IDENTIFICATION AND CONTROL OF DYNAMIC SYSTEMS USING NEURAL NETWORKS

by

Eliezer Colina Morles
M.Sc., Systems Eng.

A thesis presented to the
UNIVERSITY OF SHEFFIELD
for the degree of
DOCTOR OF PHILOSOPHY
in the Faculty of Engineering

Department of Automatic Control and Systems Engineering,
University of Sheffield.

DECEMBER, 1993

DECLARATION

No part of the research referred to in this thesis has been submitted in support of an application for another degree or qualification at this or any other university or other institution of learning.

During the course of this research the following publications were presented

- E. Colina-Morles; N. Mort, "Self-Tuning Control Via Neural Network Parametric System Identification".
  Presented in the one day colloquium on "Application of Neural Networks in Control and Modelling of Industrial Processes". Sir George Cayley Institute of the Polytechnic of Central London, April 1991.

- E. Colina-Morles; N. Mort, "Identification and Control of Dynamic Systems via Adaptive Neural Networks".
  Department of Automatic Control and Systems Engineering, Research Report Number 433, July 1991.

- E. Colina-Morles; N. Mort, "On-Line Control of Dynamic Systems Using Feedforward Neural Networks".
  Department of Automatic Control and Systems Engineering, Research Report Number 457, August 1992.

- E. Colina-Morles; N. Mort, "Neural Network-based Adaptive Control Design".
  Journal of Systems Engineering, Vol. 3, Number 1, pp 9-14, London 1993.

- E. Colina-Morles "On-Line Control of Dynamic Systems Using Feedforward Neural Networks".
  2-Day Symposium on Postgraduate Research in Control and Instrumentation, The University of Nottingham, March 1993.

- H. Sira-Ramirez, E. Colina-Morles, "A Sliding Mode Strategy for Adaptive Learning in Adalines".
  Submitted for publication to IEEE Trans. on Systems, man, and Cybernetics, 1993.

- E. Colina-Morles; N. Mort, "Inverse Model Neural Network-Based Control of Dynamic Systems".
  Submitted to the IEE Fourth International Conference on Control-94.

ACKNOWLEDGEMENTS

The author would like to thank and express his sincere indebtedness to all who have helped him in this work . Very particularly he would like to thank the University of Los Andes in Mérida, Venezuela, for the award in supporting his work. The author is also grateful to his supervisor, Dr. N. Mort of the Automatic Control and Systems Engineering Department at Sheffield University, for his encouragement in his work.

Finally, the author would like to thank his family for their continuing and sustaining support over the years.

To my beloved wife, Rosa Ibett;
my daughter Ibett Saraí,
my son Elzer;
and all my family.

ABSTRACT

The aim of this thesis is to contribute in solving problems related to the on-line identification and control of unknown dynamic systems using feedforward neural networks. In this sense, this thesis presents new on-line learning algorithms for feedforward neural networks based upon the theory of variable structure system design, along with mathematical proofs regarding the convergence of solutions given by the algorithms; the boundedness of these solutions; and robustness features of the algorithms with respect to external perturbations affecting the neural networks' signals.

In the thesis, the problems of on-line identification of the forward transfer operator, and the inverse transfer operator of unknown dynamic systems are also analysed, and neural networks-based identification schemes are proposed. These identification schemes are tested by computer simulations on linear and nonlinear unknown plants using both continuous-time and discrete-time versions of the proposed learning algorithms.

The thesis reports about the direct inverse dynamics control problems using neural networks, and contributes towards solving these problems by proposing a direct inverse dynamics neural network-based control scheme with on-line learning capabilities of the inverse dynamics of the plant, and the addition of a feedback path that enables the resulting control scheme to exhibit robustness characteristics with respect to external disturbances affecting the output of the system. Computer simulation results on the performance of the mentioned control scheme in controlling linear and nonlinear plants are also included.

The thesis also formulates a neural network-based internal model control scheme with on-line estimation capabilities of the forward transfer operator and the inverse transfer operator of unknown dynamic systems. The performance of this internal model control scheme is tested by computer simulations using a stable open-loop unknown plant with ouput signal corrupted by white noise.

Finally, the thesis proposes a neural network-based adaptive control scheme where identification and control are simultaneously carried out.

Contents

Chapter 1

INTRODUCTION

1.1 IDENTIFICATION AND CONTROL OF DYNAMIC SYSTEMS USING NEURAL NETWORKS

The identification of dynamic systems using neural networks addressed in this thesis may be succinctly formulated as follows:

Given a sequence of time-indexed input output measurements "u(t), f(u(t))", $t \in [0, \infty)$, obtained from a plant "f", and given a feedforward neural network represented by the function "$\hat{f}(W(t), u(t))$" that depends upon the adaptable real-valued matrix "W(t)" and the variable "u(t)", find the parameters "$W^*(t)$" such that

$$\|\hat{f}(W^*(t), u(t)) - f(u(t))\| \leq \epsilon \tag{1.1}$$

for all admissible values of "W(t)", and $\epsilon > 0$.

Observe that this identification problem may be interpreted as the problem of finding a function "$\hat{f}(.,.)$" to approximate the function "f(.)" in the best possible way, by means of a learning or adaptation process.

The existence of a best approximation is influenced by the learning process used to adapt the values of the adjustable matrix "W(t)". In this thesis, both continuous-time and discrete-time learning algorithms for the adaptation of "W(t)" will be presented. These new learning algorithms are based on the theory of variable structure control design [43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56] and may be used on-line for making feedforward neural networks emulate the forward transfer operator that represents an unknown dynamic system.

The thesis also reports about the problem of using these learning algorithms

for the on-line training of neural networks in order to represent the inverse transfer operator corresponding to a certain time history of input-output measurements obtained from an unknown dynamic system.

Both the forward dynamics and the inverse dynamics identification problems have been extensively reported in the literature using different neural networks topologies and a variety of learning algorithms [7, 15, 35, 39,40, 41, 42, 59, 60, 61, 62, 63]. In most of the reported works however, the identification of the forward dynamics or the inverse dynamics of an unknown system using multilayer neural networks involves an off-line training phase where the adaptable network parameters are adjusted with every input-output pair presentation to the network by using an algorithm that solves the problem of minimizing an error function of these parameters.

In contrast to the off-line neural network training algorithms for system identification, in the on-line learning algorithms that will be presented in this work there is no need for a training phase, and instead of minimizing an error function of the adjustable parameters, the algorithms operate by forcing an error function of the parameters to go to zero.

The information about the unknown system dynamics or its inverse provided by the on-line trained neural network may be incorporated into control schemes to actually control the system. There is a large number of neural networks-based control schemes that have been recently reported in the literature [15, 16, 36, 60, 61, 65, 69, 70, 72, 75, 76] among many others.

The formulation of the direct inverse dynamics control problem that is considered in this thesis may be simplified as follows:

Given the input-output measurements "u(t), f(u(t))" obtained from the dynamic system "f", and given a desired plant output "$y_r(t)$" that may be assumed that is the output of a known reference model; we want to determine an approximation of the inverse transfer operator "$\hat{f}^{-1}$" of "f" that enables the control law "u(t)", $t \geq t_0$, to be generated in such a way that

$$\lim_{t \to \infty} |f(u(t)) - y_r(t))| \leq \zeta \tag{1.2}$$

for some specified constant value $\zeta \geq 0$.

From the neural networks applications viewpoint it has been shown that using neural network trained versions of the inverse transfer operator of the system being considered enables the control input "u(t)" mentioned above to be implemented with excellent control performance [7, 16, 75, 76, 83, 84].

In the neural network-based direct inverse control scheme that will be proposed

in this thesis the control action is generated based upon an on-line approximation of the inverse transfer operator of the plant.

The thesis also contemplates a brief study of the internal model control problem [74, 75, 79] and formulates an on-line trained neural network-based internal model control scheme that is tested by computer simulations performed on an open-loop stable linear unknown dynamic system.

Other control application of on-line trained neural networks that is analysed in this work is the model reference adaptive control problem [15, 80, 82]. In this sense, a neural network-based adaptive control scheme is proposed and its performance is checked by computer simulations using unknown linear and nonlinear plants.

1.2 THESIS ORGANIZATION

In order to formulate the on-line variable structure control-based learning algorithms, and to study the identification and control problems referred to in the previous section, this thesis is organised as follows:

Chapter 2 presents an overview of neural networks that contains a historical evolution of the field; a taxonomy of neural networks in terms of their topological configurations and the type of learning algorithms supported; and a categorization of the learning algorithms with a short explanation of the Widrow-Hoff delta rule, and the μ-least mean square algorithm. These two algorithms constitute the basis for other learning algorithms like the backpropagation algorithm. Finally, this chapter also contains a succinct exploration on the salient features of some of the most popular neural network structures: the multilayer perceptron networks, the hopfield or recurrent networks, and the self-organizing or Kohonen networks.

The purpose of chapter 3 is to develop the variable structure control-based learning algorithms for feedforward neural networks.In this sense, a very short introduction to both continuous-time and discrete-time variable structure control design is provided. The chapter includes a completely new type of continuous-time learning algorithms for on-line training of single layer, two layer, and three layer feedforward neural networks. Mathematical proofs on the convergence properties, on the boundedness of solutions, and on the robustness features with respect to external perturbations of these new learning algorithms are also presented.

The discrete-time versions of the learning algorithms that are formulated here, include a generalization of the algorithms developed in [60] that takes into account time-varying neural networks' input and teaching signals. As in the continuous-time cases, the discrete-time version of the algoritms are developed for single layer, two

layer, and three layer neural networks; and mathematical proofs on the convergence characteristics, boundedness of solutions, and robustness features with respect to external perturbations on the networks' signals are highlighted.

Chapter 4 covers the problems of on-line identification of the forward transfer operator and the inverse transfer operator of unknown dynamic systems. Both linear and nonlinear systems are considered, and examples on the performance of the continuous-time and discrete-time learning algorithms embedded into neural network-based identification schemes are reported. In these examples, the robustness features of the identification schemes under the presence of perturbations on the neural networks signals are studied. The chapter also presents computer simulation results showing the convergence characteristics of the output of the neural networks. Here, the approximation capabilities of single layer, two layer, and three layer neural networks trained on-line with our proposed learning algorithms are illustrated.

In chapter 5, a direct inverse model neural network-based control scheme for dynamic unknown systems is presented. The robustness characteristics with respect to external perturbations affecting the system output that the proposed control scheme exhibits are possible thanks to the on-line learning capabilities of the neural network that emulates the inverse transfer operator of the unknown plant, and the existence of a feedback path.

The chapter also includes an interpretation of the existence of the inverse transfer operator of a linear system in terms of satisfying the controllability condition. Here, the performance of the control scheme is tested by computer simulations using a linear and a nonlinear plants. Both the continuous-time and the discrete-time learning algorithms are used for the on-line training of the neural networks inverse transfer operator emulators.

Chapter 6 covers other control applications of on-line trained neural networks. In particular, a neural network-based internal model control scheme, and a neural network-based adaptive control scheme are presented and their performances are tested by computer simulations performed on unknown dynamic systems.

The last chapter is a summary of conclusions and recommendations for further research work.

Chapter 2

OVERVIEW OF NEURAL NETWORKS

2.1 INTRODUCTION

Since the nineteenth century, there have been different methods for studying the operation of the human brain in a systematic way. Neuro-psychology is based upon a method of analysing the relationships between anatomical features of the brain and aspects of human behaviour. Using such methods it was shown that the motor functions of the brain and its senses are precisely localised in its structure [1]. In contrast, there are other methods which explain the operation of the brain as a whole, taking a more global approach. The newest methods of analysis attempt to study the brain in more detail, descending to the molecular level of the physical and chemical processes involved in its operation.

Neurons are nerve cells that constitute the primordial elements of the central nervous system. In general, neurons are able to receive signals coming from other neurons, process these signals, generate nerve pulses, conduct these pulses, and transmit them to other neurons. Morphologically, a neuron has a pyramidial or spherically shaped cell body, which contains the nucleus. The cell body carries out the necessary transformations to the life of the neuron. Surrounding the cell body there are tubular extensions called dendrites, which are the receptors of the neuron. Finally, the axon which differs from the dendrites in shape, is the outgoing connection for signals emitted by the neuron. Figure 2.1 illustrates a typical neuron. In a human brain, neurons are interconnected in complex spatial topologies to form the central nervous system.

The operation of the neuron is usually explained as a process in which the

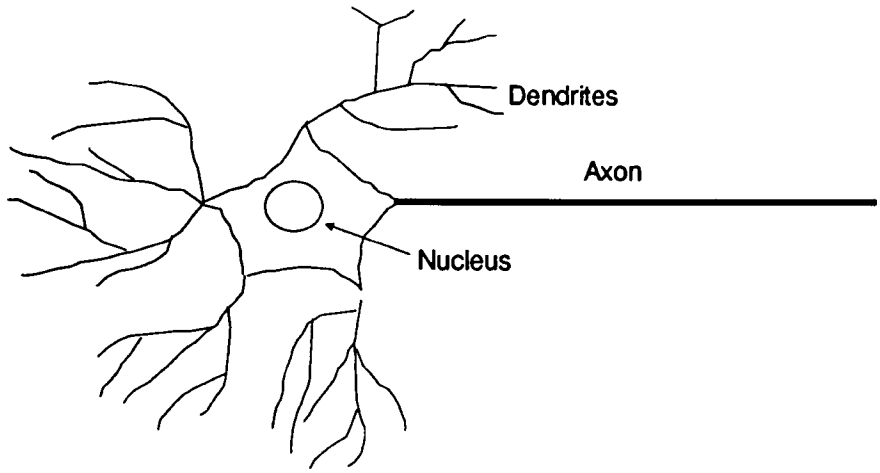


Figure 2.1: Neuron and its components.

cell carries out a summation of the signals arriving on the dendrites. When this summation is greater than a certain threshold, the neuron responds by transmitting a pulse through its axon. If the summation is less than the threshold, the neuron is inactive. In this simple conceptual model, the operation of the neuron is interpreted as an electrical phenomenon. Figure 2.2 shows the electric analog model of figure 2.1.

An artificial neural network is a model designed to emulate some of the functions of the human brain. These types of models include both the functional characteristics as well as topological configurations of neurons in the brain. The historical origins of this study area are diverse. The next section summarizes some facts in the evolution of artificial neural networks.

2.2 HISTORICAL EVOLUTION

The pioneering work on artificial neural networks was put forward in 1943 by McCulloch and Pitts [2], with the first mathematical models to study the capabilities of interconnected neurons to calculate certain logical functions. In 1949, Hebb published a study indicating the relevance of the connection between synapses to the process of learning, and pointing out the adaptation laws involved in neural systems. The ideas of McCulloch and Pitts, Hebbs, and others were influential in the work of Rosenblatt, who published his first technical report about the perceptron in 1957

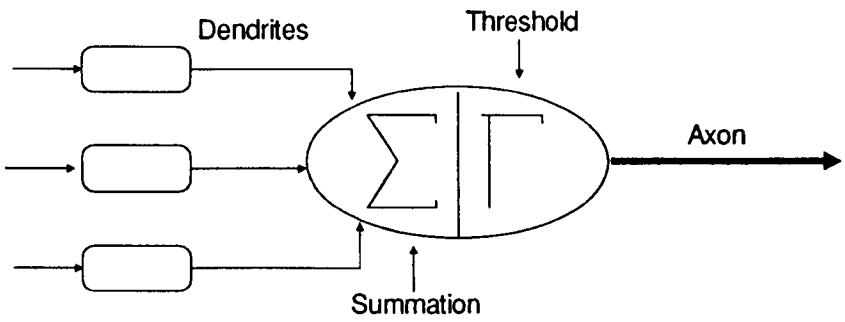


Figure 2.2: Electric model of a neuron.

[3, 4]. In his first report, Rosenblatt used the term "perceptron" to refer to an automation, instead of a class of models of the central nervous system. The aim of his work however, was to show the feasibility of designing an adaptive neural network with a rich interconnectivity and synapse-like nonlinearity to mimic some defined functions. The initial era of neural networks ended with the publication of a work by Minsky and Papert in 1969 [5]. This paper showed the theoretical limitations of single layer perceptrons to solve the "exclusive or" logical problem.

Despite the fact that the idea of merging control engineering, information science, and neural science under the banner of cybernetics, was first proposed by Wiener in 1948 [6] , the tendency during the sixties was centrifugal for each of these disciplines. Since then, the research on artificial neural networks has kept some of the control engineering tools, like the use of gradient methods, mean-square error techniques, and measurements with numerical or logical values, made of some state process or object, together with the tradition in artificial intelligence of addressing problems that are not formulated with a high degree of mathematical structure [7]. In 1976, Grossberg published a work based on biological and psychological evidence, where neural feature detectors were introduced to exploit novel characteristics of different configurations of nonlinear dynamic systems [8]. New impetus was given to the field of neural networks when Hopfield published a paper entitled "Neural networks and physical systems with emergent collective computational abilities" in 1982 [9]. In this paper, Hopfield proposed a model which was capable of implementing a content-addressable memory. His idea, although it was based on many

previous results, suggested the existence of a stable dynamic behaviour of the nervous system, where neighbouring states tend to approach a stable state. Also, he showed that during the evolution of this stable dynamic behaviour, an energy function is locally minimized, emulating the behaviour of spin isinglasses, and enabling some results from statistical physics to be used in neural networks. Another strong impulse to the research interest in the area of neural networks was propelled by the new results and the rediscovery of Werbos' backpropagation algorithm [10], by LeCum in 1985 [11], Parker in 1985 [12], and Rumelhart, Hinton, and Williams in 1986 [13]. The collection of papers by Anderson and Rosenfeld [14], is an excellent source of information to follow the development of models of neural networks. Today, the first practical applications of neural networks are emerging in a variety of engineering fields ranging from signal processing and pattern recognition devices to experimental robot controllers. In the context of control and identification of dynamic systems the paper by Narendra and Parthasarathy [15], the compilation book by Miller, Sutton, and Werbos [7], and the survey paper by Hunt, Sbarbaro, Zbikowski, and Gawthrop [16], among many others, represent an updated panorama of the state of the art in research activities in these fields.

In the literature, different names have been used to represent neural networks. Titles like connectionist models, parallel distributed processing models, and neuromorphic systems, are used as synonyms for neural networks, which has became a multi-disciplinary area of research with thousands of periodical publications in journals covering biology, psychology, mathematics, physics, and electronics among other disciplines.

The next section presents a classification of neural networks in terms of their input signals, training environment, and topological configurations.

