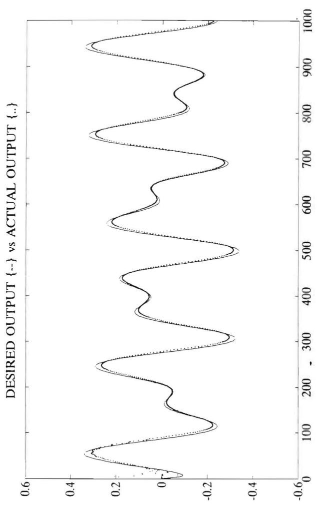

Figure 18. The Output of the Controlled System Compared To That of the Reference Model for System Model 4

B. EXPERIMENTING WITH THE BNN DMRAC FOR UNKNOWN NONMINIMUM PHASE SYSTEMS

In this section, the BNN DMRAC is extended to nonminimum phase systems. For experiment purposes, Model 2 is considered.

The pole placement algorithm [Ref.7] which was used for LTI nonminimum phase systems was tested on nonlinear nonminimum phase systems. But the experiment showed that nonlinearities affect the results and cause the system to go unstable especially in the nonminimum phase case.

So, it was decided not to consider the pole placement approach for this case and the BNN DMRAC is directly applied to a nonlinear nonminimum phase system. The simulations yield fairly reasonable results which are displayed in Figure 19 and in Figure 20.

The unknown nonlinear nonminimum phase system used is chosen as

$$y(t+1) = \frac{y(t)\,y(t-1)\,[y(t)+2.5]}{1+y(t)^2+y(t-1)^2} + u(t) + 2\,u(t-1)\ . \tag{4-6}$$

From the following plots we see that in some cases the output of the system is able to track the desired reference signal. However, in several runs, the nonminimum phase system does not perform as desired and it might become unstable. The extension of a neural network controller to this class of systems is still an open problem in the control system literature.

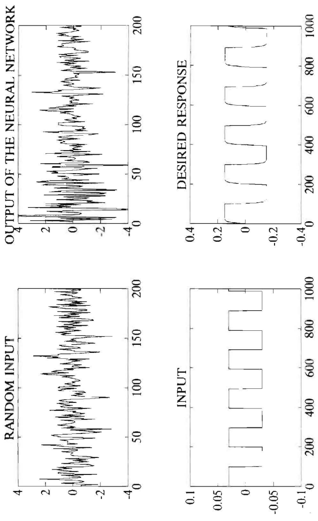

Figure 19. The Training Input, the Resulting Output, the External Input and the Desired Response for the Nonlinear Nonminimum Phase Unknown System

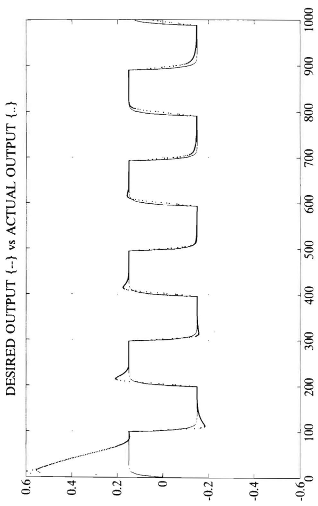

Figure 20. The Output of the Controlled System Compared To That of the Reference Model for the Nonlinear Nonminimum Phase System

C. IMPLEMENTATION OF THE BNN SOFTWARE SIMULATOR

For this thesis research, a software BNN simulator was developed. Until neural network hardware systems or neurocomputers become commonly (and economically) available, most researchers will work with software simulators for neural networks. The software approach offers the full flexibity for development, allowing the user to exercise and experiment freely with the various features of the neural network [Ref.1]. The main drawback of this approach is the slow learning process of the simulator.

By using the appropriate MATLAB[2] codes, the following functions have been developed and used in simulations:

⋆ INITIAL 2 : This function initializes the BNN. It takes as input variables the parameters describing the number of inputs and number of outputs, number of neurons in the hidden layer(s). It yields as outputs the initial conditions of the weights and biases of the network.

⋆ NET 2: This function sets up the data structure for a 2 layer BNN. It takes as input variables a parameter describing the number of inputs and neurons in the hidden and output layer. It also takes as another input variable a parameter specifying the spreading range of the biased inputs.

⋆ BP 2: This function backpropagates the output errors to the input layer. It takes as input variables the parameter describing the weights and biases, and the random input, learning rate and the actual input. It yields updated weights and biases.

⋆ SIGMOID : This function introduces the nonlinearity to the neural network.

⋆ DSIG : This function takes the derivative of the sigmoid function.

[2]MATLAB is a registered trademark of the Math Works, Inc.

These functions can be all implemented recursively, and imbedded in any iterative loop. The source codes of these functions used in this thesis research are provided in Appendix C.

V. OBSERVATIONS, DISCUSSIONS AND CONCLUSIONS

A. SUMMARY

A Direct Model Reference Adaptive Controller (DMRAC) for LTI systems was developed and applied to both minimum phase and nonminimum phase LTI systems for pole placement. The DMRAC for LTI systems was then extended to nonlinear systems by training a BNN to emulate a suitable nonlinear regression form that describes the system under consideration.

