INVESTIGATION OF DIFFERENT APPROACHES FOR IDENTIFICATION AND CONTROL OF COMPLEX AND NONLINEAR SYSTEMS USING NEURAL NETWORKS

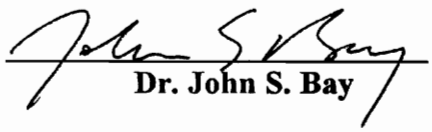
By

Nishith D. Tripathi

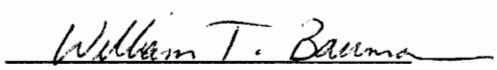
Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE
in
Electrical Engineering

APPROVED:

Dr. Hugh F. VanLandingham, Chairman



Dr. John S. Bay



Dr. William T. Baumann

November, 1994
Blacksburg, Virginia

INVESTIGATION OF DIFFERENT APPROACHES FOR IDENTIFICATION AND CONTROL OF COMPLEX AND NONLINEAR SYSTEMS USING NEURAL NETWORKS

by

Nishith D. Tripathi
Dr. Hugh F. VanLandingham, Chairman
The Bradley Department of Electrical Engineering

ABSTRACT

System identification deals with the problem of building mathematical models of dynamical systems based on observed data from the systems. Most of the conventional techniques of system identification, in general, require some amount of *a priori* knowledge about the structure of the systems. Also, they are only useful either with linear or linearized systems. There are numerous control principles working nicely in industry. However, they are less effective for MIMO systems or complex nonlinear systems. The need to control, in a better way, increasingly complex dynamical systems under significant uncertainty has made the need for new methods quite apparent. This thesis investigates different approaches for *identification* and *control* of *complex nonlinear* systems using neural networks. For system identification and control, ANN properties of generalization and their capability of extracting complex relationships among inputs presented to them are useful. Two different techniques, called *whole region method* (WRM) and the *separate regions method* (SRM) technique, have been developed and applied to two classes of nonlinear systems. Different connectionist control techniques such as adaptive control and neuro-PID control have been developed and applied to the robotic manipulators.

Acknowledgments

I express my gratitude to Dr. H. F. VanLandingham for his incessant guidance and for being the prime mover of my thesis research. Thanks are also due to Dr. J. S. Bay for his help in control simulations, Dr. W. T. Baumann for assisting me in certain aspect of system identification, and Dr. K. Ramu for his inspiration. I would like to thank EE Department for providing workstation facilities.

I sincerely appreciate and acknowledge each kind of support provided by my parents (Mr. Dhananjay C. Tripathi and Mrs. Damini D. Tripathi).

Nishith D. Tripathi

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

System identification deals with the problem of building mathematical models of dynamical systems based on observed data from the systems. System identification is a tool for many problems in science and engineering. The value of the tool has been apparent from the many applications in diverse fields. Successful system identification applications have been reported from such different fields as process control, biomedicine, seismology, environmental systems, aircraft dynamics, macro economy and signal processing, to name a few. Most of the conventional techniques of system identification, in general, require some amount of a priori knowledge about the structure of the systems. There are two phases for system identification using conventional methods. The first phase is the selection of a model which represents the class of systems to be identified. The second phase is the estimation of the parameters for the model chosen in the first phase. There are several models available to choose from in the first phase. Models for linear time-invariant (LTI) systems are included in transfer functions models (such as ARX models, ARMAX models, Output Error models, Box-Jenkins models etc.) and state-space models. Models for linear time-varying systems include state-space models. Models for nonlinear systems include linear regression and nonlinear state-space models. After a particular model is chosen, we need to estimate the associated model parameters. Two general techniques available for parameter estimation are prediction error identification (such as least-squares method) and correlation methods (such as the instrumental variables method). There also exists a conventional method, related to "pseudo observability

indices" (POI), which does not require any structural information about the system, but its scope is limited to linear systems. The conventional techniques for system identification, such as the method of POI, are very effective. However, they are only useful either with linear systems (or nonlinear systems linearized around an operating point) or nonlinear systems whose structure is known. Such structural information is very difficult to obtain in the case of complex nonlinear systems.

There are numerous control principles working nicely in industry. Conventional control techniques include frequency-response methods and time-domain methods. They are well-suited to linear SISO systems and quite successful in satisfying a given performance criterion. However, they are less effective for MIMO systems or complex nonlinear systems. Modern control theory was developed to fill this deficit. Optimal control theory and adaptive control can be viewed as techniques representative of modern control theory. Ever-increasing technological demands necessitate the development of innovative approaches to highly challenging control problems. Also, the control of nonlinear systems, whose parameters change with time as well as with different operating conditions, poses a problem. In a nutshell, the need to control, in a better way, increasingly complex dynamical systems under significant uncertainty has led to a reevaluation of conventional methods, and has made the need for new methods quite apparent.

This thesis investigates different approaches for ***identification*** and ***control*** of complex *nonlinear* systems using one of the tools of artificial intelligence (AI) - neural networks. There has been resurgence of interest in artificial neural networks (ANNs) recently because of some new ANN topologies and algorithms. They are being applied to

numerous problems which are nearly intractable by conventional techniques. In the case of system identification and control, ANN properties of generalization and their capability of extracting complex relationships among inputs presented to them are useful. Typically, data are presented to the ANN and the ANN is trained using which is called "supervisory" training. The data set used in training the ANN is called a training set and it consists of an input set and a target set. The ANN learns the functional relationship between input vectors and vectors in the target set.

Nonlinear System Identification. Here, the nonlinear system identification problem is viewed as a problem of finding system parameters such that the knowledge of inputs presented to the system is sufficient to determine its outputs. These system parameters need not represent actual parameters of the system as long as the system outputs are predicted correctly. Typically, for system identification, the ANN input set contains system inputs with delays as well as delayed system outputs while the target set contains the corresponding desired system outputs. Hence, after the ANN is trained, it can predict the outputs of the system when information about system inputs is provided. Two different techniques have been developed for *system identification.* One technique, called the ***whole region method*** (**WRM**), uses the input and output data to obtain the model of the system in one operation. In WRM, the ANN has to learn both the linear and nonlinear relationships between inputs and outputs. Another technique, called the ***separate regions method*** (**SRM**), first extracts the linear relationship between inputs and outputs, and, separately, the behavior of the system due to nonlinearity in the system is captured by the neural network. The advantage of SRM is that in several systems, the linear relationship between system inputs and outputs can be accurately obtained by the powerful conventional methods of system identification. Hence, the neural network has to

learn only the nonlinear relationship between inputs and outputs. This simplifies the learning process of neural network. Both these methods can produce good results, with SRM giving somewhat better results than WRM.

Control of Nonlinear Systems. The use of neural networks in control systems can be viewed as a natural step in the evolution of control methodology to meet new challenges. Neural networks represent one of the new approaches. There are several configurations of neural networks which are capable of learning the functional relationship between system inputs and outputs. For control applications, the ANN input set contains system outputs with delays and delayed system inputs and the target set contains system inputs (which are also control inputs) . This can be viewed as a problem of *"inverse system identification"*. Hence, after the ANN is trained, it can predict the inputs to be given to the system so that desired system outputs can be achieved. In open loop control, the ANN, which has been trained off-line, is used as a controller. No information about the actual outputs of the systems is made available to the ANN. The motivation behind this arrangement is that open loop control is the simplest control and may suffice for simple nonlinear systems. However, usually feedback is important in systems, and the second and third approaches use the information of actual outputs to adapt the parameters of the controller to the actual . Second approach, i.e. neuro-PID control, uses the ANN to put the actual values of outputs close to the desired outputs and then the PID controller reduces the error between desired and actual outputs to zero very fast. Thus, PID controller gains can be set accordingly. The motivation behind this idea is that a PID controller alone may not be able to control the system in the whole region of system operation. The ANN can help the PID controller focus on a narrow range of operation. The PID controller uses the actual measurements of system outputs. In the third

approach, i.e. adaptive control, the ANN, which has been trained off-line, changes its parameters based on its performance so that new system behavior or the system behavior uncaptured during off-line training can now be learned on-line.

This thesis demonstrates that the neural network represents one of the excellent tools for the problems of nonlinear system identification and control, especially when effective techniques for these problems do not exist. Two techniques of system identification using ANNs were investigated for two different classes of nonlinear systems. A novel control technique, referred to as neuro-PID control in the thesis, has been developed. The neuro-PID control combines the conventional PID control with the ANN control to achieve better performance by exploiting advantages of the two conceptually different control techniques.

The thesis is organized as follows. Chapter 2 gives a brief introduction to the conventional system identification techniques. In particular, the method of pseudo-observability indices (POI) is described comprehensively since some of its concepts are used in one of the approaches to the system identification in this thesis. Also, chapter 2 introduces ANNs, and describes the basic steps to be followed for system identification using neural networks. Simulation results are presented and evaluated in Chapter 3. Chapter 4 states the control problems undertaken for the thesis and discusses different connectionist control techniques. Chapter 5 presents the simulation results corresponding to the control of systems under consideration. Finally, Chapter 6 discusses the scope of the application of different approaches investigated for system identification and control of nonlinear systems. Future work and possible improvements in the approaches used in the thesis are also suggested in Chapter 6.

Chapter 2
System Identification Using Neural Networks

System identification deals with the problem of building mathematical models of dynamical systems based on observed data from the systems. System identification is a tool for many problems in science and engineering. The value of the tool has been apparent from many applications in diverse fields. Interesting discussion of usefulness of system identification can be found in [24] and [13]. Successful system identification applications have been reported from such different fields as process control [27], biomedicine [22], seismology [52], environmental systems [31], aircraft dynamics [26], macro economy [51] and signal processing [23], to name the few [47]. This chapter provides a brief introduction to system identification and application of ANNs to the system identification. A conventional method called the method of pseudo observability indices (POI) is described. This method does not require any structural information about the system but its scope is limited to linear systems. The concepts of this method are used later in neural network approaches to system identification. The problem of nonlinear system identification can be viewed as a problem of finding the system parameters so that the knowledge of inputs presented to the system suffices in predicting the outputs of the system. These system parameters don't represent actual parameters of the system but they are simply the parameters of the trained neural network. These parameters are the elements of the weight matrices and bias vectors associated with the neural network. The nonlinear systems to be identified are defined. ANNs are introduced. Basic steps for system identification using neural networks are illustrated. These steps are then performed

for the systems under consideration. The use of neural networks for system identification appears in several papers e.g. [43], [54], [62], [67], [66] etc.

2.1 System Identification: Background

Typically, there are three major steps: (i) Collection of Data, (ii) Consideration of the set of models (iii) Determination of the "best" model in the set, and (iv) Model validation. The data might have been obtained by setting up an experiment or by recording the variables during the normal operation of the system. There are two different types of model sets. The model set, whose parameters are basically viewed as vehicles for adjusting the fit to the data and do not reflect physical considerations in the system, is called a *black box*. The model set with adjustable parameters with physical interpretation may be called a *gray box*. Determination of the "best" model is the *identification method.* The assessment of model quality is typically based on how the models perform when they attempt to reproduce the measured data. After the "best" model is obtained, it remains to test whether this model is valid for its purpose. A model can at best be regarded as a good enough description of certain aspects that are of particular interest to us. Several models useful for linear time-invariant systems include *transfer function models* (such as equation error model, ARMAX model, output error model, Box-Jenkins model etc. [47]) and *state space models*. System models for linear time variant systems can be obtained by using time-varying matrices in the state space model of linear time-invariant systems. Two widely used models for nonlinear systems are *linear regression structure* and *nonlinear state space model*[47]. After a set of candidate models has been selected and parametrized as a model structure using a parameter vector θ, we need to estimate θ. There are many methods for finding the parameter vector and they are known as *parameter estimation*

methods. There are some methods (called nonparametric methods) which determine transfer functions of linear time-invariant model using time-domain and frequency domain techniques. They can be found in [70]. There are two general procedures for estimation methods- prediction error identification approach (which uses methods such as least squares method) and correlation approach (e.g. the instrumental variables methods). The next section describes a conventional method called the method of pseudo-observability indices (POI).

2.1.1 A Method of Pseudo Observability Indices (POI)

The method to be discussed requires no structural information. The necessary information is obtained directly from the input/output data. Pseudo-Observability Indices (POI) [10] are used to establish the resulting observable form state space model. The complete procedure of finding the system model can be found in [65]. Here, the concepts, which are directly used in the neural network approaches are discussed.

The set of POI,$\{\nu_i\}$, $1 \le i \le p$, is any set of numbers satisfying $1 \le \nu_i \le n-p+1$ and $\sum_{i=1}^{p} \nu_i = n$. The set of POI is admissible if rank of

$$\left[(c_1)^T (c_1A)^T \ldots (c_1A^{\nu_1-1})^T ;\ldots: (c_p)^T (c_pA)^T \ldots (c_pA^{\nu_p-1})^T\right]^T$$

is n. $\mathbf{c}_i$ is the i^{th} row of **C**. Consider a system with order n=7, m=2 inputs and p=3 outputs. Suppose that $\nu = \nu_i = \{3,1,3\}$ is an admissible POI.

A crate diagram is used to visualize the selection of linearly independent rows from the given output data.[34]. For example, with the columns of the crate diagram being associated with particular output strings, the selected POI indicates that the independent elements are rows beginning with $y_1(0)$, $y_2(0)$, $y_3(0)$, $y_1(1)$, $y_3(1)$, $y_1(2)$, and $y_3(2)$. For

above POI, crate diagram (shown on next page) is prepared and then several "selection vectors" are created.

From the crate diagram, following "selection vectors" are generated.

* By omitting the first row, the vector v_i is created by selecting the non-blank elements row-wise:

$v_i=[1\ 0\ 1\ 1\ 1\ 0\ 0]^T$

* From vi, the binary complement is formed, and denoted as v_a:

$v_a=[0\ 1\ 0\ 0\ 0\ 1\ 1]^T$

* By considering the blank elements to zero, v_{li} is formed with the first row included:

$v_{li}=[1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0]^T$

The crate diagram is:

{3,1,3}		
1	1	1
1	0	1
1		1
0		0

* By taking the blank elements to be unit valued, v_{ld} is formed , with the first row included:

$v_{ld}=[0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1]^T$;

The above selector vectors are uniquely determined by the particular set of POI, or equivalently, the location of the unity elements in the corresponding crate diagram. From

these selection vectors, the "selector matrices" are derived from the associated selector vectors by a corresponding selection of columns from an appropriately dimensioned identity matrix.

$$S_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T ; S_{li} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T ;$$

$$S_a = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T ;$$

$$S_{ld} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T ; \qquad (2.1.1)$$

A relationship, called Identification Identity, exists [65] which can be represented by

$$\mathbf{y}_{2k} = [\mathbf{N}_r \ \mathbf{A}_r]\mathbf{z}_k, \qquad (2.1.2)$$

where $z_k = \begin{bmatrix} u_k \\ y_{1k} \end{bmatrix}$

Let $r=\nu_m = \max\{\nu_i\}$. and $h=(\nu_m+1)m+n$. Here, $\mathbf{y}_{2k}$ contains samples $\mathbf{y}_i(k+\nu_i)$, $i=[1,p]$, of the output vector $\mathbf{y}(k)$ and:

* $(\nu_m+1)m$ dimensional vector uk containing samples of the input vectors $\mathbf{u}(k+j)$, $j=[1,\nu_m]$ and

* n dimensional vector $\mathbf{y}_{1k}$ containing the samples $\mathbf{y}_i(k+j)$, $j=[0,\nu_i-1]$ of the output vector $\mathbf{y}(k)$.

where ν is a set of admissible POI.

We can concatenate the term [Nr Ar] in the Identification Identity.

Wl=[Nr Ar]; (2.1.3)

The Wl can be viewed as "linear" weights, given by the conventional method. When certain inputs and outputs (vector $\mathbf{z}_k$) are multiplied by this **Wl** , we get the value of y_{2k}- the predicted output by conventional method. **Wl** can be found as follows:

Concatenate the vectors $\mathbf{y}_{2k}$ and $\mathbf{z}_k$ corresponding to the samples k=0,1,...,q-1, where it is assumed that

$h<q$ and $q+\nu_m<N$,(N is the total number of samples) into (p x q) and (h x q) matrices $\mathbf{Y}_2$ and **Z**, respectively, yielding:

$$Y_2 = [Wl]\, Z, \text{ where } Z = \begin{bmatrix} U \\ Y_1 \end{bmatrix}; \quad (2.1.4)$$

U has (ν_m+1)m rows and $\mathbf{Y}_1$ has n rows. The structure of **U** is:

$$U_k = \begin{bmatrix} u(0) & u(1) & \dots & u(q-1) \\ u(1) & u(2) & \dots & u(q) \\ . & . & \dots & . \\ u(r) & u(r+1) & \dots & u(q+r-1) \end{bmatrix}; \quad (2.1.5)$$

where $r = \nu_m$

$\mathbf{Y}_1$ and $\mathbf{Y}_2$ appearing in Eq.2.1.4 may be obtained by pre multiplying the following [(r+1)p x q] matrix $\mathbf{Y}_k$,

$$Y_k = \begin{bmatrix} y(0) & y(1) & \dots & y(q-1) \\ y(1) & y(2) & \dots & y(q) \\ . & . & . & . \\ y(r) & y(r+1) & \dots & y(q+r-1) \end{bmatrix} \quad (2.1.6)$$

by selector matrices $\mathbf{S}_{li}{}^T$ and $\mathbf{S}_{ld}{}^T$, respectively, i.e.

$$\mathbf{Y}_1 == \mathbf{S}_{li}{}^T\, \mathbf{Y}_k \text{ and } \mathbf{Y}_2 = \mathbf{S}_{ld}{}^T\, \mathbf{Y}_k \quad (2.1.7)$$

Using the selector matrices and the value of r, following auxiliary selector vectors and then the auxiliary selector matrices are formed:

$\tilde{v}_{li} = [1...1\ v_{li}]$ where the number of 1's is (k+1)m and $\mathbf{v}_{li}$ has (k+1)p elements. Similarly, $\tilde{v}_{ld} = [0...0\ v_{ld}]$ Then,

$Z = \tilde{S}_{li}^{T} Z_k$ *and* $Y_2 = \tilde{S}_{ld}^{T} Z_k$.If the condition number of **Z** is not acceptable, another POI set is chosen. Now, we can solve for $\mathbf{Wl}\ \mathbf{Z} = \mathbf{Y}_2$.

Following observations can be made from above discussion of a conventional method of system identification:

* The selection of a POI set indicates the number of delayed inputs and delayed outputs needed to identify the system. This is evident from Eq.2.1.2. The definition of $\mathbf{y}_{1k}$, $\mathbf{y}_{2k}$ and $\mathbf{u}_k$ in the Eq. 2.1.2 gives clue to the number of delayed inputs and outputs required to identify the system.

* From the conventional method, we can get the weight matrix **Wl** which ,when multiplied by above determined delayed inputs and outputs , predicts the output consistent with the present inputs and history of inputs and outputs.

Both these facts are used in the neural network approaches to system identification.

2.1.2 Some Notes on Conventional Techniques of System Identification

In this section, the conventional methods for system identification were briefly introduced. There are two major steps : selection of a model and estimation of parameters for a selected model. The conventional methods (other than POI method) ,in general, requires structural information of the systems to be identified. Also, the methods for nonlinear systems are not as comprehensive as the methods for LTI systems. Typically, when we are faced with the identification of nonlinear systems without any structural information about the system, these methods cease to be useful. However, the

concepts from the conventional methods, at times, can be useful even while identifying the nonlinear systems. This will become apparent when the some of the concepts from the method of POI will be used for nonlinear system identification using neural networks.

2.2 Nonlinear Systems

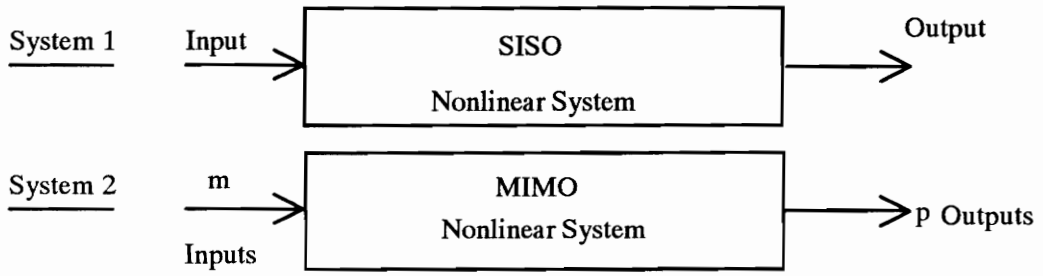
Two different classes of nonlinear systems are taken into account. One class of systems (henceforth called Class I systems) is the class where the nonlinear systems operate as a linear system around some operating point in some region of operation. However, the system gives up its linear behavior as it is moved away from this operating point. Such class of nonlinear systems is exemplified by pendulum and cart type systems. When the pendulum angle is small, the pendulum behaves like a linear system. When the pendulum angle becomes large, the pendulum no longer acts linearly. When the nonlinear system is linearized around an operating point, the nonlinear system in the vicinity of such operating point behaves linearly and we possess an arsenal of conventional techniques for the analysis of such systems. However, the system will behave like a nonlinear system when it moves away from this operating point. Such systems can be classified as Class I systems if we know the extent of the "linear" region. A second class of systems, Class II systems, is characterized by the fact that we don't have any information about the existence of the "linear" region of operation. The only information we have is the input and output data of the nonlinear system. Examples of both the classes of systems are defined next which will be used in simulations. Figure 2.2.1 shows both the classes of nonlinear systems.

2.2.1 Class I Systems

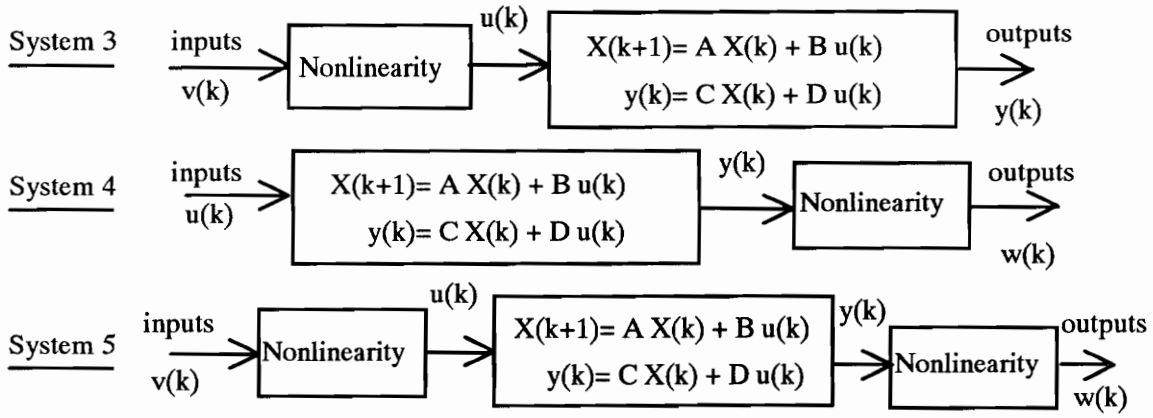
A class of nonlinear systems can be represented by the following relationships:

$$X(k+1) = f(X(k), u(k))$$
$$y(k) = g(X(k), u(k)) \qquad (2.2.1)$$

where **X** is a n x 1 state vector, **u** is an m x 1 input vector, **y** is an p x 1 output vector, and **f** and **g** are functions. Functions **f** and/or **g** may be nonlinear functions. The system represented by Eq.(2.2.1), is assumed to be linear in some region and outside this region, the system is nonlinear.



Systems used as Class I Systems in Simulations



Systems used as Class II Systems in Simulations

Figure 2.2.1 Systems used as Class I and Class II Nonlinear Systems in Simulations

The SISO and MIMO systems , used as systems to be identified, are given here:

$$y(k) = 0.5y(k-1) + u(k-1) + 2\,u^2(k-1) \tag{2.2.2}$$

$$\begin{aligned} X(k+1) &= AX(k) + Bu(k) + E(k) \\ y(k) &= CX(k) + Du(k) \end{aligned} \tag{2.2.3}$$

where,

$$A = \begin{bmatrix} 1.1 & 0.8 & 1 & 0.8 \\ -0.8 & 0.1 & 0 & 0.6 \\ -0.6 & -0.8 & -0.5 & 0.2 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}; B = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ -1 & 0 \\ 0 & 1 \end{bmatrix};$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}; D = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}; E(k) = \begin{bmatrix} 0.001u_1(k)^2 \\ 0.0015u_2(k)^3 \\ 0.003u_3(k)u_2(k) \\ 0.0007u_1(k)u_2(k)u_3(k) \end{bmatrix}$$

Thus, Eq.(2.2.2) is the nonlinear SISO system (System 1) and Eq.(2.2.3) represents the nonlinear MIMO system (System 2).

Two input-output data sets are generated. One data set corresponds to data in "linear" region of the system operation and another data set corresponds to data outside the "linear" region of the system operation. By controlling the magnitude of E(k), we can generate data corresponding to linear or nonlinear region. Though the systems used in simulations are linear in states, the method SRM is still applicable since the practical systems usually have a linearizable region.

2.2.2 Class II Systems

Three different cases of the systems have been covered under this class. System 3 has nonlinearity in inputs. System 4 has nonlinearity in outputs. System 5 has nonlinearity

in both the inputs and outputs. Thus, system 5 can be claimed to be representative of a general nonlinear system.

Above three systems were obtained by using the following linear system as basis:

$$X(k+1) = A\,X(k) + B\,u(k) \tag{2.2.4}$$

$$y(k) = C\,X(k) + D\,u(k) \tag{2.2.5}$$

where,

$$A = \begin{bmatrix} 1.1 & 0.8 & 1 & 0.8 \\ -0.8 & 0.1 & 0 & 0.6 \\ -0.6 & -0.8 & -0.5 & 0.2 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}; B = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ -1 & 0 \\ 0 & 1 \end{bmatrix};$$
$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}; D = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.2.6}$$

Each nonlinearity block is defined as follows:

$$nl\_var(k) = lin\_var(k) + 0.01\,lin\_var^2(k) + 0.001\,lin\_var^3(k) \tag{2.2.7}$$

where, lin_var(k) is the input variable to the nonlinearity block and nl_var(k) is the output variable of the nonlinearity block in Figure 2.2.1.

2.3 Introduction to ANNs

Artificial Neural Networks (ANNs) are characterized by their topology, i.e. by the number of interconnections, the node characteristics that are classified by the type of nonlinear elements used, and the kind of learning rules implemented. A clear and concise general introduction to ANNs is given in [46]. In [2], [3], [6], [74], collection of ANN papers with emphasis on control applications have

appeared. ANNs can be used to model the input/output behavior of dynamical systems. It can also be used as a controller. ANNs can be combined to both identify and control the plant, thus implementing an adaptive controller. They can be used to detect and identify system failures, and to help store information for decision making- providing the ability to decide when to switch to a different controller among a finite number of controllers.

2.3.1 Artificial Neural Network (ANN)

An ANN is composed of many simple elements (called neurons, nodes, cells or units) operating in parallel. These neurons are biologically inspired. A neuron with bias and a neuron without bias are shown in Figure 2.3.1.