2.3 TAXONOMY OF NEURAL NETWORKS

The basic constituent element of most neural network configuration is the adaptive neuron, also known as perceptron or adaline. In general, these adaptive neurons represent the processing elements of the network, and may be considered to have the following components:

- a weighted input summer junction,
- a nonlinear activation function,
- a learning rule.

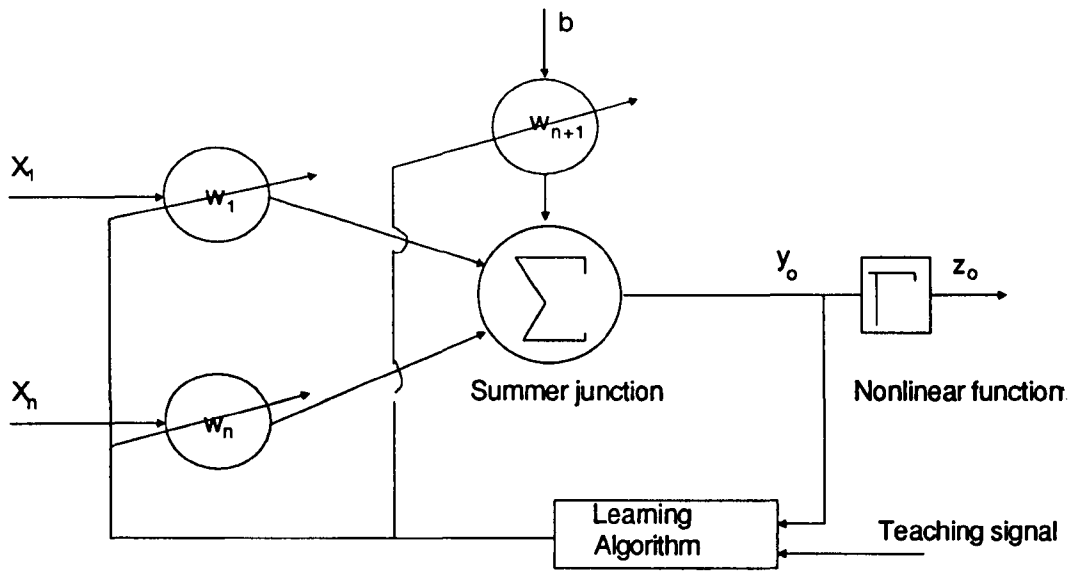


Figure 2.3: Adaptive neuron.

Figure 2.3 illustrates a neuron as an adaptive element. Observe that from figure 2.3,

$$y_o(t) = \sum_{i=1}^{n} w_i(t)\, x_i(t) + w_{n+1}(t)\, b, \tag{2.1}$$

$$z_o(t) = \Gamma(y_o(t)). \tag{2.2}$$

The computational power, or the representation capabilities of a neural network depends upon the interconnectivity among its constituent neurons and the learning rule used to adjust its weights. It is possible to think of a neural network as a classifier designed to perform a specific task. For example, the classical decision theory problem of identifying which class best represents an input pattern can be tackled with a multilayer perceptron neural network; or the problem of retrieving data given an incomplete input pattern, can be solved by designing a recurrent neural network that operates like a content-addressable memory. Similarly, a self organizing feature map neural network can be used to deal with image and speech transmission problems of data compression. The taxonomy presented by Lippmann [17], shows six topologies of classifier neural networks for fixed patterns. This taxonomy is arranged in a tree, with the upper branch divided between networks with binary-valued inputs, and networks with continuous-valued inputs. Below this branch, networks are grouped between those trained with supervision, and those trained in an unsupervised environment. Figure 2.4 illustrates this taxonomy. Observe that the algorithms listed at the bottom of the tree are those classical algorithms which

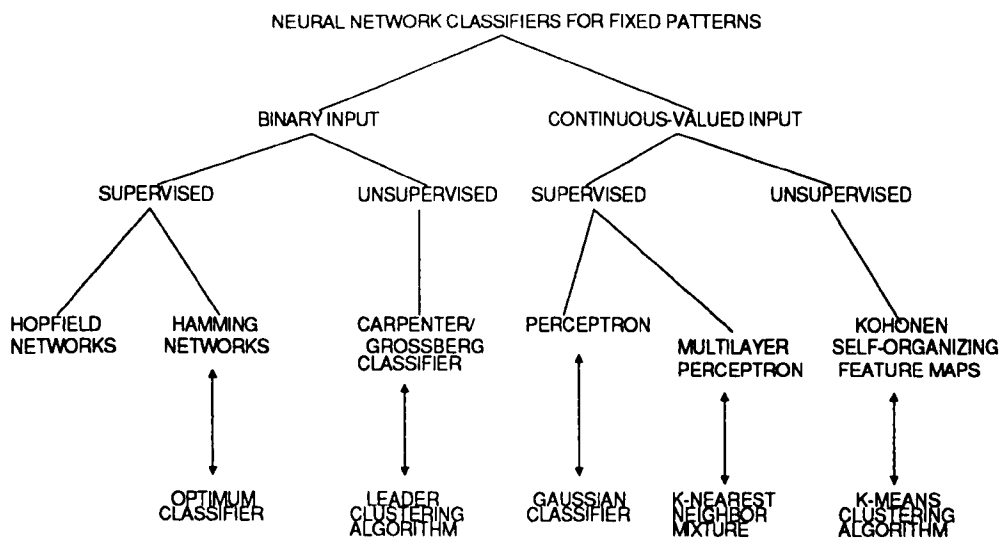


Figure 2.4: Taxonomy of six classifier neural networks.

are similar to, or perform the same function as their corresponding neural networks. For example, the Hamming network [17] is a neural network implementation of the optimum classifier for binary input patterns corrupted by random noise, whereas Kohonen networks [18] form a pre-specified number of clusters as in the k-means algorithm, where "k" refers to the number of clusters formed. The classification of neural networks by Lippmann does not differentiate between adaptive and fixed training rules.

Another way to classify neural networks is by using the unifying model proposed by Hunt et al [16]. This model contemplates that many of the basic processing elements of the network, are formed by three components:

- a weighted summer,
- a linear dynamic siso system,
- and a non-dynamic nonlinear function.

A schematic representation of the unifying model of a neuron is depicted in figure 2.5. From figure 2.5, the weighted summer is described by

$$v_i(t) = \sum_{j=1}^{n} a_{ij}\, y_j(t) + \sum_{k=1}^{m} b_{ik}\, u_k(t) + w_i, \tag{2.3}$$

which can be conveniently expressed in vector-matrix notation as

$$V(t) = A\,Y(t) + B\,U(t) + W, \tag{2.4}$$

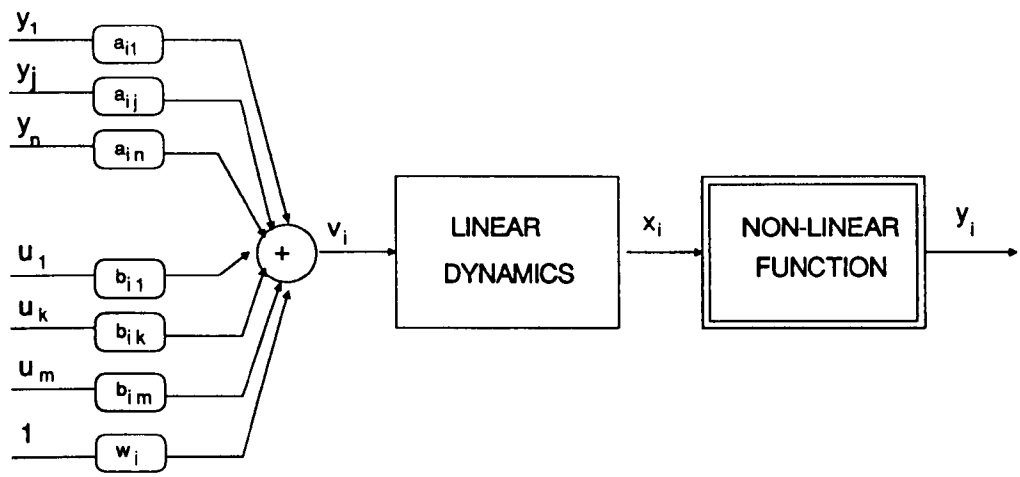


Figure 2.5: Unifying model of a neuron.

where $A \in \Re^{n \times n}$, $Y(t)$ is a vector of outputs from other neurons, $B \in \Re^{n \times m}$, $U(t)$ is a vector of external inputs, and W is a vector of constant elements. The linear dynamic siso system on the other hand, may be described by its transfer function as

$$x_i(s) = H(s)\, v_i(s), \tag{2.5}$$

or equivalently, in the time domain, by the equation

$$x_i(t) = \int_{-\infty}^{t} h(t-\tau)\, v_i(\tau)\, d\tau, \tag{2.6}$$

where $H(s)$ and $h(t)$ form a Laplace transform pair. A possible choice for $H(s)$ is the following

$$H(s) = \frac{1}{\alpha_0 s + \alpha_1}, \tag{2.7}$$

which is equivalent to selecting

$$h(t) = \frac{1}{\alpha_0}\, exp^{-(\frac{\alpha_1}{\alpha_0})t}, \tag{2.8}$$

that corresponds to the time domain input-output relationship

$$\alpha_0\, \dot{x}_i(t) + \alpha_1\, x_i(t) = v_i(t). \tag{2.9}$$

Finally, the non-dynamic nonlinear function gives the neuron output y_i as a function of the linear dynamics output x_i. This is

$$y_i = g(x_i), \tag{2.10}$$

where usual choices for $g(.)$ are threshold functions, sigmoid functions, gaussian functions, etc.

Depending on the selection of the linear dynamic siso system, neural networks may be classified as static and dynamic networks. For example, if the transfer function of the linear dynamic system is $H(s) = 1$, then an assembly of neurons can be represented by a set of algebraic equations of the type

$$X(t) = AY(t) + BU(t) + W, \tag{2.11}$$

$$Y(t) = g(X(t)), \tag{2.12}$$

with the dimensions of A and B depending on the number of the outputs and external inputs to the neural network, respectively. If on the other hand, the linear dynamic system is described, for example by $H(s) = \frac{1}{Ts+1}$ then, the neural network mathematical model can be written as the differential equation

$$T\dot{X}(t) + X(t) = AY(t) + BU(t) + W, \tag{2.13}$$

$$Y(t) = g(X(t)). \tag{2.14}$$

In terms of particular topologies, equations 2.11 and 2.12 may be regarded as representing a static multilayer feedforward network, whereas equations 2.13 and 2.14 may be considered as representing a recurrent network. Other network architectures are viewed as extensions or refinement of the ones described above.

Neither the neural network taxonomy proposed by Lippmann [17] nor the classification in the paper by Hunt et al [16], are general enough to encompass the variety of neural network topologies and learning algorithms used to train them. As an example, the neural network structures used in this work can operate both with discrete-time or continuous-time inputs, in a supervised learning environment which supports adaptive on-line training, for a multilayer perceptron topology whose neurons have continuous time variable weights. Figure 2.6 illustrates the type of neural networks used in this work. Notice that the training environment for the multilayer perceptron network shown in figure 2.6, includes an error correction based algorithm. There are other types of training algorithms, based upon different performance objectives. The next section presents a categorization of learning rules for neural networks.

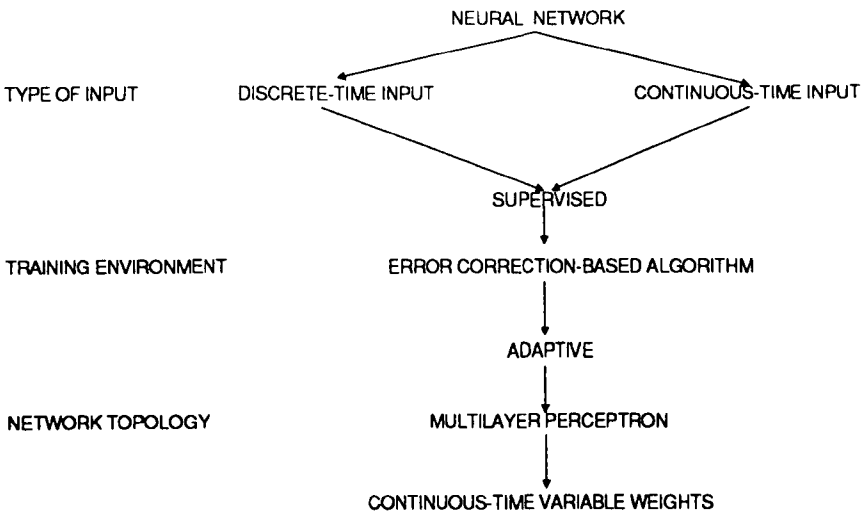


Figure 2.6: A particular type of neural networks.

2.4 CATEGORIZATION OF LEARNING ALGORITHMS

Perhaps the most important characteristic of neural networks is their ability to learn by adjusting their connection weight values to capture information that can be recalled. All learning methods can be grouped into two categories:

- supervised learning methods,
- unsupervised learning methods.

In supervised learning, which can be further classified into structural learning and temporal learning, a teacher guides the network at each stage of learning, indicating the correct result. The aim in supervised structural learning is to find the best possible input-output relationship corresponding to each input pattern, as in pattern matching and pattern classification problems. Supervised temporal learning on the other hand, is concerned with capturing a sequence of patterns necessary to achieve a final goal, such as in prediction and control problems. Examples of supervised learning algorithms include error-correction learning, reinforcement learning, stochastic learning, and hardwired systems. The unsupervised learning is a self organizing process which relies on local information with no need of any external teacher. Examples of unsupervised learning algorithms are the hebbian learning, principal component learning, differential hebbian learning, min-max learning, and

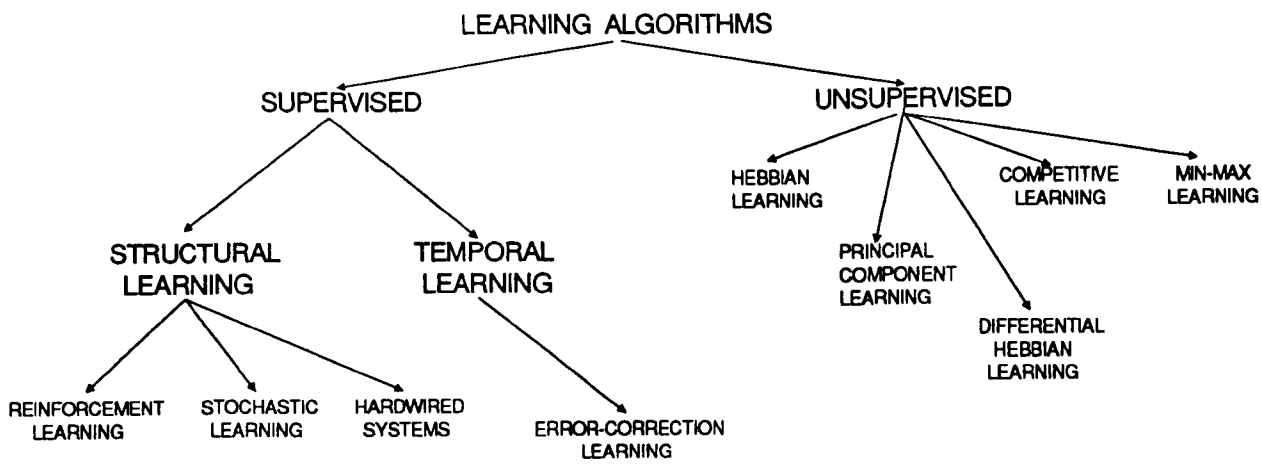


Figure 2.7: Categorization of learning algorithms.

competitive learning.

A good summary of information regarding neural network learning has been published by Simpson [19]. Figure 2.7 illustrates a categorization of learning algorithms for neural networks.

2.4.1 LEARNING ALGORITHMS FOR PERCEPTRON NETWORKS

Usually, the creation of new learning rules, or the variation of the existing ones, must rely upon the principle of minimal disturbance. That is, make adjustments to decrease the output error for the presented training pattern, with minimal disturbance to responses alrready learned. Learning algorithms for perceptron networks satisfy this principle and may be divided into two groups:

- Error-correction algorithms, which adapt the weights of a network to correct error in the output response to the presented input pattern.
- Gradient descent algorithms, which adjust the weights of a network during each pattern presentation by steepest descent with the objective of reducing the mean-square error, averaged over all training patterns.

By virtue of the differences in objectives, these two types of algorithms have different learning characteristics. An excellent categorization of learning algorithms for perceptron networks can be found in a paper by Widrow and Lehr [20]. Figure 2.8

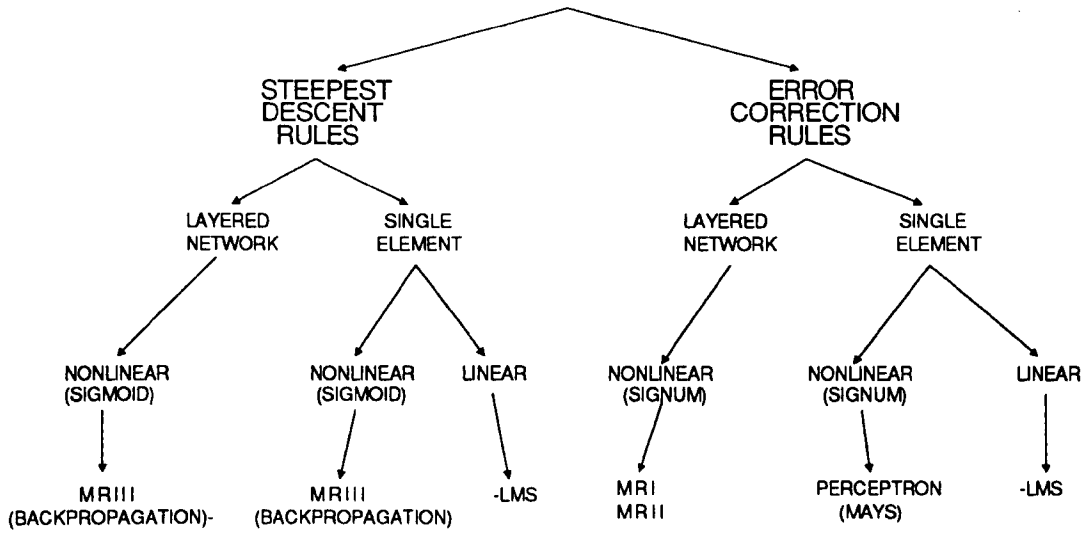


Figure 2.8: Learning rules for perceptron networks.

illustrates this categorization. Next, the α-least mean square (α-LMS) algorithm, which is an error-correction based algorithm; and the μ-least mean square (μ-LMS) algorithm, which is a gradient descent based algorithm, are presented. These formulations enable us to see the different learning characteristics of both algorithms.