Later, the control of four general classes of unknown nonlinear systems, modelled in discrete-time, using the BNN DMRAC was considered. Lastly, an experiment for unknown nonlinear nonminimum phase systems was conducted.

B. OBSERVATIONS, DISCUSSIONS AND CONCLUSIONS

A DMRAC designed with a least-squares estimator, assuming the system is LTI, failed to work for most nonlinear systems. Hence, the BNN DMRAC is an effective technique in controlling nonlinear systems where the conventional technique fails.

Some experiments conducted showed that the BNN DMRAC performs its control function reasonably well for various types of inputs. However, for inputs with high frequency components (with respect to the sampling rate), the controlled system became quite oscillatory, so that it would not be able to track the reference model properly. In addition, the controller would sometimes saturate during training and therefore it failed to control the system [Ref.1]. It is suggested that the solution to this problem is to increase the sampling rate with or without increasing the number of neurons and hidden layers used in the BNN. Off-line training with more appropriate data (i.e. training signals containing similar frequency characteristics as the actual signals experienced by the controlled system), and adjusting the learning parameters also helps to improve the tracking performance and avoid saturation [Ref.1]. Unfortunately, there is still no general rule to help select the most appropriate learning parameters. Hence, a great deal of experimentation is usually required.

The BNN DMRAC designed for unknown nonminimum phase systems is not promising compared with their minimum phase counterpart. A nonminimum phase system is usually hard to stabilize adaptively by a direct approach.

C. FURTHER RESEARCH AND DEVELOPMENT

In this thesis research, the emphasis was to develop a structure for direct adaptive control of certain classes of unknown nonlinear systems using the BNN. The results of the experiments clearly showed that the BNN DMRAC is very effective in controlling unknown nonlinear systems. On the other hand, nonminimum phase systems at the moment do not yield satisfactory results and need more research efforts.

Once more, in the design of the BNN, the selection of the number of layers, neurons, the type of nonlinear transformation, etc., is still at the discretion of the designer. This area definitely needs further research.

APPENDIX A

DMRAC DESIGN FOR UNKNOWN MINIMUM PHASE LTI SYSTEMS

In this Appendix we address the problem of designing an adaptive controller for a LTI system based on the algebraic approach. The unknown LTI system to be controlled is described by the following polynomial operators:

$$A(q) \quad q^2 - 0.2q + 0.9, \qquad B(q) = 3q. \tag{A-1}$$

In order to track the reference model

$$y_m(t) - 0.8y_m(t-1) = v(t-1), \tag{A-2}$$

the desired closed loop polynomial, $P^*(q)$ and the characteristic polynomial of the observer, $\alpha(q)$ are chosen arbitrarily and with roots in the stable region as

$$P^*(q) = \frac{1}{3}3q(q - 0.8) = q(q - 0.8), \qquad \alpha(q) = q^2 - q + 0.25. \tag{A-3}$$

Using the algorithm in Chapter 1 Section B, the Diophantine equation can be formed as follows:

$$(h_1 q + h_2)(q^2 - 0.2q + 0.9) + (k_1 q + k_2)3q = (q^2 - q + 0.25)\,[q - 0.2q + 0.9 - q(q - 0.8)] \tag{A-4}$$

or

$$(h_1 q + h_2)(q^2 - 0.2q + 0.9) + (k_1 q + k_2)3q = 0.6q^3 + 0.3q^2 - 0.75q + 0.225. \quad \text{(A-5)}$$

The MATLAB simulation for the above system yields the following steady-state gains:

$$\begin{aligned} h_1 &= 0.61234, \\ h_2 &= 0.24935, \\ k_1 &= 0.05678, \\ k_2 &= -0.42982. \end{aligned}$$

These results can be checked by using the Sylvester matrix formulations for this system which in this case is given by

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.2 & 1 & 3 & 0 \\ 0.9 & -0.2 & 0 & 3 \\ 0 & 0.9 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.3 \\ -0.75 \\ 0.225 \end{bmatrix}. \quad \text{(A-6)}$$

By solving equation (A-6) we obtain the following values:

$$\begin{aligned} h_1 &= 0.6, \\ h_2 &= 0.25, \\ k_1 &= 0.0567, \\ k_2 &= -0.4133. \end{aligned}$$

Hence, the estimated gains needed to control the system are found to be very close to the steady-state gains obtained from the Sylvester matrix equations.

The simulation results of the experiment for the system above are displayed in Figures (A-1) thru (A-3).

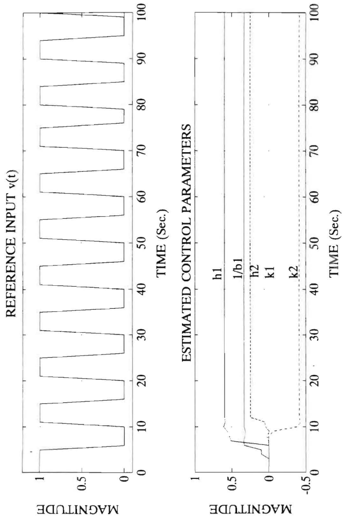

Figure A-1 . Reference Input and Estimated Control Parameters for LTI Minimum Phase Unknown System