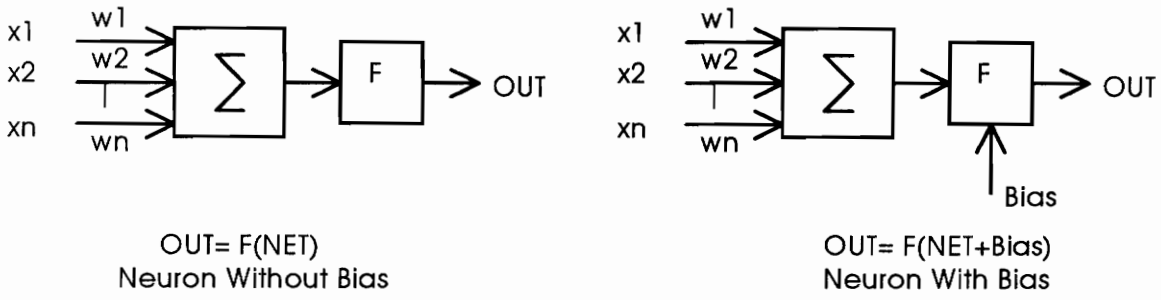


Figure 2.3.1 An Artificial Neuron With and Without Bias

X is an input vector containing x1, x2,...xn. These inputs correspond to the signals into the synapses of a biological neuron [69]. Each input is multiplied by an associated weight w1, w2.,wn, before it is applied to the summation block Σ. Each weight corresponds to the "strength" of a single biological synaptic connection. The set of these weights is W. In Figure 2.3.1,

$$NET = X\,W \qquad (2.3.1)$$

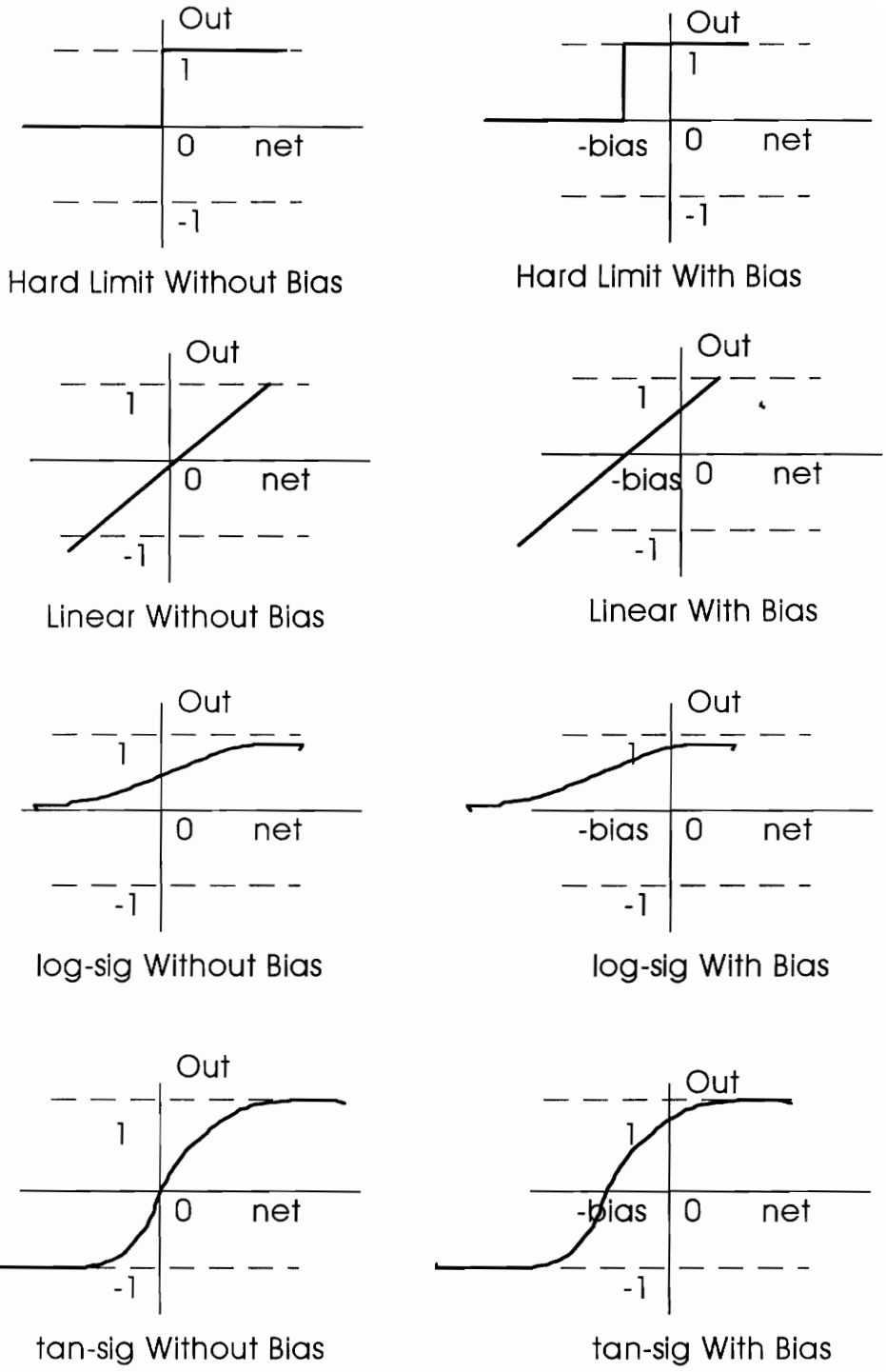


Figure 2.3.2 Different Activation Functions

F is an activation function or transfer function. The commonly used activation functions are hard limit, linear and sigmoid (logarithmic and hyperbolic

tangent) functions. In Figure 2.3.2, they are shown without bias, but they are used with bias also.

The hard limit transfer function is often used with neurons that make decisions such as classification. The linear transfer function is used with linear approximators. The sigmoid transfer functions, logsig (sigmoidal logarithmic) and tan-sig (sigmoidal hyperbolic tangent), are widely used in backpropagation networks.

Feedforward and Recurrent Networks

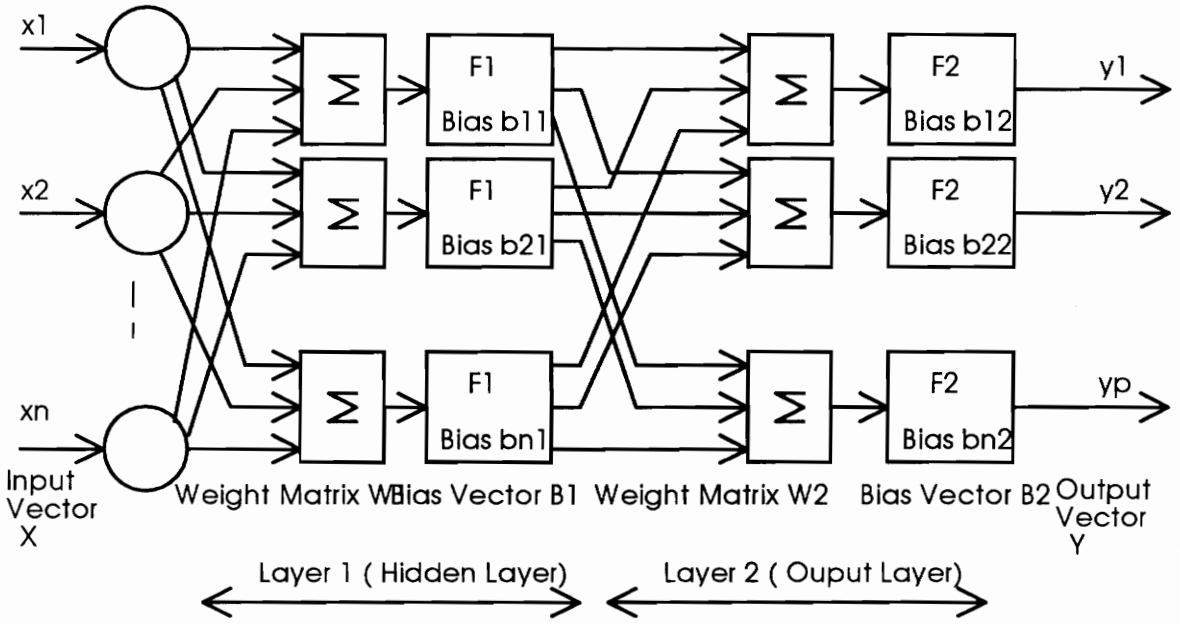


Figure 2.3.3 2-Layer Feedforward ANN

Though a single neuron shown in Figure 2.3.1 can perform some functions, the power of neural computation is achieved when such neurons form an ANN

consisting of several such neurons. There are two types of multilayer ANNs- Feedforward ANN (Figure 2.3.3) and Recurrent ANN (Figure 2.3.4). In Figure 2.3.3, F1 and F2 are activation functions. Multilayer feedforward ANNs can be constructed by cascading a group of single layers as shown in Figure 2.3.3. As seen from the Figure 2.3.3, the ANN has no feedback connections, i.e., connections through weights extending from the outputs of a layer to the inputs of the same or previous layer. Hence, such ANNs are called feedforward networks. These networks are widely used. They do not have any memory; their outputs are solely determined by the current inputs and their values of weights and biases. Another types of networks, recurrent networks, are shown in Figure 2.3.4. The network shown is a single layer network but multilayer recurrent can be developed easily similar to Figure 2.3.4. The recurrent networks have their outputs connected to the inputs. Hence, their outputs are determined both by their current inputs and their previous outputs. They are quite powerful because they have short-term memory and they are sequential rather than combinational (e. g. feedforward networks). The initial state of the neurons' outputs is set to the input vector. The feedback from the output to the input allows the network to exhibit temporal behavior.

Training of ANN: Supervised and Unsupervised Training

The ANN is trained so that it produces a desired set of outputs when a set of inputs is applied. In supervised training, a training pair is formed which contains input vector and a target vector representing the desired output. The output corresponding to the inputs presented to the ANN is calculated and compared with the target vector and the weights and biases are modified according to some

training algorithm that tends to minimize the error. In unsupervised training, the training set contains only input vectors.

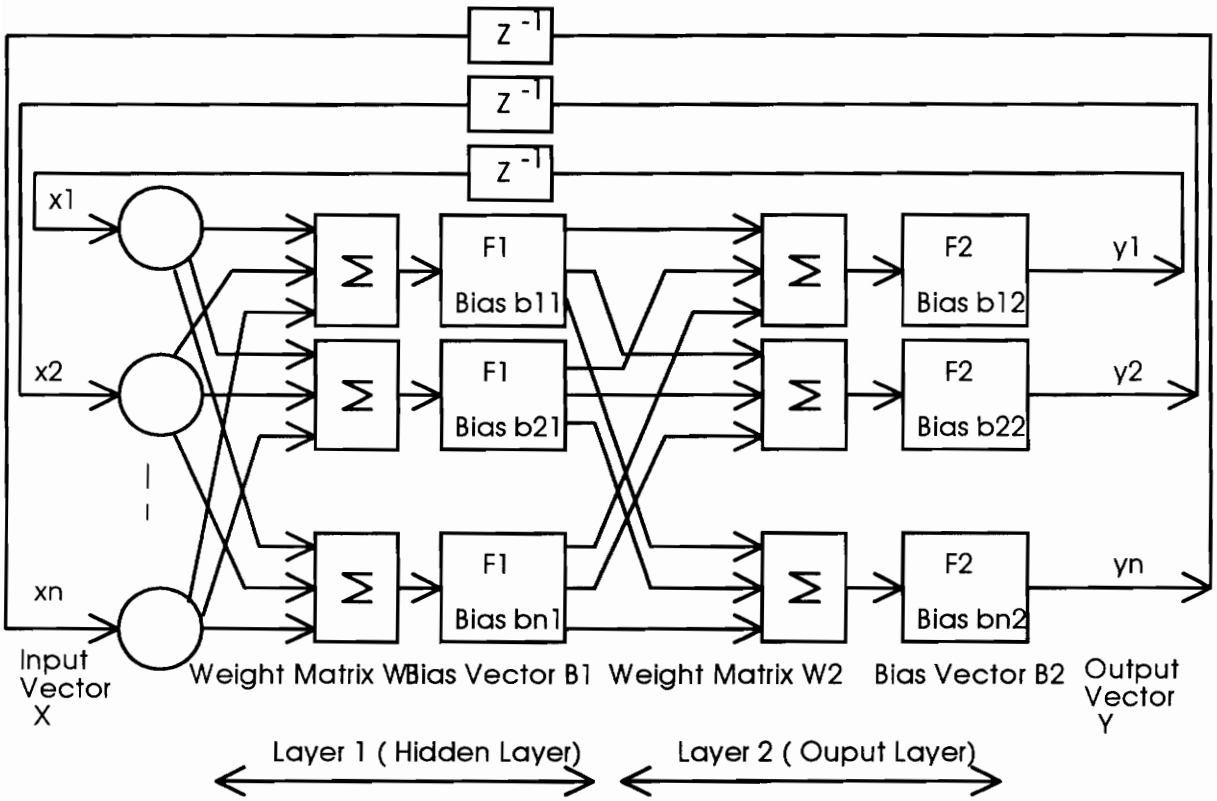


Figure 2.3.4 2-Layer Recurrent Network

The training algorithm extracts the statistical properties of the training set and groups similar vectors into classes. The outputs of such a network are transformed into a comprehensible form subsequent to the training process which is not a difficult task.

Numerous paradigms of ANN exist based on how inputs are processed (i.e., which activation functions are used) and the training algorithm (often referred to as a learning rule).

2.3.2 Applications of ANNs

ANNs are biologically inspired. Several paradigms exist such as perceptrons [58], ADALINE [72], backpropagation networks ([75] and [59]), instar [25], outstar [25], ART1 networks [39], Hopfield nets [45], counterpropagation networks ([28], [29] and [30]), BAM ([41] and [42]), Boltzmann network, cascade-correlation network, digital neural network, directed random search network, Hamming network, probabilistic network, cognitron, neocognotron etc. Discussions and applications of these ANNs can be found in [15], [17], [60]]. Backpropagation, invented independently in three separate research efforts ([70], [56] and [59]) is the most successful of the current algorithms. However, ANNs are not a panacea. Reliability is very difficult to prove in the case of ANNs. A related difficulty lies in the inability of ANNs to explain how they solve problems. Despite these drawbacks, it is reasonable to expect a rapid increase of our understanding of ANNs so that new and/or improved network paradigms and applications would be achieved.

2.4 Basic Steps In System Identification

The linear system model can be given by state space representation. Complex nonlinear systems can not be represented by such elegant way. Here, the nonlinear system identification problem is viewed as a problem of finding system parameters such that the knowledge of inputs present in the system is sufficient to determine its outputs. These system parameters need not represent actual parameters of the system as long as the system outputs are predicted correctly.

The basic steps in system identification using neural networks can be enumerated as (i) Collection of data set, (ii) Determination of neural network structure and training algorithm (iii) Determination of Input set and Target set for the training of the neural network (iv) Training and testing of the network. These steps are elaborated next.

2.4.1 Collection of Data Set

It is the first step in the development of neural network application. In general, the training set must provide a representative sample of the data the network will process in the finished application. For simulation purposes, it is assumed that the type of inputs encountered by the systems under consideration are random in nature. However, while getting actual data from the system, care must be taken to ensure that all types of inputs (e.g. random, sinusoidal, step etc.) , to which the system is usually exposed to, are presented in the data sets. Otherwise, neural network will work very well in the simulations but will not serve the purpose when actually implemented! In practice, collection of data sets may be easy when the variables of interest are recorded or monitored. But, again, those data should be rich enough so that observable part of the process dynamics can be captured by the data.

2.4.2 Determination of ANN Structure and Training Algorithm

The number of layers , the number of neurons, or nodes, in each layer and the squashing functions are some of the degrees of freedom which need to be chosen. It depends, to a large extent, on the complexity of the particular problem under consideration. Initially, the minimum number of neurons in each layer and two layers (one

hidden layer and one output layer) are recommended. The number of neurons in the hidden layers should be increased gradually until it gives satisfactory results. The backpropagation algorithms tend to do well at function estimation and time series tasks, especially if the desired result is easily expressed as a set of continuously varying values. It is also good at representing complex nonlinear relationships in the form of a compact, efficient network. The backpropagation learning rule follows gradient descent on the error surface to minimize network error. Changes in each weight and bias are proportional to that element's effect on the sum-squared error of the network. The standard backpropagation can be improved by using momentum, adaptive learning rate and Nguyen-Widrow initial conditions. The 2-layer sigmoid/linear network has been proven to be able to represent any functional relationship between inputs and outputs if the sigmoid layer has enough neurons.[74].

2.4.3 Determination of Input set and Target set for Training the Neural Network

When dynamics is inherent in the system, the present output of the system depends not only on the current inputs but also on the past inputs and outputs. Hence, it is useful to be able to decide which delayed inputs and outputs should be presented to the network. If no clue is available, simulations can be run with minimum delays in inputs and outputs and then delays can be added gradually until satisfactory results are obtained. Simulation results for Class II systems will provide a good insight in this regard. Class I systems allow us to determine the number of delays as will be shown later. Also, the difference between two system identification approaches is based on which inputs and outputs are presented to the neural network .

2.4.4 Training and Testing of the Network

After deciding about the neural network structure and the input set and target set, training of neural network can be started. Different training parameters can be tried first to get a general idea about the effect of the training parameters. Typically, the backpropagation algorithm takes long time to converge to a solution. When a complex system is under consideration. there will be many local minima and the solution we get may not correspond to a global minima. However, we are not much concerned about global optimality of the solution as long as we get the accuracy we need. Also, there has been no theoretical developments to prove or guarantee the optimality of a solution obtained using the backpropagation algorithm. In general, the error reduces as training progresses. However, there is a potential danger of overtraining in which case the neural network will memorize the patterns instead of generalizing the relationship between the patterns. To avert such problem, it is suggested that the performance of the neural network be checked periodically during training. For this purpose, a fraction of available data set (henceforth called "test data set") should be kept only for testing the neural network. This "test data set" should not be used in training the network and thus it will be "unseen" by the network. Also, while testing the network, the neural network parameters should not be modified.

It should be noted that Input set and Target set (containing desired outputs) should be properly scaled before they are used with neural network. Scaling is done at the front end of the neural network and "descaling" is done at the output of the neural network. Scaling results in better training and hence it is a prerequisite of training the network.

2.5 Performing The Basic Steps of System Identification for the Systems Under Consideration

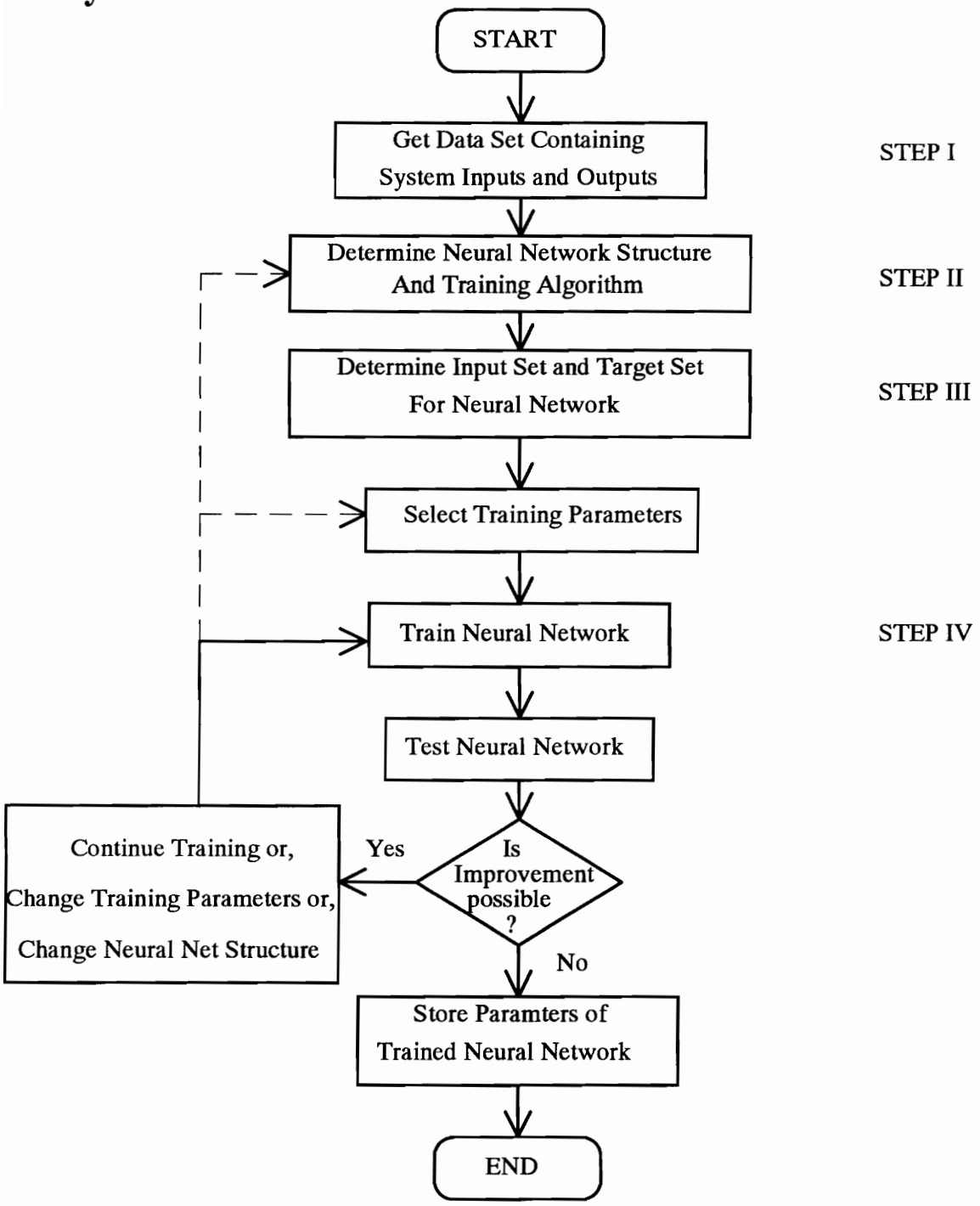
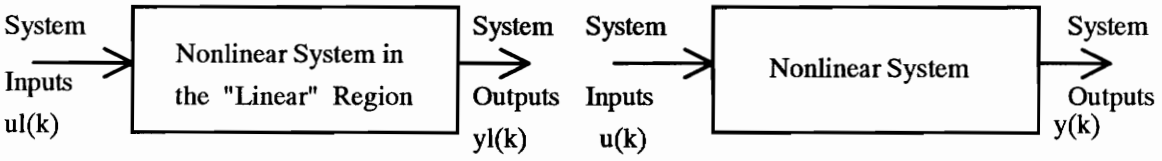


Figure 2.5.1 Flowchart of System Identification Using Neural Networks

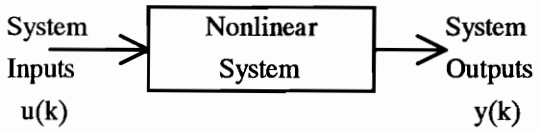
The basic steps briefly introduced in the previous section were performed for the systems under consideration. They are described here. The flowchart for system identification using neural networks is shown in Figure 2.5.1.

2.5.1 Collection of Data Set

The systems to be identified are defined in section 2.2. Random inputs were applied to the systems and the system inputs and outputs were observed. Figure 2.5.2 shows how data sets are collected for Class I and Class II systems.



Collection of Data Sets ul(k) & yl(k) and u(k) & y(k) for Class I Systems



Collection of Data Set u(k) & y(k) for Class II Systems

Figure 2.5.2 Collection of Data Sets for Class I and Class II Systems

For Class I systems two different data sets were obtained. One data set corresponds to the data in the linear region (ul(k) and yl(k)) and another data set corresponds to the comprehensive region (which includes region outside the linear region, u(k) and y(k)). As seen from the Eqs. (2.2.2) and (2.2.3), the nonlinearity was introduced into the system inputs so that two data sets corresponding to two regions can be obtained by controlling magnitude of the system inputs. Small magnitude of inputs gives the data

set corresponding to the linear region and large magnitude of inputs gives the data set corresponding to the comprehensive region.

For Class II systems, we have only one data set (u(k) and y(k)) to play with, which is the data set corresponding to the comprehensive region.

The next step is to set aside a fraction of the collected data set for the purpose of testing the neural network. Total 500 data points were collected. 400 data points were used for training the network and the remaining 100 data points were used for testing the network. The set of 400 data points is used in forming the training data set and the set of 100 data points is used in forming the test data set.

2.5.2 Determination of ANN Structure and Training Algorithm

The ANN structure used for system identification is shown in Figure 2.5.3.

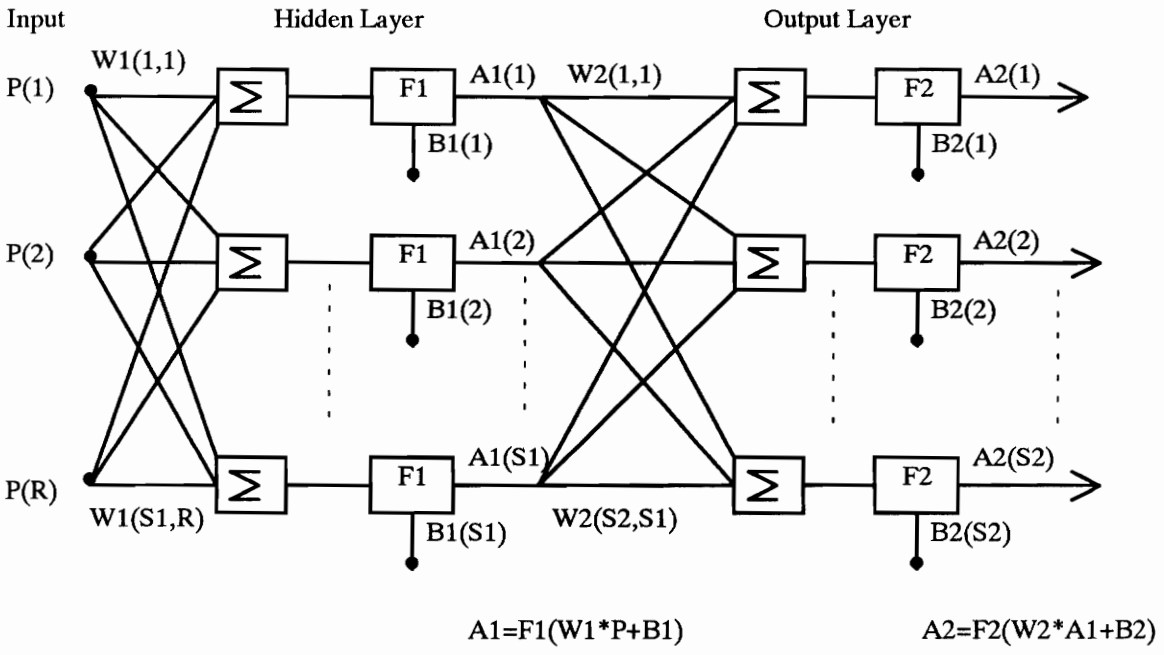


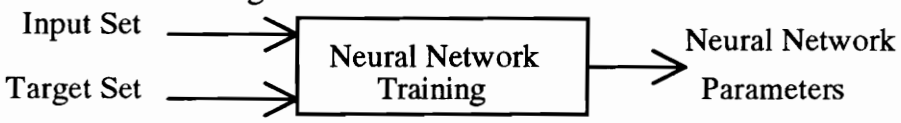
Figure 2.5.3 Structure of Neural Network Used in System Identification

It is a 2-layer backpropagation network; the first layer is a hidden layer and the second layer is the output layer. R is the number of inputs to the network. P is the input pattern. The blocks containing Σ and F's (i.e. F1 and F2) constitute a single neuron. Σ is the summation and F1 and F2 are transfer functions or activation functions. F1 is a hyper tangent sigmoid (also called tan-sigmoid) function and F2 is linear function. S1 and S2 are hidden layer and output layer neurons respectively. W(i,j) is the strength of the connection from j^{th} input to the i^{th} neuron. B(i) is the bias associated with the neuron.