2.4.1.1 α-LEAST MEAN SQUARE ALGORITHM

The α-LMS algorithm, also known as the Widrow-Hoff delta rule, is used to adapt the weights of a single linear neuron such as the one shown in figure 2.9. The linear error $e(k)$ represents the difference between the desired response y_d and the linear output $y_o(k)$. This is

$$e(k) = y_d - y_o(k), \tag{2.15}$$

where $y_o(k)$ is defined by

$$y_o(k) = W^T(k)\,X(k). \tag{2.16}$$

A change in the weights yields a corresponding change in the error

$$e(k+1) - e(k) = -(W(k+1) - W(k))^T X(k), \tag{2.17}$$

and if the change in the weights is selected as

$$W(k+1) - W(k) = \frac{\alpha\, e(k)\, X(k)}{\|X(k)\|^2}, \tag{2.18}$$

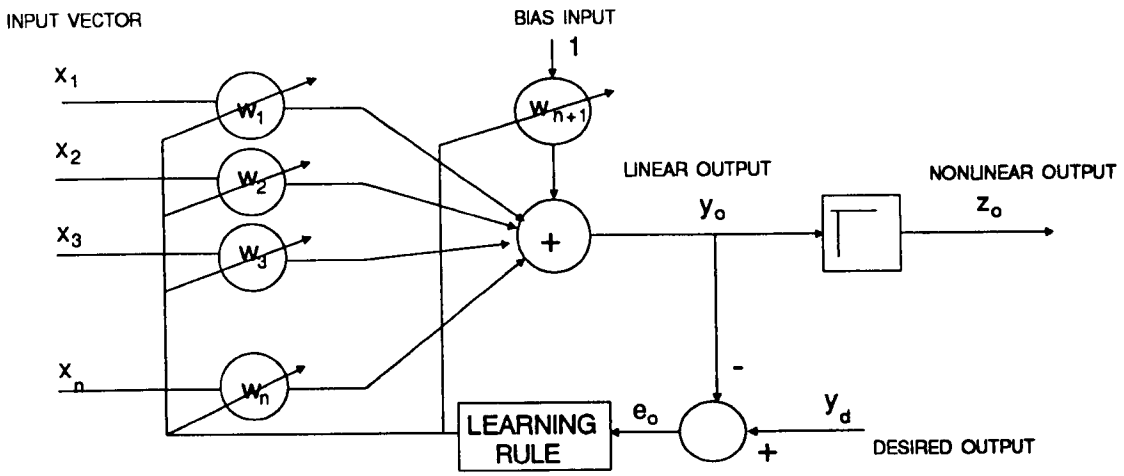


Figure 2.9: Adaptive linear element.

then equation 2.17 yields

$$e(k+1) - e(k) = -\frac{\alpha\, e(k)\, X^T(k)\, X(k)}{\|X(k)\|^2} = -\alpha\, e(k). \tag{2.19}$$

Therefore, if the input pattern X is kept fixed, the linear error e is reduced by a factor α as the weights are adapted. For input patterns independent over time, the value of α to guarantee stability must belong to the interval $0 < \alpha < 2$. Observe that this choice of α does not depend on the magnitude of the input signals. The weight update is collinear with the input pattern and of a magnitude inversely proportional to $\|X(k)\|^2$. This algorithm also operates for binary inputs.

2.4.1.2 μ-LEAST SQUARE ALGORITHM

A very common approach to reduce a mean square error function is based upon the method of steepest descent. In terms of a neural network, the gradient of the mean-square error function is measured and the neural network weights are adjusted in the direction corresponding to the negative of the measured gradient. This is

$$W(k+1) = W(k) + \mu(-\nabla(k)), \tag{2.20}$$

where μ is a parameter that control stability and rate of convergence, and $\nabla(k)$ is the value of the gradient at a point on the mean-square error surface corresponding to $W(k)$. The mean-square error surface is a convex hyperparaboloidal surface which

may be obtained in the following way. At the k-th iteration, squaring and expanding error equation 2.15 yields

$$\begin{aligned} e^2(k) &= (y_d - X^T(k)\,W(k))^2 \\ &= y_d^2 - 2\,y_d\,X^T(k)\,W(k) + W^T(k)\,X(k)\,X^T(k)\,W(k). \end{aligned} \tag{2.21}$$

Now, averaging equation 2.21 over the ensemble, yieds

$$E[e^2(k)] = E[y_d^2] - 2\,E[y_d\,X^T(k)]\,W(k) + W^T(k)\,E[X(k)X^T(k)]\,W(k) \tag{2.22}$$

Let P be the correlation vector between the desired response y_d and the input vector X, and let R be the input correlation matrix. This is

$$P^T = E[y_d\,X^T(k)], \tag{2.23}$$

$$R = E[X(k)\,X^T(k)]. \tag{2.24}$$

Substituting equations 2.23 and 2.24 into equation 2.22 yields the quadratic form

$$E[e^2(k)] = E[y_d^2] - 2\,P^T\,W(k) + W^T(k)\,R\,W(k), \tag{2.25}$$

which represents the mean-square error surface.

The μ-LMS algorithm works by performing approximate steepest descent on the mean-square error surface represented by equation 2.25. Since this equation is a quadratic function of the weights and is convex, it has a unique minimum. An instantaneous value of the gradient $\nabla(k)$ is obtained as

$$\begin{aligned} \hat{\nabla}(k) &= \frac{\partial e^2(k)}{\partial W(k)} \\ &= \begin{bmatrix} \frac{\partial e^2(k)}{\partial w_1(k)} \\ \vdots \\ \frac{\partial e^2(k)}{\partial w_{n+1}(k)} \end{bmatrix} \end{aligned} \tag{2.26}$$

Performing the differentiation in equation 2.26 and replacing it into equation 2.20, yields

$$\begin{aligned} W(k+1) &= W(k) - 2\,\mu\,e(k)\,\frac{\partial e(k)}{\partial W(k)} \\ &= W(k) - 2\,\mu\,e(k)\,X(k). \end{aligned} \tag{2.27}$$

For input patterns independent over time, convergence in the mean-square sense of the weight vector is guaranteed if

$$0 < \mu < \frac{1}{Trace(R)}, \tag{2.28}$$

where $Trace(R) = \sum$(diagonal elements of $E[X\,X^T]$). Observe that both the α-LMS and the μ-LMS algorithms are based on instantaneous gradient values. The α-LMS algorithm however, is self normalizing, with the parameter α determining the fraction of the instantaneous error to be corrected with each adaptation; whereas in the $\mu - LMS$ algorithm, μ is a constant coefficient.

For error-correction based algorithms, the same fundamental idea of instantaneous error correction was used by Rosenblatt [21] in his α-perceptron algorithm considering a binary nonlinear error function. For multi-element networks, the madaline rule I and the madaline rule II are also error correction based algorithms [20].

Other gradient descent based rules are the backpropagation algorithm [10, 11, 12, 13], and the madaline rule III by Andes et al [22].

2.5 NEURAL NETWORK STRUCTURES

Three of the most popular neural networks today are multilayer perceptron networks, Hopfield networks, and the self organizing feature maps of Kohonen. There are several other neural network structures suitable for a variety of engineering problems ranging from pattern operations (classification, matching, and completion), to noise removal, optimization, and control. This section presents an outline of important characteristics of some commonly used neural networks. Only those networks most frequently used in identification and control of dynamic systems will be summarized. the summary includes three aspects:

- type of inputs,
- learning rules and training environment,
- network topology.

2.5.1 MULTILAYER PERCEPTRON NETWORKS

Multilayer perceptrons are feedforward networks which accept both continuous-valued inputs or binary inputs. These networks are trained in a supervised environment that may support adaptive learning. Structural learning is achievable when gradient descent algorithms are used. On the other hand, temporal learning can be easily obtained by implementing error-correction-based algorithms. Topologically, a multilayer perceptron network is a layer like configuration of cascaded, interconnected processing units or nodes. Figure 2.10 shows a three layer perceptron network with two hidden layers of nodes. Provided an adequate selection of the in-

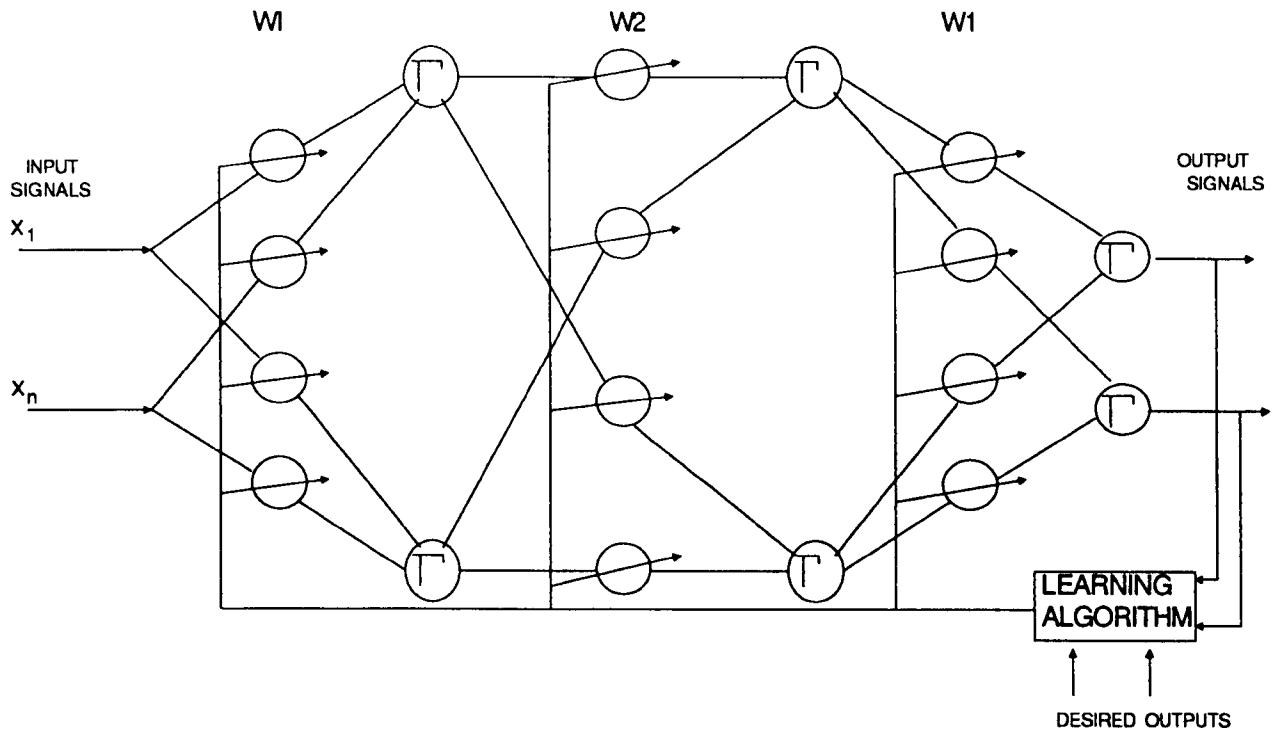


Figure 2.10: A three layer perceptron neural network.

put signals are made, the approximation capability of a multilayer network depends on its number of hidden layers, its number of processing units, and also, it depends on whether its processing units are linear or nonlinear elements. Kolmogorov's theorem states that any continuous function of "n" variables can be computed using only linear summations and nonlinear but continuously increasing functions of one variable [23]. This theorem can be used to explain the potential approximation capabilities of multilayer neural networks. It has been shown by Hornik et al [24], that a two layer network with an arbitrarily large number of nodes in the hidden layer can approximate any continuous real valued function $f \in C(\Re^n, \Re^m)$ over a compact subset of $\Re^n$.

The backpropagation algorithm, which is a generalization of the least mean-square algorithm, is often used to train multilayer perceptron networks. This algorithm operates by minimizing a cost function equal to the mean-square difference between the desired and the actual network outputs. The algorithm uses a gradient descent search technique. Details about this algorithm can be found in [13]. The next chapter presents an error-correction type algorithm, used as learning rule to train multilayer networks. This learning rule is based on the theory of variable structure control system design.

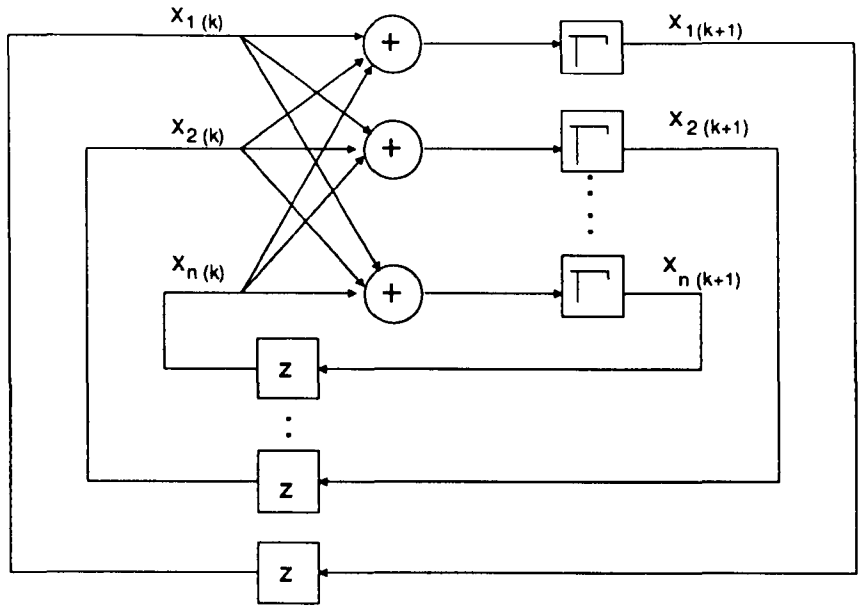


Figure 2.11: The Hopfield network.

2.5.2 HOPFIELD OR RECURRENT NETWORKS

According to Hopfield [9], the human nervous system attempts to find stable states which are attractors, in its state space. This implies that neighbouring states tend to approach a stable state, enabling errors to be corrected and providing the ability to fill in information that is missing. A Hopfield or recurrent network is an implementation of these properties and therefore, it represents a content-addressable memory.

Hopfield networks are usually operated with binary inputs. they are less appropriate when input values are continuous, because a fundamental representation problem must be addressed to convert the analog quantities to binary values. In recurrent networks, the learning process takes place in a supervised environment using Hebb's rule [25], which consists of increasing the weight of a connection between two neurons every time that the two neurons are simultaneously active. Adaptive on-line learning is not supported, and therefore the process of "memorising" stable states (prototypes) must be done off-line.

A Hopfield network is topologically arranged as a single layer network, included in a feedback configuration, as shown in figure 2.11. The discrete-time version of the Hopfield network may be modeled by

$$X(k+1) = \Gamma(X(k)); \;\; X(0) = X_0, \tag{2.29}$$

where $\Gamma(.)$ represents the nonlinear activation function, and X_0 is an initial condition. The network state evolves to an equilibrium state if $\Gamma(.)$ is suitably chosen. The set of

initial conditions in the neighbourhood of X_0 which converge to the same equilibrium state is then identified with that state.

In the continuous-time case, every feedback path includes a transfer function of the type $\frac{1}{(s+\alpha)}$. The network may be modeled by the equation

$$\dot{X}(t) = -\alpha X(t) + \Gamma(X(t)) + I, \tag{2.30}$$

where $X \in \Re^n$ is the network state, and $I \in \Re^n$ is a constant input vector. The determination of the weights by the Hebb rule in Hopfield networks, may introduce undesirable "rubbish states" that may form strong attractors. Hopfield proposes a model where the network is randomly initialised and when it converges, the state into which it stabilises is slightly "unlearned" by applying the Hebb rule in the reverse direction. This process of "unlearning" [26] allows the attractiveness of rubbish states to be decreased, whilst increasing that of the desired states. Another method to improve the avoidance of undesirable states in a Hopfield network is to use the simulated annealing algorithm introduced by Kirkpatrick et al [27]. In this algorithm, which uses an analogy with thermodynamics, the state of a system consisting of a large number of particles is characterised as the data, given the state of all these particles; for example, the position of the magnetic moment of each atom. The probability of finding this system in a given state "e" is proportional to the Boltzmann factor $exp(-\frac{J(e)}{T})$, where $J(e)$ represents the energy of this state, and "T" is a given temperature. The probability of two states e_1 and e_2 occurring is related by the equation

$$\frac{p(e_1)}{p(e_2)} = exp(\frac{-J(e_1) - J(e_2)}{T}). \tag{2.31}$$

In the simulated annealing algorithm, $J(.)$ represents a cost function to be minimized, and T is an input parameter which is initially set to a high value so that the system explores a large number of states. When the system has stabilised, T is gradually lowered, until $T = 0$. In a Hopfield network operating according to the simulated annealing principle, each neuron changes states in the sense of increasing the energy function, as a function of some parameter T. The Boltzmann machine [28] is a recurrent neural network whose learning algorithm is based upon the simulated annealing principle.

2.5.3 SELF ORGANIZING NEURAL NETWORKS

Self organizing neural networks or Kohonen feature maps [29], are designed based on the organizing principle of sensory pathways in the brain. According to this principle,

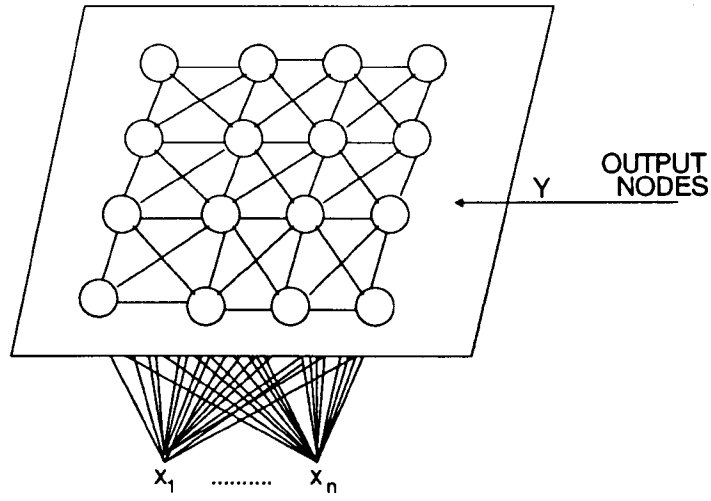


Figure 2.12: Kohonen feature map.

the placement of neurons is orderly and often reflects some physical characteristic of the external stimulus being sensed.