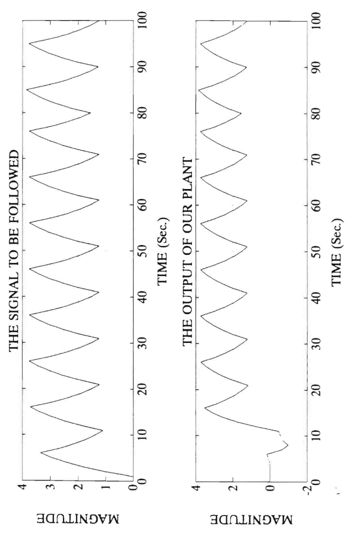

Figure A-2. The Signal To Be Followed and the Actual Output For LTI Minimum Phase Unknown System

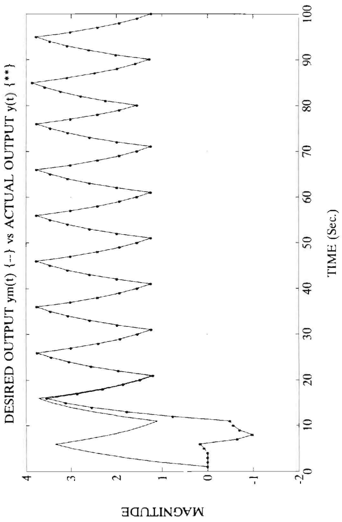

Figure A-3 . Desired Output vs Actual Output For LTI Minimum Phase Unknown System

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% A DMRAC DESIGN FOR UNKNOWN MINIMUM PHASE LTI SYSTEMS
%
% Designing a controller  using recursive least squares estimates
%   of the compensator parameters.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
! del *.met
clear,clg,clc

% Generating the reference input signal
t=0:100;
v=square(t/(0.5*pi));
kmax=length(v)+1;
for i=1:kmax-1
   if v(i)<=0
      v(i)=0;
   end
end

% Initialization of the parameters
u=zeros(1,kmax);
ubar=zeros(1,kmax);
y=zeros(1,kmax);
ybar=zeros(1,kmax);
ym=zeros(1,kmax);
th=zeros(5,kmax);
th(5,4)=0.1;
P=10000*eye(5);

% Generating the reference model
for j=2:kmax-1
   ym(j)=0.8*ym(j-1)+v(j-1);
end

% Designing the controller
for k=4:kmax-1
   h1=th(1,k);
   h2=th(2,k);
   k1=th(3,k);
   k2=th(4,k);
   ib1=th(5,k);
   u(k)=(h1+1)*u(k-1)+(h2-0.25)*u(k-2)+k1*y(k-1)+k2*y(k-2)+...
       (v(k)-v(k-1)+0.25*v(k-2))*ib1;
   ubar(k)=u(k-1)-u(k-2)+0.25*u(k-3);
   y(k)=0.2*y(k-1)-0.9*y(k-2)+3*u(k-1);
   ybar(k)=y(k)-1.8*y(k-1)+1.05*y(k-2)-0.2*y(k-3);
   phit(k,:)=[u(k-2) u(k-3) y(k-2) y(k-3) ybar(k)];
   K(:,k)=(P*phit(k,:)')/(1+phit(k,:)*P*phit(k,:)');
   P=P-(P*phit(k,:)'*phit(k,:)*P)/(1+phit(k,:)*P*phit(k,:)');
   th(:,k+1)=th(:,k)+K(:,k)*(ubar(k)-phit(k,:)*th(:,k));
end

% Plotting the results
subplot(211)
axis([0 100 -0.1 1.2]);
plot(v(1,1:100));
title('REFERENCE INPUT v(t)');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid,pause
axis
clear t

t=1:100;
subplot(212)

plot(t,th(1,1:100),t,th(2,1:100),t,th(3,1:100),t,th(4,1:100),...
    t,th(5,1:100));
title('ESTIMATED CONTROL PARAMETERS');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid
gtext('h1');
gtext('1/b1');
gtext('h2');
gtext('k1');
gtext('k2');
pause
meta 1
clg
subplot(211)
plot(ym(1,1:100));
title('THE SIGNAL TO BE FOLLOWED');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
```

```
grid,pause
subplot(212)
plot(y(1,1:100));
title('THE OUTPUT OF OUR PLANT');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid,pause
meta 2
clg
axis([0 100 -2 4])
plot(ym(1,1:100));
hold on
plot(y(1,1:100),'*');
plot(y(1,1:100));
title('DESIRED OUTPUT ym(t) {--} vs ACTUAL OUTPUT y(t) {**}');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid
hold off
meta 3
axis('normal');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

APPENDIX B

DMRAC DESIGN FOR UNKNOWN NONMINIMUM PHASE LTI SYSTEMS

In this Appendix we address the problem of designing an adaptive controller for a nonminimum phase LTI system based on the algebraic approach. The unknown LTI system to be controlled is described by the following polynomials:

$$\begin{aligned} A(q) &= q^2 + q + 1, \\ B(q) &= q + 2, \end{aligned} \tag{B-1}$$

where $A(q)$ is stable and monic, and $B(q)$ is monic and causes the system to be nonminimum phase. In order to track the reference model

$$y_m(t) - 0.8y_m(t-1) = v(t-1), \tag{B-2}$$

the desired closed loop polynomial, $P^*(q)$ and the characteristic polynomial of the observer, $\alpha(q)$ are chosen arbitrarily as

$$\begin{aligned} P^*(q) &= (q - 0.8)^2, \\ \alpha(q) &= q^2 - q + 0.25. \end{aligned} \tag{B-3}$$