The training algorithm used was backpropagation algorithm with the improvements such as adaptive learning rate, momentum and Nguyen-Widrow initial conditions. This backpropagation algorithm was taken from MATLAB. In the beginning of training, we have to initialize the weights and biases in the network. The common approach is to use some random values for initialization. Nguyen and Widrow found a way to pick initial weights and biases that are better than purely random values. They did this by analyzing how a 2-layer network performs a function approximation. Use of their method leads to a satisfactory network in less training time. The method was applied to get the weights and biases in the first layer of a tan-sigmoid /linear network. Small random values were used to initialize the weights in second layer. Simulations were run with different number of hidden layer neurons. The number of output layer neurons is fixed by the number of outputs in the target patterns. For each number of hidden layer neurons, simulations were run several times because of two reasons. One reason is that we may get better results for one set of random values than another set of values. The second reason is that it can give an overall picture of how the network performs under different initial conditions.

2.5.3 Determination of Input set and Target set for Training the Neural Network

Determination of delays in inputs and outputs. When a system has dynamics, the prediction of outputs of the system requires not only the present inputs but also the past history of inputs and outputs. The number of delays in inputs and outputs depends on the complexity of the dynamics of the system. In the case of Class I systems, we have a data set in the linear region of the system. There are many methods for identification of linear systems as described in Chapter 2. One of the powerful methods is the method of POI. The application of POI method gives us exact number of delays required in inputs and outputs for accurate prediction of outputs. Hence, in case of Class I systems, the method of POI was used to get information about delays and then simulations were carried out taking into account the delay information. In case of Class II systems, we do not know even the existence of linear region and hence simulations have to be done by guessing the delays. Thus, the help of proven conventional methods for linear system identification was taken wherever possible. Two approaches discussed here are Whole Region Method (WRM) and Separate Regions Method (SRM). Figure 2.5.4 shows the input set and target set used in off-line training for WRM.



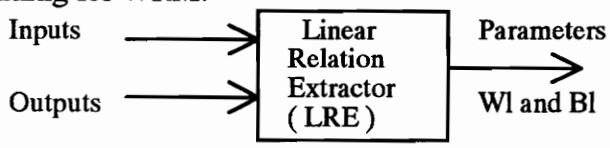
Input Set: set of patterns { u(k) u(k-1)...u(k-du) y(k-1)...y(k-dy) }
Target Set: set of { y(k) }

du: maximum delay in system inputs
dy: maximum delay in system outputs

Off-line Training of Neural Network for WRM

Figure 2.5.4 Determination of Input Set and Target Set for WRM

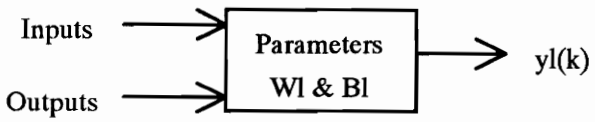
Here, u(k) and y(k) are system inputs and outputs. The input set for WRM contains the system inputs with proper delays and delayed system outputs. The target set for WRM contains the system outputs. Figure 2.5.4 shows the input set and target set used in off-line training for WRM.



* Class I Systems:
  Inputs: ul(k) ul(k-1)...ul(k-du)
  Outputs: yl(k) yl(k-1)...yl(k-dy)
  Form of LRE: POI Method

* Class II Systems:
  Inputs: u(k) u(k-1)...u(k-du)
  Outputs: y(k) y(k-1)...y(k-dy)
  Form of LRE: Least Square Solution

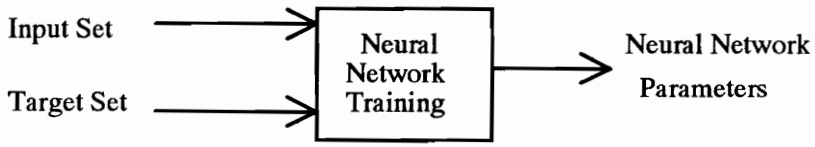
Extracting Linear Relationship Between System Inputs & Outputs



Inputs: u(k) u(k-1)...u(k-du)
Outputs: y(k-1)...y(k-dy)

ynl(k)=y(k)-yl(k)

Determining Output Component Due to Nonlinearity Alone



Input Set: set of patterns { u(k) u(k-1)...u(k-du) ynl(k-1)...ynl(k-dy) }
Target Set: set of { ynl(k) }

Off-line Training of Neural Network for SRM

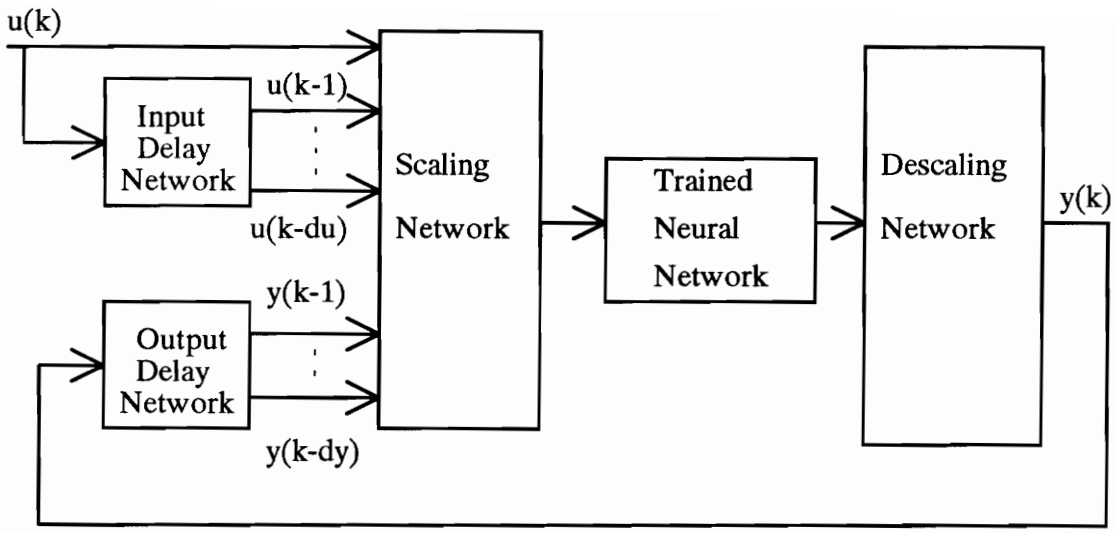
du: maximum delay in system inputs
dy: maximum delay in system outputs

Figure .2.5.5 Determination of Input set and Target Set for SRM

The determination of input set and target set for SRM consists of two phases (Figure 2.5.5). First, the linear relation extractor (LRE) extracts the linear relationship between the system inputs and outputs is extracted. For Class I systems, the method of POI can be used as LRE since we have the data set corresponding to the linear region. For Class II systems, we don't have such information and hence a simple least square solution is used. At the end of this first phase, we possess the weight and bias which represent the model of the behavior of the nonlinear system in the linear region. The second phase is the separation of the component of the system outputs which is due to nonlinearity alone. Since we have the linear model of the system, we can now easily determine the output due to nonlinearity by subtracting the output due to linear behavior of the system from the "total" output of the system. Then the input set contains the system inputs with delays and the delayed outputs due to nonlinearity in the system. The target set contains the system outputs due to nonlinearity.

2.5.4 Training and Testing of the Network

The input set and target set determined in the previous section are supplied to the 2-layer backpropagation network shown in Figure 2.5.3. Periodically the neural network is checked to see how accurately the neural network is able to predict the outputs. This training is continued until the prediction of outputs ceases to improve and the optimum parameters (i.e. those weights and biases which gave the least error when used to predict the outputs with the test data) of the trained neural network are stored. These optimum parameters should be used in a finished application to predict the outputs of the system. The arrangement for prediction of outputs by WRM is shown in Figure 2.5.6.



y(k): Output Predicted by Trained Neural Network (WRM)

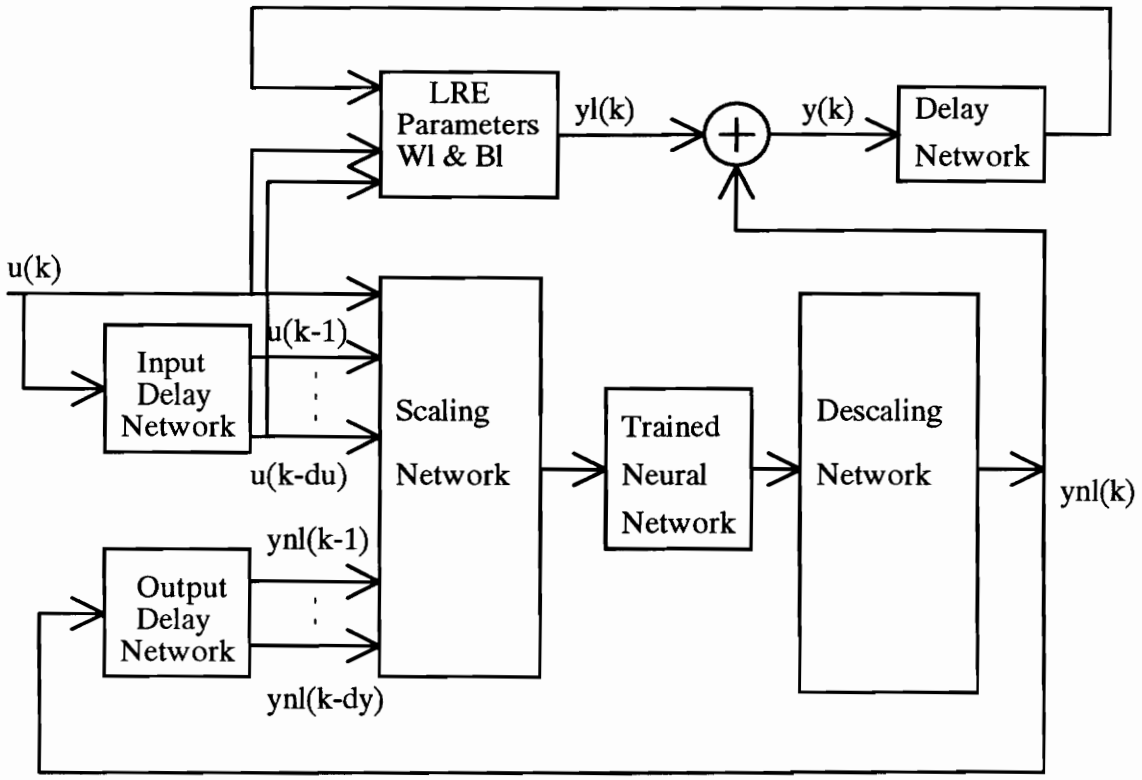
Figure 2.5.6 Arrangement for Output Prediction by WRM

Figure 2.5.6 explicitly shows the Scaling Network and Descaling Network. Figure 2.5.7 shows the arrangement for output prediction by SRM. The nonlinear component of the outputs is predicted by the neural network and the linear component of the outputs is calculated using the LRE parameters.

2.6 Summary

Conventional methods of system identification were briefly visited. In particular, a method of POI was described in detail because some of its concepts are used in neural network approaches to system identification. The nonlinear systems to be identified were defined. The basic steps of system identification using neural networks were enumerated. Two different classes of systems were considered. For both the classes of systems, system inputs and outputs were collected. In case of Class I systems, additional data sets

corresponding to the linear region of the system behavior were also obtained. The 2-layer feedforward backpropagation neural network was used.



ynl(k): Component of Output Predicted by Trained Neural Network
yl(k): Component of Output Contributed by LRE Parameters
y(k): Output Predicted by SRM

Figure 2.5.7 Arrangement for Output Prediction by SRM

The approaches used for system identification include WRM (Whole Region Method) and SRM (Separate Regions Method). In WRM, the system inputs and outputs in the comprehensive region are used for neural network training. In SRM, the linear relationship between system inputs and outputs is extracted separately and the neural network is trained to provide the output due to nonlinearity. The linear relationship

between system inputs and outputs can be determined very accurately using the conventional method of POI for Class I systems because we possess a linear data set. In Class II systems, the linear relationship was obtained using the least square solution. The performance of neural network being trained was checked periodically using Test Data to avoid the possibility of excessive training and optimum parameters of the neural network for the systems under consideration were obtained.

Chapter 3
Simulation Results for System Identification

The basic steps for system identification described in the previous chapter were followed and simulation results were obtained. In this chapter, simulation results are presented and evaluated. Both the WRM and SRM proved to be effective in predicting the outputs of the systems under investigation. The results indicate the strength of the neural networks in their application to the problem of nonlinear system identification. SRM gives somewhat better results than WRM. Some terms used in evaluating the performance of the identification methods are explained first. The results are revealed for both the classes of systems.

3.1 Background

Several terms are introduced here .

Actual and Predicted Output. As mentioned in Chapter 3, test data set is used in evaluating how well the neural networks have learned during training. The test data set consists of the inputs and outputs of the system. These outputs are referred to as Actual Outputs. The inputs, constituents of the test data set, are presented to the trained neural network. The neural network, based on its parameters (i. e. weight matrices and bias vectors), calculates the outputs of the system. These outputs are referred to as Predicted Outputs.

Frobenius Norm of Error. A norm is used to evaluate the closeness of two functions. Since we are interested in determining how close the predicted outputs and the actual outputs are, a Frobenius norm of the difference between the actual and predicted outputs is used.

Epoch. An epoch here is an entire pass through all of the input training vectors. Each input training pattern comprises of scaled current and delayed system inputs and scaled delayed outputs.

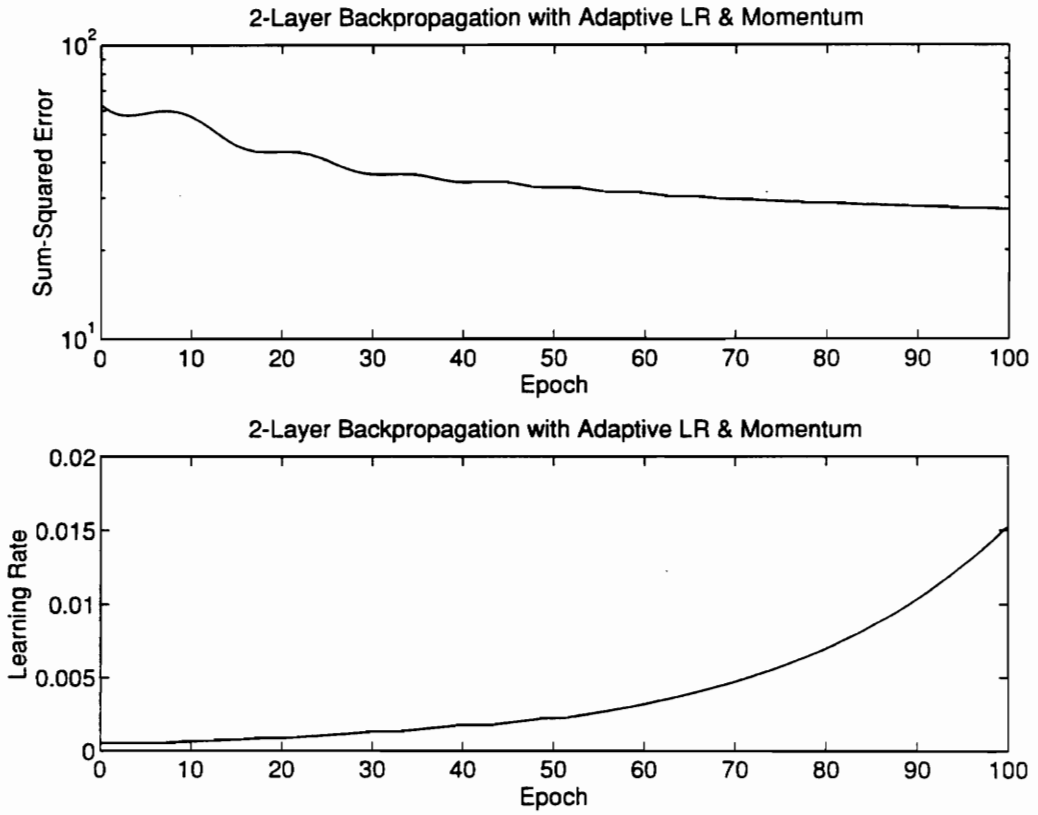


Figure 3.1.1 Variation of Training Data Error and Adaptive Learning Rate with Training Time

An improved backpropagation algorithm with adaptive learning rate and momentum is used for training the neural network. A typical graph of the error in the training data and the learning rate is shown in Figure 3.1.1.

It should be noted that the sum-squared error is the error in the scaled training data presented to the neural network and will generally keep reducing as the training progresses. The learning rate adapts itself depending upon the sum-squared error.

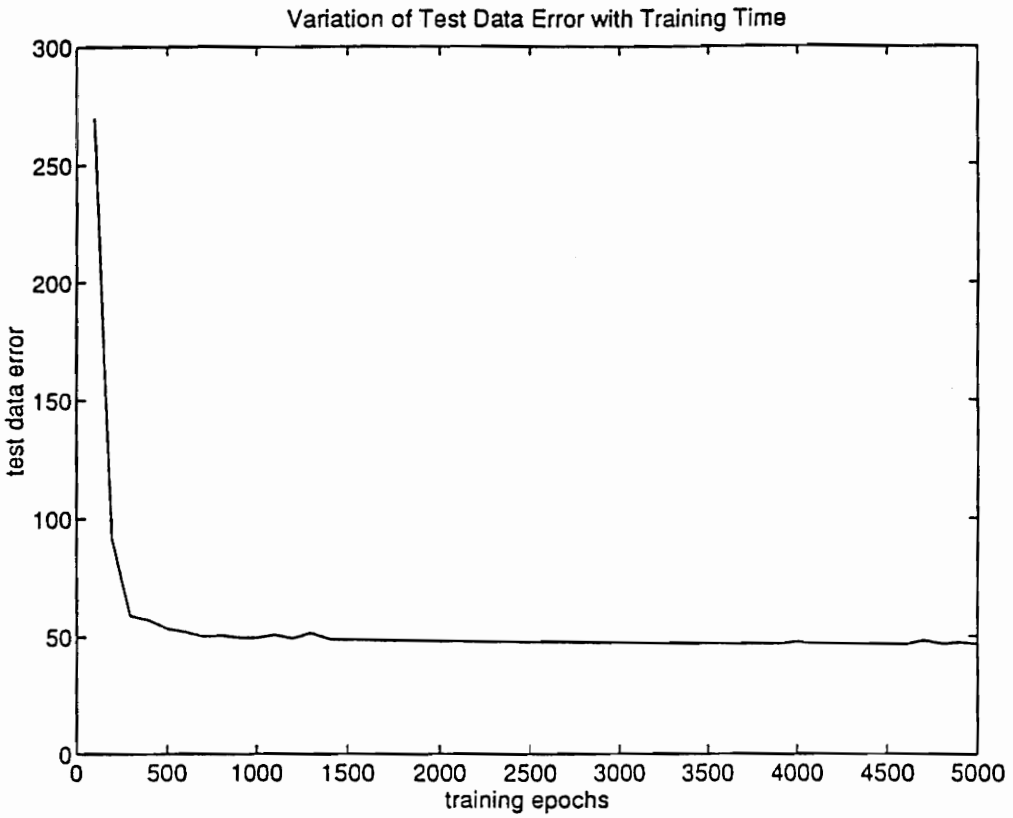


Figure 3.1.2 Typical Variation of Test Data Error with Training Time

The variation of test data error as a function of training time appears in Figure 3.1.2. The training may be terminated prematurely if it is felt that further training will not

significantly decrease error in test data. As seen from the Figure 3.1.2, the test data error reduces drastically in the beginning as the neural network starts getting feedback from its own predicted outputs. Then, the reduction in test data error is not significant. It should be kept in mind that excessive training may result in "memorizing", rather than "generalizing", in which case the test data error will increase as the degree of excessive training increases.

3.2 Identification of Class I Systems

Since we know the linear region of operation of Class I systems, conventional method of POI can be applied to determine the order of the system and also the appropriate number of delays in inputs and outputs in order to be able to predict the outputs more accurately. The method of POI was applied to the data set representing the linear region of system operation. The number of delays required in inputs and outputs were found to be 1 and 2 for the System 1 and System 2 respectively. Hence, this information was used while carrying out the training of the neural networks for Class I systems..

The results for Class I Systems, System 1 and System 2, are presented. Table 3.2.1 summarizes the results corresponding to System 1. From Table 3.2.1, we can conclude that for a given complexity (i.e. no. of hidden layer neurons), SRM gives better error performance than the WRM. Since an SISO system is simple in dynamics compared to MIMO systems, the error in test data keeps on reducing as the training progresses. The training was terminated after several epochs as tabulated here. Another salient feature stemming from the table is that the test data error reduces as the number of hidden layer neurons increases to a certain point and then the test data error does not reduce further.

In fact, test data error increases if there are excessive number of neurons. This "excessive" number of neurons cannot be determined theoretically. Running some simulations can easily trace this "excessive" number of neurons. In the present case, four neurons in the hidden layer gave the best error performance for both WRM and SRM. Hence, when the neural network is to be used as System Identifier in actual implementation, the optimum parameters corresponding to four hidden layer neurons should be used. Each row in the table corresponds to simulation with different initial random weights.

Table 3.2.1 Simulation Results for System 1 by WRM and SRM

No. of Hidden Layer Neurons	Whole Region Method (WRM)		Separate Regions Method (SRM)	
	Training Epochs	Frobenius Norm of Error in Test Data	Training Epochs	Frobenius Norm of Error in Test Data
2	50000	0.2147	50000	0.0373
		0.2170		0.0375
		0.2150		0.0377
		0.2155		0.0377
		0.2160		0.0377
		0.2168		0.0382
3	180000	0.0292	180000	0.0244
		0.0311		0.0246
		0.0721		0.0246
		0.1497		0.0248
		0.1559		0.0255
		0.1561		0.0259
4	150000	0.0224	150000	0.0101
		0.0228		0.0122
		0.0318		0.0262
		0.0336		0.0272
		0.0558		0.0268
		0.1591		0.0282
5	150000	0.0257	150000	0.0133
		0.0267		0.0201
		0.0524		0.0249
		0.0808		0.0268
		0.1597		0.0278
		0.0546		0.0287

Figure 3.2.1 shows the actual and predicted output for an SISO System 1 using the WRM. When SRM is used, the linear relationship between system inputs and outputs is extracted first. The conventional method of system identification, the method of POI, was used to extract the linear weights.

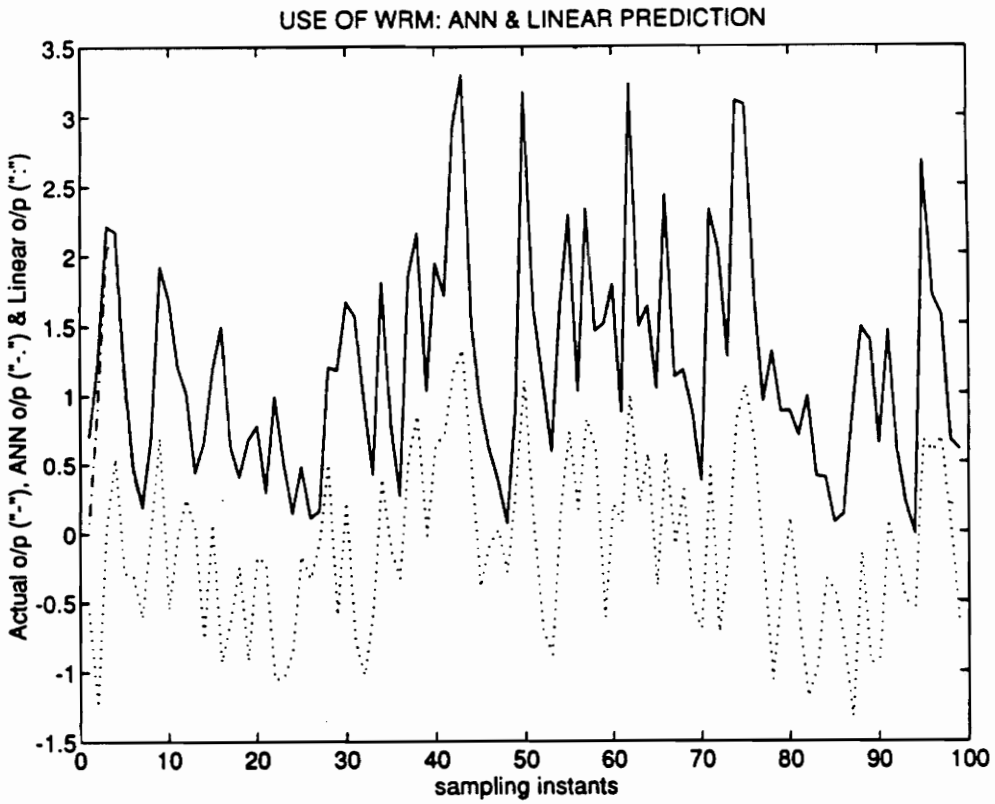


Figure 3.2.1 Comparison of Actual Output and Output Predicted by WRM for System 1

Figure 3.2.2 shows the actual and predicted output in the linear region of the system operation and it is evident that the linear region has been modeled quite well. The use of POI method has provided an excellent model of the nonlinear system in the "linear" region and it gives the SRM an edge over the WRM for Class I systems. This analysis will be revisited after presenting the results for Class II systems.

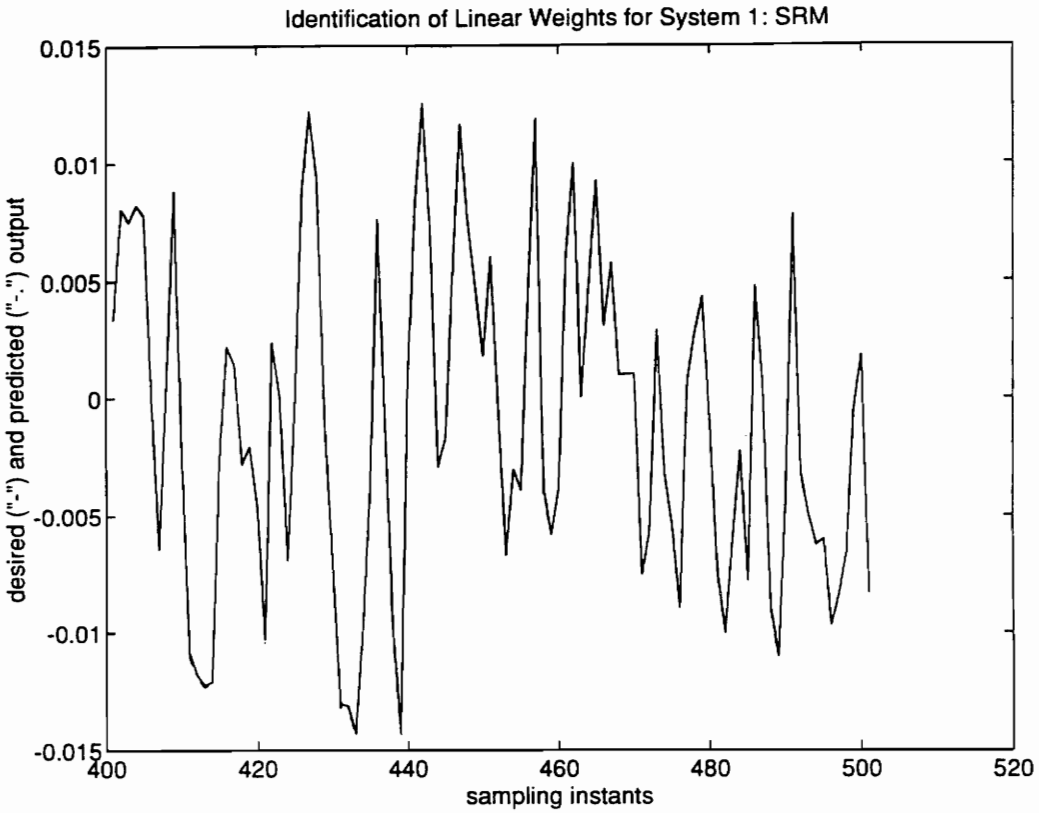


Figure 3.2.2 Identification of Linear Weights by SRM for System 1

The results of using SRM for the predicting of the outputs of the system outside the linear region are shown in Figure 3.2.3. Since accurate prediction of the outputs require the delayed outputs due to the dynamics in the system, the actual and predicted outputs in Figures 3.2.1 and 3.2.3 are not matching for first several samples. However, after the neural network starts getting the information about the outputs due to feedback connection to itself, it "locks on" and predicts the outputs accurately.