The inputs to a Kohonen network are continuous-valued signals presented sequentially in time, in a unsupervised environment which supports structural learning. The learning mechanism is based on the fact that the Hebb rule, when neuron activation can take only positive values, cannot reduce the value of the weight of connections when one or the two neurons are inactive. This implies that the learning mechanism cannot contribute to the phenomenon of forgetting as a result of either activity or inactivity of neurons. Kohonen's algorithm creates a vector quantizer by adjusting weights from common input nodes to output nodes.

The architecture shown in figure 2.12, illustrates a possible topology of the network. This architecture takes account of external data arriving at the network input, and of internal connections of the network. In figure 2.12, $Y = (y_1, y_2, \ldots, y_n)^T$ is the output vector of the "n" neurons in the network, $X = (x_1, x_2, \ldots x_m)^T$ is the input vector from the "m" external inputs to the network, M is the matrix of weights on the connections from external inputs to the neurons, and N is the matrix of weights of the connections between neurons in the network. Mathematically speaking, the Kohonen model may be characterised by the following dynamic state equations

$$\dot{Y} = F_1(X, Y, M, N), \tag{2.32}$$

$$\dot{M} = F_2(X, Y, M), \tag{2.33}$$

$$\dot{N} = F_3(Y, N). \tag{2.34}$$

Equations 2.32, 2.33, and 2.34 describe the operation of the network, and its learning rules for external and internal connection weights, respectively. Considering the effects of the lateral interaction, the neuron rule of operation can be described by the equation

$$y_i(t) = f(x_i(t) + \sum_{k=-m,+m} I_k\, y_{i+k}(t)), \tag{2.35}$$

where f is a sigmoid activation function, $x_i(t)$ is the total external input to neuron "i" at time "t", and I_k represents the weights of the internal connections that have lateral interaction with neuron "i". Note that the lateral interactions are considered in a neighbourhood around each neuron. In order to guarantee that the activation level of each neuron is directly proportional to the "resemblance" between the current input and the input for which that neuron was trained, the network must find, from the "n" neurons, the neuron "c" such that

$$\|X - M_c\| = min(\|X - M_i\|), \;\; 1 \leq i \leq n. \tag{2.36}$$

On the other hand, to increase the activation of the selected neuron, and the surrounding group of neurons, the following general form of the learning rule must be applied.

$$\dot{N}_{ij} = \begin{cases} K(t)(X_j - N_{ij}) & \text{for neurons } i \subset V_c \\ 0 & \text{for neurons } i \supset V_c \end{cases} \tag{2.37}$$

where V_c is a neighbourhood around neuron "c", and $K(t)$ is a function which linearly decreases with learning time, ensuring that the learning terminates in finite time.

There are several other self organizing neural networks with potential applications in the identification and control of dynamic systems. Among these, it is worth mentioning the cognitron and neocognitron of Fukushima [30, 31, 32], and the adaptive resonance theory network of Carpenter and Grossberg [33, 34]. Also, other neural networks with applicability to identification and control, that operate in a supervised environment, are the radial basis function networks [35], and the cerebellar articulation controller network [36].

2.6 SUMMARY

This chapter has presented a summary review of neural networks. This review has covered some historical aspects of the evolution of neural networks, as well as two

classifications that allow a systematic study of a variety of network structures. In the taxonomy proposed by Lippmann [17], neural networks have been grouped taking into consideration whether their inputs were continuous-valued or binary-valued signals. Also, in this taxonomy neural networks have been further divided according to their training environment into supervised and unsupervised training networks. Examples of network topologies have also been presented to support the classification. In the unifying model for neural networks that has been suggested by Hunt et al [16], the processing elements were formed by three components: a weighted summer, a linear siso system, and a non-dynamic nonlinear function. In this unifying model, neural networks have been classified depending upon the selection of the linear siso system, into static and dynamic networks.

Also, this chapter has included a categorization of learning algorithms for training neural networks. Two groups of learning algorithms have been described: supervised and unsupervised learning. Examples of each of these groups have been reported. A more exhaustive analysis of learning algorithms for perceptron networks, has been presented, and details on the deduction of the α-LMS algorithm and the μ-LMS algorithm have been contemplated to remark the differences between error-correction based algorithms and gradient descent-based algorithms.

Finally, this chapter has summarized salient characteristics of three important neural network structures: multilayer perceptron networks, Hopfield or recurrent networks, and self organizing maps or Kohonen networks, that are commonly used to propose novel control and identification system applications.

Chapter 3

VARIABLE STRUCTURE CONTROL-BASED-LEARNING ALGORITHMS FOR FEEDFORWARD NEURAL NETWORKS

3.1 INTRODUCTION

It is based upon the parallel processing capability of the human brain that artificial neural networks are designed. This parallel processing capability is performed by the dynamics of interconnected neurons with learning ability. In the context of artificial neural networks, learning means the ability to adapt a network so that the output responses to some input patterns are as close as possible to their corresponding responses. In supervised learning, artificial neurons adjust their connection weights depending on the input signals which they receive and on the error signals obtained from the difference between associated teaching signals and the actual network outputs. In an unsupervised environment on the other hand, neurons modify their connection weights depending upon their input signals and internal states.

As a general rule, supervised learning algorithms are designed relying on the principle of minimal disturbance. That is, the output error for the current training pattern is reduced by adapting the connection weights so that the responses already learned are minimally disturbed [20]. Two types of supervised learning rules are:

- error-correction rules,

- gradient descent rules.

In error-correction rules the weights of the network are adapted by controlling the convergence to zero of an error equation. If the error correction is proportional to the error itself, the learning rule is a linear one, as in the Widrow-Hoff delta rule [20]. Otherwise, it is a nonlinear error-correction rule, as in the perceptron learning rule of Rosenblatt [3]. Error correction algorithms are suitable for learning temporal sequences such as in prediction and control problems. In gradient descent rules, on the other hand, the connection weights of the network are adjusted during each pattern presentation, based on the method of steepest descent or other gradient method, with the objective of minimizing a mean-square error, averaged over all training patterns [20, 37, 38]. A widely used gradient descent algorithm, which has both temporal and structural learning capabilities, is the generalized delta rule or backpropagation algorithm [13].

The learning process in an artificial neural network can be viewed as a process of estimating a mapping that transforms input signals into corresponding output signals. In supervised learning, a set of examples of input-output pairs of the mapping to be learned is provided to the neural network. In this context, learning is interpreted as an approximation problem [35, 39, 40]. That is, the problem of finding a function "$\hat{f}(W, X)$" to approximate a continuous or discrete mapping "$f(X)$" by suitable selection of the parameters "W", which belong to some set "P". When the function "$\hat{f}$" is given, the approximation problem reduces to finding the set of parameters "W" that provide the best possible approximation of "$f(X)$" on the set of examples. Mathematically speaking, the problem can be formulated as follows: Given a function "$f(X)$" defined on a set "$\mathcal{X}$", and an approximating function "$\hat{f}(W, X)$" that depends on $W \in P$ and the real valued vector "X", find the parameters "W^*" such that

$$\|\hat{f}(W^*, X) - f(X)\| \leq \epsilon \tag{3.1}$$

$\forall W^* \in P$, and $\epsilon > 0$.

Observe that the existence of a best approximation depends firstly upon the class of approximating function "$\hat{f}$" used [35], and secondly, on the learning algorithm used to find the appropriate values of the parameters "W" for the given choice of "$\hat{f}$". Typical approximating functions $\hat{f} : \Re^n \rightarrow \Re$, which represent neural networks are the following:

- Single layer neural networks: the corresponding approximating function, which is linear, is described by

$$f(W, X) = W^T X. \tag{3.2}$$

- Two layer neural networks: the approximating function is linear with respect to a basis of functions $\{\Gamma_i\}_{i=1}^m$ of the original inputs "X".

$$f(W, X) = W^T \Gamma(X). \tag{3.3}$$

Note that these approximating functions may be regarded as spline interpolation, or as extensions in series of orthogonal polynomials.

- Multilayer neural networks (more than two layers): the approximating function is a nested nonlinear function of the type

$$f(W, X) = \Gamma(W_I^T \Gamma(W_{H_p}^T \Gamma(\ldots \Gamma(W_{H_1}^T X)\ldots))) \tag{3.4}$$

where "W_I^T", "$W_{H_p}^T$", and "$W_{H_1}^T$" represent the input weight matrix, the "p-th" hidden weight matrix, and the output weight matrix, respectively. A common choice of the nonlinear function "$\Gamma(.)$" is the sigmoid function. Other choices include saturation functions, threshold functions, etc. It has been proved that this type of network, with a layer of hidden units, can approximate arbitrarily well any continuous multivariable function [24].

Other functions that may be readily realised in three layer networks are radial basis functions described by

$$f(W, X) = c_0 + \sum_{i=1}^{n} c_i \, \phi(\|X - W^i\|), \tag{3.5}$$

where the input nodes contain the vector variable "X". The hidden layer has "n" nodes; one for each centre "$W^{(i)}$". The components of "$W^{(i)}$", say "w_{ij}", are the values that link the i-th input node to the j-th hidden node. For example, the j-th hidden node "z_j" is obtained as

$$z_j = \|X - W^{(i)}\|^2 = \sum_{i=1}^{p} (x_i - w_{ij})^2. \tag{3.6}$$

The radial basis function "ϕ" is a continuous function from $\Re^+$ to $\Re$, which is applied to "$(z_j)^{1/2}$" in order to contribute in producing an output of the network at the third layer, according to equation 3.5. The parameters "c_i" represent the connection weights between the hidden layer and the output layer. Some of the commonly used radial basis functions are the following

$$\phi(z) = z^2 \log(z), \tag{3.7}$$

$$\phi(z) = \exp(-\frac{z^2}{2}), \tag{3.8}$$

$$\phi(z) = (z^2 + k^2)^{1/2}. \tag{3.9}$$

The functions 3.7. 3.8, and 3.9 represent thin plate splines, gaussian basis functions, and multiquadratic basis functions, respectively. Important contributions on the applications of radial basis functions neural networks have been reported by Broomhead and Lowe [41], and Chen et al [42], among many others.

In this work, the general class of approximating functions to be used are the linear functions described by equation 3.2, that represents single layer networks; and the nested nonlinear functions of the type described by equation 3.4, that represents multilayer neural networks. On the other hand, the learning algorithms to be used are derived from a continuous-time and a discrete-time variable structure control (VSC) framework [43, 44]. These algorithms operate by adjusting the connection weights of the networks so that a sliding regime, or a quasi-sliding regime, respectively, is induced on the learning error equations of the networks. Before presenting the derivation of the variable structure control-based-learning algorithms in section 3.3, the next section contains an outline of VSC design topics.

3.2 VSC DESIGN

The design of VSC systems and their associated sliding regimes have become a powerful methodology for dealing with the robust control of nonlinear dynamical systems which present both parametric and unmodeled dynamics uncertainties. Detailed expositions on the state of the art and the potential applicabilities of this design methodology can be found in survey articles by Utkin [44, 45], Sira-Ramirez [43, 46, 47], and several books [48, 49].

In variable structure systems design, it is possible to induce the system dynamics to evolve in a given surface, or manifold, that results in a dynamical behaviour of lower order than the original system. Once on the surface, the system dynamics are largely determined by the design parameters and equations defining the manifold, and gives the opportunity of exploiting new properties originally absent from the system. The design of a variable structure controller involves two steps:

- Selection of a feedback control law to accomplish manifold reachability, and
- once on the manifold, the selected control law must be able to maintain the evolution of the system constrained to this manifold.

3.2.1 CONTINUOUS-TIME VSC DESIGN

Mathematically, the variable structure control design for continuous-time nonlinear systems may be summarized as follows.

Consider the single-input dynamic system described by

$$x^{(n)} = f(X) + g(X)u; \quad X(t_0) = X_0 \tag{3.10}$$

where $X \in \Re^n$ is the state vector, "$f(X)$" and "$g(X)$" are nonlinear functions of "X". Let the tracking error in the variable "x" be defined by $\tilde{x} = x - x_d$, where x_d represents a desired state variable, possibly time-varying. Furthermore, let the scalar equation $s(X,t) = 0$ define a time-varying manifold in the state-space $\Re^n$ described by

$$s(X,t) = (D + \lambda)^{n-1}\,\tilde{x} \tag{3.11}$$

where $D = \frac{d}{dt}$ is the operator differentiation, and λ is a strictly positive constant. Observe that the problem of tracking x_d is equivalent to the problem of reaching and remaining on the surface $s(X,t)$ for all $t > t_0$. Reaching the surface $s(X,t)$ can be achieved by choosing the control law u of 3.10 such that the following condition is satisfied

$$\frac{1}{2}\,D(s^2) \leq -\eta\,|s| \tag{3.12}$$

where η is a strictly positive constant.

Condition 3.12 is called the sliding condition, and can be geometrically interpreted as a movement of trajectories off the surface pointing towards the surface. Figure 3.1 illustrates the sliding condition. It is important to note that if $X_d(t_0) \neq X(t_0)$, satisfying condition 3.12 guarantees that the surface "$s(X,t)$" is reached in a finite time smaller than $(|s(X(t_0),t_0)| + \eta\,t_0)/\eta$ [50].

The behaviour of the system on the sliding surface is known as sliding regime, and is defined by the equation

$$D\,s(X,t) = 0 \tag{3.13}$$

The equivalent control law is the control action needed to maintain the sliding regime, and therefore is obtained by solving equation 3.13 for the control input "u".

A VSC law is obtained by letting the control function "u" take one of two feedback values according to the sign of "$s(X,t)$". This is

$$u = \begin{cases} u^+(X) & \text{for } s(X,t) > 0 \\ u^-(X) & \text{for } s(X,t) < 0 \end{cases} \tag{3.14}$$

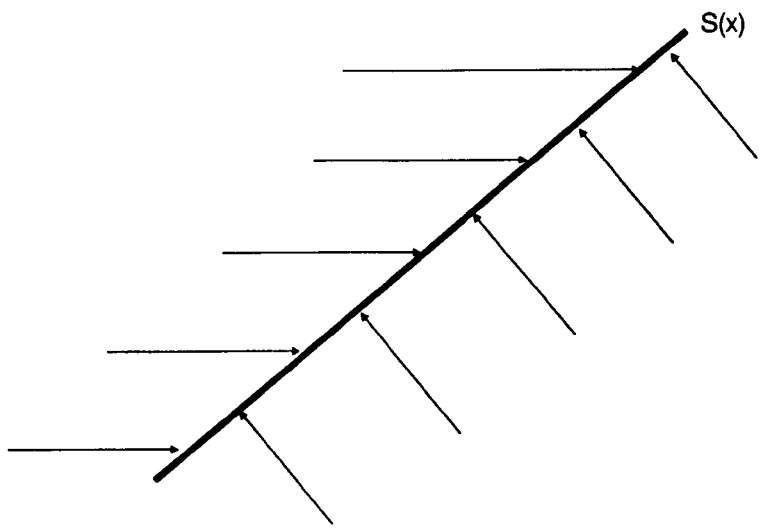


Figure 3.1: The sliding condition

with $u^+(X) \neq u^-(X)$.

In summary, variable structure system design involves selecting a suitable switching function "$s(X,t)$" of the tracking error, according to equation 3.11, and then choosing a feedback control law "u" such that the induced system dynamics remain stable despite the presence of model imprecision and of disturbances. This objective can be achieved by designing a control law that is discontinuous across "$s(X,t)$", and allows the sliding condition 3.12 to be verified. Due to imperfections in implementing the associated switching controls, the sliding dynamics present chattering motions close to the sliding manifold.

3.2.2 DISCRETE-TIME VSC DESIGN

The main developments in the theory of dicrete-time variable structure systems design have been put forward by the contributions of Miloslavjevic [51], in the context of sample data systems; Opitz [52], Magaña and Zak [53], and Sarpturk et al [54] for discrete-time linear systems, and more recently by the contributions of Drakunov and Utkin [55], Furuta [56], and Sira-Ramirez [43].

For discrete-time systems, the extension of the continuous-time condition for the existence of a sliding regime do not necessarily guarantee sliding dynamics with chattering motions close to the manifold. The term quasi-sliding regime was introduced by Miloslavjevic [51] to characterize sliding dynamics in discrete-time systems. It has been found that a quasi-sliding regime exists on the zero level set of an out-

put switching function, if and only if the nonlinear discrete-time system has relative degree equal to one [43]. The relative degree determines the time delay experienced by the input signals of a system before they influence its outputs.

The design of quasi-sliding regimes for discrete-time nonlinear systems may be summarized as follows:

Consider a smooth single-input single-output nonlinear system described by

$$x(k+1) = F(x(k), u(k)); \quad k = 0, 1, 2, ... \tag{3.15}$$

$$y(k) = h(x(k)), \tag{3.16}$$

where $x \in \mathcal{X} \subset \Re^n$, $u \in \Re$, $y \in \Re$, and the mappings "F" and "h" are assumed to be analytic. The sliding manifold is defined as a smooth curve described by the level set

$$h^{-1}(0) = \{x \in \mathcal{X} : h(x) = 0\} \tag{3.17}$$

A variable structure feedback control law for the system described by equations 3.15 and 3.16 is obtained by letting

$$u = \begin{cases} u^+(x) & \text{for } h(x) > 0 \\ u^-(x) & \text{for } h(x) < 0 \end{cases} \tag{3.18}$$

with $u^+(x) > u^-(x)$. A necessary condition for the existence of convergent quasi-sliding dynamics about "$h^{-1}(0)$" is that a quasi-sliding regime exists about such a manifold. A convergent quasi-sliding regime exists on "$h^{-1}(0)$" if and only if

$$|y(k+1)\, y(k)| < y^2(k) \tag{3.19}$$

Furthermore, if the system described by 3.15 and 3.16 has relative degree equal to 1, then there exists a variable structure feedback control law of the form 3.18 which creates a quasi-sliding regime on "$h^{-1}(0)$" [43]. On the other hand, the equivalent control "$u_{eq}(x)$" is the control function that maintains the system trajectories on the manifold $y = h(x) = 0$.

It is important to point out that under condition 3.19 the chattering behaviour of the dynamic system about the suggested quasi-sliding manifold may or may not exist at all. Moreover, the quasi-sliding regime may be achieved without discontinuous controller of the form 3.18. Illustrative examples of these situations are presented in [43].