Using the algorithm in Chapter 1 Section C, the Diophantine equation can be formed as follows: (B-4)

$$(h_1q + h_2)(q^2 + q + 1) + (k_1q + k_2)(q + 2) \quad (q^2 - q + 0.25)[q^2 + q + 1 - (q^2 - 1.6q + 0.64)],$$

or,

$$(h_1q + h_2)(q^2 + q + 1) + (k_1q + k_2)(q + 2) = 1.8q^3 - 0.96q^2 - 0.39q + 0.21. \qquad \text{(B-5)}$$

Also, $P^*(q)\alpha(q)$ is obtained as

$$(q^2 - q + 0.25)(q^2 - 0.8q + 0.16) = q^4 - 1.8q^3 + 1.21q^2 - 0.36q + 0.04. \qquad \text{(B-6)}$$

The MATLAB simulation for the above system yields the following gains,

$$\begin{aligned} h_1 &= 1.7927 \\ h_2 &= -2.121 \\ k_1 &= -0.6184 \\ k_2 &= 1.1666 \\ b_1 &= 0.0000 \\ b_2 &= -0.6642 \\ c_1 &= 0.6642 \\ c_2 &= 0.3358. \end{aligned}$$

where b_1, b_2, c_1 and c_2 are the coefficients of the polynomials $b(q)$ and $c(q)$ respectively.

These results can be checked with Sylvester matrix solution for this system which can be written as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 2 & 1 \\ 0 & 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} 1.8 \\ -0.96 \\ -0.39 \\ 0.21 \end{bmatrix}. \qquad \text{(B-7)}$$

By solving equation (B-7) we obtain the following values:

$$\begin{aligned} h_1 &= 1.8, \\ h_2 &= -2.15, \\ k_1 &= -0.61, \\ k_2 &= 1.18. \end{aligned}$$

Hence, the estimated gains needed to control the system are found to be very close to the gains obtained from Sylvester matrix equations. In fact, only h_1, h_2, k_1 and k_2 are needed in the control phase while b_1, b_2, c_1 and c_2 are needed only in the estimation phase of the entire design process.

The simulation results of the experiment for the system above are displayed in Figures (B-1) thru (B-3).

The experiment then, is conducted for an unknown nonminimum phase and unstable LTI system. It is observed that the algorithm first stabilizes the system with the adaptive pole placement process and then provides the necessary input to track the desired reference signal. The results of this experiment for the nonminimum phase and unstable system are displayed in Figures (B-4) thru (B-6). For the experiment $A(q)$ is chosen to be as follows:

$$A(q) = q^2 + q + 2. \tag{B-8}$$

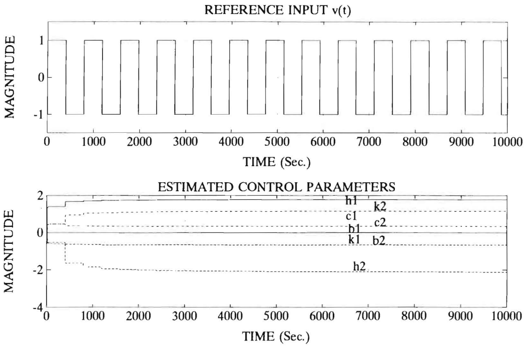

Figure B-1 . Reference Input and Estimated Control Parameters for LTI Nonminimum Phase Unknown Stable System

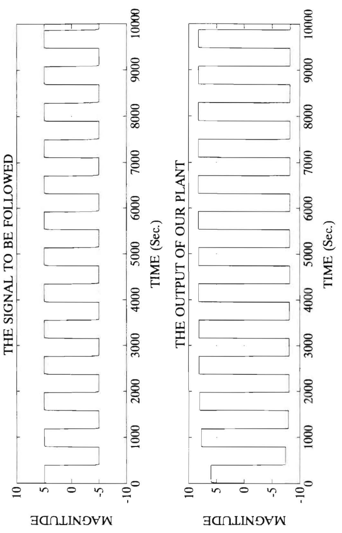

Figure B-2 . The Signal To Be Followed and the Actual Output For LTI Nonminmum Phase Unknown Stable System.

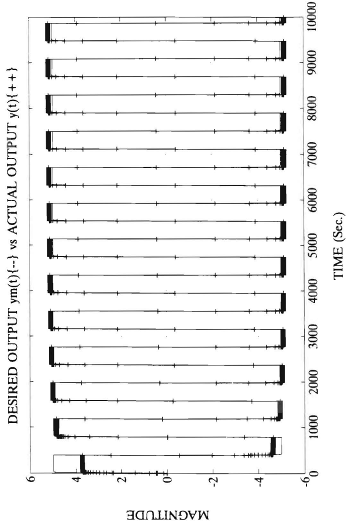

Figure B-3 .Desired Output vs Actual Output For LTI Nonminimum Phase Unknown Stable System

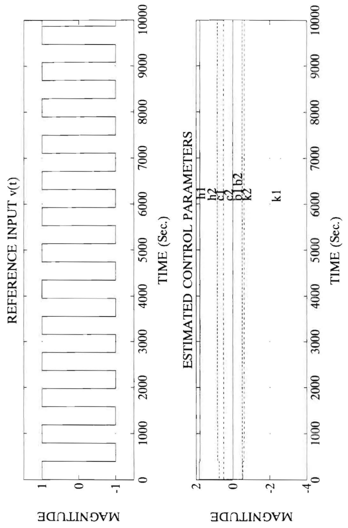

Figure B-4. Reference Input and Estimated Control Parameters for LTI Nonminimum Phase Unknown Unstable System.