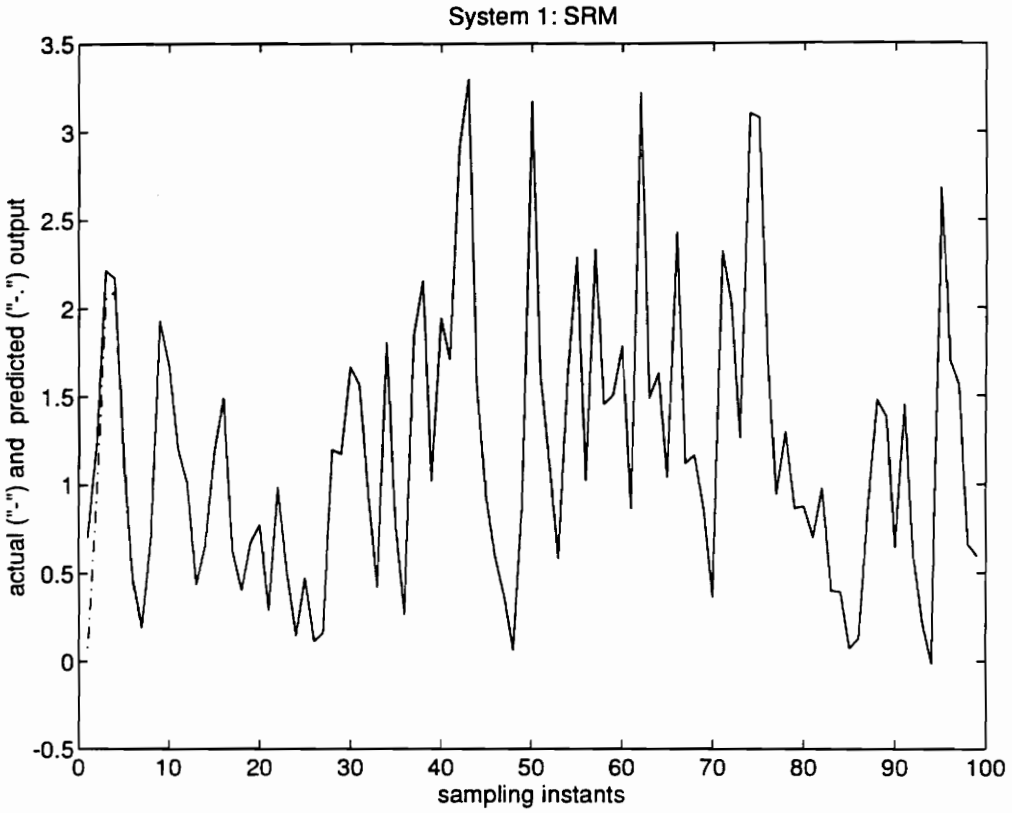


Figure 3.2.3 Comparison of Actual Output and Output Predicted by SRM for System 1

The results corresponding to the MIMO System 2 are recapitulated in Table 3.2.2.

Table 3.2.2 Simulation Results for System 2 by WRM and SRM

Magnitude of Nonlinearity	No. of Hidden Layer Neurons	W R M		S R M	
		Training Epochs	Frobenius Norm of Error in Test Data	Training Epochs	Frobenius Norm of Error in Test Data
No Nonlinearity	7	1400	1.1475	900	0.8654
		1400	1.5142	300	1.0248
		2100	1.2459	500	1.1343
		2200	1.1678	500	0.8802
		3300	1.6180	600	0.9216
		4600	1.4198	1100	0.9147
"small" Nonlinearity ("small" magnitude inputs)	5	30000	2.8187	500	2.0415
		4400	3.3771	200	2.6963
		7700	3.2770	300	2.9696
		8300	3.3333	400	2.5797
		9200	3.5588	500	2.4999
		29300	3.1802	700	2.8502
	7	14000	3.4869	1800	2.3537
		900	3.1752	200	3.4325
		2800	3.9948	300	2.6257
		3800	3.8848	500	2.5405
		4400	3.0411	900	3.0053
		17300	3.7782	1600	2.8494
"large" Nonlinearity ("large" magnitude inputs)	5	1200	13.8763	5900	13.6731
		1300	16.4838	200	16.3069
		1900	20.0638	500	13.6174
		2600	17.5173	600	16.0068
		3500	17.2990	2600	13.3499
		4200	18.2433	7200	13.6136
	7	3500	13.8897	3900	13.7125
		1500	17.0301	200	13.0159
		1500	19.6756	300	13.8183
		4800	18.8564	500	13.8319
		5900	18.6197	3100	16.0662
		7500	17.3797	4800	13.4137

As mentioned in the previous chapter, the magnitude of nonlinearity entering into the system was gradually increased to analyze the behavior of the WRM and SRM under

different "size" of the region of system operation. From Table 3.2.2, it can be concluded that SRM gives better error performance than WRM. Also, the training time required to obtain optimum weights for a given system is usually less for SRM. When the nonlinearities become predominant, there is no significant training time savings for SRM, though the accuracy of SRM will still be better than WRM. For Class I systems, we can conclude that SRM is generally better than WRM for nonlinear system identification. The actual and predicted outputs by WRM for System 2 are graphed in Figure 3.2.4 .

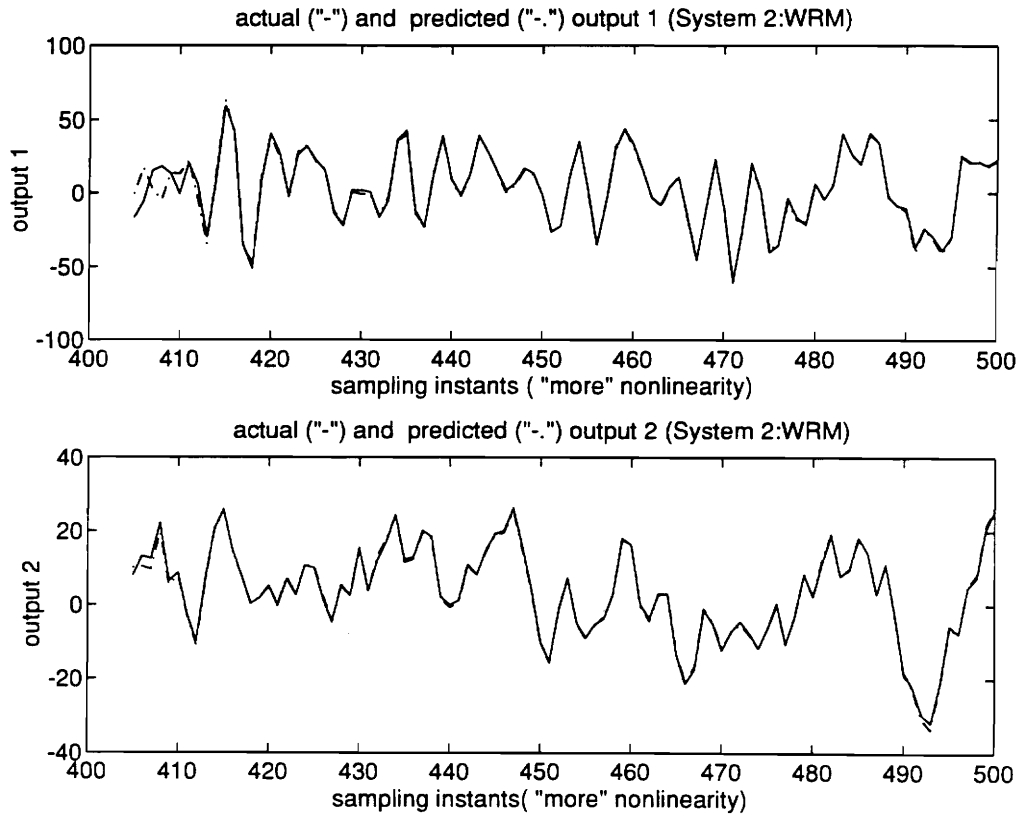


Figure 3.2.4 Comparison of Actual Outputs and Outputs Predicted by WRM for System 2

There is a very good match between the actual outputs and predicted outputs. The results shown in Figure 3.2.4 correspond to the case of "large" or "more" nonlinearity in the system. Similar good results were obtained for the cases of " no nonlinearity" and "small " nonlinearity when WRM was applied to the System 2.

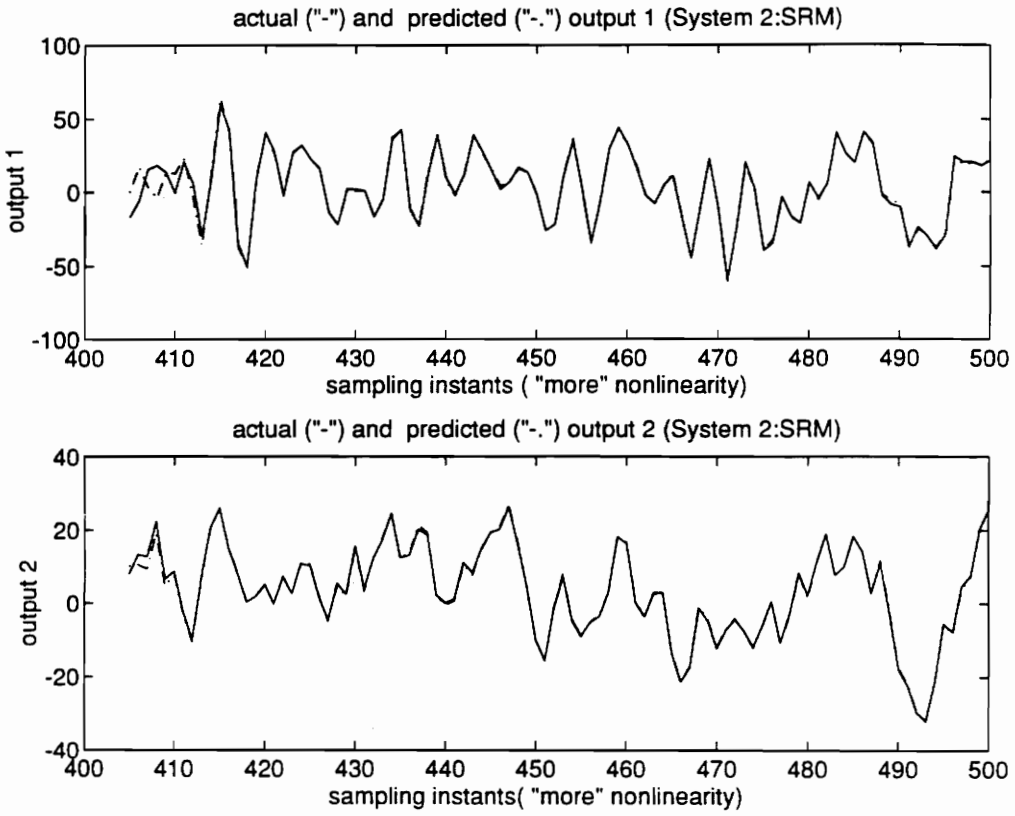


Figure 3.2.5 Comparison of Actual Outputs and Outputs Predicted by SRM for System 2

Here also, there is a very good match between the actual outputs and predicted outputs. The results shown in Figure 3.2.5 correspond to the case of "large" or "more" nonlinearity in the system. Similar good results were obtained for the cases of " no nonlinearity" and "small " nonlinearity when SRM was applied to the System 2.

Figure 3.2.6 presents histogram of error in prediction of outputs by SRM. The horizontal axis is the error magnitude and the vertical axis is the number of output points at a particular error magnitude. Again, the large errors appearing in the histogram are due to insufficient information available initially when the neural network is "switched on" to predict the outputs.

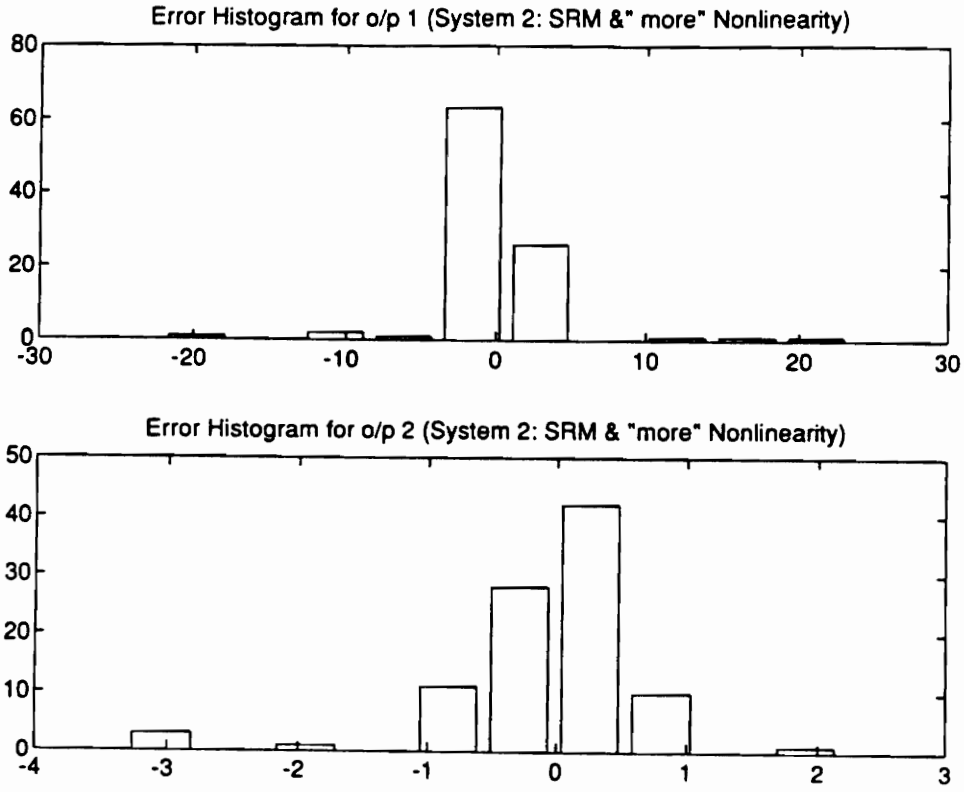


Figure 3.2.6 Error Histogram by SRM for System 2 with "large" (or, more) Nonlinearity

3.3 Identification of Class II Systems

Since we don't have information about (or even existence of) the linear region of system operation for Class II systems, we have to take into account possibilities of different number of delays. Thus, simulations were accomplished using different delays in the inputs and outputs. The results of applying WRM to the systems 3, 4 and 5 are epitomized in the Tables 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.3.5 and 3.3.6. Also, the results corresponding to application of SRM to the systems 3, 4 and 5 are shown are in the Tables 3.3.7, 3.3.8, 3.3.9, 3.3.10, 3.3.11 and 3.3.12.

Table 3.3.1 Simulation Results for System 3 by WRM (No. of Delays: 1 & 2)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
1	3	89.9063	1000
		91.3197	2000
		90.5531	1000
		91.0915	69000
		90.0478	2000
		91.7936	1000
	4	89.6502	75000
		90.6444	2500
		91.2384	1000
		90.5682	1000
		89.6894	29000
		91.8057	14000
	8	87.8720	2300
		89.3972	400
		88.3618	1600
		88.7692	200
		88.4751	500
	12	83.4710	19900
		87.9272	1100
		86.3299	200
		89.4392	500
		87.0109	800
		90.7103	1000
2	3	3.7711	18100
		3.8991	48100
		3.9559	49100
		3.8967	49100
		3.7641	38100
		3.9004	47100
	4	3.4427	41100
		3.0255	38100
		3.8467	45100
		3.8344	70100
		3.9592	71100
		3.1375	26100
	8	3.8166	2900
		3.2367	34600
		3.1520	8400
		3.4387	40700
		3.9342	8600
		3.2480	48300
	12	3.9123	9400
		3.2567	11100
		3.1437	7900
		3.1686	11100
		3.0237	15600

Table 3.3.2 Simulation Results for System 3 by WRM (No. of Delays: 3 & 4)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
3	3	3.1435	1200
		3.3924	74700
		3.1601	2500
		3.9939	6500
		3.2219	5400
		3.3210	1800
	4	3.0640	3100
		3.9386	74900
		3.3161	2900
		3.6280	1100
		3.1170	1800
		3.4511	2300
	8	3.2574	3100
		6.2067	4500
		3.8708	2400
		3.6530	4000
		3.6886	2700
		3.7833	1200
	12	3.9322	1400
		6.8014	34800
		3.1671	3100
		3.6054	6800
4	3	6.7216	1000
		7.7153	71700
		7.3114	1000
		6.9780	700
		8.0900	8500
		7.0616	600
	4	6.4435	700
		7.1283	800
		7.9113	1600
		6.9115	600
		7.5394	1300
		7.8371	3900
	8	3.8557	700
		7.7280	1000
		7.0907	1200
		6.7782	800
		6.7514	1000
		7.1775	1000
	12	6.5111	700
		7.2557	1300
		7.2620	900
		8.0231	1000
		8.3223	1600
		7.6169	1500

Table 3.3.3 Simulation Results for System 4 by WRM (No. of Delays: 1 & 2)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
1	3	103.3777	200
		103.5425	1000
		106.6917	71300
		103.5786	500
		103.6490	500
		103.7039	7100
	4	102.1673	44400
		103.0260	9400
		103.7481	4500
		106.0962	900
		103.5650	400
		106.7599	600
	8	102.2831	67100
		103.2028	2800
		103.8993	200
		106.8392	900
		103.8352	1800
		103.7271	600
	12	103.3742	500
		103.6063	400
		103.1920	800
		106.3959	600
		103.3325	200
		106.3543	500
2	3	13.1560	70000
		13.2133	68000
		13.1614	70000
		13.2127	66000
		13.1950	64000
		13.1690	74000
	4	10.6466	72000
		13.7262	13000
		12.6357	71000
		13.9305	19000
		10.7754	73000
		10.9710	10000
	8	7.9218	33700
		8.4449	75000
		9.2958	60600
		9.7768	39600
		9.8574	71900
		9.6696	74500
	12	7.5841	99800
		7.8117	99700
		9.1781	53900
		9.3230	47200
		8.3969	57900
		9.1451	73400

Table 3.3.4 Simulation Results for System 4 by WRM (No. of Delays: 3 & 4)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
3	3	12.1421	7000
		13.8771	73600
		12.5817	71800
		12.8441	73900
		12.5411	74300
		12.9141	73100
	4	10.7811	4100
		11.4041	5400
		12.9166	20700
		11.4605	11400
		11.3602	12500
		11.2247	9200
	8	7.8629	54800
		11.1130	14900
		9.4282	14300
		9.8536	16500
		8.7697	6200
	12	11.6037	18100
		11.6726	4800
		12.9316	73900
		12.6153	74500
		12.6126	71800
		12.7325	72800
4	3	11.3345	43200
		11.4835	74500
		11.6668	74500
		11.4727	72900
		11.3589	62300
		11.4908	74500
	4	10.4576	46300
		10.7603	27800
		11.0965	29100
		10.8147	15200
		10.7414	19900
		10.9479	7400
	8	9.3249	72700
		12.3001	66300
		10.5182	12100
		11.8866	58600
		10.4601	9300
		9.7589	3600
	12	10.2570	35200
		12.1927	14000
		12.4297	74800
		11.4831	17700
		11.0709	64600
		10.8925	27800

Table 3.3.5 Simulation Results for System 5 by WRM (No. of Delays: 1 & 2)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
1	3	107.6030	5300
		109.7495	16300
		110.5175	4800
		109.7880	8200
		110.5693	2100
		112.4799	500
	4	107.0290	1400
		108.3489	200
		110.5451	200
		110.1275	200
		108.7913	700
		107.5887	1100
	8	103.4044	6600
		108.9139	19700
		109.4539	1500
		109.0319	200
		103.5240	3900
		109.1779	2200
	12	103.2319	4200
		107.6136	900
		107.1427	1200
		103.3129	30300
		106.2159	1600
		103.6566	2000
2	3	13.0950	99100
		13.6635	100000
		13.6065	96800
		13.6128	91100
		13.6426	90000
		13.6161	99900
	4	10.7004	91800
		10.9700	99400
		13.0702	7000
		12.6633	96300
		13.4988	7400
		11.0229	89100
	8	7.6577	106600
		9.9642	86300
		9.1160	23500
		9.8713	43100
		9.6352	72600
		9.4368	53300
	12	7.4035	36500
		8.5698	147100
		9.3410	73000
		7.7866	71200
		8.5012	149600
		9.3201	50900

Table 3.3.6 Simulation Results for System 5 by WRM (No. of Delays: 3 & 4)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
3	3	13.0834	8100
		13.7298	74100
		16.6421	1300
		13.2181	70900
		13.5666	19800
		13.9876	66500
	4	10.3425	74700
		11.7827	3600
		13.2942	15200
		12.5439	6200
		11.7087	10400
		11.4347	74600
	8	8.2495	74700
		9.7240	22600
		9.7714	7400
		9.8665	65400
		10.6192	8000
		11.2627	23200
	12	8.4461	20300
		8.9839	9600
		10.4950	19100
		9.3025	54800
		9.8736	21900
		10.7412	31600
4	3	12.1786	73100
		12.4617	73800
		12.4997	74800
		13.3053	20000
		12.5080	70600
		12.6409	72900
	4	10.8272	8100
		11.0878	16700
		11.6133	74700
		11.9114	23100
	8	10.0821	6100
		10.8815	7200
		11.6061	13600
		11.7654	7000
		10.5567	10200
		10.4619	20200
	12	10.9358	14400
		11.6950	6000
		11.8172	38000
		12.4158	17500
		12.0996	7300
		12.8570	2800

Table 3.3.7 Simulation Results for System 3 by SRM (No. of Delays: 1 & 2)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
1	3	86.9346	1800
		88.9354	20400
		87.4791	100
		89.5823	200
		90.1126	800
		89.9408	600
	4	83.1825	100
		89.5620	200
		91.5193	4300
		89.2912	200
		91.0402	24000
		88.6911	2000
	8	83.2530	10600
		88.2032	21100
		89.4863	6000
		91.4023	4600
		88.3916	2500
		83.7084	22500
	12	87.1293	1600
		89.6604	7000
		88.5604	400
		88.7886	4100
		88.7679	600
		90.3858	300
2	3	3.0355	6900
		3.1069	9000
		3.0519	9900
		3.1167	800
		3.1936	500
		3.1936	500
	4	3.0671	500
		3.2561	500
		3.4428	1000
		3.1672	800
		3.3735	200
		3.1930	400
	8	3.2072	500
		3.4803	900
		3.3283	500
		3.3747	500
		3.2959	600
		3.4366	500
	12	3.3330	300
		3.7681	500
		3.55456	1000
		3.5919	1100
		3.4398	1400

Table 3.3.8 Simulation Results for System 3 by SRM (No. of Delays: 3 & 4)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
3	3	3.9033	300
		3.1768	200
		3.2036	74900
		3.0881	1900
		3.1649	400
		3.0409	400
	4	3.9639	74000
		3.0635	74600
		3.1807	700
		3.1558	400
		3.1756	74800
		3.0840	73500
	8	3.1624	75000
		3.0868	74900
		3.4712	400
		3.2078	73500
		3.1341	500
	12	3.3030	75000
		3.3546	500
		3.4937	1100
		3.5375	800
		3.5767	700
		3.3392	900
4	3	7.7761	62300
		7.7866	1200
		7.8699	1000
		7.8388	71300
		7.8471	800
	4	7.4767	200
		7.5804	500
		7.8138	1100
		7.7246	900
		7.6961	400
	8	7.5378	600
		7.8588	74900
		7.8982	74200
		7.9190	500
		7.8853	500
		7.7627	74800
	12	7.5643	600
		7.8099	500
		7.6925	400
		8.6108	700
		8.4764	600
		7.6529	1300

Table 3.3.9 Simulation Results for System 4 by SRM (No. of Delays: 1 & 2)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
1	3	97.3815	10500
		106.6226	400
		106.1307	1700
		100.2349	1900
		103.3448	100
		101.2700	2700
	4	99.1824	25000
		102.5120	800
		101.6292	24400
		101.4427	23700
		103.3523	2900
		103.9393	1700
	8	101.3614	24700
		102.6229	1400
		103.9560	400
		106.4567	600
		103.4556	1100
		103.6910	600
	12	98.3622	14800
		99.1315	24900
		106.1112	400
		106.4325	200
		103.2857	1400
2	3	12.2745	900
		13.7033	1300
		12.5621	13200
		13.6970	1200
		12.8117	3000
		13.9092	700
	4	13.1087	600
		13.6591	200
		13.1132	900
		13.6909	300
		13.0267	1500
		13.4418	800
	8	8.3048	7400
		8.4253	24100
		9.5447	6500
		8.6884	12100
		10.6229	2900
		8.8572	11300
	12	7.2161	5500
		7.2242	28900
		7.8468	7200
		9.1405	16900
		8.6519	4100

Table 3.3.10 Simulation Results for System 4 by SRM (No. of Delays: 3 & 4)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
3	3	7.7978	67500
		7.8274	800
		13.8584	200
		13.2477	900
		13.6372	300
		13.3867	800
	4	13.2320	37900
		13.0390	800
		13.3917	500
		13.6818	500
		13.6337	800
		13.1222	200
	8	7.6831	8400
		10.3354	74700
		8.7785	5300
		8.9113	60600
		8.0294	57800
		10.0169	8300
	12	8.6103	8200
		8.7147	3000
		9.9265	18600
		10.8209	2500
		8.7988	10000
4	3	13.2186	19100
		13.3942	7300
		13.5591	70400
		13.3160	200
		13.3774	800
		13.6765	74300
	4	10.7415	2200
		11.2921	5700
		13.1968	900
		16.0590	500
		13.1754	900
		12.4228	74300
	8	10.2215	1600
		10.2575	22100
		11.9894	6000
		11.5476	1800
		9.6620	6400
		10.4612	4200
	12	10.0718	10300
		10.5078	10800
		10.9957	2200
		11.5472	700
		11.7583	1700
		11.1060	73800

Table 3.3.11 Simulation Results for System 5 by SRM (No. of Delays: 1 & 2)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
1	3	106.4106	700
		107.8622	1300
		108.7412	24700
		108.4341	10800
		107.9186	2700
		107.5622	600
	4	103.1832	100
		107.3657	400
		109.5523	200
		110.0464	200
		107.2038	8200
		103.4409	2600
	8	102.3006	2800
		107.4289	1900
		108.1551	3000
		109.1224	200
		108.3009	300
	12	106.6841	500
		108.5244	500
		109.7239	4100
		107.8123	800
		108.1165	800
		108.3630	300
2	3	13.9977	2600
		13.2728	3000
		13.6908	1500
		13.2910	1400
		13.0082	500
		13.6383	1500
	4	13.7558	2700
		16.5641	13600
		13.4119	800
		13.7453	400
		17.9643	400
		13.8878	19400
	8	7.4364	8800
		10.9349	25000
		9.0273	13300
		8.8387	24200
		9.5120	6500
		8.9833	13100
	12	8.6847	6300
		11.0528	25000
		11.7560	25000
		9.2612	19500
		9.1634	15800

Table 3.3.12 Simulation Results for System 5 by SRM (No. of Delays: 3 & 4)

No. of Delays	No. of Hidden Layer Neurons	Frobenius Norm of Error in Test Data	Training Epochs
3	3	13.4835	500
		16.3801	800
		13.5171	400
		13.6224	1300
		13.5312	500
	4	12.6996	22300
		13.2795	400
		16.0627	400
		17.4410	400
		17.7769	1100
		13.6716	300
	8	9.0064	75000
		9.3265	74900
		10.1931	3900
		10.1969	22800
		11.5358	200
		9.6031	3000
	12	7.8748	74900
		8.9978	7400
		8.6989	5400
		11.3937	58500
		9.4912	5000
		9.7001	6300
4	3	11.5603	56400
		13.1582	16400
		17.0112	2200
		13.3097	22800
		16.2889	1400
		13.8668	300
	4	11.3605	71600
		11.6657	15100
		11.5021	7600
		17.2890	55000
		12.6522	59900
		12.0083	6500
	8	9.2290	10100
		11.4476	1500
		12.8004	800
		10.5006	2400
		11.1418	21900
		12.6750	1600
	12	10.3110	67500
		11.6785	5300
		12.4258	6200
		11.0357	5600
		10.4170	2200
		10.9279	38600

For each delay, simulations were run using different number of neurons in the hidden layer. To get an overall picture about the effectiveness of the methods, WRM and SRM, simulations were run using different initial conditions (initial weights) for the same number of hidden layer neurons. Again, it was observed that increasing number of hidden layer neurons decrease test data error to a certain point, and then test data error increases as the number of hidden layer neurons increase beyond this point. Different initial conditions lead to local minima after different training time.