3.3 VSC LEARNING ALGORITHMS

In section 3.1, it has been emphasized that the supervised learning process in artificial neural networks may be interpreted as the approximation problem of finding the

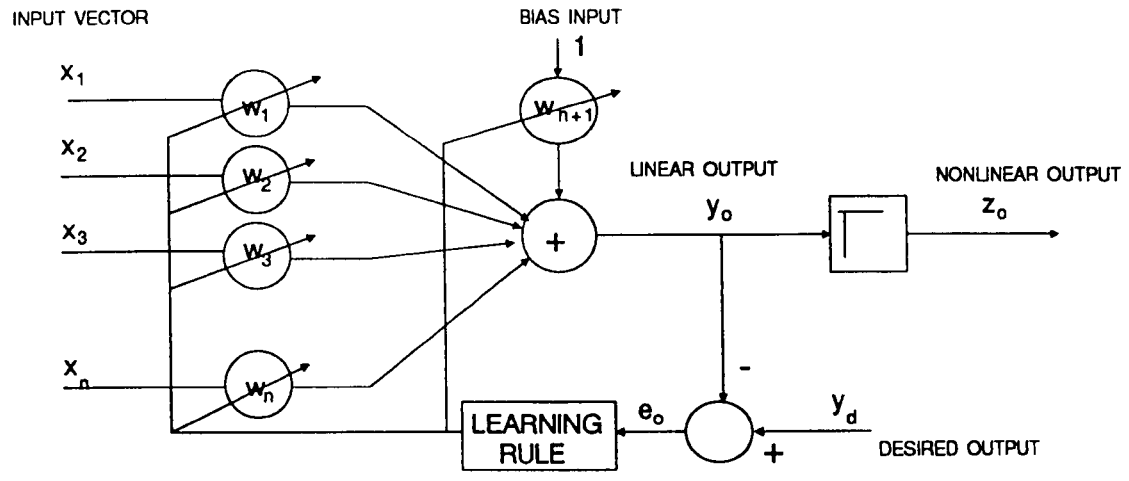


Figure 3.2: Linear adaptive neuron

appropriate approximating function, or when this is given, finding the parameters of that function to track another function in the best possible way. On the other hand, in section 3.2 the idea of inducing the trajectories of a dynamic system to evolve on a defined manifold has been introduced. This manifold may be selected as a function of an error signal that depends on the difference between the output of the system and a desired response. The design of variable structure systems can be incorporated in the learning process of a neural network to achieve function approximations.

3.3.1 CONTINUOUS-TIME VSC LEARNING RULES

3.3.1.1 THE ADALINE CASE

Consider the adaptive linear neuron depicted in figure 3.2. It is easy to verify that the following equations hold true

$$y_o(t) = X^T(t)W(t) + b\,w_{n+1}(t) \tag{3.20}$$

$$e(t) = y_d(t) - y_o(t), \tag{3.21}$$

where $y_o(t) \in \Re$ is the neuron output, $X(t) \in \Re^n$ is the neuron input vector, $W(t) \in \Re^n$ and $w_{n+1}(t) \in \Re$ represent variable weight values, $y_d(t) \in \Re$ is the desired output, "b" represents a constant bias or threshold input signal, and $e(t) \in \Re$ is the error between the desired and the actual neuron outputs.

It is possible to think of the weights of the neuron as a dynamic system modeled

by an equation of the form

$$\dot{W}_a(t) = F(U(t)), \tag{3.22}$$

where $W_a^T(t) = (W(t)\ w_{n+1}(t))^T$ is the state vector, and "$U(t)$" is a control input that must be interpreted as an updating function. The output of this dynamic system is assumed to be a function of the type

$$e(t) = h(W_a(t)) \tag{3.23}$$

In particular, the dynamic system to model the behaviour of the weights and the proposed output are described by the equations

$$\dot{W}_a(t) = U(t) \tag{3.24}$$

$$e(t) = y_d(t) - y_o(t) \tag{3.25}$$

It can be seen that from equation 3.11, the sliding manifold for the dynamic system represented by equations 3.24 and 3.25 may be described by

$$s(W_a, t) = e(t) \tag{3.26}$$

Observe that the sliding condition 3.12 is equivalent to satisfying

$$\dot{s}(W_a, t) \leq -\eta\, sign(s(W_a, t)), \tag{3.27}$$

where "$sign(s(.))$" is a function defined by

$$sign(s(.)) = \begin{cases} +1 & \text{if } s(.) > 0 \\ 0 & \text{if } s(.) = 0 \\ -1 & \text{if } s(.) < 0 \end{cases} \tag{3.28}$$

Let the augmented vector "$X_a(t)$" be defined by $X_a(t) = (X(t)\ b)^T$. It is assumed that both the augmented input vector "X_a" and the desired output "$y_d(t)$" are bounded signals, and present bounded time derivatives. This is,

$$\| X_a(t) \| \leq B_x; \qquad \| \dot{X}_a(t) \| \leq B_{\dot{x}}; \quad \forall t \tag{3.29}$$

$$\| y_d(t) \| \leq B_y; \qquad \| \dot{y}_d(t) \| \leq B_{\dot{y}}; \quad \forall t \tag{3.30}$$

Similarly, the weight vector "$W_a(t)$" is assumed to be bounded by means of

$$\| W_a(t) \| \leq B_w \quad \forall t \tag{3.31}$$

The boundedness of "$W_a(t)$" will be proved later.

THEOREM 1.

If the control input "$U(t)$" for the adaptation law described by equation 3.24 is selected according to the equation

$$U(t) = \frac{X_a(t)}{(X_a(t))^T (X_a(t))}(k\, sign(e(t)) \tag{3.32}$$

with $k > \eta + B_{\dot{y}} + B_{\dot{x}} B_w$, then given any initial condition "$e(0)$", the learning error "$e(t)$" converges to zero in finite time "t_r" estimated by

$$t_r \leq \frac{|\, e(0)\,|}{\eta}, \tag{3.33}$$

and a sliding regime is sustained on $e(t) = 0$ for all $t < t_r$.

PROOF.

Consider a Lyapunov function candidate given by

$$v(e(t)) = \frac{1}{2} e^2(t) \tag{3.34}$$

The time derivative of "$v(e(t))$" is given by

$$\begin{aligned}
\dot{v}(e(t)) &= e(t)\,(\dot{y}_d(t) - \dot{X}_a^T(t)\, W_a(t) - X_a^T(t)\, \dot{W}_a(t)) \\
&= e(t)\,(\dot{y}_d(t) - \dot{X}_a^T(t)\, W_a(t) - X_a^T(t)\, U(t)) \\
&= e(t)\,(\dot{y}_d(t) - \dot{X}_a^T(t)\, W_a(t) - k\, sign(e(t))) \\
&= e(t)\,(\dot{y}_d(t) - \dot{X}_a^T(t)\, W_a(t)) - k\,|e(t)| \\
&\leq (B_{\dot{y}} - B_{\dot{x}}\, B_w)\,|e(t)| - k\,|e(t)| = (B_{\dot{y}} - B_{\dot{x}}\, B_w - k)|e(t)| \leq 0
\end{aligned} \tag{3.35}$$

Thus, the controlled trajectories of the learning error converge to zero. In order to show that such a convergence takes place in a finite time "t_r", and that a sliding regime exists on $e(t) = 0$, note that the sliding condition 3.27 can be written as

$$\dot{e}(t) \leq -\eta\, sign(e(t)), \tag{3.36}$$

and on the other hand,

$$\begin{aligned}
\dot{e}(t) &= \dot{y}_d(t) - \dot{X}_a^T(t)\, W_a(t) - X_a^T(t)\, \dot{W}_a(t) \\
&= \dot{y}_d(t) - \dot{X}_a^T(t)\, W_a(t) - k\, sign(e(t))
\end{aligned} \tag{3.37}$$

Note that $|e(t)| = e(t)\, sign(e(t))$ and that

$$\dot{e}(t)e(t) \leq (B_{\dot{y}} - B_{\dot{x}} B_w)|e(t)| - k|e(t)| = (B_{\dot{y}} - B_{\dot{x}} B_w - k)|e(t)|,$$

and if $k > \eta + B_{\dot{y}} + B_{\dot{x}}\, B_w$ then inequality 3.36 is verified.

Observe that

$$\int_0^\tau \dot{e}(t)\,dt \leq -\eta \int_0^\tau sign(e(t))\,dt \tag{3.38}$$

and for $\tau < t_r$

$$e(\tau) - e(0) \leq -\eta\,\tau\,sign(e(0)) \tag{3.39}$$

At time $t = t_r$, the value of "$e(t)$" is zero and therefore

$$e(0) \leq \eta\,t_r\,sign(e(0)) \tag{3.40}$$

Multiplying both sides of inequality 3.40 by "$sign(e(0))$" yields

$$|e(0)| \geq \eta\,t_r \tag{3.41}$$

and therefore $t_r \leq \frac{|e(0)|}{\eta}$.

It is important to point out that if the vector "$\dot{X}_a(t)$" is measurable, the control input "$U(t)$" shown in equation 3.32 may be rewritten in the following terms

$$U(t) = \frac{X_a(t)}{(X_a(t))^T\,(X_a(t))}(k\,sign(e(t)) - \dot{X}_a^T(t)\,W_a(t)) \tag{3.42}$$

Equation 3.42 represents a more relaxed variable structure feedback control action, with "k" being a positive design constant satisfying $k > \eta + B_{\dot{y}}$.

3.3.1.2 BOUNDEDNESS OF SOLUTIONS FOR THE WEIGHTS

This section contains an analysis of the average behaviour of the controlled weight variables, and includes the consideration of the invariance conditions satisfied after the sliding regime starts on the sliding manifold.

The invariance conditions may be expressed as the verification of the equations

$$e(t) = 0 \tag{3.43}$$

$$\dot{e}(t) = 0 \tag{3.44}$$

Equation 3.44 implies

$$\dot{y}_d(t) - \dot{X}_a^T(t)\,W_a(t) - X_a^T(t)\dot{W}_a(t) = 0 \tag{3.45}$$

The equivalent weight vector "$W_{a_{eq}}(t)$" is a virtual vector variable used to describe the regulated evolution of any error learning trajectory satisfying the condition $e(t) = 0$, with $t > t_r$. This is

$$\dot{W}_{a_{eq}}(t) = -\frac{X_a(t)}{(X_a(t))^T(X_a(t))}(\dot{y}_d(t) - \dot{X}_a^T(t)\,W_{a_{eq}}(t)) \tag{3.46}$$

$$= -\left(\frac{X_a(t)\,\dot{X}_a^T(t)}{X_a^T(t)\,X_a(t)}\right) W_{a_{eq}}(t) + \left(\frac{X_a(t)}{X_a^T(t)\,X_a(t)}\right) \dot{y}_d(t) \tag{3.47}$$

Note that the average weight vector trajectory satisfies a linear time-varying vector differential equation with forcing function represented by the bounded function "$\dot{y}_d(t)$". The boundedness of the vector of variable weights, after the sliding regime occurs is exclusively dependent upon the variation of the augmented input vector "$X_a(t)$" and that of the desired signal "$y_d(t)$". In particular, observe that if $X_a(t) = X_a$ and $y_d(t) = y_d$ are constant, the equivalent adaptation law 3.47 would satisfy $\dot{W}_{a_{eq}}(t) = 0$ and therefore, $W_{a_{eq}}(t) = W_a = constant$. If on the other hand, only $X_a(t) = X_a$ is constant, then the equivalent adaptation law would satisfy

$$\dot{W}_{a_{eq}}(t) = \left(\frac{X_a(t)}{X_a^T(t)\, X_a(t)} \right) \dot{y}_d(t) \tag{3.48}$$

In this case $W_{a_{eq}}(t) = (\frac{X_a(t)}{X_a^T(t) X_a(t)})\, y_d(t)$ which means that the minimum norm solution "$W_{a_{eq}}(t)$" of $e(t) = 0 = y_d(t) - X_a^T(t)\, W_{a_{eq}}(t)$ is also a solution of the differential equation defining "$W_{a_{eq}}(t)$". In order to prove boundedness in the general case, when both "$X_a(t)$" and "$y_d(t)$" are time-varying functions the following definition is necessary [57].

DEFINITION.

Denote by "$F(t)$" the time-varying matrix

$$F(t) = \frac{X_a(t)\, \dot{X}_a^T(t)}{X_a^T(t)\, X_a(t)} \tag{3.49}$$

The differential equation $\dot{W}_{a_{eq}}(t) = F(t)\, W_{a_{eq}}(t)$ is said to be uniformly stable if there exists a positive constant "γ" such that, for all "t_0" and all $t > t_0$, the state transition matrix "$\Phi(t, t_0)$", corresponding to the matrix "$F(t)$", satisfies

$$\| \Phi(t, t_0) \| < \gamma \tag{3.50}$$

This definition allows to formulate the following proposition.

PROPOSITION.

Suppose the system $\dot{W}_{a_{eq}}(t) = F(t)\, W_{a_{eq}}(t)$ is uniformly stable and let "$\dot{y}_d(t)$" be absolutely integrable. Then, the solutions to equation 3.47 are bounded.

PROOF.

Consider the inequalities

$$\| X_a(t) \| \geq 1 \tag{3.51}$$

$$\int_{t_0}^{\infty} \frac{|\dot{y}_d(t)|}{\| X_a(t) \|}\, dt \leq \int_{t_0}^{\infty} |\dot{y}_d(t)|\, dt = \beta \tag{3.52}$$

and assume that the initial states "$W_{a_{eq}}(t_0)$" are bounded by a constant "W".

From the variation of constants formulae, the solutions of the linear time-varying differential equation 3.47 are written as

$$W_{a_{eq}}(t) = \Phi(t,t_0)\,W_{a_{eq}}(t_0) + \int_{t_0}^{t} \Phi(t,\tau)\,\frac{X_a(\tau)}{X_a^T(\tau)\,X_a(\tau)}\dot{y}_d(\tau)\,d\tau \tag{3.53}$$

by virtue of equation 3.50, the norm of "$W_{a_{eq}}(t)$" satisfies

$$\begin{aligned} \|W_{a_{eq}}(t)\| &\leq \|\Phi(t,t_0)\|\|W_{a_{eq}}(t_0)\| + \|\int_{t_0}^{t} \Phi(t,\tau)\frac{X_a(\tau)}{X_a^T(\tau)X_a(\tau)}\dot{y}_d(\tau)d\tau\| \\ &\leq \|\Phi(t,t_0)\|\|W_{a_{eq}}(t_0)\| + \int_{t_0}^{t} \|\Phi(t,\tau)\|\frac{|\dot{y}_d(\tau)|}{\|X_a(\tau)\|}d\tau \\ &\leq \gamma(W+\beta); \quad \forall t > t_0. \end{aligned} \tag{3.54}$$

3.3.1.3 ROBUSTNESS FEATURES WITH RESPECT TO EXTERNAL PERTURBATIONS

A key feature of sliding mode control is the insensitivity of the regulated variables with respect to external bounded perturbations affecting the underlying system. In this analysis, it is assumed that the external perturbation input vector $\Xi(t) = (\xi_1(t),...,\xi_n(t))^T$ has a bounded norm not larger than the norm of the input vector "$X(t)$". It is also considered that the norm of the time derivative of the external perturbation vector is bounded. In other words

$$\|\Xi(t)\| = \sqrt{\xi_1^2(t) + ... + \xi_n^2(t)} \leq B_\xi < B_x \quad \forall t \tag{3.55}$$

$$\|\dot{\Xi}(t)\| = \sqrt{\dot{\xi}_1^2(t) + ... + \dot{\xi}_n^2} \leq B_{\dot{\xi}} \quad \forall t \tag{3.56}$$

The augmented external perturbation vector "$\Xi_a(t)$" is defined as

$$\Xi_a(t) = (\xi_1(t),...,\xi_n(t),0)^T \tag{3.57}$$

Equation 3.57 means that the constant input "b" to the bias weight "$w_{n+1}(t)$" is not influenced by the perturbation signal "$\Xi(t)$".

If the external perturbation affects the values of the input signal "$X(t)$" to the adaptive neuron, say in an additive way, then the perturbed learning error $\hat{e}(t) = y_d(t) - y_o(t)$ is given by

$$\hat{e}(t) = y_d(t) - (X_a(t) + \Xi_a(t))^T\,W_a(t) \tag{3.58}$$

Since the time derivative of perturbed input signal is not available for measurements, the adaptation law for the weights is selected as the type proposed in equation 3.32. This is:

$$\dot{W}_a(T) = \frac{(X_a(t) + \Xi_a(t))}{(X_a(t) + \Xi_a(t))^T(X_a(t) + \Xi_a(t))}(k\,sign(\hat{e}(t))) \tag{3.59}$$

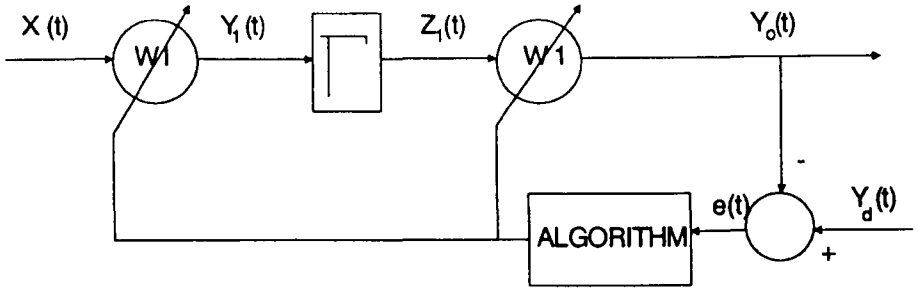


Figure 3.3: Two layer feedforward neural network.

THEOREM 2.

If the adaptation law for the augmented weight vector "$W_a(t)$" in an adaptive neuron influenced by an additive perturbation at its input signal is chosen according to equation 3.59, with "k" being a positive constant satisfying

$$k > \eta + B_{\dot{y}} + B_w(B_{\dot{x}} + B_{\dot{\xi}}), \tag{3.60}$$

then, given an arbitrary initial condition "$\hat{e}(0)$", the perturbed learning error "$\hat{e}(t)$" converges to zero in finite time "$\hat{t}_r$" estimated by

$$\hat{t}_r \leq \frac{|\hat{e}(0)|}{\eta} \tag{3.61}$$

in spite of all bounded values of the perturbation inputs and its time derivatives. Moreover, a sliding motion is sustained on $\hat{e}(t) = 0$ for all $t > \hat{t}_r$.

PROOF.

Using equation 3.58 with condition 3.60, the proof is identical as the proof of theorem 1.