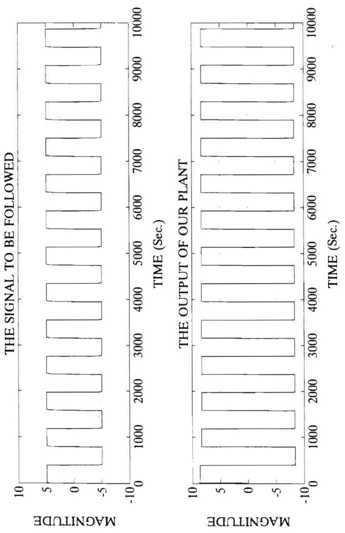

Figure B-5. The Signal To Be Followed and the Actual Output For LTI Nonminimum Phase Unknown Unstable System.

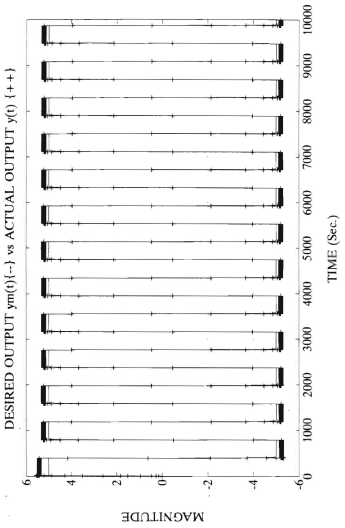

Figure B-6. Desired Output vs Actual Output For LTI Nonminimum Phase Unknown Unstable System.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%  A DMRAC DESIGN FOR UNKNOWN NONMINIMUM PHASE LTI SYSTEMS
%
%  Designing a controller  using recursive least squares estimates
%    of the compensator parameters.
%
%  Experimenting with a stable system
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

! del *.met
clear,clg,clc

% Generating the reference input signal

w=10000;
t=0:w;
v=square(t/(40*pi));

% Initialization of the parameters

kmax=length(v)+1;
u=zeros(1,kmax);
ubar=zeros(1,kmax);
uF=zeros(1,kmax);
uFmin1=zeros(1,kmax);
y=zeros(1,kmax);
yF=zeros(1,kmax);
yFmin1=zeros(1,kmax);
ym=zeros(1,kmax);
th=zeros(8,kmax);
P=w*eye(8);

% Generating the reference model

for j=2:kmax-1
   ym(j)=0.8*ym(j-1)+v(j-1);

end
```

```
% Designing the controller

for k=6:kmax-1
   h1=th(1,k);
   h2=th(2,k);
   k1=th(3,k);
   k2=th(4,k);
   b1=th(5,k);
   b2=th(6,k);
   c1=th(7,k);
   c2=th(8,k);
   u(k)=(h1+1)*u(k-1)+(h2-0.25)*u(k-2)+k1*y(k-1)+k2*y(k-2)+v(k)-v(k-1)+0.25*v(k-2);
   ubar(k)=u(k-2)-u(k-3)+0.25*u(k-4);
   uF(k)=u(k)-1.8*u(k-1)+1.21*u(k-2)-0.36*u(k-3)+0.04*u(k-4);
   uFmin1(k)=u(k-1)-1.8*u(k-2)+1.21*u(k-3)-0.36*u(k-4)+0.04*u(k-5);
   y(k)=-y(k-1)-y(k-2)+u(k-1)+2*u(k-2);
   yF(k)=y(k)-1.8*y(k-1)+1.21*y(k-2)-0.36*y(k-3)+0.04*y(k-4);
   yFmin1(k)=y(k-1)-1.8*y(k-2)+1.21*y(k-3)-0.36*y(k-4)+0.04*y(k-5);
   phit(k,:)=[u(k-3) u(k-4) y(k-3) y(k-4) uF(k) uFmin1(k) yF(k) yFmin1(k)];
   K(:,k)=(P*phit(k,:)')/(1+phit(k,:)*P*phit(k,:)');
   P=P-(P*phit(k,:)'*phit(k,:)*P)/(1+phit(k,:)*P*phit(k,:)');
   th(:,k+1)=th(:,k)+K(:,k)*(ubar(k)-phit(k,:)*th(:,k));
end