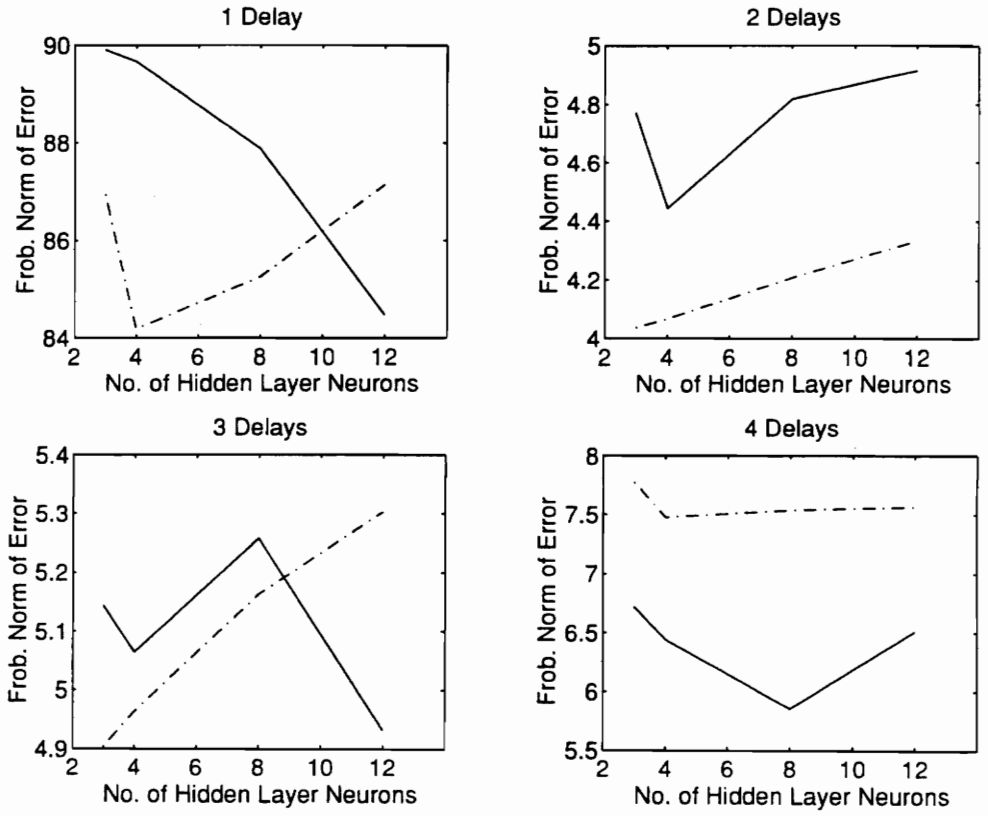


Figure 3.3.1 Comparison of Error Performance of WRM (-) and SRM (-.) for System 3

The results given by WRM and SRM for the Systems 3, 4 and 5 are compared in the Figures 3.3.1, 3.3.2 and 3.3.3, the basis of comparison being the Frobenius Norm of Error corresponding to the test data.

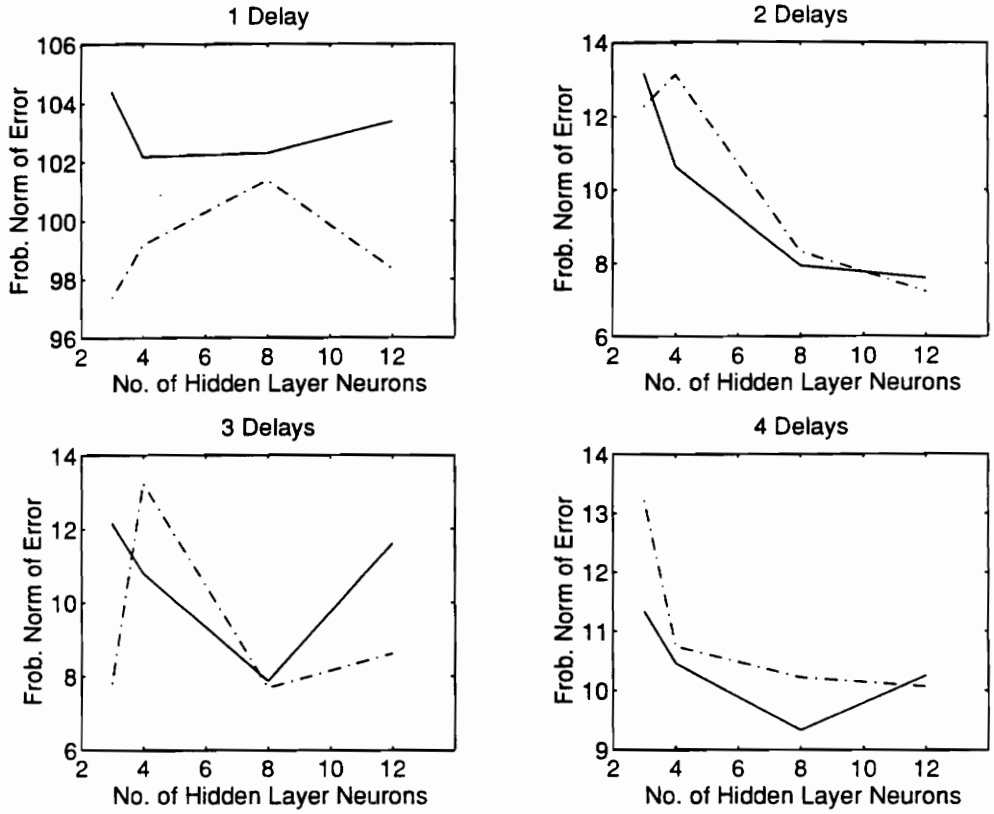


Figure 3.3.2 Comparison of Error Performance of WRM (-) and SRM (-.) for System 4

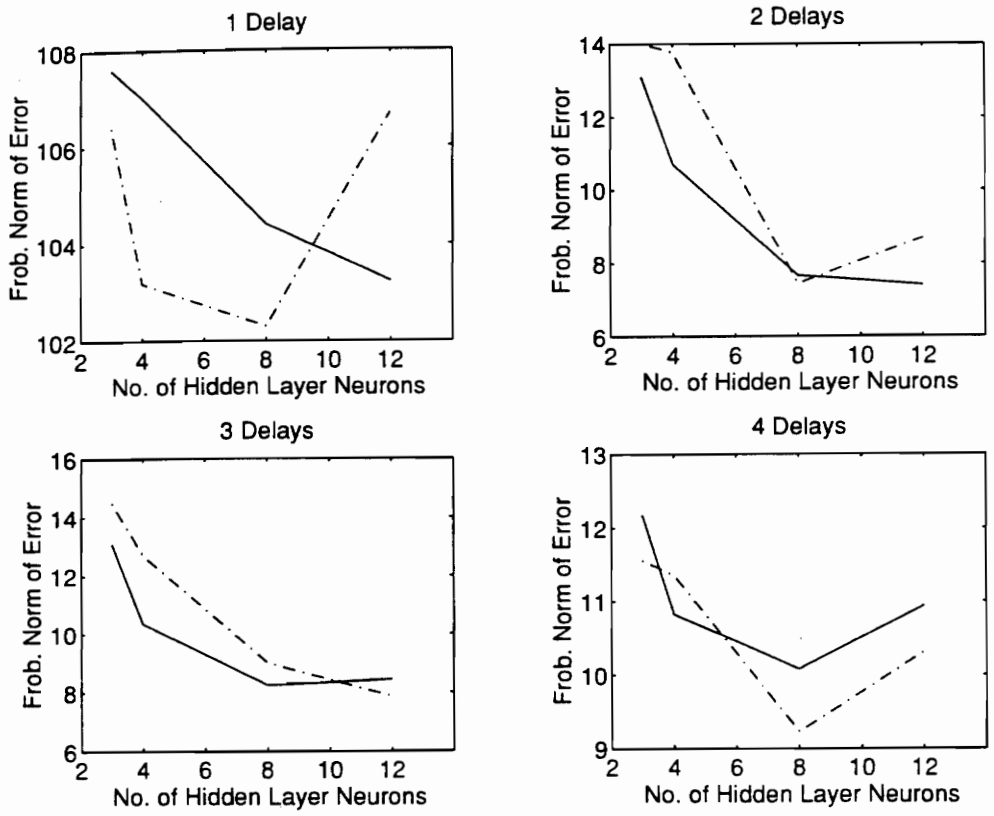


Figure 3.3.3 Comparison of Error Performance of WRM (-) and SRM (-.) for System 5

It can be concluded that SRM gives a slightly better prediction though the improvement is not significant. Also, the best results are obtained when two delays in inputs and outputs are used for all the systems. It can be observed that increasing the hidden layer neurons contributes to the error reduction till certain point and further increasing the number of neurons does not help reduce the error. On the contrary, the error grows large as the number of neurons become large, barring some exceptions. This

behavior can be attributed to the fact that excessive number of neurons might result in memorizing the patterns rather than generalizing the patterns.

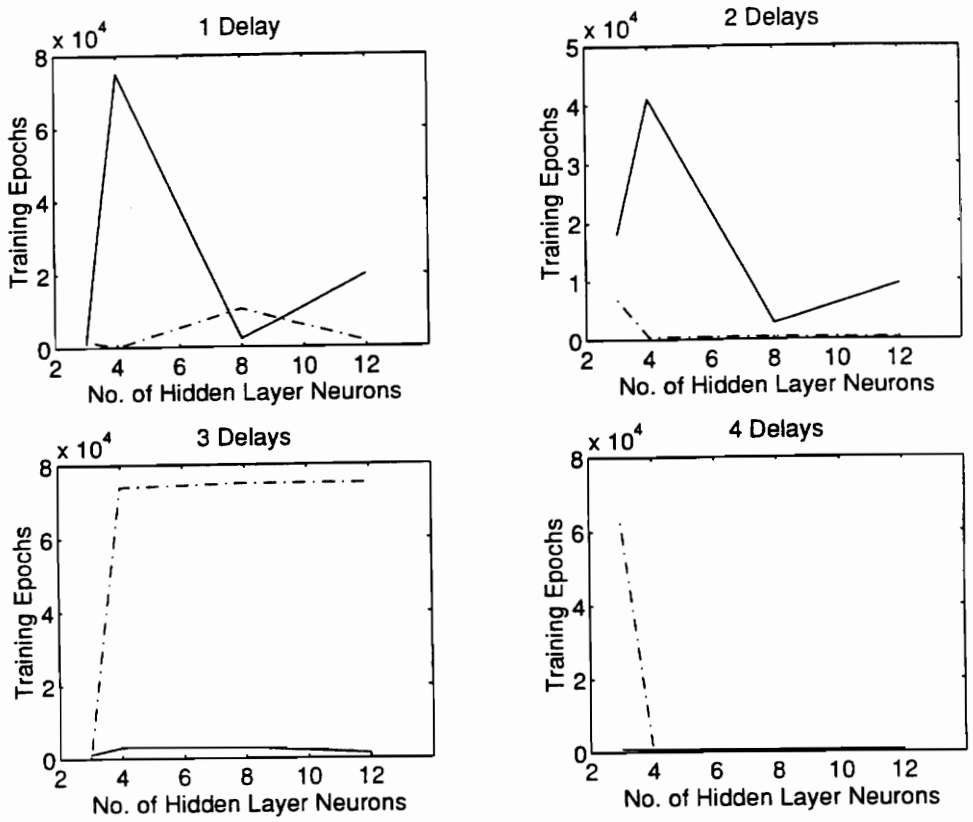


Figure 3.3.4 Comparison of Training Time Performance of WRM (-) and SRM (-.) for System 3

The results given by WRM and SRM for the Systems 3, 4 and 5 are compared in the Figures 3.3.4, 3.3.5 and 3.3.6, the basis of comparison being the time taken by the neural network to be trained to its best given the fixed training parameters.

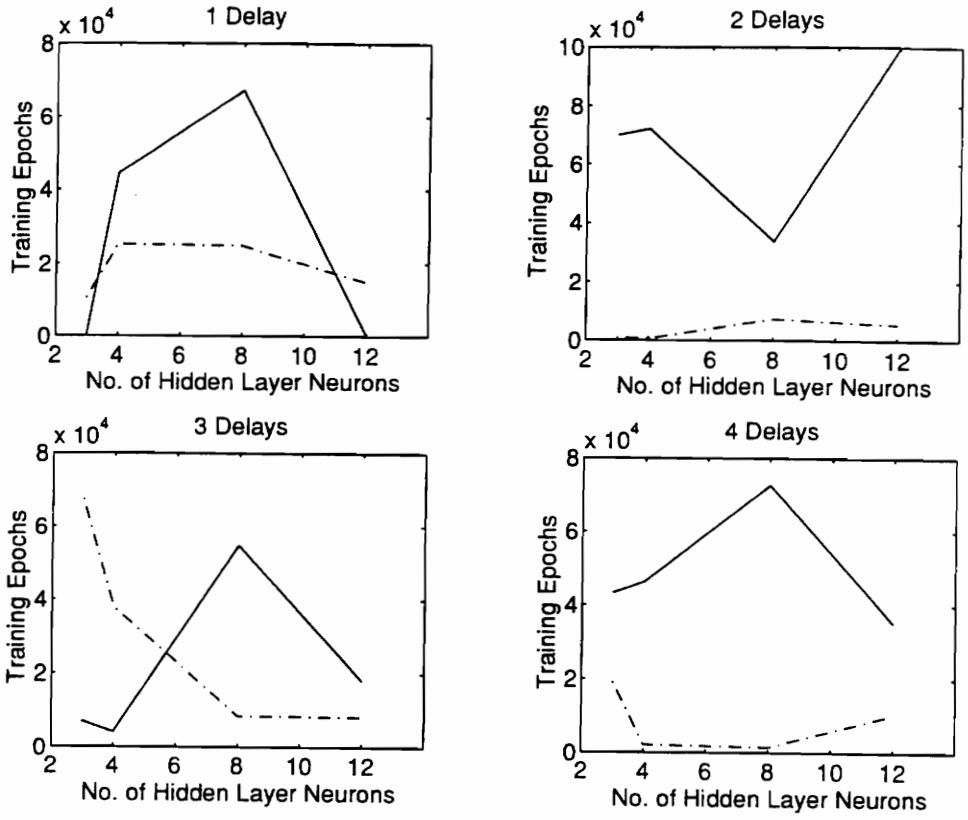


Figure 3.3.5 Comparison of Training Time Performance of WRM (-) and SRM (-.) for System 4

Though there is not a "stand-out" winner, SRM has less training time in the case when we are interested, i.e. the least Frobenius Norm of Error results corresponding to two delays in inputs and outputs.

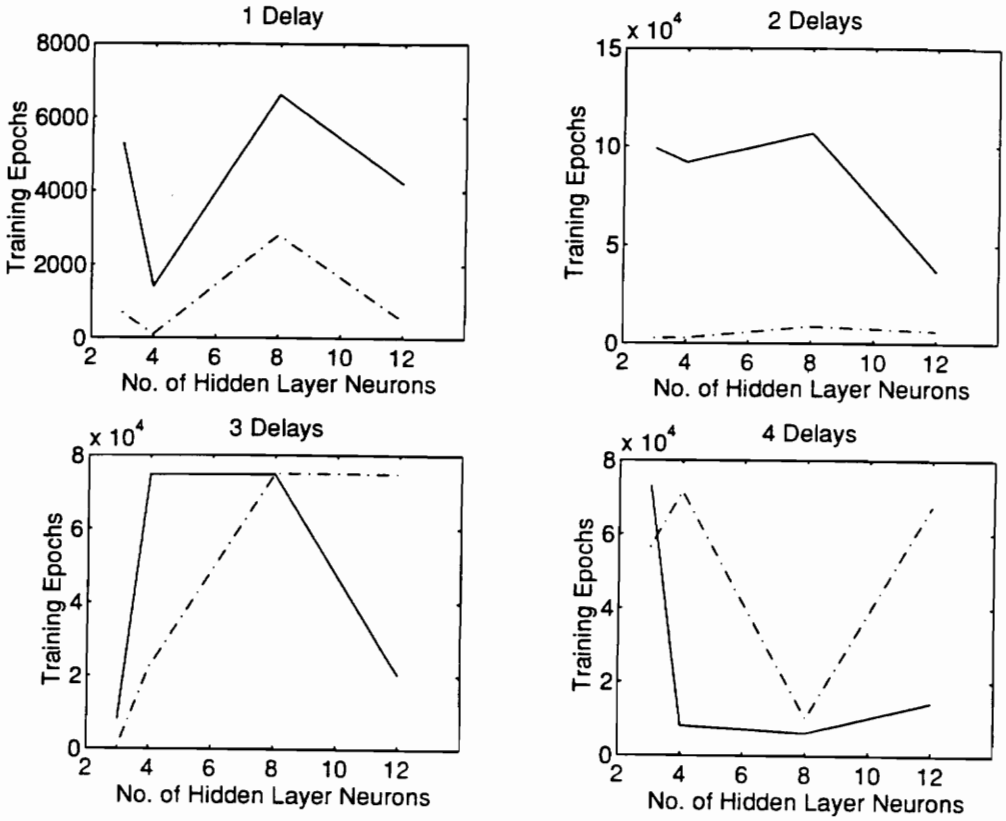


Figure 3.3.6 Comparison of Training Time Performance of WRM (-) and SRM (-.) for System 5

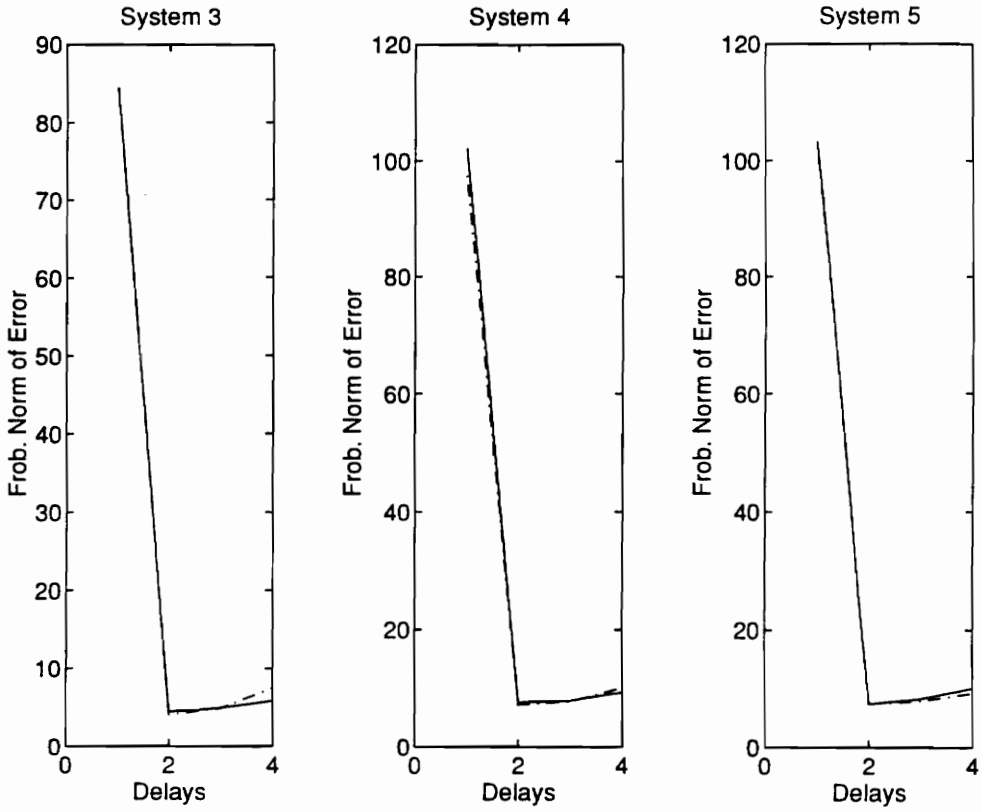


Figure 3.3.7 Effect of Delays (in Inputs and Outputs) on Error Performance of WRM (-) and SRM (-.) for Class II Systems

The effect of delays is readily evident from the Figure 3.3.7. Since the results corresponding to Delay 2 are substantially improved over the results corresponding to Delay 1 , we know that the actual nonlinear system would be nicely identified if we use two delays in inputs and outputs for neural networks. However, to be on safe side, we can increase one more delay and corroborate our earlier decision of using two delays. Four delays were used only to see how the identification is affected if "more than necessary" information used. In actual practice, simulations corresponding to this delay would not be necessary.

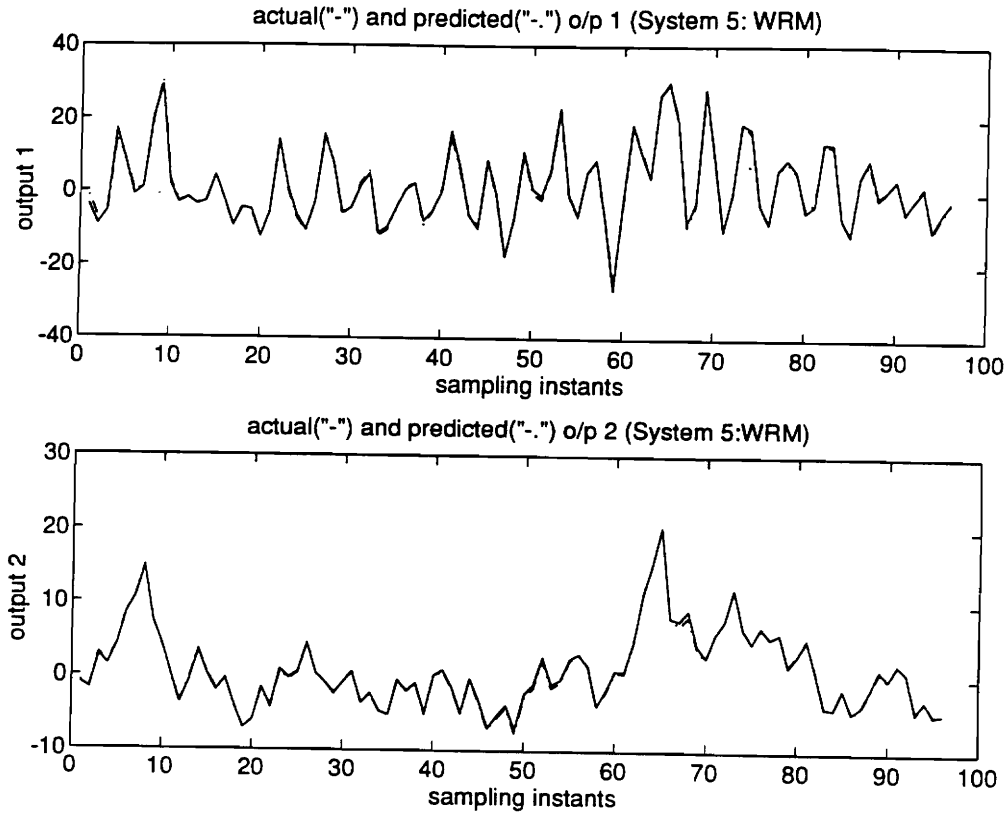


Figure 3.3.8 Comparison of Actual Outputs and Outputs Predicted by WRM for System 5

The actual and predicted outputs for System 5 using WRM and SRM are shown in Figures 3.3.8 and 3.3.9 respectively. In both the cases, there is a very good match between the actual outputs and predicted outputs.

3.4 Summary

The simulation results for identification of Class I and Class II systems using WRM and SRM were presented. Both the methods gave very good prediction of outputs. This proves that neural networks can capture nonlinear behavior of systems, which evades conventional techniques of system identification. In general, SRM gives somewhat better results than WRM. In particular, SRM has a clear upper hand in case of Class I systems.