3.3.1.4 THE TWO LAYER NEURAL NETWORK CASE

Consider the two layer neural network illustrated in figure 3.3. Notice that the following equations hold valid

$$Y_o(t) = (W1(t))^T Z_1(t); \quad W1(t) \in \Re^{n1 \times p} \tag{3.62}$$

$$Z_1(t) = \Gamma(Y_1(t)); \qquad Z_1(t) \in \Re^{n1} \tag{3.63}$$

$$Y_1(t) = (WI(t))^T X_a(t); \quad WI(t) \in \Re^{(n+1)\times n1} \tag{3.64}$$

The input vector $X_a(t) = (x_1(t), x_2(t), ..., x_n(t), b)^T$ is an "$(n+1)$" dimensional array whose (n+1)-th component is a fixed bias value connected through a variable weight to every neuron in the input layer. This bias value is also connected through variable weights to every neuron in the network. The nonlinear activation function "$\Gamma(.)$" is assumed to be a differentiable function. Note that the output error is defined by the vector equation

$$E(t) = Y_d(t) - Y_o(t), \tag{3.65}$$

where $E(t) \in \Re^p$. Here, it is assumed that both the input vector "$X_a(t)$" and the desired output vector "$Y_d(t)$" are bounded vectors with bounded time derivatives. This is

$$\|X_a(t)\| < B_x \qquad \|\dot{X}_a(t)\| < B_{\dot{x}} \tag{3.66}$$

$$\|Y_d(t)\| < B_y \qquad \|\dot{Y}_d(t)\| < B_{\dot{y}} \tag{3.67}$$

The time derivative of "$E(t)$" may be expressed as

$$\begin{aligned}
\dot{E}(t) &= \dot{Y}_d(t) - (\dot{W}1(t))^T Z_1(t) - (W1(t))^T \dot{Z}_1(t) \\
&= \dot{Y}_d(t) - (\dot{W}1(t))^T Z_1(t) - (W1(t))^T \left(\frac{\partial \Gamma(Y_1(t))}{\partial Y_1(t)}\right) \dot{Y}_1(t) \\
&= \dot{Y}_d(t) - (\dot{W}1(t))^T Z_1(t) - (W1(t))^T \left[\frac{\partial \Gamma(Y_1(t))}{\partial Y_1(t)} \left((\dot{W}I(t))^T X_a(t)\right.\right. \\
&+ \left.\left.(WI(t))^T \dot{X}_a(t)\right)\right] \\
&= \dot{Y}_d(t) - (W1(t))^T \left(\frac{\partial \Gamma(Y_1(t))}{\partial Y_1(t)}\right) (WI(t))^T \dot{X}_a(t) - (\dot{W}1(t))^T Z_1(t) \\
&- (W1(t))^T \left(\frac{\partial \Gamma(Y_1(t))}{\partial Y_1(t)}\right) (\dot{W}I(t))^T X_a(t)
\end{aligned} \tag{3.68}$$

The weight matrices dynamics are modelled by the following matrix differential equations

$$\dot{W}I(t) = UI(t), \tag{3.69}$$

$$\dot{W}1(t) = U1(t), \tag{3.70}$$

where $UI(t) \in \Re^{(n+1)\times n1}$, and $U1(t) \in \Re_{n1\times p}$. The columns of "$WI(t)$" and "$W1(t)$" are considered bounded by means of $\|WI_i(t)\| < B_{WI_i}$ and $\|W1_i(t)\| < B_{W1_i}$, respectively.

THEOREM 3.

If the control input matrices "$UI(t)$" and "$U1(t)$" of the dynamic equations 3.69 and 3.70 are respectively chosen as

$$UI(t) = \frac{X_a(t)}{(X_a(t))^T(X_a(t))}\left[(Z_1(t))^T\left(\frac{\partial\Gamma(Y_1(t))}{\partial Y_1(t)}\right)^{-1}\right], \tag{3.71}$$

$$U1(t) = -W1(t) + \frac{\Gamma(Y_1(t))}{(\Gamma(Y_1(t)))^T(\Gamma(Y_1(t)))}\left[K\,SIGN(E(t))\right], \tag{3.72}$$

where "K" is a diagonal matrix whose diagonal elements satisfy the condition

$$k_i > \eta_i + B_{\dot{y}} + B_{W1_i} B_{WI_i} B_{\dot{x}} \tag{3.73}$$

then, given an arbitrary initial condition "$E(0)$", the learning error "$E(t)$" converges to zero in finite time "t_{r_i}" estimated by

$$t_{r_i} \leq \frac{|e_i(0)|}{\eta_i} \tag{3.74}$$

and a sliding regime is sustained on the manifolds "$E(t) = 0$" for all $t > t_{r_i}$.

PROOF.

Let the Lyapunov function candidate "$v(e_i(t))$" for the i-th learning error "$e_i(t)$" be defined by

$$v(e_i(t)) = \frac{1}{2}e_i^2(t) \tag{3.75}$$

where "$e_i(t)$" represents the i-th component of the error vector. This is

$$e_i(t) = Y_{d_i}(t) - Y_{o_i}(t) \tag{3.76}$$

Observe that the time derivative of "$v(e_i(t))$" may be expressed as

$$\begin{aligned}\dot{v}(e_i(t)) &= e_i(t)\,\dot{e}_i(t)\\ &= e_i(t)\left(\dot{y}_{d_i}(t) - (W1_i(t))^T\left(\frac{\partial\Gamma(Y_1(t))}{\partial Y_{1_i}(t)}\right)(WI_i(t))^T\dot{X}_a(t)\right.\\ &- \left.(\dot{W}1_i(t))^T\Gamma(Y_1(t)) - (W1_i(t))^T\left(\frac{\partial\Gamma(Y_1(t))}{\partial Y_{1_i}(t)}\right)(\dot{W}I_i(t))^T X_a(t)\right)\end{aligned} \tag{3.77}$$

Substituting equations 3.69, 3.70, 3.71, and 3.72 into equation 3.78 yields

$$\begin{aligned}\dot{v}(e_i(t)) &= e_i(t)(\dot{y}_{di}(t) - (W1_i(t))^T\left(\frac{\partial\Gamma(Y_1(t))}{\partial Y_{1_i}(t)}\right)(WI_i(t))^T\dot{X}_a(t)\\ &- k_i\,sign(e_i(t)))\\ &\leq |e_i(t)|(B_{\dot{y}} - B_{W1_i}B_{WI_i}B\dot{x}) - k_i|e_i(t)|\\ &\leq (B_{\dot{y}} - B_{W1_i}B_{WI_i}B\dot{x} - k_i)|e_i(t)| \leq 0.\end{aligned} \tag{3.78}$$

Therefore, the trajectories of the learning error converge to zero. The proof of the second part of the theorem follows the same guidelines as theorem 1.

Observe that the matrix "$(\frac{\partial\Gamma(Y_1(t))}{\partial Y_{1_i}(t)})$" in equation 3.69 is a diagonal matrix. Also, notice that if the derivative of the input signal vector "$\dot{X}_a(t)$" is available for measurements, then a sliding regime can be induced on the learning error manifolds $E(t) = 0$, for any initial condition "$E(0)$" in a finite time, if the weight matrices "$WI(t)$" and "$W1(t)$" are selected according to

$$\dot{W}I(t) = -\left[\frac{(\dot{X}_a(t))^T(X_a(t))}{(X_a(t))^T(X_a(t))}\right] WI(t) \tag{3.79}$$

$$\dot{W}1(t) = \frac{Z_1(t)}{(Z_1(t))^T(Z_1(t))}\,[K\,SIGN(E(t))]^T \tag{3.80}$$

with "K" a diagonal matrix whose diagonal elements satisfy

$$k_i = \eta_i + B\dot{y}. \tag{3.81}$$

It is important to remark that the boundedness of the weight elements of the matrices "$WI(t)$" and "$W1(t)$", as well as the robustness characteristics with respect to external input perturbations of this algorithm for two layer neural networks, are similar as in the adaline case.

3.3.1.5 THE THREE LAYER NEURAL NETWORK CASE

Consider the three layer neural network illustrated in figure 3.4. Here, it is assumed that the boundedness conditions represented by equations 3.66 and 3.67 of the previous neural network case are valid. The following equations can be easily verified

$$Y_o(t) = (W1(t))^T Z_2(t); \qquad W1(t) \in \Re^{n2\times p} \tag{3.82}$$

$$Z_2(t) = \Gamma(Y_2(t)); \qquad Z_2(t) \in \Re^{n2} \tag{3.83}$$

$$Y_2(t) = (W2(t))^T Z_1(t); \qquad W2(t) \in \Re^{n1\times n2} \tag{3.84}$$

$$Z_1(t) = \Gamma(Y_1(t)); \qquad Z_1(t) \in \Re^{n1} \tag{3.85}$$

$$Y_1(t) = (WI(t))^T X_a(t); \qquad WI(t) \in \Re^{(n+1)\times n1} \tag{3.86}$$

Notice that $X_a(t) \in \Re^{(n+1)}$ represents the augmented input vector. From the output error equation $E(t) = Y_d(t) - Y_o(t)$ it follows that

$$\begin{aligned}\dot{E}(t) &= \dot{Y}_d(t) - (\dot{W}1(t))^T Z_2(t) \\ &- (W1(t))^T \left(\frac{\partial\Gamma(Y_2(t))}{\partial Y_2(t)}\right)(W2(t))^T \left(\frac{\partial\Gamma(Y_1(t))}{\partial Y_1(t)}\right)(\dot{W}I(t))^T \dot{X}_a(t)\end{aligned}$$

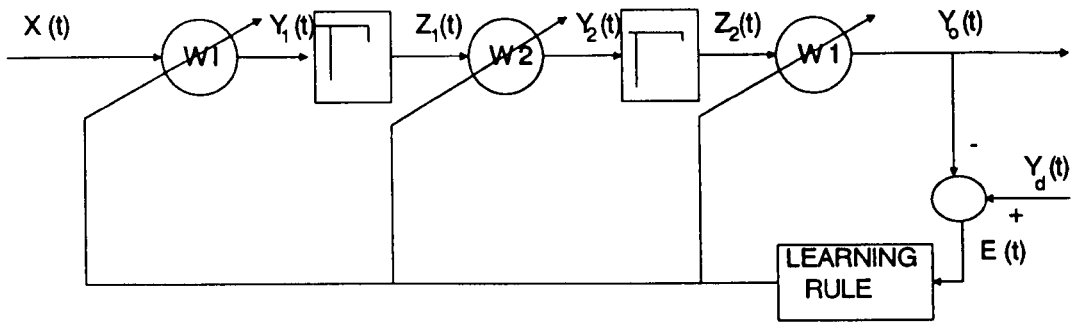


Figure 3.4: Three layer feedforward neural network.

$$- \quad (W1(t))^T \left(\frac{\partial \Gamma(Y_2(t))}{\partial Y_2(t)}\right) (\dot{W}2(t))^T Z_1(t) \tag{3.87}$$

$$- \quad (W1(t))^T \left(\frac{\partial \Gamma(Y_2(t))}{\partial Y_2(t)}\right) (W2(t))^T \left(\frac{\partial \Gamma(Y_1(t))}{\partial Y_1(t)}\right) (\dot{W}I(t))^T X_a(t)$$

The weight matrices dynamics are modelled by the following matrix differential equations

$$\dot{W}I(t) = UI(t), \tag{3.88}$$

$$\dot{W}2(t) = U2(t), \tag{3.89}$$

$$\dot{W}1(t) = U1(t), \tag{3.90}$$

with "$UI(t)$", "$U2(t)$", and "$U1(t)$" control matrices of appropriate dimensions. The boundedness conditions on the columns of "$WI(t)$", "$W2(t)$", and "$W1(t)$" are given by $\|WI_i(t)\| < B_{WI_i}$, $\|W2_i(t)\| < B_{W2_i}$, and $\|W1_i(t)\| < B_{W1_i}$, respectively.

THEOREM 4.

If the control input matrices "$UI(t)$", "$U2(t)$", and "$U1(t)$" for adapting the weights in the dynamic equations described by 3.88, 3.89, and 3.90 are respectively chosen as

$$UI(t) = \frac{X_a(t)}{(X_a(t))^T(X_a(t))} \left[(Z_1(t))^T \left(\frac{\partial \Gamma(Y_1(t))}{\partial Y_1(t)}\right)^{-1}\right] \tag{3.91}$$

$$U2(t) = -W2(t) + \frac{Z_1(t)}{(Z_1(t))^T(Z_1(t))} \left[(Z_2(t))^T \left(\frac{\partial \Gamma(Y_2(t))}{\partial Y_2(t)}\right)^{-1}\right] \tag{3.92}$$

$$U1(t) = -W1(t) + \frac{Z_2(t)}{(Z_2(t))^T(Z_2(t))}\,[K\,SIGN(E(t))] \tag{3.93}$$

where "K" is a diagonal matrix whose diagonal elements satisfy the condition

$$k_i \leq \eta_i + B_{\dot{y}} + B_{W1_i}\,B_{W2_i}\,B_{WI_i}\,B_{\dot{x}} \tag{3.94}$$

then given any initial condition "$E(0)$", the learning error "$E(t)$" converges to zero in finite time "t_{r_i}" estimated by

$$t_{r_i} \leq \frac{|e_i(0)|}{\eta_i}, \tag{3.95}$$

with "$e_i(t)$" the i-th component of the vector "$E(t)$"; and a sliding regime is sustained on the vector $E(t) = 0$ for all $t > t_{r_i}$.

PROOF.

The proof is similar to the one given for the previous theorem.

It should be noticed that the matrices "$\left(\frac{\partial\Gamma(Y_1(t))}{\partial Y_1(t)}\right)$" and "$\left(\frac{\partial\Gamma(Y_2(t))}{\partial Y_2(t)}\right)$" are diagonal matrices. Also, similar to the previous cases, the control laws defined by equations 3.91, 3.92, and 3.93 can be relaxed by assuming that the derivative of the input vector "$X_a(t)$" is available for measurements. In this case the control laws become

$$UI(t) = -\left[\frac{(X_a(t))(\dot{X}_a(t))^T}{(X_a(t))^T(X_a(T))}\right] WI(t) \tag{3.96}$$

$$U2(t) = \frac{Z_1(t)}{(Z_1(t))^T(Z_1(t))}\left[(Z_2(t))^T\left(\frac{\partial\Gamma(Y_2(t))}{\partial Y_2(t)}\right)^T\right] \tag{3.97}$$

$$U1(t) = -W1(t) + \frac{Z_2(t)}{(Z_2(t))^T(Z_2(t))}\,(K\,SIGN(E(t))) \tag{3.98}$$

and if the elements of the diagonal matrix "K" are selected according to

$$k_i > \eta_i + B_{\dot{y}} \tag{3.99}$$

then, regardless of the initial condition "$E(0)$" the learning error trajectories converge to zero in finite time, and a sliding regime is induced on the manifolds $E(t) = 0$.

Again, the boundedness of the elements of the weight matrices, and the robustness of the algorithm with respect to external bounded perturbations can be proved using analogous arguments to the ones given for the adaline case.

3.3.2 DISCRETE-TIME VSC LEARNING RULES

A different situation from the continuous-time case occurs in the discrete-time variable structure control-based learning algorithm where a quasi-sliding regime may be

induced in the dynamic behaviour of the weights, by selecting a control law that forces the output error between the desired and the actual outputs to satisfy an asymptotically stable difference equation. In this case the differentiability condition imposed upon the nonlinear activation function is not required any more.

3.3.2.1 THE ADALINE CASE

Consider again the adaptive linear neuron depicted in figure 3.2. In this case, it is assumed that the function "$\Gamma(.)$" is any nonlinear function that satisfies the property

$$\Gamma(-V(t)) = -\Gamma(V(t)) \tag{3.100}$$

Examples of the function "$\Gamma(.)$" are the following

$$\Gamma(v_i(t)) = sign(v_i(t)) = \begin{cases} 1 & \text{if } v_i(t) > 0 \\ -1 & \text{if } v_i(t) < 0 \end{cases} \tag{3.101}$$

$$\Gamma(v_i(t)) = sat(v_i(t)) = \begin{cases} 1 & \text{if } v_i(t) > 1 \\ v_i(t) & \text{if } v_i(t) \in [-1,1] \\ -1 & \text{if } v_i(t) < 1 \end{cases} \tag{3.102}$$

$$\Gamma(v_i(t)) = sig(v_i(t)) = \frac{1}{1 + exp^{-v_i(t)}} \tag{3.103}$$

It is also assumed that the vector "X", the desired output "y_d", and the actual output "y_o" take values at discrete time intervals "kT"; $k = 0, 1, 2, ...$, with "T" a fixed time quantity. For the sake of simplicity, the parameter "T" will be omitted in the notation. The augmented vector $X_a(k) = (X(k), b)^T$ represents the input vector to the adaline, with "b" being a constant bias connected to the adaline through the variable weight "$w_{n+1}(k)$".

The following equations may be easily verified

$$y_o(k) = X^T(k)W(k) + bw_{n+1}(k) \tag{3.104}$$

$$= X_a^T(k)W_a(k) \tag{3.105}$$

$$z_o(k) = \Gamma(y_o(k)) \tag{3.106}$$

The augmented weight vector "$W_a(k)$" includes the weight of the bias input. This is $W_a(k) = (W(k), w_{n+1}(k))^T$. The weights of the adaptive neuron may be modeled by a dynamic system of the form

$$W_a(k) = F(W_a(k-1), U(k-1)) \tag{3.107}$$

$$e(k) = h(W_a(k)) \tag{3.108}$$

where "$W_a(k)$" represents the state vector, "$U(k)$" is a control input that represents the updates for the values of the weight vector, and "$e(k)$" may be viewed as the output of the dynamic system. In particular, the system may be modeled by the equations

$$W_a(k) = W_a(k-1) + U(k-1) \tag{3.109}$$

$$e(k) = y_d(k) - y_o(k) \tag{3.110}$$

Consider now the following level curve of the output map

$$s(W_a(k), k) = \left\{ W_a(k) \in \Re^{n+1} : e(k) = h(W_a(k)) = 0 \right\} \tag{3.111}$$

THEOREM 5.

If the weight updates "$U(k-1)$" of the dynamic system represented by equations 3.109 and 3.110 are selected as

$$\begin{aligned} U(k-1) = & \frac{\Gamma(X_a(k))}{(X_a(k))^T(\Gamma(X_a(k)))} [\alpha e(k-1) + y_d(k) - y_d(k-1) \\ & - (X_a(k) - X_a(k-1))^T W_a(k-1)] \end{aligned} \tag{3.112}$$

with $0 < \alpha < 2$, then for an arbitrary initial condition "$e(0)$", the learning error equation "$e(k)$" tends to zero asymptotically, and a quasi-sliding mode is induced on the manifold $e(k) = 0$.

PROOF.