% Plotting the results
subplot(211)
axis([0 w -1.5 1.5]);
plot(v(1,1:w));
title('REFERENCE INPUT v(t)');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid,pause
axis
clear t
subplot(212)
t=1:w;
plot(t,th(1,1:w),t,th(2,1:w),t,th(3,1:w),t,th(4,1:w),...
     t,th(5,1:w),t,th(6,1:w),t,th(7,1:w),t,th(8,1:w));
title('ESTIMATED CONTROL PARAMETERS');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid;
gtext('h1');
gtext('k2');
gtext('c1');
gtext('c2');
gtext('b1');
gtext('k1');
gtext('b2');
gtext('h2');
pause
meta 1b

clg
subplot(211)
axis([0 w -10 10]);
plot(ym(1,7:w));
title('THE SIGNAL TO BE FOLLOWED');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid,pause
subplot(212)
plot(y(1,1:w));
title('THE OUTPUT OF OUR PLANT');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid,pause
meta 2b

clg
axis([0 w -6 6]);
plot(ym(1,1:w));
hold on
plot(0.625*y(1,1:w),'+');
plot(0.625*y(1,1:w));
title('DESIRED OUTPUT ym(t){--} vs ACTUAL OUTPUT y(t){++}');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid
hold off
meta 3b
axis

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%   A DMRAC DESIGN FOR UNKNOWN NONMINIMUM PHASE LTI SYSTEMS
%
%   Designing a controller  using recursive least squares estimates
%     of the compensator parameters.
%
%   Experimenting with a unstable system
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
! del *.met
clear,clg,clc

% Generating the reference input signal
w=10000;
t=0:w;
v=square(t/(40*pi));
% Initialization of the parameters
kmax=length(v)+1;
u=zeros(1,kmax);
ubar=zeros(1,kmax);
uF=zeros(1,kmax);
uFmin1=zeros(1,kmax);
y=zeros(1,kmax);
yF=zeros(1,kmax);
yFmin1=zeros(1,kmax);
ym=zeros(1,kmax);
th=zeros(8,kmax);
P=w*eye(8);
% Generating the reference model
for j=2:kmax-1
   ym(j)=0.8*ym(j-1)+v(j-1);
end

% Designing the controller
for k=6:kmax-1
   h1=th(1,k);h2=th(2,k);k1=th(3,k); k2=th(4,k); b1=th(5,k);b2=th(6,k);
   c1=th(7,k);c2=th(8,k);
   u(k)=(h1+1)*u(k-1)+(h2-0.25)*u(k-2)+k1*y(k-1)+k2*y(k-2)+v(k)-v(k-1)+0.25*v(k-2);
   ubar(k)=u(k-2)-u(k-3)+0.25*u(k-4);
   uF(k)=u(k)-1.8*u(k-1)+1.21*u(k-2)-0.36*u(k-3)+0.04*u(k-4);
   uFmin1(k)=u(k-1)-1.8*u(k-2)+1.21*u(k-3)-0.36*u(k-4)+0.04*u(k-5);
   y(k)=-y(k-1)-2*y(k-2)+u(k-1)+2*u(k-2);
   yF(k)=y(k)-1.8*y(k-1)+1.21*y(k-2)-0.36*y(k-3)+0.04*y(k-4);
   yFmin1(k)=y(k-1)-1.8*y(k-2)+1.21*y(k-3)-0.36*y(k-4)+0.04*y(k-5);
   phit(k,:)=[u(k-3) u(k-4) y(k-3) y(k-4) uF(k) uFmin1(k) yF(k) yFmin1(k)];
   K(:,k)=(P*phit(k,:)')/(1+phit(k,:)*P*phit(k,:)');
   P=P-(P*phit(k,:)'*phit(k,:)*P)/(1+phit(k,:)*P*phit(k,:)');
   th(:,k+1)=th(:,k)+K(:,k)*(ubar(k)-phit(k,:)*th(:,k));

end

% Plotting the results
subplot(211)
axis([0 w -1.5 1.5]);
plot(v(1,1:w));
title('REFERENCE INPUT v(t)');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid,pause
axis
clear t
subplot(212)
t=1:w;
plot(t,th(1,1:w),t,th(2,1:w),t,th(3,1:w),t,th(4,1:w),...
    t,th(5,1:w),t,th(6,1:w),t,th(7,1:w),t,th(8,1:w));
title('ESTIMATED CONTROL PARAMETERS');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid;
gtext('h1');gtext('h2');gtext('c1');gtext('c2');
gtext('b1');gtext('b2');gtext('k2');gtext('k1');
pause
meta 1b
clg
subplot(211)
axis([0 w -10 10]);
plot(ym(1,7:w));
title('THE SIGNAL TO BE FOLLOWED');
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid,pause
subplot(212)
plot(y(1,1:w));
title('THE OUTPUT OF OUR PLANT');
```

```
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid,pause
meta 2b
clg
axis([0 w -6 6]);
plot(ym(1,1:w));
hold on
plot(0.625*y(1,1:w),'+');
plot(0.625*y(1,1:w));
title('DESIRED OUTPUT ym(t){--} vs ACTUAL OUTPUT y(t)'{++});
xlabel('TIME (Sec.)');
ylabel('MAGNITUDE');
grid
hold off
meta 3b
axis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

APPENDIX C

BNN SOFTWARE SIMULATOR

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% SOFTWARE SIMULATOR FOR BNN
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! del *.met
clear,clg,clc
rand('normal');

% Random input to train the system.
num=15000;
u1=rand(1,num);
k=0:num;

% Actual input to the system.
v=0.04*sin(k/(9*pi))-0.03*cos(k/(5*pi));
kmax=length(v);