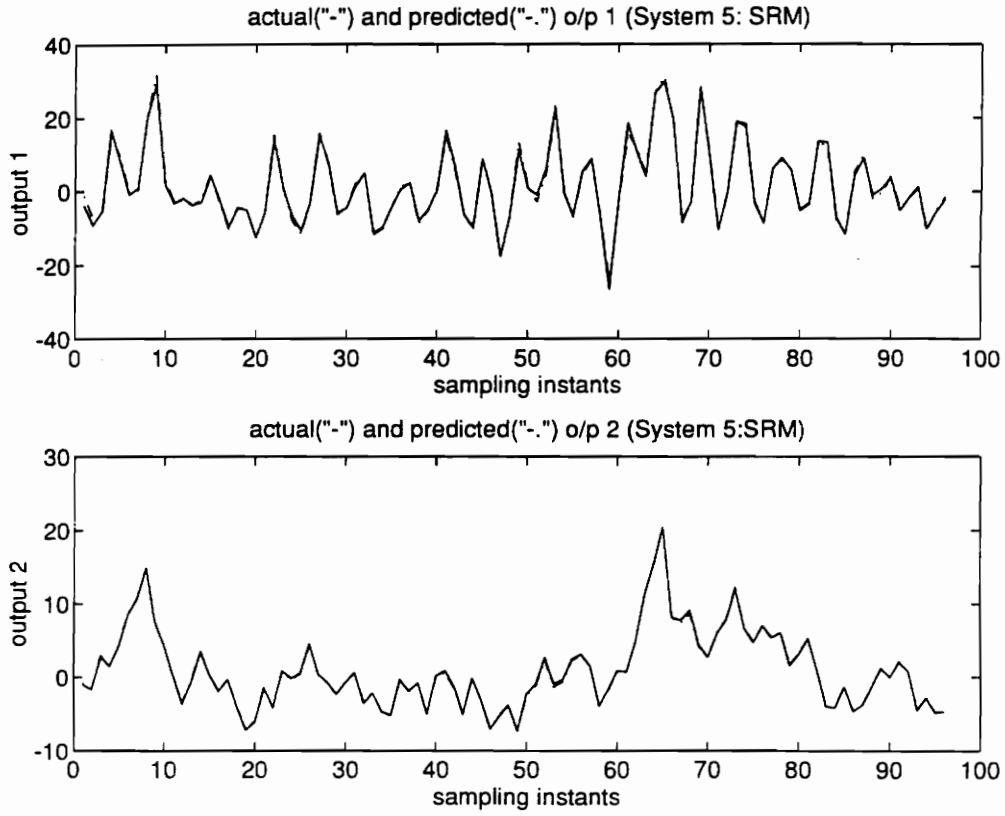


Figure 3.3.9 Comparison of Actual Outputs and Outputs Predicted by SRM for System 5

This is due to the fact that we knew the linear region of the nonlinear system and a powerful method of POI gave us an excellent model of the nonlinear system in the linear region. SRM usually gives better performance in terms of error and training time. Also, the number of simulations required to correctly predict the outputs of the nonlinear systems is reduced in case of Class I systems. The method of POI gives information about appropriate delays in inputs and outputs and hence simulations need to be run only for certain delays. However, in case of Class II systems we don't have such freedom. Thus, we can run simulations for several different delays and the magnitude of the error in test data would quickly indicate the right number of delays required to identify the system accurately.

Chapter 4
Automatic Control And Neural Networks

Though automatic control has been considered as a mature discipline, newer techniques continue to emerge. Conventional control techniques include frequency-response methods and time-domain methods. They are well-suited to linear SISO systems and quite successful in satisfying given performance criterion. However, they are less powerful for MIMO systems or complex nonlinear systems. Modern control theory was developed to fill this deficit. Optimal control theory and adaptive control can be viewed as representatives of modern control theory. Also, artificial intelligence has begun to appear in the domain of control. The emerging techniques of Fuzzy logic and Artificial Neural Networks ([2], [3], [6], [73], [76] etc.) have been attractive since they do not require structural information about the systems. This is significant in view of the increasing complexity of today's systems. This chapter first portrays the evolution of control technology. Nonconnectionist adaptive control methods have been explained since one of these techniques has been used later in a single-link manipulator control problem. There are several configurations of neural networks which are capable of learning the functional relationship between the inputs presented to them. After an ANN is trained, it can predict the inputs to be applied to the system so that desired system outputs can be achieved. The techniques to be investigated here are open loop control, neuro-PID control and adaptive connectionist control. ANNs will be used in developing a controller using these techniques. The basic steps of connectionist approach to control and several techniques and their importance are discussed.

4.1 Automatic Control: Primary Concepts and Evolution

A control system is an interconnection of components with a system which form a configuration that results in a desired response of the system configuration. A system or a process to be controlled can be represented by a block diagram shown in Figure 4.1.1.

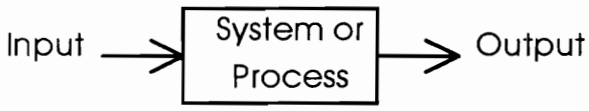


Figure 4.1.1 Representation of a System or a Process

Two fundamental types of control systems are: open loop control system and a closed loop control systems. They are shown in Figure 4.1.2.

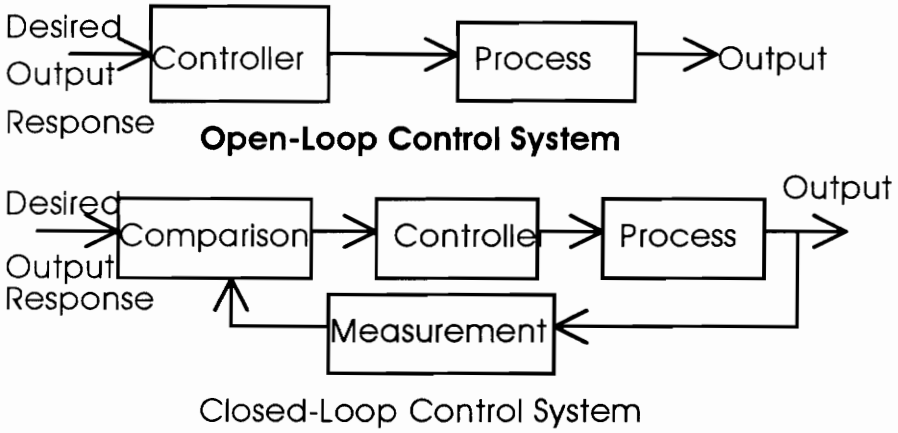


Figure 4.1.2 Open-Loop and Closed-Loop Control Systems

As seen from Figure 4.1.2, an open-loop control uses a controller to obtain desired output. The output is not fed back for comparison with the input. Open-loop control can

be used only if the relationship between the input and the output is known and remains the same. Closed-loop systems are called feedback systems. In the closed-loop system, the actuating error signal, which is the difference between the input signal and the output signal, is fed back to the controller so as to reduce error and bring the output to a desired value.

Automatic control has played an important role in the advancement of science and engineering. It is an integral part of numerous systems [55] such as space-vehicle systems, missile-guidance systems, aircraft-auto piloting systems, robotics systems, manufacturing processes etc. The first application of feedback control was used in the development of float regulator mechanisms in Greece in the period 300 to 1 B. C. ([49] and [50]). James Watt's centrifugal governor for speed control of a steam engine in the eighteenth century has been considered as the first significant work in automatic control and it is considered as the first automatic feedback controller used in an industrial process. Minorsky worked on automatic controllers for steering ships and evinced how stability could be determined using the differential equations pertaining to the system in 1922. In 1932, a procedure for determining the stability of a closed loop system on the basis of open-loop response to steady state sinusoidal inputs was devised by Nyquist. In 1934, Hazen discussed the design of relay servomechanisms capable of closely following a changing input.

In the 1940s, *frequency response methods* enabled engineers to design a linear closed-loop control systems that satisfied certain performance requirements. Before the end of 1950s, the *root-locus method* was developed by Evans. These two techniques forms the core of the *classical control theory*. The frequency domain technique was used primarily in the U.S.A. by Bode, Nyquist, and Black at the Bell Telephone Laboratories ([11], [12]). The time domain techniques using differential equations were developed in the U.S.S.R. During the 1950s, the utilization of both analog and digital computers for

control became possible. Modern control theory has been gradually developed to tackle the complex problems. The techniques within the domain of modern control theory include optimal control theory and adaptive control. More recently, the concepts from the area of Artificial Intelligence (AI) such as expert systems, neural networks and fuzzy logic have been utilized to control nonlinear and/or complex systems. The advent of optimal control theory renewed interest in the time domain methods due to Liapunov, Monorsky and several others.

Summary. In this section, the domain of control techniques was visited. The classical control theory has powerful techniques in time-domain and frequency-domain which are very well established, well understood and proven in practical applications. PID controller is popular in the industrial control. However, the scope of conventional control techniques is limited to linear SISO systems and hence these techniques cannot be extended to MIMO or complex nonlinear systems. Moreover, though they satisfy performance requirements, their behavior is not *optimal.* The optimal control methods provide control which is optimal in some predefined sense. Adaptive control is particularly suitable for the systems (linear or nonlinear) whose structure is known but their parameters are unknown. ANNs are attractive when the systems are complex and nonlinear and no information about the structure of the system is available. They can provide the desired behavior of the systems. However, it is extremely difficult to prove the stability of ANNs. Fuzzy logic simulates the human thinking in trying to perform the task. The ANNs and Fuzzy logic have been combined in novel ways to take the advantages of each one. The next section discusses nonconnectionist adaptive control.

4.2 Adaptive Control

Many dynamic systems are characterized by uncertain parameters that are constant or slowly-varying. Adaptive control is an approach to the control of such systems. An adaptive control can be regarded as a control system with on-line parameter estimation. Theoretical advances in the last decade, together with the availability of cheap computation, have lead to many practical applications in areas such as robotics manipulation, aircraft and rocket control, chemical processes, power systems, ship steering, and bioengineering [63]. An adaptive controller differs from an ordinary controller in that the adaptive controller parameters are adaptive (i.e. variable) , and there exists a mechanism to adjust these parameters on-line bases on the signals in the system. There are two main approaches for constructing adaptive controllers- Model-Reference Adaptive Control (MRAC) and Self-Tuning Control (STC). They are briefly sketched here.

Model Reference Adaptive Control (MRAC)

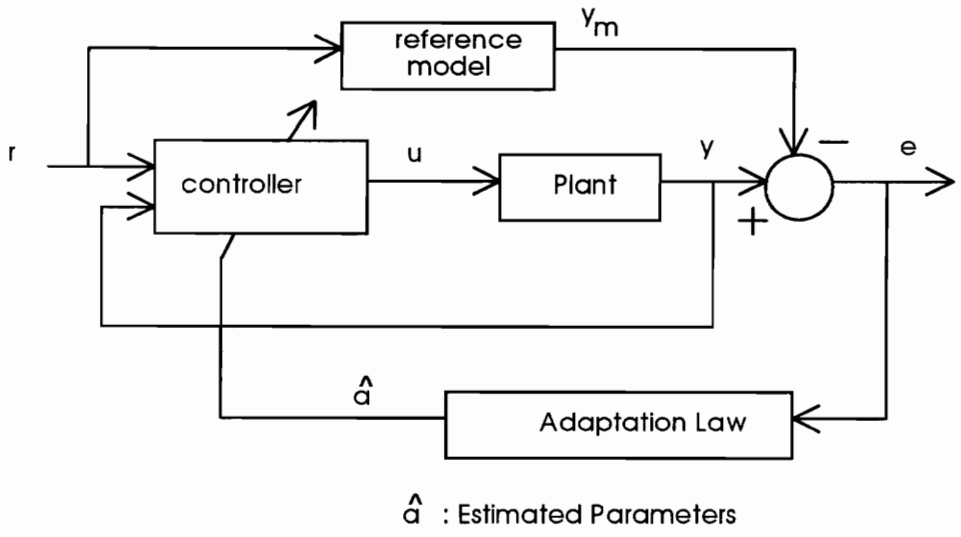


Figure 4.2.1 Model-Reference Adaptive Control (MRAC) System

The aim of the MRAC scheme is to nullify the differences between the behavior of the controlled process and the behavior of the reference model by using a suitable adaptation mechanism. The block diagram of a closed loop MRAC system is shown in Figure 4.2.1.

It consists of four parts: (i) a plant, (ii) a reference model, (iii) control law, and, (iv) an adaptation mechanism. The *plant* is assumed to have a known structure, although the parameters of the plant are unknown. It means that the number of poles and zeros of a linear plant or the structure of the dynamic equations of nonlinear plants (with unknown coefficients, of course) is known. Any model that assures the fulfillment of desired input-output performance can be chosen as a *reference model.* The choice of the reference model is part of the adaptive control system design. It should reflect the performance specifications and it should be achievable for the adaptive control system. The *controller* should have perfect tracking capacity meaning that the controller parameters should make the plant output identical to that of the reference model when the plant parameters are known. The existing adaptive control designs require that controller be parameterized by a number of adjustable parameters to guarantee stability and tracking convergence. The adaptation mechanism adjusts the parameters in the control law. The objective of adaptation is to make the tracking error converge to zero. The formalisms used for adaptation design include Lyapunov theory, hyper stability theory, and passivity theory. The details about MRAC can be obtained from [53].

Self Tuning Control

A controller obtained by coupling a controller with an on-line (recursive) plant parameter estimator is called a Self Tuning controller. Thus, an STC performs system identification of unknown plant. The block diagram of STC is shown in Figure 4.2.2.

At each time instant, the estimator sends to the controller a set of estimated plant parameters which is computed based on the past plant input u and y. The computer finds the corresponding controller parameters and computes a control input u based on the controller parameters and measured signals. This control input causes a new plant output to be generated, and the whole cycle of parameters and input/output updates is repeated.

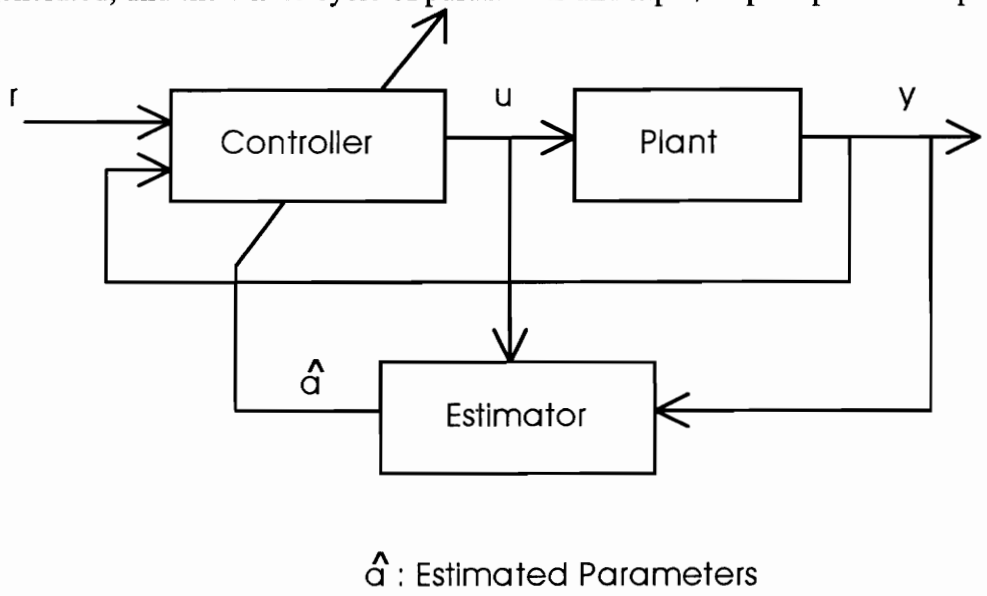


Figure 4.2.2 Self-Tuning Controller (STC)

Here, the controller parameters are estimated as if they were true plant parameters. Such an idea is referred to as *certainty equivalence principle* [63]. Since plant parameters are estimated first, and then they need to be translated into the controller parameters, STC is often called an *indirect adaptive control.* STC details can be found in [4].

4.3 Basic Steps for Control Using Neural Networks

Figure 4.3 shows a schematic diagram of the problem of controlling a nonlinear system.

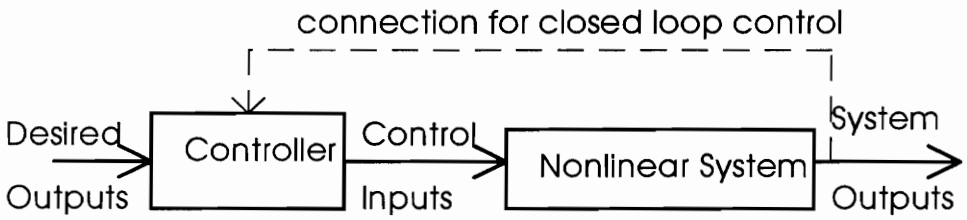


Figure 4.3.1 Control of a Nonlinear System

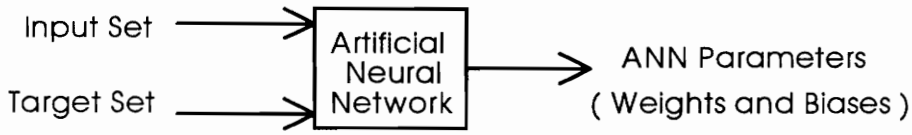
Thus, we need to find a control law such that the control inputs determined using the control law force the system to produce the desired outputs. for example, the control law may be specified by the proportional gain, integral gain and derivative gain in case of a PID controller. When we do not have information about the structure of the nonlinear system or when the system parameters change with time or operating regions, ANN can be used since the use of an ANN is a *model-free* approach. Followings are the basic steps to be performed to control a system using ANN (the approach which uses ANNs is called a connectionist approach.). These steps are very similar to those described in conjunction with the system identification.

(i) Collection of data set.

It is the elementary though very important step. Input variables and output variables of the system are observed. Since an ANN has to learn the relationship between the system inputs and outputs, sufficient care must be exercised to ensure that the collected data is representative of the actual system. The knowledge of the system may be of help in determining the variables to be measured.

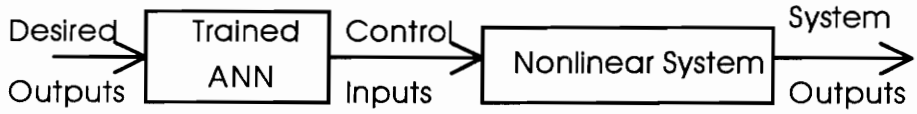
(ii) Off-line Training of an ANN

A suitable ANN can be chosen from several ANN configurations introduced in Chapter 2. The issue of choosing a network topology was discussed while dealing with the system identification. Those considerations apply here too since the problem in control is to find the "inverse" system identification, which is nothing but the process of extracting the relationship between system inputs and outputs. In general, the 2-layer backpropagation network should suffice for most applications. The primitive form of a controller using ANN is shown in Figure 4.3.2. For control application, Input Set contains system outputs (along with delays and delayed system inputs, if necessary) and Target Set contains system inputs (which are also control inputs) .



Input Set: System Outputs (and delayed system inputs & outputs, if necessary)
Target Set: Control Inputs or System Inputs

Training of an ANN



Trained ANN as a Controller

Figure 4.3.2 Primitive Form of a Connectionist Controller (Open Loop Controller)

The ANN is first trained off-line. The motivation behind the idea of off-line training is due to several reasons. The relationship between system inputs and outputs of the system may not be easy to extract in real-time. Also, the training backpropagation learning algorithm is, in general, slow. If an ANN is trained off-line, it can be trained well since there are not severe time-constraints and training can be extended to a longer period to ensure good generalization property if necessary. After it is trained off-line, it will have the weights and biases that would be good representation of the relationship between system inputs and outputs. Then, these parameters may be altered *on-line* depending upon the performance of the controller (which is nothing but the trained ANN). Since we already have a reasonably good representation of the system (in form of off-line trained ANN), the training time to modify the ANN parameters only slightly will be very short (compared to the time it would have taken if it had not been trained off-line.). Another advantage is that the parameters of the off-line trained ANN represent the overall behavior of the system over the entire range of operation (in which the system will be operating). If an ANN is trained *only on-line*, we may get a very good representation of the system over small operating range of the system and as we continue to shift the operating point of the system, the system behavior may change significantly, thereby demanding "large" variations in the ANN parameters to accommodate the new system behavior. This will prolong the training time and may even result in unstable learning.

(iii) Selection of a Control Technique

After an ANN is trained off-line, we can explore several possible control strategies to get the desired performance of the system. As seen from Figure 4.4.2, no information from the actual output of the system is utilized in determining the control inputs to the system. Hence, this strategy is called open loop control. If the system is nonlinear but simple in dynamics, this strategy may suffice. It has several advantages that It is simple

and does not require any additional computations. However, it is not suitable for systems which are complex and whose parameters are variable. Then, we need to find some mechanism that will allow us to control the system in the anticipated range of operation. The next two sections describe two techniques, Neuro-PID control and Adaptive Connectionist Control, which incorporate feedback (from actual outputs) to improve controller performance.

4.4 Neuro-PID Controller

This approach combines the classical control approach with the recent ANN approach. The PID controllers are very popular in the industrial control. Usually, they are used in such a manner that the tuning of their parameters results in a control system satisfying some performance criterion [72]. They are used around some operating point of the system. They try to reduce the difference between the desired output and the actual output to zero by employing proportional, integral and derivative actions. However, they are useful only in some region around the operating point since they make the assumption of linear system. An off-line trained ANN, discussed earlier, may not give the perfect performance (for example due to insufficiently rich inputs presented during training, or a small change in system parameter). However, they have captured the overall behavior of the system. Thus, the off-line trained ANN can determine the control input to be applied to the system that will force the system to produce the output that will be close to the desired output. We can employ a PID controller to reduce the difference between the actual output and the desired output. Since, the PID controller has to operate only in a small region, its gains can be set accordingly to give the fast correction to the control input given by the ANN. The arrangement of Neuro-PID control is shown in Figure 4.4.1.

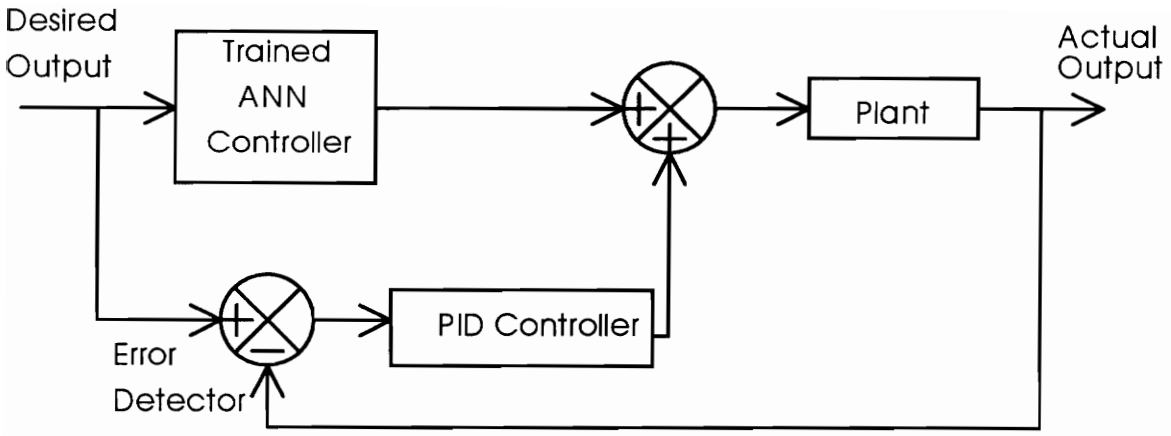


Figure 4.4.1 Neuro-PID Control

Thus, the function of the PID controller here can be considered as fine-tuning. This approach is suitable to only single-output systems because PID controller can handle only a single variable at time. In MIMO systems, there are interactions between several system inputs and outputs and hence this approach may not be useful. However, since the industry is dominated by PID controllers, this approach is applicable to the systems where a PID controller is currently used. This approach has a significant advantage in that the range of system operation can be enhanced to a great extent without worrying about nonlinear behavior of the system. Also, the settings of PID controller can be made using the simulation itself, and hence the *tuning* of PID controller becomes easy. Moreover, the gains of the controller will be relatively small (since majority of work is done by ANN controller). Also, we can pick up better gains of PID controller since we are only concerned with small range of PID controller operation. If we had not used the ANN, then more care would have to exercised since a large range of operation had to be considered.

4.5 Adaptive Connectionist Controller

This approach is generic in the sense that it can be applied not only to the single output systems but also to the multivariable systems. The arrangement for adaptive connectionist control appears in Figure 4.4.1.

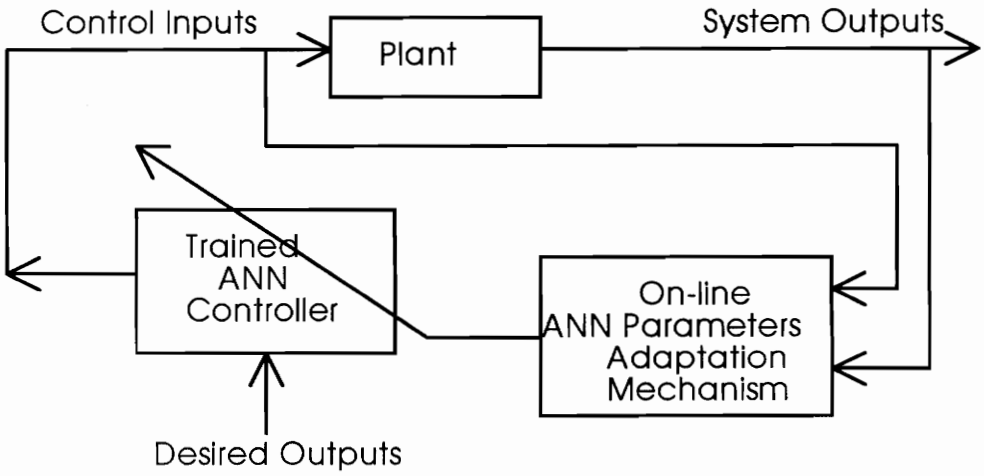


Figure 4.5.1 Adaptive Connectionist Control

In this approach, the control inputs calculated by the ANN controller are applied to the plant. The control inputs and actual outputs are measured and the ANN controller's parameters are modified by the on-line ANN parameters adaptation mechanism according to the performance of the ANN controller. The on-line ANN parameters adaptation mechanism can be easily implemented in software. It has the form of a backpropagation training algorithm which uses the actual outputs as input set and control inputs as target set. An additional watch should be put on this adaptation mechanism such that it does not modify the parameters of ANN if they are giving good performance. This approach is quite general and hence applicable to a wide variety of control problems including the

complex nonlinear systems with varying parameters. By virtue of adaptation mechanism, the adaptive connectionist controller can adapt to the new "state" of the system. Thus, initial training need not be very good. This indicates that this approach is quite fault-tolerant since it tolerates the insufficiency in training data presented to the ANN during off-line training.

4.6 Summary

The basic steps for connectionist control were outlined. Three forms of control techniques, Open Loop Control, Neuro-PID control and Adaptive Connectionist Control, were discussed. Open loop control is applicable to simple systems where we have enough confidence about our knowledge of the system behavior. Usually, some form of feedback is necessary in practical systems. The other two connectionist approaches employ feedback to improve the controller performance. In neuro-PID control, the off-line trained ANN makes the system give an output close to the desired value of the output. The PID controller then fine-tunes the output. This approach is applicable to nonlinear single output systems and can be viewed as an *operating range enhancer* of the existing PID controllers. The last approach, adaptive connectionist approach, is quite generic and can be used with nonlinear complex time-varying systems.

Chapter 5
Simulation Results of Control Problems

The previous chapter presented some connectionist techniques. Those techniques were applied to two systems, a single-link manipulator and a single-link robot with joint flexibility and damping. The results obtained after application of control techniques are presented here. It will be noted later that the connectionist control approaches proved to be very effective. In fact, the connectionist approach could solve the problem which was not solvable by adaptive control technique with parameter estimation. It must, however, be emphasized that the success of ANNs in dealing with challenging control problem is , to a great extent, dependent on the available data. If the input and output data of a complex nonlinear system is available, an ANN is definitely an excellent tool to be experimented with. The nicety of the connectionist approach is that we do not require the structural information of the system. Such information, even if partially available, can of significant help in that it can help determine the input set and target set of the nonlinear complex system. In general, only minimal information about the system would be necessary while following a connectionist approach to the control problem.

5.1 Problem Statement

Two systems to be controlled are a single-link manipulator and a single-link robot with joint flexibility and damping. These systems and the control objectives are defined next.

5.1.1 System 1: A Single-link Manipulator

A single-link manipulator can be represented by the following equation:

$$I\,\ddot{q}(t) + M g L \sin(\theta) = u \qquad (5.1.1)$$

where,

I = Inertia

M = Mass

g = Gravitational acceleration

L = Length of the link

u = Control input

I, M and L are unknown. A simple schematic diagram of the link is shown in Figure 5.1.1.