Note that $e(k-1) = y_d(k-1) - y_o(k-1)$ and by substituting equation 3.109 into the difference equation $e(k) - e(k-1)$, yields

$$e(k) - e(k-1) = y_d(k) - y_d(k-1) - (X_a(k) - X_a(k-1))^T W_a(k-1) - X_a^T(k) U(k-1) \tag{3.113}$$

Substituting equation 3.112 into equation 3.113 yields $e(k) - e(k-1) = \alpha e(k-1)$ and therefore

$$e(k) = (1 - \alpha) e(k-1). \tag{3.114}$$

Clearly, for $0 < \alpha < 2$, $\lim_{k \to \infty} e(k-1) = 0$, which means that the learning error equation "$e(k)$" tends to zero asymptotically, regardless of any intial condition.

On the other hand, notice that the quasi-sliding condition 3.19, (ie. $|e(k+1)e(k)| < e^2(k)$) is satisfied. This is

$$|e(k+1)e(k)| = |(1-\alpha)e^2(k)| < |e(k)|^2 = e^2(k) \tag{3.115}$$

and therefore, a quasi-sliding regime exists on $e(k) = 0$.

3.3.2.2 BOUNDEDNESS OF SOLUTIONS FOR THE WEIGHTS

Similar to the continuous-time case, here it is assumed that the input vector "$X_a(k)$" and the desired output "$y_d(k)$" are bounded signals by means of

$$\|X_a(k)\| < B_x \qquad \|y_d(t)\| < B_y \quad \forall k \tag{3.116}$$

Substituting equations 3.110 and 3.112 into equation 3.109 yields

$$\begin{aligned} W_a(k) &= \left\{ I - \frac{\Gamma(X_a(k))}{(X_a(k))^T(\Gamma(X_a(k)))} [X_a(k) + (\alpha - 1)X_a(k-1)]^T \right\} W_a(k-1) \\ &+ \frac{\Gamma(X_a(k))}{(X_a(k))^T(\Gamma(X_a(k)))} (y_d(k) + (\alpha - 1)y_d(k-1)) \end{aligned} \tag{3.117}$$

In general, equation 3.117 represents a linear time-varying discrete-time vector equation with forcing function given by the bounded signal $y_d(k) + (\alpha - 1)y_d(k-1)$. Observe that if the input vector "$X_a(k)$" and the desired output "$y_d(k)$" are kept constant during the adaptation cycle "k", (ie. $X_a(k) = X_a(k-1) = X_a$ and $y_d(k) = y_d(k-1) = y_d$) then equations 3.112 and 3.117 can be rewritten as

$$U(k-1) = \frac{\Gamma(X_a)}{(X_a)^T(\Gamma(X_a)} (\alpha\, e(k-1)) \tag{3.118}$$

$$\begin{aligned} W_a(k) &= \left[I - \alpha \frac{(\Gamma(X_a)(X_a)^T}{(X_a)^T(\Gamma(X_a))} \right] W_a(k-1) \\ &+ \alpha \left[\frac{\Gamma(X_a)}{(X_a)^T(\Gamma(X_a))} \right] y_d \end{aligned} \tag{3.119}$$

Note that if the function "$\Gamma(.)$" is the identity operator, then equation 3.119 becomes the Widrow-Hoff delta rule [20].

Let the matrix "A" and the vector "b" be defined as follows

$$A = \left(I - \alpha \frac{(\Gamma(X_a))(X_a)^T}{(X_a)^T(\Gamma(X_a))} \right) \tag{3.120}$$

$$b = \alpha \frac{\Gamma(X_a)}{(X_a)^T(\Gamma(X_a))} \tag{3.121}$$

By virtue of the boundedness of vector "X_a", and the fact that "$\Gamma(x_i)$" is also bounded by means of $|\Gamma(x_i)| \leq 1$, the elements of the matrix "A" and those of the vector "b" are also bounded. For example, a typical element in the main diagonal of the matrix "A" can be written as

$$A_{ii} = 1 - \alpha \frac{x_i \Gamma(x_i)}{\sum_{i=1}^{n+1} x_i \Gamma(x_i)} \tag{3.122}$$

Observe that for $0 < \alpha < 2$, $|A_{ii}| \leq 1$. A similar argument can be used to prove the boundedness of elements off the main diagonal of "A", and for the elements of the vector "b".

It is known that the solution of the linear time-invariant vector difference equation $W_a(k) = A\,W_a(k-1) + b\,y_d$ can be expressed in terms of the initial condition "$W_a(0)$" as

$$W_a(k-1) = A^{k-1}W_a(0) + \sum_{j=1}^{k-1} A^{k-1-j}\,b\,y_d \tag{3.123}$$

hence

$$\|W_a(k-1)\| < \|A\|^{k-1}\|W_a(0)\| + B_y \sum_{j=1}^{k-1} \|A\|^{k-1-j}\|b\| \tag{3.124}$$

Since all quantities in the right hand side of equation 3.124 are bounded then "$W_a(k-1)$" is also bounded.

3.3.2.3 ROBUSTNESS FEATURES WITH RESPECT TO EXTERNAL PERTURBATIONS

It has been established that the output error "$e(k)$" is obtained by substracting the desired output "$y_d(k)$" from the actual neuron output "$y_o(k)$". In general, the measurements of the output error may be corrupted by noise. This noise influences the accuracy of the function approximation performed by the neuron, and in some cases, it may have an adverse effect on the boundedness of the weight vector. In this analysis, it is considered that a measurement noise signal "$\xi(k)$", which is bounded by $|\xi(k)| < B_\xi$, affects the output error additively. This is

$$\hat{e}(k) = e(k) + \xi(k) \tag{3.125}$$

with "$e(k)$" the error signal defined in equation 3.110. Notice that under this circumstance, equation 3.117 becomes

$$\begin{aligned} W_a(k) &= \left\{ I - \frac{\Gamma(X_a(k))}{(X_a(k))^T(\Gamma(X_a(k)))} [X_a(k) + (\alpha-1)X_a(k-1)]^T \right\} W_a(k-1) \\ &+ \frac{\Gamma(X_a(k))}{(X_a(k))^T(\Gamma(X_a(k)))} [y_d(k) + (\alpha-1)y_d(k-1) + \alpha\xi(k-1)] \end{aligned} \tag{3.126}$$

It can be readily proved that, under the boundedness restrictions on the input "$X_a(k)$" and the noise signal "$\xi(k-1)$" the weight vector "$W_a(k)$" is also kept bounded. However, the function approximation performed by the neuron deviates from the desired output "$y_d(k)$", at least by the amount "$|\xi(k-1)|$". Indeed, notice

that substituting "$e(k)$" by "$\hat{e}(k)$" in equation 3.112, and replacing this result into the following equation

$$\begin{aligned} \hat{e}(k) - \hat{e}(k-1) &= y_d(k) - y_d(k-1) + \xi(k) - \xi(k-1) \\ &- (X_a(k) - X_a(k-1))^T W_a(k-1) - X_a^T(k) U(k-1) \end{aligned} \tag{3.127}$$

yields

$$e(k) = (1-\alpha)\, e(k-1) - \alpha \xi(k-1) \tag{3.128}$$

The solution of equation 3.128 can be written as

$$e(k-1) = (1-\alpha)^{k-1} e(0) - \sum_{j=1}^{k-1} (1-\alpha)^{k-1-j}\, \alpha\, \xi(k-1) \tag{3.129}$$

and for $0 < \alpha < 2$, $\lim_{k\to\infty} |e(k-1)| \leq |\xi(k-1)|$.

It is worth mentioning the result obtained by Hui and Zak [58], regarding the selection of the reduction factor "α" in single perceptrons subjected to additive measurement noise in the output error. In their work, in order to guarantee convergence of the weight vector in the Widrow-Hoff delta rule, the reduction factor "α" must be selected as any sequence $\{\alpha(k)\}_{k=0}^{\infty}$, $0 \leq \alpha < 1$, satisfying $\prod_{k=0}^{\infty}(1-\alpha(k)) = \varepsilon$, for $\varepsilon \in [0,1)$.

3.3.3 THE TWO LAYER NEURAL NETWORK CASE

Consider the two layer neural network depicted in figure 3.3, and assume that both the input vector and the desired output vector take values at discrete time intervals "kT". Similar to the previous case, the augmented input vector "$X_a(k)$" includes a constant bias component connected to every neuron of the input layer through variable weights. The following equations can be readily verified.

$$Y_o(k) = (W1(k))^T Z_1(k); \qquad W1(k) \in \Re^{n1 \times p} \tag{3.130}$$

$$Z_1(k) = \Gamma(Y_1(k); \qquad Z_1(k) \in \Re^{n1} \tag{3.131}$$

$$Y_1(k) = (WI(k))^T X_a(k); \qquad WI(k) \in \Re^{(n+1)\times n1} \tag{3.132}$$

The output error is a vector equation defined by

$$E(k) = Y_d(k) - Y_o(k) \tag{3.133}$$

Let the weight matrices "$WI(k)$" and "$W1(k)$" be defined in terms of the following dynamic systems

$$WI(k) = WI(k-1) - UI(k-1) \tag{3.134}$$

$$W1(k) = W1(k-1) - U1(k-1) \tag{3.135}$$

with "$UI(k-1)$" and "$U1(k-1)$" control inputs that represent the updates for the values of "$WI(k-1)$" and "$W1(k-1)$", respectively.

The difference between "$E(k)$" and "$E(k-1)$" may be expressed as

$$\begin{aligned} E(k)-E(k-1) &= Y_d(k)-Y_o(k)-Y_d(k-1)+Y_o(k-1) \\ &= Y_d(k)-Y_d(k-1)+(W1(k-1))^T(\Gamma(Y_1(k-1)) \\ &- (W1(k-1)+U1(k-1))^T(\Gamma[(WI(k-1)+UI(k-1))^T X_a(k)] \\ &= Y_d(k)-Y_d(k-1)+(W1(k-1))^T\{\Gamma(Y_1(k-1)) \\ &- \Gamma[(WI(k-1)+UI(k-1))^T X_a(k)]\} \\ &- (U1(k))^T\Gamma[(WI(k-1)+UI(k-1))^T X_a(k)] \end{aligned} \tag{3.136}$$

THEOREM 6.

If the update matrices "$UI(k-1)$" and "$U1(k-1)$" of the dynamical systems 3.134 and 3.135 are chosen as

$$\begin{aligned} UI(k-1) &= -\frac{X_a(k)}{(X_a(k))^T(X_a(k))}[(WI(k-1))^T X_a(k) \\ &+ Y_1(k-1)]^T \end{aligned} \tag{3.137}$$

$$\begin{aligned} U1(k-1) &= -2\,W1(k-1)-\frac{\Gamma(Y_1(k-1))}{(\Gamma(Y_1(k-1)))^T(\Gamma(Y_1(k-1)))}[A\,E(k-1) \\ &+ Y_d(k)-Y_d(k-1)]^T \end{aligned} \tag{3.138}$$

then, for any initial condition "$E(0)$", the learning error equation satisfies the following asymptotically stable difference equation

$$E(k)=(I-A)E(k-1) \tag{3.139}$$

where "A" is an "$p \times p$" matrix with eigenvalues inside the unit circle in the z-plane. Furthermore, a quasi-sliding regime is sustained on the manifolds "$E(k)=0$".

PROOF.

Substituting the transpose of the matrices "$UI(k-1)$" and "$U1(k-1)$" into equation 3.136 and knowing that equation 3.100 holds true, yields the error equation 3.139. If the eigenvalues of the matrix "A" lay inside the unit circle of the z-plane, then equation 3.139 is asymptotically stable. (ie. Choose the matrix "A" as a diagonal matrix with elements "α_i" such that $|1-\alpha_i|<1, i=1,2,..p$. On the other hand, observe that the i-th component of the vector "$E(k)$" is "$e_i(k)$" and, for simplicity, assuming that the matrix "A" is diagonal,

$$|e_i(k+1)e_i(k)| = |(1-\alpha))e_i^2(k)| < e_i(k)|^2 = e_i^2(k), \tag{3.140}$$

and therefore, the quasi-sliding regime condition is satisfied.

3.3.4 THE THREE LAYER NEURAL NETWORK CASE

Consider the three layer neural network shown in figure 3.4, and assume that both the input vector "X_a" and the desired output "Y_d" take values at discrete time intervals "kT", with k=0,1,2,..., and "T" a fixed quantity. As before, notice that $X_a(k) = (X(k), b)^T$ contains a constant bias element connected to every neuron in the input layer. The following equations hold valid,

$$\begin{aligned} Y_o(k) &= (W1(k))^T Z_2(k); \qquad W1(k) \in \Re^{n2 \times p} && (3.141)\\ Z_2(k) &= \Gamma(Y_2(k)); \qquad Z_2(k) \in \Re^{n2} && (3.142)\\ Y_2(k) &= (W2(k))^T Z_1(k); \qquad W2(k) \in \Re^{n1 \times n2} && (3.143)\\ Z_1(k) &= \Gamma(Y_1(k)); \qquad Z_1(k) \in \Re^{n1} && (3.144)\\ Y_1(k) &= (WI(k))^T X_a(k); \qquad WI(k) \in \Re^{(n+1) \times n1} && (3.145) \end{aligned}$$

The output error is the vector equation defined in 3.133. Let the weight matrices "$WI(K)$", "$W2(k)$", and "$W1(k)$" be defined in terms of the following dynamic equations,

$$\begin{aligned} WI(k) &= WI(k-1) + UI(k-1), && (3.146)\\ W2(k) &= W2(k-1) + U2(k-1), && (3.147)\\ W1(k) &= W1(k-1) + U1(k-1), && (3.148) \end{aligned}$$

with "$UI(k-1)$", "$U2(k-1)$", and "$U1(k-1)$" representing update matrices for the values of "$WI(k-1)$", "$W2(k-1)$", and "$W1(k-1)$", respectively. The difference between "$E(k)$" and "$E(k-1)$" may be written as,

$$\begin{aligned} E(k) - E(k-1) &= Y_d(k) - Y_d(k-1) + (W1(k-1))^T \{\Gamma(Y_2(k-1))\\ &- \Gamma\left\{[W2(k-1) + U2(k-1)]^T\right.\\ &* \left.\Gamma\left\{[WI(k-1) + UI(k-1)]^T X_a(k)\right\}\right\}\\ &- (U1(k-1))^T \Gamma\left\{[W2(k-1) + U2(k-1)]^T\right.\\ &* \left.\Gamma\left\{[WI(k-1) + UI(k-1)]^T X_a(k)\right\}\right\} \qquad (3.149) \end{aligned}$$

THEOREM 7.

If the weight update matrices "$UI(k-1)$", "$U2(k-1)$", and "$U1(k-1)$" of the dynamic systems represented by equations 3.146, 3.147, and 3.148 are respectively chosen as

$$UI(k-1) = -\frac{X_a(k)}{(X_a(k))^T (X_a(k))}[(WI(k-1))^T X_a(k)$$

$$+ \quad Y_1(k-1)]^T \tag{3.150}$$

$$U2(k-1) = -\frac{2(Z_1(k-1))}{(Z_1(k-1))^T(Z_1(k-1))}[Y_2(k-1)]^T \tag{3.151}$$

$$U1(k-1) = \frac{Z_2(k-1)}{(Z_2(k-1))^T(Z_2(k-1))}[AE(k-1) + Y_d(k) - Y_d(k-1)]^T \tag{3.152}$$

then, for any initial condition "$E(0)$", the learning error equation satisfies the following asymptotically stable difference equation

$$E(k) = (I - A)E(k-1), \tag{3.153}$$

where "A" is an "$p \times p$" matrix with eigenvalues inside the unit circle in the z-plane. Furthermore, a quasi-sliding regime is sustained on the manifold $E(k) = 0$.

PROOF.

Substituting the transposes of the matrices "$UI(k-1)$", "$U2(k-1)$", and "$U1(k-1)$" into equation 3.149, and knowing that equation 3.100 holds valid, yields equation 3.153, which is asymptotically stable if the matrix "A" is chosen with eigenvalues inside the unit circle of the z-plane. The proof of the existence of a quasi-sliding regime on $E(k) = 0$ is identical to the proof given in the second part of the previous theorem.

It is important to notice that if the nonlinear function "$\Gamma(.)$" is chosen as the "$Sign(.)$" function (see equation 3.101), then the terms "$(Z_1(k-1))^T(Z_1(k-1))$", and "$(Z_2(k-1))^T(Z_2(k-1))$" in equations 3.151 and 3.152, represent the number of neurons in the hidden layer, and in the output layer, respectively. This is, $(Z_1(k-1))^T(Z_1(k-1)) = n1$ and $(Z_2(k-1))^T(Z_2(K_1)) = n2$. Also, observe that if the input vector "$X_a(k)$" and the desired output vector "$Y_d(k)$" are kept constant during the adaptation cycle "k", the algorithm represented by equations 3.150, 3.151, and 3.152 is similar to the algorithm presented by Sira-Ramirez and Zak [59].

Finally, it is pertinent to clarify that the task of the bias term "b", included as an element into the augmented input vector "$X_a(k)$" is to avoid division by zero in the equations of the proposed algorithms. In fact, these bias terms may be eliminated in the formulation of the algorithms by a suitable selection of the component of the input vector "$X(k)$". For example, in the discrete-time case, where the structure of the dynamic equations for the weights is given by $W(k) = W(k-1) + U(k-1)$ the bias terms can be avoided by selecting [60]

$$U(k-1) = \begin{cases} G(X(k), Y_d(k), E(k-1)) & \text{if} \quad X^T(k)X(k) \neq 0 \\ 0 & \text{if} \quad X^T(k)X(k) = 0 \end{cases} \tag{3.154}$$

where "$G(X(k), Y_d(k), E(k-1))$" represents an update formulae (ie. equation 3.138, or 3.150, etc).

3.4 IMPLEMENTATION OF THE LEARNING ALGORITHMS

Appendices A, B, and C contain computer programs for the implementation of the continuous-time and discrete-time versions of the proposed algorithms for an adaptive linear element, a two layer neural network, and a three layer neural network, respectively. These programs were written in the "*Simnon*" computer simulation language and will be used in successive chapters to implement neural network identification and control-based schemes.

3.5 SUMMARY

This chapter has presented a sliding mode control approach for adaptive learning algorithms in continuous-time and discrete-time feedforward neural networks.

In the continuous-time case, the learning algorithms that have been proposed represent a robust mechanism for achieving finite time reachability of a zero level learning error manifold, and therefore provide a way of implementing on-line approximations of unknown functions. It has been proved that these algorithms are insensitive to bounded external perturbations, and that the evolution of the weight elements of the neural network is guaranteed to be bounded.