% Calculating the reference output.
ym=zeros(1,kmax-1);
y=zeros(1,kmax-1);
utm1=zeros(1,kmax-1);
mu=0.04;
for i=2:kmax-1
   ym(i)=0.8*ym(i-1)+v(i-1);
end
for j=2:kmax-1
   %model1
   y(j+1)=0.5*y(j)+0.4*y(j-1)+u1(j)^3+0.3*u1(j)^2-0.1*u1(j);
   %model2 y(j+1)=(y(j)*y(j-1)*(y(j)+2.5)/(1+y(j)^2+y(j-1)^2))+u1(j);
   %model3        y(j+1)=y(j)/(1+y(j)^2)+u1(j)^3;
   %model4 y(j+1)=(y(j)*y(j-1)*y(j-2)*u1(j-1)*(y(j-2)+1)+u1(j))/(1+y(j-1)^2+y(j-2)^2);
end
% Initialization of the neural network.
```

```
[w3,w2,b2,w1,b1]=initial2(1,18,18,5);

% Training the neural network.
for t=4:kmax-1
   u=[u1(t-2) u1(t-3) y(t-2) y(t-3) (y(t)-0.8*y(t-1))]';
   utm1(t-1)=net2(w3,w2,b2,w1,b1,u);
   [w3,w2,b2,w1,b1]=bp2(w3,w2,b2,w1,b1,u,u1(t-1),mu);
end

% Plots of the desired signals.
subplot(221)
axis([0 200 -4 4]);
plot(u1(1:200));
title('RANDOM INPUT');
grid

subplot(222)
axis([0 200 -4 4]);
plot(utm1(1:200));
title('OUTPUT OF THE NEURAL NETWORK');
grid

subplot(223)
axis([0 1000 -0.1 0.1]);
plot(v(1:1000));
title('INPUT');
grid

subplot(224)
axis([0 1000 -0.4 0.4]);
plot(ym(1:1000));
title('DESIRED RESPONSE');
grid,pause
meta h

% Controlling the plant with a neural identifier-controller.
clg
clear u; clear y;clear t;
u=zeros(1,10000);
y=zeros(1,10000);
i1=zeros(5,10000);
i2=zeros(5,10000);
```

```
for t=4:9999;
   i1(:,t)=[u(t-1) u(t-2) y(t-1) y(t-2) v(t)]';
   u(t)=net2(w3,w2,b2,w1,b1,i1(:,t));
   y(t)=(y(t-1)*y(t-2)/(1+y(t-1)^2+y(t-2)^2))+u(t-1);
   i2(:,t)=[u(t-2) u(t-3) y(t-2) y(t-3) (y(t)-0.8*y(t-1))]';
   [w3,w2,b2,w1,b1]=bp2(w3,w2,b2,w1,b1,i2(:,t),u(t-1),mu);
end

% Ploting the results.

subplot(111)
n=1:999;
axis([0 1000 -0.6 0.6]);
plot(n,ym(1,1:999),n,y(1,1:999),'.');
title('DESIRED OUTPUT {--} vs ACTUAL OUTPUT {..}');
grid,pause
meta hh

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%
%    BNN DMRAC WITH UNKNOWN NONLINEAR NONMINIMUM
%      PHASE SYSTEM
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
! del *.met

clear,clg,clc
rand('normal');
% Random input to train the system.
num=1000;
u1=rand(1,500000);
k=0:num;
% Actual input to the system.
v=0.03*square(k/(10*pi));
%v=0.04*sin(k/(9*pi))-0.03*cos(k/(5*pi));
kmax=length(v);
% Calculating the reference output.
ym=zeros(1,kmax-1);
```

```
y=zeros(1,kmax-1);
utm1=zeros(1,kmax-1);
mu=0.099;
for i=2:kmax-1
   ym(i)=0.8*ym(i-1)+v(i-1);
end
for j=2:kmax-1
%model2
y(j+1)=(y(j)*y(j-1)*(y(j)+2.5)/(1+y(j)^2+y(j-1)^2))+u1(j)+2*u1(j-1);
%model3          y(j+1)=y(j)/(1+y(j)^2)+u1(j)^3;
%model4 y(j+1)=(y(j)*y(j-1)*y(j-2)*u1(j-1)*(y(j-2)+1)+u1(j))/(1+y(j-1)^2+y(j-2)^2);
%model1 y(j+1)=0.5*y(j)+0.4*y(j-1)+u1(j)^3+0.3*u1(j)^2-0.1*u1(j);
end

% Initialization of the neural network.

[w3,w2,b2,w1,b1]=initial2(1,20,20,5);

% Training the neural network.

for t=4:kmax-1
   u=[u1(t-2) u1(t-3) y(t-2) y(t-3) (y(t)-0.8*y(t-1))]';
   utm1(t-1)=net2(w3,w2,b2,w1,b1,u);
   [w3,w2,b2,w1,b1]=bp2(w3,w2,b2,w1,b1,u,u1(t-1),mu);
end

% Plots of the desired signals.

subplot(221)
axis([0 200 -4 4]);
plot(u1(1:200));
title('RANDOM INPUT');
grid

subplot(222)
axis([0 200 -4 4]);
plot(utm1(1:200));
title('OUTPUT OF THE NEURAL NETWORK');
grid

subplot(223)
axis([0 1000 -0.1 0.1]);
plot(v(1:1000));
```

```
title('INPUT');
grid

subplot(224)
axis([0 1000 -0.4 0.4]);
plot(ym(1:1000));
title('DESIRED RESPONSE');
grid,pause
meta h