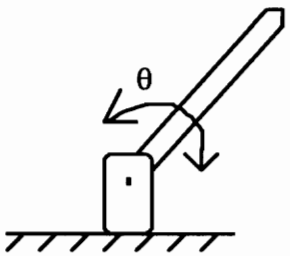


Figure 5.1.1 A Single-link Manipulator

The objective is to determine the control input u such that it follows a particular trajectory during a time period of 10 seconds. The range of the reference trajectory is 0 to 0.8 radian.

5.1.2 System 2: A Single-link Robot with Flexible Joint and Damping

A single-link robot arm with flexible joint and damping can be represented by the following equations:

$$I\ddot{\theta}_1 + B_1\dot{\theta}_1 + MGL\sin(\theta_1) + k(\theta_1 - \theta_2) = 0 \qquad (5.1.2)$$

$$J\ddot{\theta}_2 + B_2\dot{\theta}_2 - k(\theta_1 - \theta_2) = u \qquad (5.1.3)$$

where,

I, J = Inertia

B_1,B_2 = Viscous friction coefficients

M = Mass

G = Gravitational acceleration

L = Length of the link

k = Spring stiffness

θ_1 = Link angle

θ_2 = Motor shaft angle

I, J, M, L, k, B_1 and B_2 are unknown. Figure 5.1.2 shows a robot link with joint flexibility and damping The objective is to make the link angle and motor angle follow a desired trajectory as closely as possible. The range of the reference trajectory is 0 to 0.8 radian.

The next section presents the simulation results obtained for above two systems.

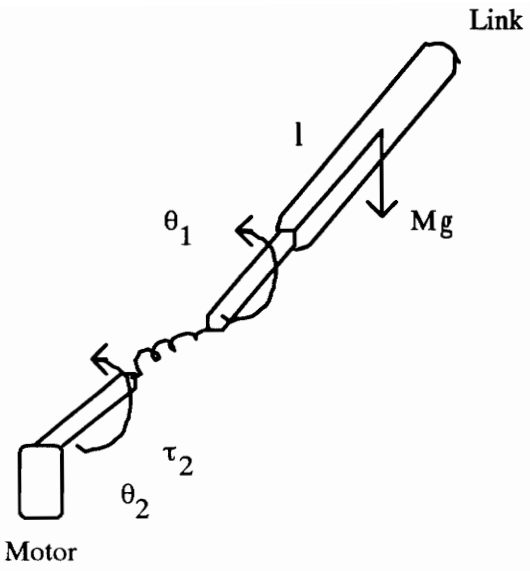


Figure 5.1.2. A Single-link Robot Arm with Joint Flexibility and Damping

5.2 Simulation Results for System 1

Several techniques such as nonconnectionist adaptive control (using Slotine & Li algorithm), open loop connectionist control, neuro-PID control, and adaptive connectionist control were applied. The results are summarized in Table 5.2.1.

Table 5.2.1 Comparison of Performance of Different Control Strategies for System 1

Control Strategy	Frobenius Norm of Error over Entire Range of Operation	Final Position Error
Nonconnectionist Adaptive	1.077	0.0129
Open Loop Connectionist	3.8959	0.0116
Neuro-PID	1.1990	0.0084
Adaptive Connectionist	0.5374	0.0074

The results corresponding to each technique are presented first and they are compared in the last subsection.

5.2.1 Nonconnectionist Adaptive Control

The adaptive control using the algorithm of Slotine and Li was implemented. Following control law was used:

$$u = \hat{I}\,\dot{v} + M\hat{g}L\sin(\theta) - K_D r \tag{5.2.1}$$

$$v = \dot{\theta}^d - \Gamma(\theta - \theta^d) \tag{5.2.2}$$

$$r = \dot{\theta} - v = \dot{e} + \Lambda e \tag{5.2.3}$$

$$e = \theta - \theta^d \tag{5.2.4}$$

where the caret indicates the estimated value of the parameter.

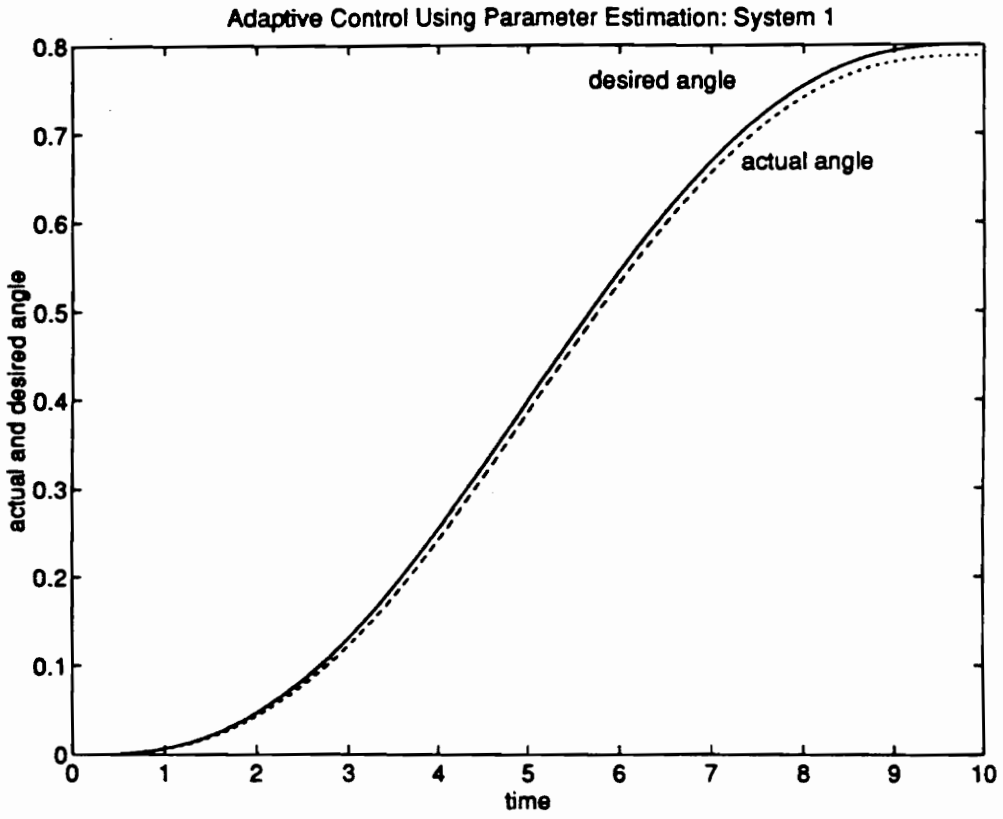


Figure 5.2.1 Nonconnectionist Adaptive Control for System 1

Figure 5.2.1 shows the desired trajectory and the actual trajectory. The angles are in radians. There is a good tracking performance. The tracking error over the entire trajectory and the steady-state error are listed in Table 5.2.1.

5.2.2 Open Loop Connectionist Control

An ANN was trained using the data set in which position, velocity and acceleration constituted an input set and the control input formed a target set. These data sets were obtained by applying random inputs to the simulated system.

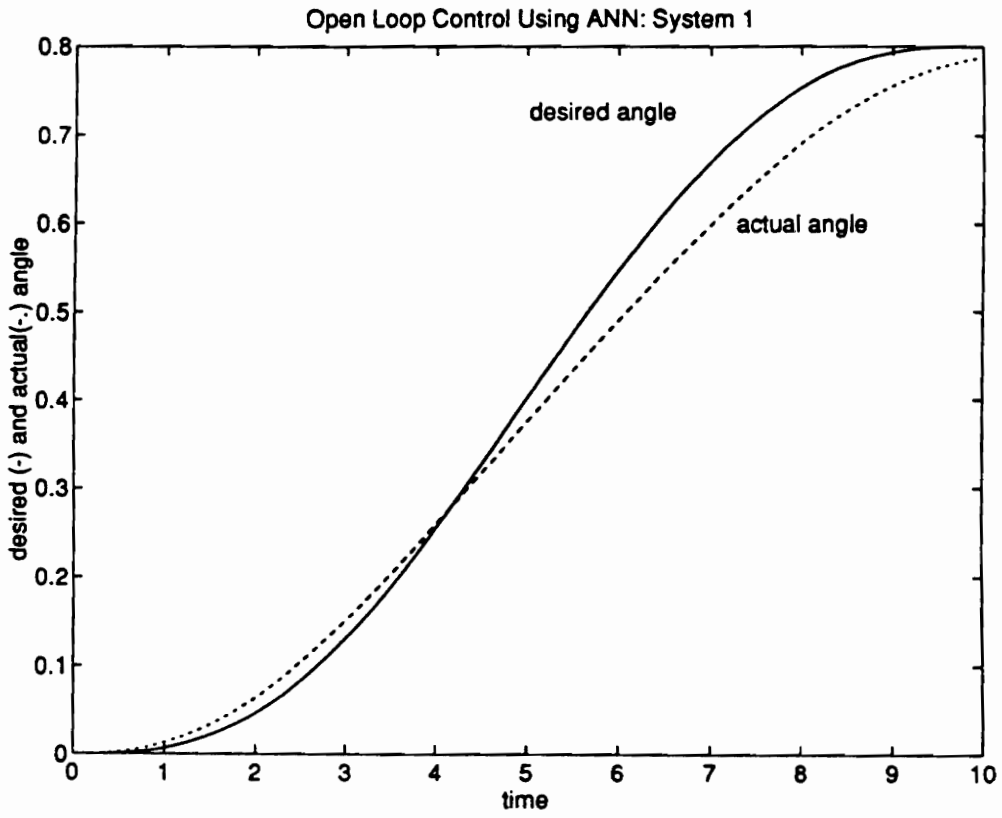


Figure 5.2.2 Open Loop Connectionist Control for System 1

A 2-layer backpropagation network was used, with tangent-sigmoid and linear transfer function in the first and second layer respectively. 5 hidden layer neurons gave satisfactory performance with test data. The trained ANN was used to calculate the control input to be given to the plant (i.e. the link). The desired angle and the actual angle are shown in Figure 5.2.2. The actual angle does not follow the desired angle very closely. However, it follows the trend of the desired angle. This gives the clue that some adaptive mechanism might capture the system dynamics which evaded the off-line training. The tracking error over the entire trajectory and the steady-state error are listed in Table 5.2.1.

5.2.3 Neuro-PID Control

The concepts of connectionist control and PID control were combined to develop this control. Here, the off-line trained ANN (used in open loop control) gives one component of the control input. Another control input comes from the PID controller whose output is based on the difference between the actual angle and the desired angle. The proportional gain of 5, the integral gain of 2.5 and the derivative gain of 0.6 along with an additional gain of 1.5 gave good results, as seen from Figure 5.2.3. Again, the tracking error over the entire trajectory and the steady-state error are listed in Table 5.2.1.

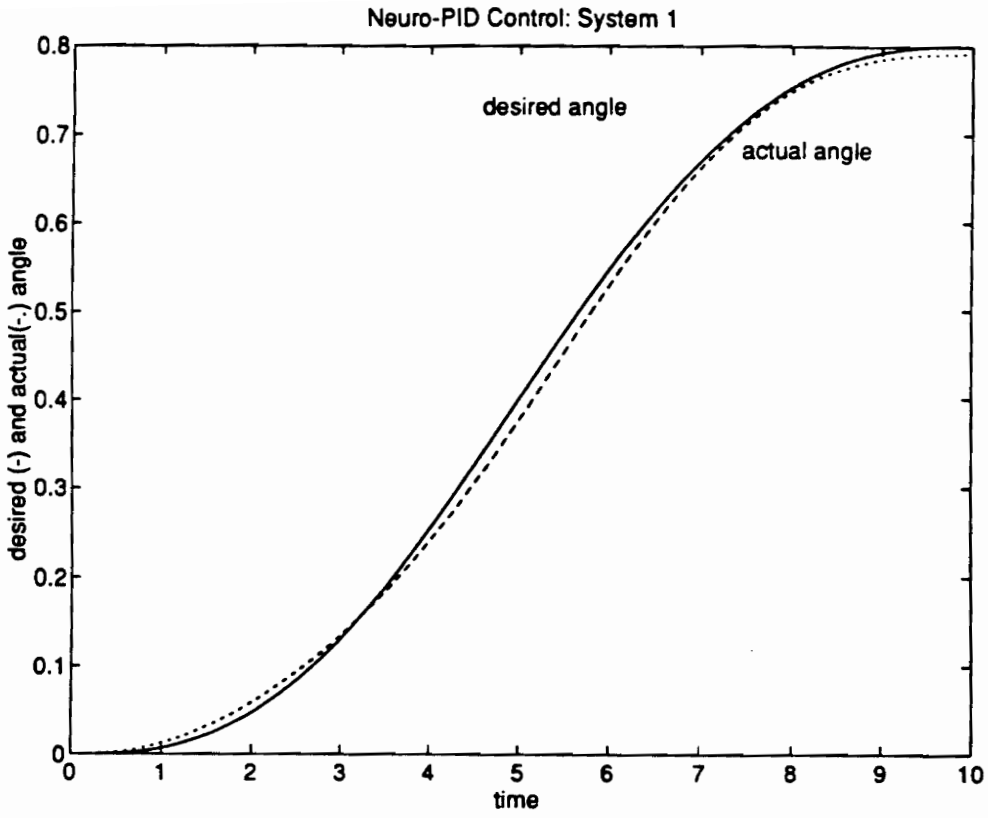


Figure 5.2.3 Neuro-PID Control for System 1

5.2.4 Adaptive Connectionist Control

This strategy utilizes still another form of feedback from actual output. The off-line trained ANN calculates the control input to be supplied to the link. Also, periodically its parameters are changed based on the measurements of actual output and the control input. The results corresponding to this strategy appear in Figure 5.2.4. Also, the tracking error over the entire trajectory and the steady-state error are listed in Table 5.2.1.

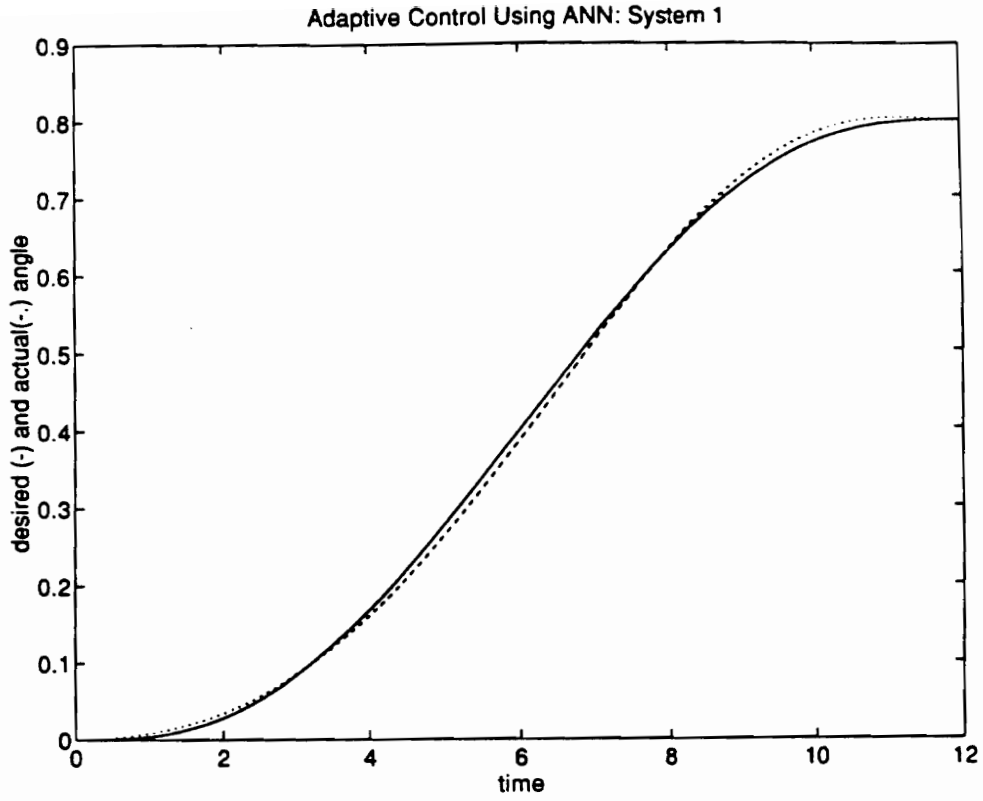


Figure 5.2.4 Adaptive Connectionist Control for System 1

There is, indeed, a very good match between the desired angle and the actual angle. The history of control input is shown in Figure 5.2.5.

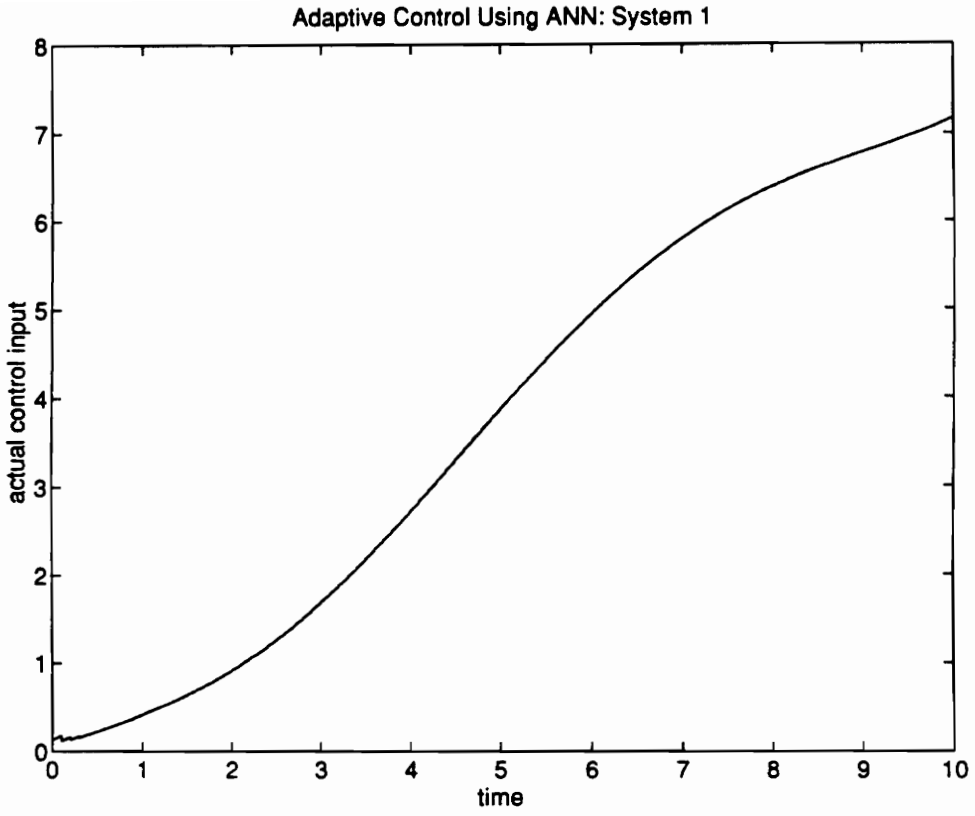


Figure 5.2.5 History of Control Input: Adaptive Connectionist Control for System 1

5.2.5 Comparison of Control Strategies

The nonconnectionist adaptive control gives good results, but it should be noted that it requires some knowledge concerning the approximate values of system parameters . If the structure of the system dynamics is not known, this method can not be applied at all. Open loop control was not good because of insufficient information present during the training. Neuro-PID controller gives very good performance taking into account both the

computational effort and accuracy. It requires only the off-line training of an ANN and the PID gains can be set very easily using the simulation itself. In cases where on-line training of an ANN (i.e. the adaptive connectionist approach) does not produce good results due to unavailability of sufficient training time, neuro-PID control may be the best choice for a single output systems. The adaptive connectionist control technique gives the best performance in terms of both the error over the whole range of operation and the final error. It is reasonable to expect this because the parameters of the ANN controller are modified on-line, capturing the system dynamics which were not learned during the off-line training.

5.3 Simulation Results for System 2

Several techniques such as adaptive nonconnectionist control, open loop connectionist control and adaptive connectionist control were applied to control system 2. The nonconnectionist adaptive control is unable to give the satisfactory performance. When this technique is applied to system 2, the link angle becomes oscillatory or the system shows instability [68]. Figure 5.3.1 shows the history of the link angle when adaptive nonconnectionist control technique is used.

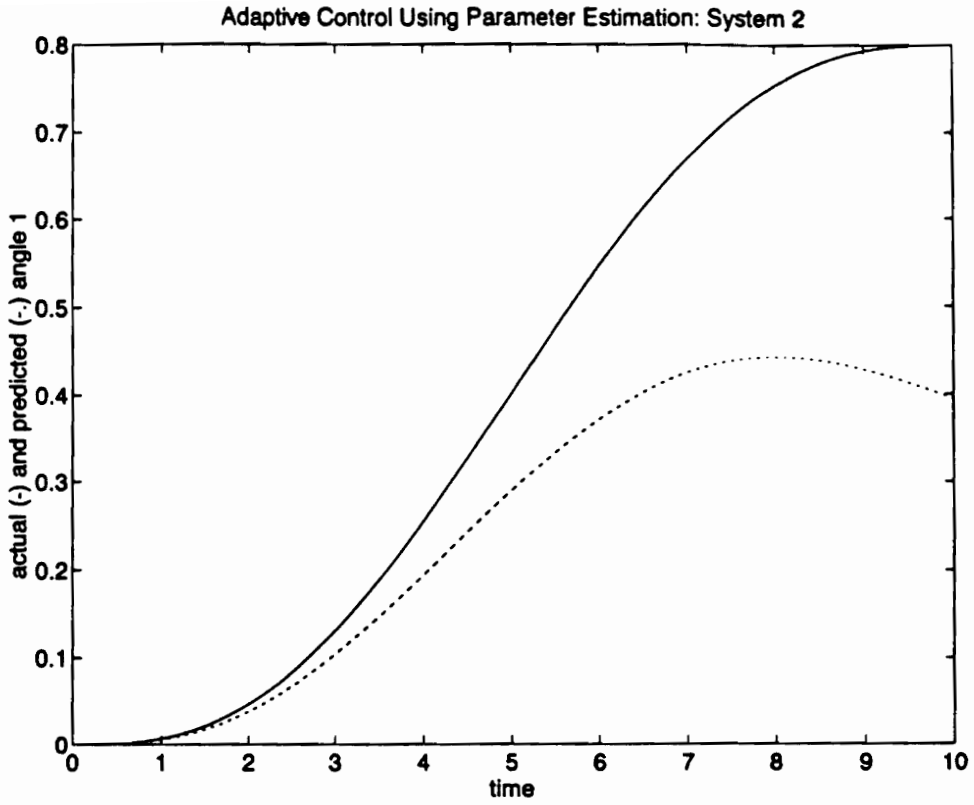


Figure 5.3.1 History of Link Angle: Adaptive Nonconnectionist Control for System 2

Figure 5.3.2 shows the history of motor angle when the adaptive nonconnectionist control technique is used.

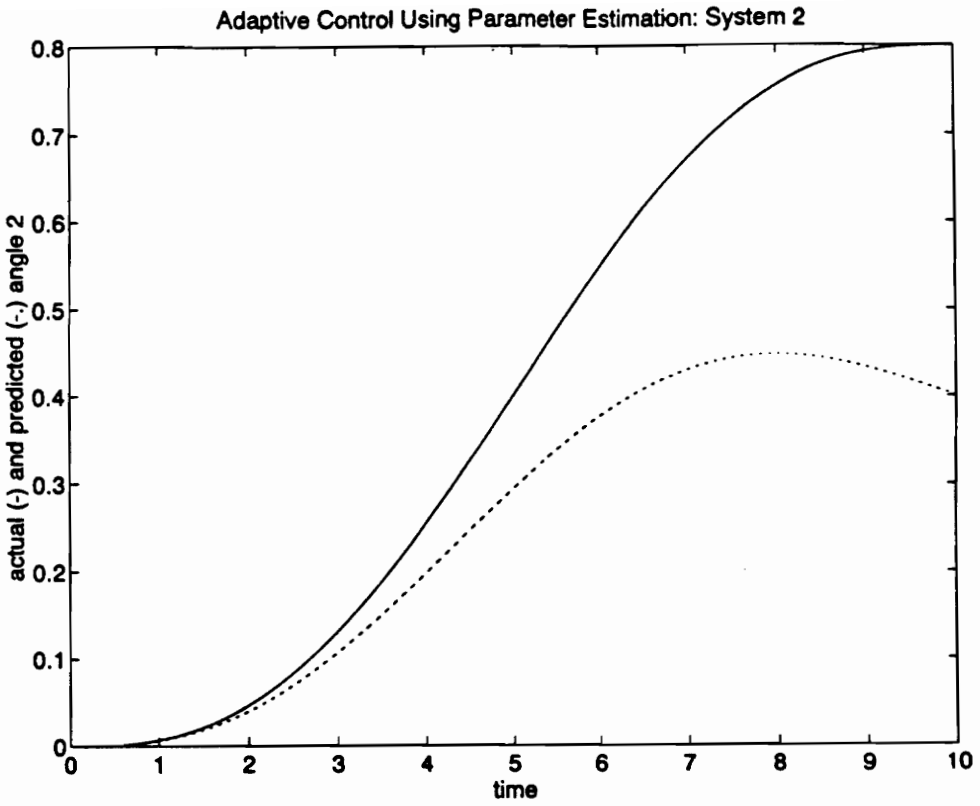


Figure 5.3.2 History of Motor Angle: Adaptive Nonconnectionist Control of System 2

Open loop connectionist approach is also ineffective. Neuro-PID control can not be used for system 2 since the system has two outputs and only one input to control both the outputs. The adaptive connectionist approach was quite successfully applied to the system 2.

As seen from Figure 5.3.3 the link angle follows the trend of the desired trajectory and does not oscillate. There is a small steady-state error.