% Controlling the plant with a neural identifier-controller.

clg
clear u
clear y
clear t
u=zeros(1,num);
y=zeros(1,num);
i1=zeros(5,num);
i2=zeros(5,num);

for t=4:999;
   i1(:,t)=[u(t-1) u(t-2) y(t-1) y(t-2) v(t)]';
   u(t)=net2(w3,w2,b2,w1,b1,i1(:,t));
   y(t)=(y(t-1)*y(t-2)/(1+y(t-1)^2+y(t-2)^2))+u(t-1);
   i2(:,t)=[u(t-2) u(t-3) y(t-2) y(t-3) (y(t)-0.8*y(t-1))]';
   [w3,w2,b2,w1,b1]=bp2(w3,w2,b2,w1,b1,i2(:,t),u(t-1),mu);
end

% Ploting the results.

subplot(111)
n=1:999;
axis([0 1000 -0.6 0.6]);
plot(n,ym(1,1:999),n,y(1,1:999),'.');
title('DESIRED OUTPUT {--} vs ACTUAL OUTPUT {..}');
grid,pause
meta hh


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [W3,W2,b2,W1,b1]=initial2(ny,M2,M1,nu)
%
%     initializes the neural network
%
%     ny=number of outputs, nu=number of inputs,
%     M2,M1=number of neurons in hidden layers
%
rand('normal')
W3=rand(ny,M2);
W2=rand(M2,M1);          b2=rand(M2,1);
W1=ones(M1,nu);          b1=rand(M1,1);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function y=net2(W3,W2,b2,W1,b1,u)
%
%    sets up the data structure for 2 layer neural network
%    u=input, y=output
%    W=weights, b=biases
%
v=sigmoid(W1*u+b1);
z=sigmoid(W2*v+b2);
y=W3*z;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [W3,W2,b2,W1,b1]=bp2(W3,W2,b2,W1,b1,u,yd,mu)
%
%    updates the weights and the biases of  a 2 layer neural network with backpropagation
%
%    u is the input to the network,
%    yd  is the desired reference model to track
%   mu is the learning rate

vb=W1*u+b1;  v=sigmoid(vb);
zb=W2*v+b2;  z=sigmoid(zb);
```

```
y=W3*z;
e=y-yd;

d3=e;
d2=diag(W3'*d3)*dsig(zb);
d1=diag(W2'*d2)*dsig(vb);

W3=W3-mu*d3*z';
W2=W2-mu*d2*v';          b2=b2-mu*d2;
W1=W1-mu*d1*u';          b1=b1-mu*d1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function y=sigmoid(x)
%
%  introduces the nonlinearity to neural network
%
x=min(x,100); x=max(x,-100);
y= (1-exp(-x))./(1+exp(-x));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function d=dsig(x)
%
%   takes the derivative of sigmoid
%
%   x is a vector
%
x=min(x,100);  x=max(x,-100);
temp=exp(-x);
d=temp ./ (1+2*temp + temp .* temp);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

LIST OF REFERENCES

1. Teo, C.H., " Back-Propagation Neural Networks in Adaptive Control of Unknown Nonlinear Systems," Master's Thesis, Naval Postgraduate School, Monterey, California, 1991.

2. Cristi, R., Notes for EC4360 (System Identification), Naval Postgraduate School, 1993 (unpublished).

3. Demuth, H. and Beale, M., *Neural Network Toolbox User's Guide*, The Math Works Inc., April 1993.

4. Lipmann, R.P., " An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp.4-22, April 1987.

5. Narendra, K.S., "Adaptive Control Using Neural Networks," in *Neural Networks for Control*, pp.115-142, MIT Press, Cambridge, Massachussets,1990.

6. Narendra, K.S. and Parthasarathy, K., " Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, Vol.1, No.1, March 1990.

7. Elliot, H., Cristi, R., and Das, M., " Global Stability of Adaptive Pole Placement Algorithms," *IEEE Transactions on Automatic Control*, Vol . AC-30, No. 4, April 1985.

DISTRIBUTION LIST

		No.Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library Code 52 Naval Postgraduate School Monterey, CA 93943-5101	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
4.	Professor R. Cristi, Code EC/Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
5.	Professor R. Hippenstiel, Code EC/Hi Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
6.	Professor R.Hutchins,Code EC/Hu Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
7.	Deniz Kuvvetleri Komutanlığı Personel Daire Başkanlığı Bakanlıklar, Ankara, TURKEY	1
8.	Deniz Harp Okulu Komutanlığı Tuzla, İstanbul, TURKEY	2

9.	Gölcük Tersanesi Komutanlığı Gölcük, Kocaeli, TURKEY	1
10.	Taşkızak Tersanesi Komutanlığı Taşkızak, İstanbul, TURKEY	1
11.	Charito Burnett SATO Leisure Travel Counselor P.O. Box # 8718 Monterey,CA 93943	1
12.	Alpay Cakarcan 108-27 Sok. No: 51/13, Ozlem apt. Camlik, Hatay, Izmir, TURKEY	1