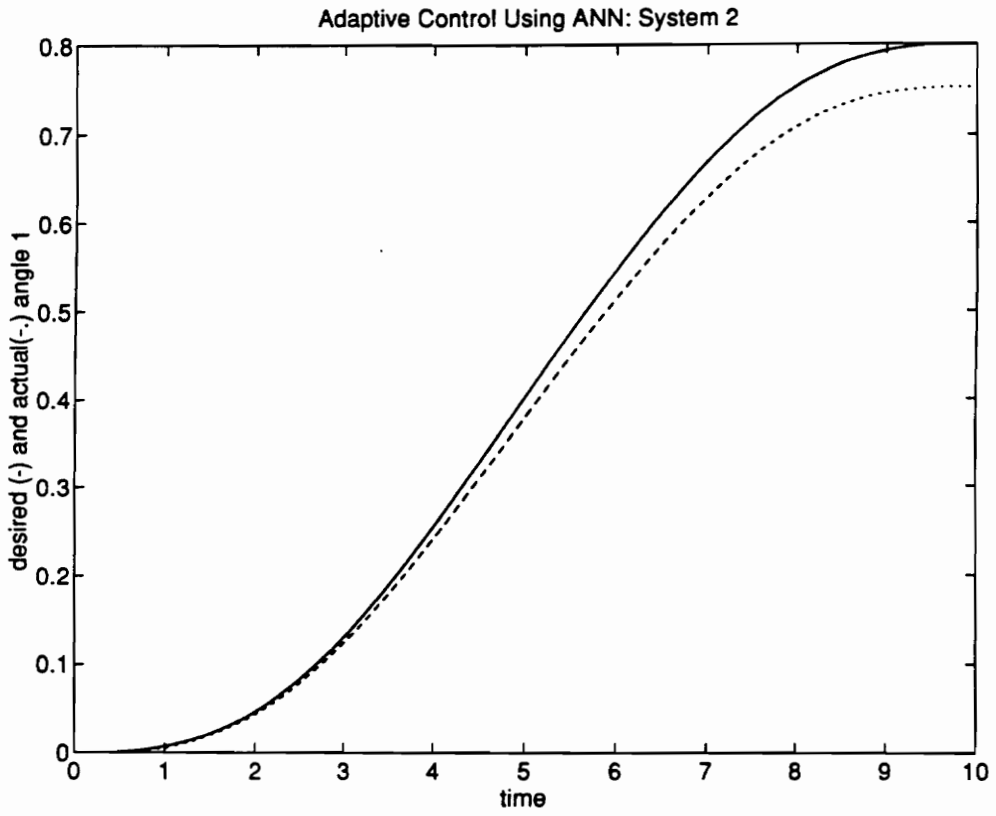


Figure 5.3.3 History of Link Angle: Adaptive Connectionist Control for System 2

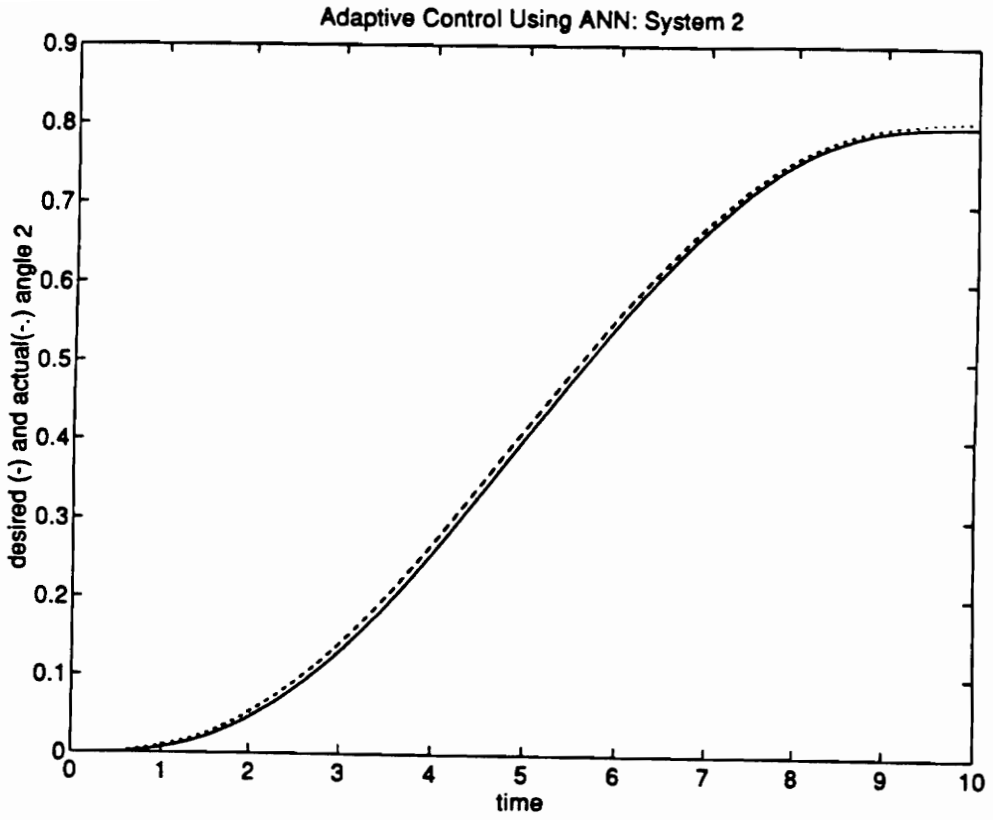


Figure 5.3.4 History of Motor Angle: Adaptive Connectionist Control of System 2

The motor angle follows the desired trajectory very nicely (Figure 5.3.4). The control input is shown in Figure 5.3.5. Since the parameters of the ANN are changed periodically, there are periodic sudden changes in the control inputs. Thus, the adaptive connectionist control gives satisfactory performance for a difficult control problem.

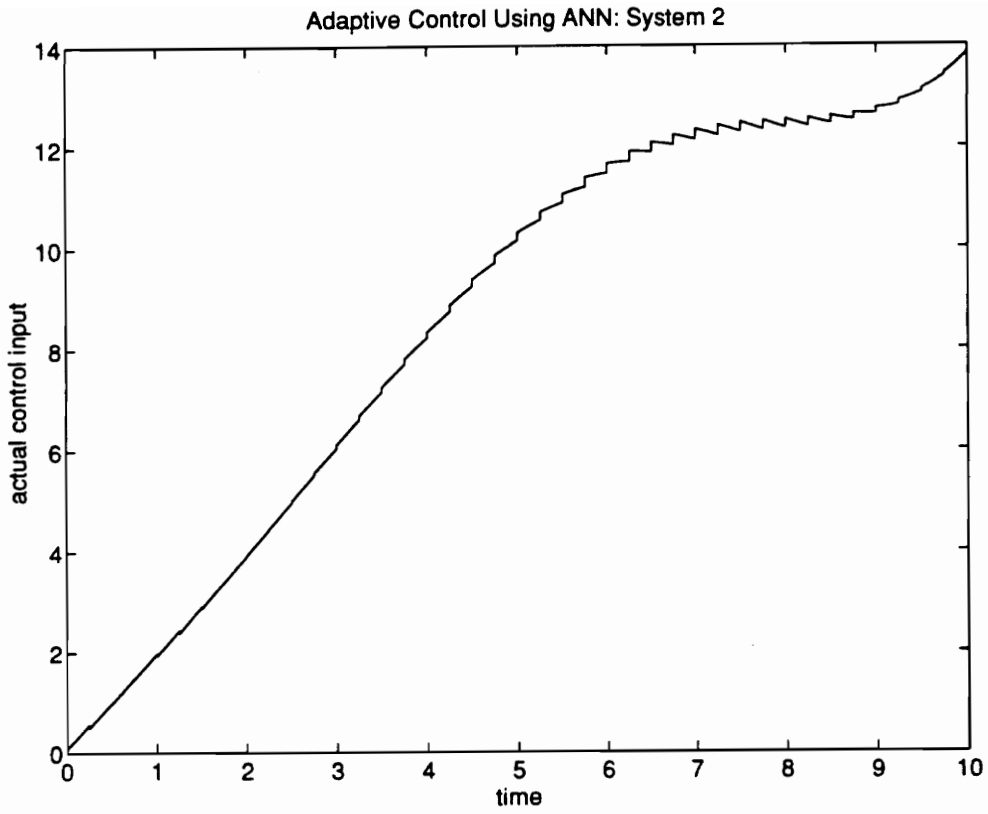


Figure 5.3.5 History of Control Input: Adaptive Connectionist Control for System 2

5.4 Summary

The several control strategies, nonconnectionist adaptive control, open loop connectionist control, neuro-PID control and adaptive connectionist control were investigated. Control of a single link and a single link with flexible joint and damping was tackled. When control of a single output nonlinear systems with unknown dynamics is to

be handled, neuro-PID control most promising, taking into account the accuracy and the computational requirements. A PID controller alone may not be able to control effectively over the whole range of the link angle as shown in Figure 5.3.6.

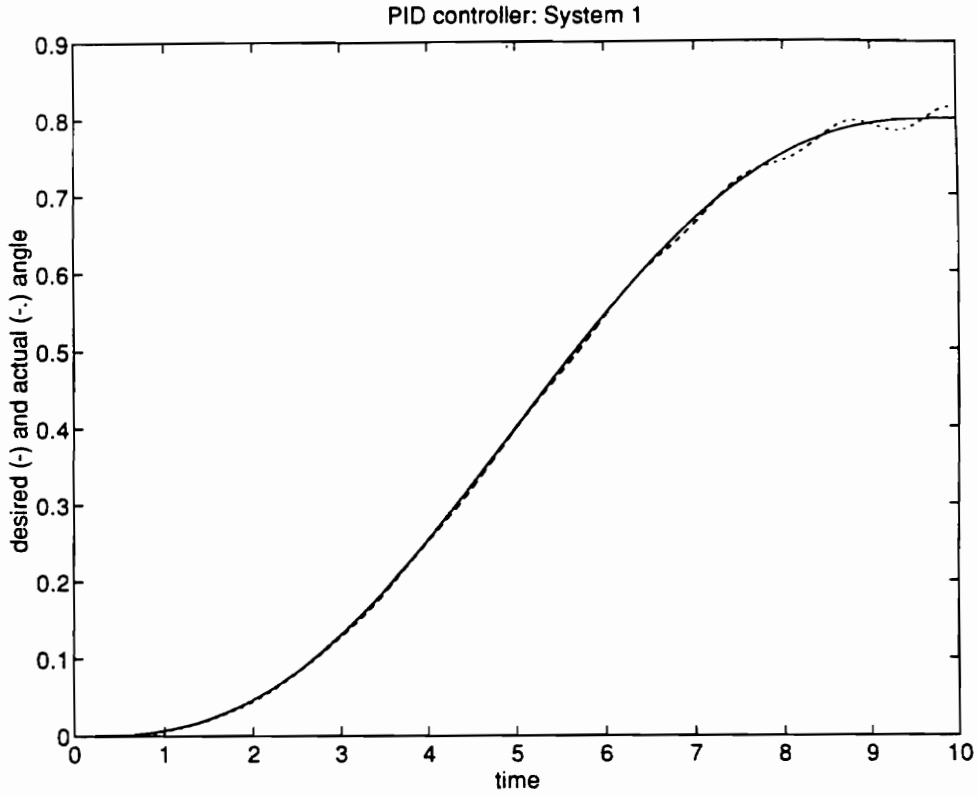


Figure 5.3.6 PID Controller for System 1

When control of complex nonlinear systems with varying parameters is under investigation, adaptive connectionist approach is the best choice. Also, the PID controller can be combined with an adaptive ANN controller if the real time implemenattion is possible.

Chapter 6
Conclusions and Future Work

This thesis dealt with two different problems- system identification and control. In this chapter, a brief overview of the work done is given, followed by the scope of the system identification and control techniques developed in the thesis. Finally, several areas of future work are suggested.

6.1 System Identification and Control: An Overview

Two techniques were investigated for identification of complex nonlinear systems. The nonlinear systems were categorized into two classes. Class I systems were defined as the nonlinear systems which operate linearly in some operating region and such region was known a priori. Class II systems are those systems whose such region of operation, if exists, is unknown. This classification was established with an important consideration in mind. Many nonlinear systems are linearized around some operating point and then classical techniques of analysis of systems can be used. However, such systems, in general, are nonlinear. Hence, a notation of "class I" systems was sought. On the other hand, some systems are inherently nonlinear and they may not exhibit linear behavior in a known region. These types of systems can then be viewed as the representatives of class II systems. Two techniques investigated in conjunction with nonlinear system identification are WRM (Whole Region Method) and SRM (Separate Region Method). In WRM, the inputs and outputs of the system were used in training an ANN. In SRM,

the linear relationship between the inputs and outputs of the system was first extracted and the ANN was trained using the inputs of the system and the system outputs due to nonlinearities. Both the techniques produced excellent results, confirming the concept that the ANNs are quite powerful in extracting abstract and nonlinear relationships. For the sake of comparison, it can be stated that SRM gives slightly better results than WRM. In particular, in case of Class I systems, SRM has a good lead over WRM. This can be attributed to two phenomena. Since the linear region of the system is known in case of Class I systems, powerful methods of conventional system identification (such as the method of POI) can be used to get a good linear model of the system. Then the ANN has to capture only nonlinearities and it is good at this task.

Several techniques such as open loop connectionist control, neuro-PID control and adaptive connectionist control were applied to two problems in the robotics domain. An ANN used in open loop control learn the "inverse" system identification so that when the desired outputs of the system are given to the ANN controller, it gives the control inputs to be applied to the system, hoping that the actual outputs of the system will be very close to the desired outputs. In neuro-PID control, the ANN controller is the same as that used in open loop control. However, now a popular PID controller is assisting the ANN controller to reduce the difference between the actual and desired output. An adaptive connectionist controller is generic. It is initially the ANN controller used in open loop control. However, its parameters are now changed according to its actual performance.

6.2 Scope of Research

The scope of system identification techniques and connectionist control techniques is explained next.

6.2.1 Scope of System Identification Techniques

As noted earlier, a complex nonlinear systems can be categorized as Class I or Class II system. The techniques of WRM and SRM can be applied to virtually any nonlinear system. Though there is not a unique solution, these techniques lead to a system model which can be used for a number of purposes such as a plant simulator. Especially, if a nonlinear system is currently linearized and then operated in the linear (and hence *constrained*) region, the technique of SRM can be used so that the system operation is not limited to a circumscribed region. Thus, this technique is the *range enhancer* of the system. Also, after a system model is obtained, we can predict the behavior of the system when the inputs to the plant are known. Thus, alarm conditions can be diagnosed much *earlier* and corrective actions can be started soon. If we had not developed the model of the system, we would not have known about the alarm conditions *until* the inputs had passed through the system. Also, we would be able to take a necessary action *after* this alarm situation is diagnosed. The nicety about connectionist approach is that we do not need the information regarding the structure of the system. Though the information about the system helps determine the direction of to be taken (e.g. when the order of the system is found using conventional technique in the linear region for class I systems, we get the information about the exact delays in inputs and outputs necessary to get good model for nonlinear system), it is not a must. Few more simulations can readily point to the right direction. In short, these system identification techniques are one of the excellent tools for analysis of nonlinear systems.

6.2.2 Scope of Control Techniques

Open loop connectionist control may suffice for simple nonlinear systems. Neuro-PID control is very promising, taking into account the accuracy and the computational requirements. It is a good compromise between the simplicity of an open loop control and potential high computational requirements of an adaptive connectionist control. Neuro-PID control is applicable to the single output systems. Adaptive connectionist control can be applied to any complex nonlinear systems. Care should be taken in determining structure of the ANN, since on-line adaptation of ANN parameters using backpropagation learning rule means that the real-time implementation may not be obtained if there are many neurons (and hence many computations).

6.3 Future Work

In this thesis, some hypothetical systems were constructed and random inputs were given to such systems to obtain the system outputs. These inputs and outputs were used subsequently for training the network. It is suggested that actual data from a physical system be taken and the results verified.

It was assumed that the inputs and outputs do not contain any noise. If the actual data contains noise, some kind of filtering becomes necessary.

The actual real-time implementation should be obtained.

Least Squares solution was used to get the linear relationship between inputs and outputs of the system in case of class II systems while using SRM. If some better method becomes available, the results may be further improved.

Several applications of system identification techniques can be investigated. For example, the system identification achieved using these techniques may be combined with an expert system to form a fault diagnosis and correction mechanism where the ANN will predict the fault at an early stage and expert system will turn on appropriate controllers based on the analysis of the fault. In such a case, the expert system may be reasonably simple in structure, containing several IF-THEN rules.

After the models of different subsystems of a large-scale industrial manufacturing process are obtained, the behavior of the overall system may be studied under different settings of the controllers to optimize the global manufacturing process. Concepts of optimal control theory may be useful with such global optimization. Simple but good ANN controllers can be developed and standardized for subsystems (such as distillation column, or a heat exchanger, or a boiler). In particular, this may be useful during the design stage of a manufacturing process. These same concepts of modular design can be applied to other areas too. For example, several robots with the identical tasks of trajectory can be controlled using the same controller if they are manufactured with identical specifications.

References

[1] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, N. J., 1979.

[2] P. J. Antsaklis, Guest Ed., Special Issue on Neural Networks in Control Systems, *IEEE Contr. Syst. Mag.*, vol. 10, no. 3, pp. 3-87, Apr. 1990.

[3] P. J. Antsaklis, Guest Ed., Special Issue on Neural Networks in Control Systems, *IEEE Contr. Syst. Mag.*,vol. 12, no. 2, pp. 8-57, Apr. 1992.

[4] K. J. Astrom and B. Wittenmark, *Adaptive Control*, Addison-Wesley, 1989.

[5] M. Athans, " The Status of Optimal Control Theory and Applications for Deterministic Systems," *IEEE Trans. Automatic Control* (1966), 580-596.

[6] B. Bavarian, Guest Ed., Special Section on Neural Networks for Systems and Control, *IEEE Contr. Syst. Mag.,* vol. 8, no. 2, pp. 3-31, Apr. 1988.

[7] R. E. Bellman, *Dynamic Programming.* Princeton, N.J.: Princeton University Press, 1957.

[8] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming.* Princeton, N.J.: Princeton University Press, 1962.

[9] R. E. Bellman and R.E. Kalaba, *Dynamic Programming and Modern Control Theory.* New York: Academic Press, 1965.

[10] S. Bingulac and D. Cooper, Use of pseudo-observability indices in identification of continuous-time multivariable models. N. K. Sinha and G. P. Rao (ed.), *Identification of Continuous-Time Systems*, Kluver Academic Publishers, The Netherlands, 1991.

[11] H. S. Black, "Inventing the Negative Feedback Amplifier," *IEEE Spectrum*, pp. 55-60, December, 1977.

[12] H. W. Bode, "Feedback- The History of an Idea," *Selected Papers on Mathematical Trends in Control Theory*, Dover, New York, pp. 106-123, 1964.

[13] T. Bohlin, Model validation, In *Encyclopedia of Systems and Control* (M. Singh, ed.), Pergamon Press, Oxford, 1987.

[14] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*, Holden-Day, San Francisco, 1970.

[15] D. J. Burr, Experiments with a Connectionist Text Reader, In Proceedings of the First International Conference on Neural Networks, eds. M. Caudill and C. Butler, vol. 4, pp. 717-24, San Diego, CA, 1987.

[16] D. M. Considine, ed., Process Instruments and Controls Handbook, Section 9, New York: McGraw-Hill, 1957.

[17] G. W. Cottrell , P. Munro, and D. Zisper, Image Compression by backpropagation: An example of extensional programming. *Advances in cognitive science* (vol. 3). Norwood, NJ: Ablex, 1987.

[18] D. P. Eckman, *Automatic Process Control*, New York: John Wiley, 1958.

[19] W. R. Evans, "Graphical Analysis of Control Systems," *Trans. AIEE*, 67, pp. 547-551, 1948. Also in G. J. Thaler, ed., *Automatic Control*, Dowden, Hutchinson, and Ross, Inc., Stroudsburg, Pa, pp. 417-421, 1974.

[20] W. R. Evans, "Control Systems Synthesis by Root Locus Method," *Trans. AIEE*, 69, pp. 1-4, 1950. Also in G. J. Thaler, ed., *Automatic Control*, Dowden, Hutchinson, and Ross, Inc., Stroudsburg, Pa, pp. 423-425, 1974.

[21] W. R. Evans, *Control System Dynamics*, McGraw-Hill, New York, 1954.

[22] P. Eykhoff, Biomedical identification: Overview, problems, and prospects. *Proc. 7th IFAC Symposium on Identification System Parameter Estimation*, York, U.K., pp. 37-45, 1985.

[23] B. Friedlanderv, System identification techniques for adaptive signal processing, *IEEE Trans. Acoustics,, Speech and Signal Processing*, vol. ASSP-30, pp. 240-246, 1982b.

[24] M. Gevers and G. Bastin, "What does system identification have to offer?" *Proc. 6th IFAC Symposium on Identification and System parameter Estimation,*. Washington, D. C., 1982.

[25] S. Grossberg, Studies of the Mind and Brain, Drodrecht, Holland, Reidel Press, 1982.

[26] G. Grubel, Uncertainty and Control-some activities at DFVLR, in *Lecture Notes in Control and Information Sciences*, vol. 70 (J. Ackermann, ed.), Springer Verlag, New York, pp. 1-47, 1985.

[27] I. Gustavsson, Survey of application of identification in chemical and physical processes. *Automatica*, vol. 11, pp. 3-25, 1975.

[28] R. Hecht-Nielse, Counterpropagation Networks, In Proceedings of the IEEE First International Conference on Neural Networks, eds. M. Caudill and C. Butler, vol. 2, pp. 19-32, San Diego, CA, 1987a.

[29] R. Hecht-Nielse, Counterpropagation Networks, *Applied Optics* 26 (23): 4979-84, 1987b.

[30] R. Hecht-Nielse, Applications of Counterpropagation Networks, *Neural Networks*: 1:131-39, 1988.

[31] A. J. Jakeman, Application of identification and system parameter estimation to environmental problems: Some recent examples. *Proc. 7th IFAC Symposium on Identification System Parameter Estimation*, York, U.K., pp. 445-450, 1985.

[32] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press, New York, 1970.

[33] R. K. Jurgen, Consumer Electronics, *IEEE Spectrum*, pp. 65-68, January, 1991.

[34] T. Kailath, *Linear Systems*. Prentice-Hall, Englewood Cliffs, N.J., 1980.

[35] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems, "*ASME Journal of Basic Engineering* (1960), 35-45.

[36] R. E., Kalman and R.S. Bucy, " New Results in Linear Filtering and Prediction Theory," *ASME Journal of Basic Engineering* (1961), 95-108.

[37] G. Kaplan, Industrial Electronics, *IEEE Spectrum*, pp. 47-48, January, 1992.

[38] D. E. Kirk, Optimal Control Theory, An Introduction, Prentice-Hall Inc., New Jersey, 1970.

[39] T. Kohonen, Self-Organization and Associative Memory, 2nd Edition, Berlin, Springer-Verlag, 1987.

[40] B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice-Hall, Englewood Cliffs, N. J., 1990.

[41] B. Kosko, Bi-directional Associative Memories, IEEE Trans. Systems, Man and Cybernatics, 18(1):49-60, 1987.

[42] B. Kosko and C. Guest, Optical Bi-directional Associative Memory, Society for Photo-optical and Instrumentation Engineers Proceedings: Image Understanding, 758:11-18, 1987.

[43] E. B. Kosmatopoulos, A. G. Chassiakos, and M.A. Christodoulou, Robot identification using neural networks. *IEEE 30th Conf. on Decision and Control*, San Diego, CA, 1991.

[44] R. E. Larson, "Dynamic Programming with reduced Computation Requirements," IEEE Trans. Automatic Control(1967), 767-774.

[45] J. Li, A. N. Michael, and W. Porod, "Analysis and Synthesis of a Class of Neural Networks: Linear Systems Operating on a Closed Hypercube," *IEEE Trans.. Circuits and Systems*, vol. 36, no. 11, pp. 1405-1422, November 1989.

[46] R. L. Lippmann, " An Introduction to Computing with Neural Nets," *IEEE Acoustics, Speech, Signal Proc. Mag.*, pp. 3-31, Apr

[47] L. Ljung, *System Identification- Theory for the User*, Prentice-Hall, Englewood Cliffs, N. J., 1987.

[48] E. H. Mamdani and T. J. Procyk, "A Linguistic Self-Organizing Process Controller," *Automatica*, 15: 15-30, 1979.

[49] O. Mayer, *The Origins of Feedback Control*, M. I. T. Press, Cambridge, M.A., 1970.

[50] O. Mayer, "The Origins of Feedback Control," *Scientific American*, 223, 4, pp. 110-118, October, 1970.

[51] R. K. Mehra, Identification in control and econometrics: Similarities and differences, Ann. Economic and Social Measurement, vol. 3, p.21, 1974b.

[52] J. Mendel, Some representation, measurement, estimation, and validation problems in reflection seismology, *Proc. 7th IFAC Symposium on Identification System Parameter Estimation*, York, U.K., pp. 115-124, 1985.

[53] K. S. Narendra and A. M. Annasswamy, *Stable Adaptive Systems*, Prentice-Hall, 1989.

[54] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical system using neural networks. *IEEE Trans. Neural Networks*, 1, 2, 4-27, 1990.

[55] K. Ogata, *Modern Control Engineering*, 2nd Ed., Prentice-Hall, Inc., Englewood Cliffs, N.J., U.S.A., 1990.

[56] D. B. Parker, Learning-logic, Invention Report, S81-64, File 1, Office of Technology Licensing, Stanford University, 1982.

[57] L .S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mischenko, The Mathematical Theory of Optimal Processes. New York: Interscience Publishers, Inc., 1962.

[58] F. Rosenblatt, *Principles of Neurodynamics*, Washington D. C., Spartan Press, 1961.

[59] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," D. Rumelhart and J. McClelland, editors, Parallel Data Processing, vol.1, chapter 8, the MIT Press, Cambridge, MA, pp. 318-362, 1986.

[60] T. J. Sejnowski and C. R. Rosenberg, Parallel networks that learn to pronounce English text, *Complex Systems*, 3:145-68, 1987.

[61] K. Self, Designing with Fuzzy Logic, *IEEE Spectrum*, pp. 42-44.

[62] R. Shoureshi, R. Chu, and M. Tenorio, "Neural Network for System Identification," Proc. 1989 Amer. Contr. Conf., June, 1989, vol.1, pp. 916-921.

[63] J. E. Slotine and W. Li, *Applied Nonlinear Control*, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[64] M. Sugeno, *Industrial Applications of Fuzzy Control*, North-Holland, Amsterdam.

[65] H. F. VanLandingham and S. Bingulac, *Algorithms for computer-aided design of multivariable control systems*, Marcel Dekker, Inc., New York, 1993.

[66] H. F. VanLandingham, S. Bingulac, and M. Tran. " Comparison of conventional and neural network approaches to system identification," C-TAT, vol. 9, No. 1, March, 1993, pp. 77-97.

[67] H. F. VanLandingham and J. Y. Choi, "Nonlinear System Identification Using Neural Networks," Fifth IFSA World Congress, 1993, pp. 58-61.

[68] M. Vidyasagar and M. Spong, *Robot Dynamics and Control*, John Wiley & Sons, Inc., 1989.

[69] P. D. Wasserman, *Neural Computing-Theory and Practice*, New York, Van Nostrand Reinhold, 1989.

[70] P. E. Wellstead, Non-parametric methods of system identification, *Automatica*, vol. 17, pp. 55-69, 1981.

[71] P. J. Werbos, *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Masters thesis, Harvard University, 1974.

[72] B. Widrow, and S. D. Sterns, *Adaptive Signal Processing*, New York: Prentice-Hall, 1985.

[73] J. G. Ziegler and N. B. Nichols, " Optimum Settings for Automatic Controllers," *Trans. ASME*, vol. 64, pp. 759, 1942.

[74] Special Section on Neural Networks for Control Systems, *IEEE Contr. Syst. Mag.*, vol. 9, no. 3, pp. 25-59, Apr. 1989.

[75] Neural Network Toolbox User's Guide, The MathWorks, Inc., June, 1992.

[76] *Fifth International Fuzzy Systems Association World Congress*, July 4-9, Seoul, Korea, 1993.

VITA

Nishith D. Tripathi was born in Ahmedabad, India on January 22, 1971. He was an undergraduate student in the Instrumentation and Control Engineering at L. D. College of Engineering, Gujarat University, India. He received his bachelors degree in August 1992. He developed an Integrated Console suitable for Distributed Process Control System in simulated environment at the Physical Research Laboratory, Ahmedabad, India in the final year of the undergraduate studies. He joined Virginia Polytechnic Institute and State University to study towards the Masters degree in the Electrical Engineering with the specialization in controls and received his Masters degree in November 1994. He is studying towards the Ph. D. degree with Dr. Hugh V. VanLandingham at Virginia Tech. His research interests are development and applications of control techniques using neural networks and fuzzy logic.

ndtripathi