In the discrete-time case on the other hand, it has been presented learning algorithms for adapting the neural networks' weights in such a way that a zero level set of a learning error variable is reached asymptotically. This asymptotic reachability is analogous to inducing the dynamics of the weights to behave in a quasi-sliding regime. Also, it has been shown that the continous-time adaptation capability of the weights produces bounded values.

Starting with an adaptive linear neuron, the continuous-time and the discrete-time versions of the sliding mode control-based learning algorithms have been extended to cover two layer and three layer feedforward neural networks. Computer programs for the digital simulation of the proposed algorithms have also been constructed and are included in appendices A and B.

Chapter 4

DYNAMIC SYSTEM IDENTIFICATION USING FEEDFORWARD NEURAL NETWORKS

4.1 INTRODUCTION

The exploitation of the potential capabilities of feedforward neural networks to approximate linear and nonlinear mappings by using a learning algorithm, has given rise to a number of schemes for dynamic system identification [7, 15, 42, 59, 61, 62]. In general terms, the problem of estimating a dynamic system involves the following steps:

- Determination of the system characterization.

  Here, the emphasis is placed on the selection of the appropriate measurable variables of the system, and of a suitable structure to model the desired system dynamics adequately. In the context of neural networks, this means choosing the types of nodes and structure of the network that produce outputs which are able to code the desired system's variable from a selected set of signal inputs to the network.

- Assessment of the ability of the proposed model to reproduce the behaviour of the system under scrutiny.

  In applications involving feedforward neural networks, it is common to measure performance by taking the average of the squared error between the outputs

of the network and the teaching signals.

- Implementation of the identification method.

  When all the data are available at one time, the identification method is performed off-line. In contrast to this, for on-line identification methods the data is processed as it becomes available from the unknown system.

Although a number of published results have shown that a multilayer feedforward neural network can approximate arbitrarily well a continuous function [24, 63], the problem of choosing the number of nodes and layers to achieve a specific degree of accuracy for the function being approximated remains an open problem. In terms of the dimension of the network inputs, the paper by Chen et al [42], showed the importance of selecting the appropriate number of input nodes in a feedforward neural network used to represent a dynamical system. In their work, it was shown that, by taking a series expansion of a sigmoidal type of activation function of the hidden nodes, the network does not generate components of higher order lagged system inputs and outputs which were not specified in the network input nodes. This work also provides an interpretation from the estimation theory framework of the modeling capabilities of multilayer neural networks, and gives a procedure to aid in verifying the validity of a neural network model by detecting its inadequacy to fit the true system. Despite all these efforts, there are still no concrete theoretical results relating to an adequate representation of a dynamic system within the structure of a neural network. Therefore, most attempts in identifying dynamic systems using neural networks assume that the chosen neural network is able to model the system under study.

Mathematically, the problem of identifying a dynamic system may be formulated in the following terms:

Let the causal dynamic system be described by the equation

$$y(t) = F(u) \tag{4.1}$$

where $u \in \mathcal{U}$, and $y(t) \in \mathcal{Y}$ represent the input and output of the system, respectively. "F" represents an operator that maps elements of the input space "$\mathcal{U}$" into the output space "$\mathcal{Y}$". Let the identification model be described by the equation

$$\hat{y}(t) = \hat{F}(u) \tag{4.2}$$

The objective is to determine "$\hat{F}$" in such a way that

$$\|y(t) - \hat{y}(t)\| = \|F(u) - \hat{F}(u)\| \leq \varepsilon \quad for \ u \in \mathcal{U} \tag{4.3}$$

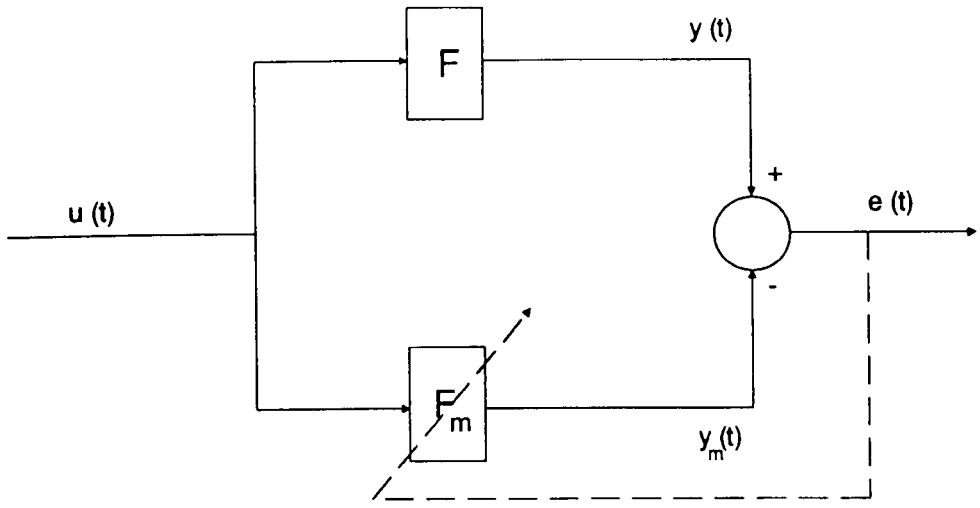


Figure 4.1: Identification of dynamic systems

where $\varepsilon > 0$ and $\|.\|$ denotes a suitably defined norm. Figure 4.1 illustrates the identification problem. Note that the error signal $e(t) = y(t) - \hat{y}(t)$, in figure 4.1, provides information regarding the accuracy of the model in copying the behaviour of the dynamic system. The work by Narendra et al [15] suggests four identification structures or models that contain multilayer neural networks as subsystems, for the estimation of unknown nonlinear dynamical systems. In this work, the plants to be identified belong to the class of bounded-input bounded-output stable systems, and the adjustment of the parameters of the identification structure, including the weights of the neural network subsystems, can be achieved using static and dynamic back-propagation methods. In order to guarantee convergence of the parameters of the model, or that the output error tends to zero, Narendra and co-worker propose a series-parallel model where the output of the unknown plant, rather than the output of the identification model, is fed back into the identification model.

The path followed in this work differs from other forward modelling design approaches in terms of the formulation of the neural network models used to emulate the dynamic behaviour of the unknown plant. That is, our main objective is not to estimate a particular set of parameters or nonlinear functions of the unknown plant, but rather, to identify the forward transfer operator (FTO), [64], that enables a transformation of the values of the input signal $u \in \mathcal{U}$ into the output signal $y \in \mathcal{Y}$. Moreover, this task is performed on-line, using the input-output data of the plant as it becomes available, and the accuracy of the neural network model in approximating the unknown FTO is assessed by forcing the output error between

the unknown plant output and the neural network output to tend to zero in the limit.

The inverse transfer operator (ITO) identification problem is also studied as the problem of finding the corresponding ITO that enables a transformation of the values of the output signal $y \in \mathcal{Y}$ into the input signal $u \in \mathcal{U}$.

These two problems are addressed in the following sections using single layer, two layer, and three layer neural networks with continuous-time and discrete-time variable structure control-based learning algorithms.

4.2 FTO MODELLING

It has been emphasized that an artificial feedforward neural network can be thought of as an adaptive nonlinear system able to perform nonlinear mapping approximations by means of a learning algorithm. When the input to the unknown plant is also the input to the neural network, and the learning objective is satisfied by making the output of the neural network emulate the behaviour of the output of the unknown plant, the internal weights adjustment experienced by the neural network contains information regarding the FTO model approximation of the real plant. This can be illustrated by considering a neural network with input signal $x(t) = u(t)$, output signal "$y_o(t)$", and a weight element "$w(t)$", designed to perform the following function approximation:

$$\lim_{t\to\infty} y_o(t) = y(t), \tag{4.4}$$

or equivalently

$$\lim_{t\to\infty} \hat{F}(x(t), w(t)) = F(u(t)) \tag{4.5}$$

where $y(t) = F(u(t))$ represents the unknown dynamic system, and $y_o(t) = \hat{F}(x(t), w(t))$ represents the neural network structure used to model the FTO of the unknown plant. The following sections contain the mathematical formulation and computer simulations results to illustrate the FTO modelling problem for continuous-time and discrete-time, causal, linear and nonlinear dynamic systems.

4.2.1 CONTINUOUS-TIME LINEAR FTO MODELLING

Consider a causal, minimum phase, undisturbed, unknown, linear system described by the following differential equation:

$$y^{(n)}(t)+a_1y^{(n-1)}(t)+...+a_{n-1}\dot{y}(t)+a_ny = b_0u^{(m)}(t)+b_1u^{(m-1)}(t)+...+b_{m-1}\dot{u}(t)+b_mu(t) \tag{4.6}$$

where $m \leq n$, and the initial conditions are given by $y^{(i)}(t_0) = y_0^{(i)}$, $i = 0, 1, ...n$. It can be easily shown that if $D = \frac{d}{dt}$ is the differential operator, then equation 4.6 can be rewritten as:

$$(D^n + a_1 D^{n-1} + ... + a_{n-1} D + a_n) y(t) = (b_0 D^m + b_1 D^{m-1} + ... + b_{m-1} D + b_m) u(t) \quad (4.7)$$

For the sake of representing equation 4.7 in terms of equation 4.1, the following symbolic operation is performed

$$y(t) = \frac{b_0 D^m + b_1 D^{m-1} + ... + b_{m-1} D + b_m}{D^n + a_1 D^{n-1} + ... + a_{n-1} D + a_n} u(t) \quad (4.8)$$

Equation 4.8 is simply a representation of the differential equation 4.6. Observe that the ratio "$\frac{y(t)}{u(t)}$" constitutes a dynamic sensitivity or gain that represents the FTO of the class of system under study. The estimation problem at hand is to propose a neural network structure to make an on-line approximation of equation 4.8 using a continuous-time variable structure control-based learning algorithm. This task is accomplished by considering that the input signal, "$u(t)$", to the linear system is used to construct the input signal to the neural network, and the output signal "$y(t)$" of the linear system is the desired signal "$y_d(t)$" for the neural network. By virtue of the linear characteristic of the system under scrutiny, it suffices to select a single layer neural network with dynamic weight adjustments to perform this function approximation. Figure 4.2 depicts the FTO identification scheme.

PROPOSITION.

The FTO of the causal, linear system described by equation 4.6 can be estimated on-line, in finite time, using an adaptive linear neuron trained with the variable structure control-based learning algorithm represented by equation 3.32.

PROOF.

Let the linear system input-output pair be $(u(t), y(t))$ and let the adaptive neuron input and desired output be $x(t) = u(t)$, and $y_d(t) = y(t)$, respectively. Let the adaptive neuron output be defined by the equation

$$y_o(t) = X_a^T(t) W_a(t), \quad (4.9)$$

with $X_a(t) = (x(t), b)^T$ and "b" a constant bias value. Finally, let the output error "$e(t)$" be defined as

$$e(t) = y_d(t) - y_o(t). \quad (4.10)$$

Observe that if "$W_a(t)$" in equation 4.9 is selected according to equation 3.32, in theorem 1 of the previous chapter, then, in a finite time $t_r \leq \frac{(|y_d(0) - y(0)|)}{\eta}$, with "$\eta$"

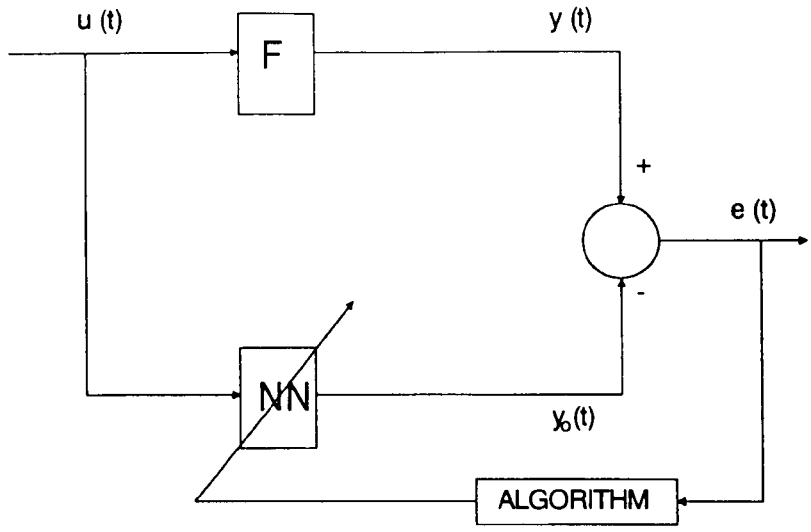


Figure 4.2: Forward transfer operator identification scheme

a positive value as defined in theorem 1, the error equation 4.10 converges to zero. That is

$$\begin{aligned} 0 &= y_d(t \geq t_r) - y_o(t \geq t_r) \\ &= y_d(t \geq t_r) - u(t \geq t_r)w_1(t \geq t_r) - bw_2(t \geq t_r) \end{aligned} \tag{4.11}$$

and

$$y_d(t \geq t_r) = y(t \geq t_r) = u(t \geq t_r)w_1(t \geq t_r) - bw_2(t \geq t_r). \tag{4.12}$$

In order to illustrate this result, consider the problem of estimating the FTO in an unknown linear system described by the equation:

$$m\,\ddot{y}(t) = -c\,\dot{y}(t) - \xi\,y(t) + u(t) \tag{4.13}$$

Equation 4.13 represents a mathematical model for the spring-mass-damper system sketched in figure 4.3; where "m" is the mass, "ξ" represents the spring stiffness, "c" is the damping coefficient, "$u(t)$" represents the input force, and "$y(t)$" is the vertical position of the mass. The corresponding dimensions of the parameters and variables involved in this model are assumed to be expressed in the metre-kilogram-second system of measurements. It is also assumed that both the input force "$u(t)$", and the vertical position of the mass "$y(t)$" are measurable variables. In this computer simulation, the values of the unknown parameters "m", "c" and "ξ" are considered to be 1.459, 5.837, and 58.37, respectively [85]; and the input force "$u(t)$" is assumed

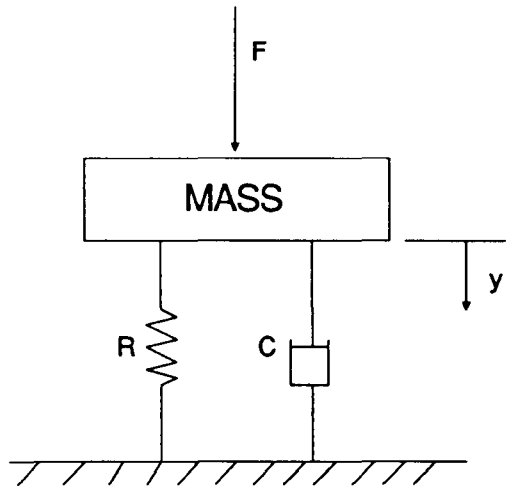


Figure 4.3: Spring-mass-damper system

be:

$$u(t) = \begin{cases} 1 & \text{if } t < 5 \\ 2 & \text{if } t > 5 \end{cases} \tag{4.14}$$

The adaptation gain "k" of the learning algorithm, (equation 3.32), is set to 10. It is important to mention that, from equations 3.32, 3.33, and 3.36, the larger the value of the adaptation gain "k" is selected, the shorter is the convergence time of the neural network output to the desired output, and the better is the tracking characteristic of the network. Figure 4.4 shows the performance of the adaptive neuron in tracking the desired output $y_d(t) = y(t)$. Observe that the corresponding tracking error goes to zero after a short time. Figure 4.5 illustrates the convergence of the adaptive neuron output "$y_o(t)$" to the desired output "$y_d(t)$". Note that the convergence time "t_r" is equal to 0.051 seconds. Finally, figure 4.6 sketches the behaviour of the neuron weights "$w_1(t)$" and "$w_2(t)$". Notice that the learning algorithm represented by equation 3.42, which constitutes a more relaxed variable structure control to update the weights of the neuron, can be easily implemented by considering that the input to the neuron is the output of a stable n-th order filter whose input is the input to the unknown linear system. A possible choice of filter design is the following:

$$\begin{aligned} \dot{z}_1(t) &= e_f(t) = -c_1 z_1(t) - c_2 z_2(t) - \ldots - c_n z_n(t) + u(t) \\ \dot{z}_2(t) &= z_1(t) \end{aligned} \tag{4.15}$$

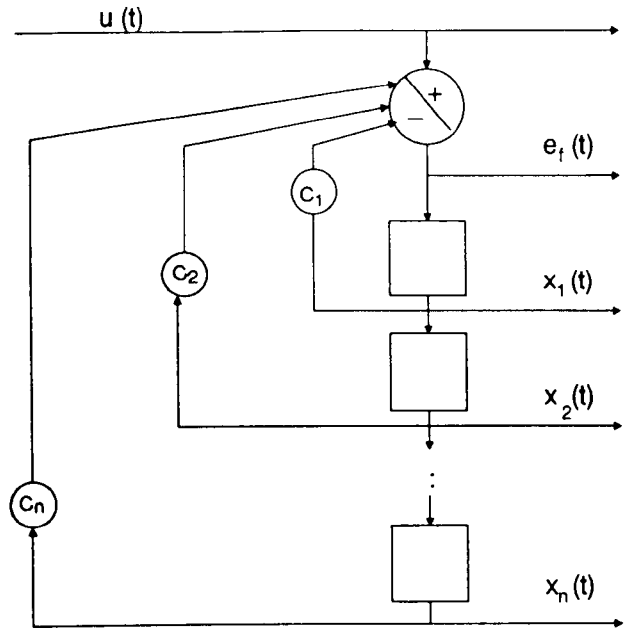


Figure 4.3a: Block diagram of the filter design.

$$\dot{z}_n(t) = z_{n-1}(t)$$

where "z_i" represents the i-th filter output, "c_i", $i = 1, 2, ..n$ are design parameters selected to make equation 4.16 a stable equation, and "$e_f(t)$" is the filter error as illustrated in figure 4.3a. Under these circumstances, the neuron input may be selected as

$$X_a(t) = (z_1(t), z_2(t), ..., z_n(t), b)^T, \tag{4.16}$$

whose time derivative is given by

$$\dot{X}_a(t) = (e_f(t), z_1(t), ..., z_{n-1}(t), 0)^T. \tag{4.17}$$

As an example, consider again the problem of estimating the FTO of the linear system represented by equation 4.13. In particular, it is assumed that the filter parameters are $c_1 = 5$, and $c_2 = 2$, and the plant parameters are the same as before. Figure 4.7 shows the performance of the neuron in tracking the desired output $y_d(t) = y(t)$. Compared to figure 4.4, there is an improvement in the tracking error. Figure 4.8 illustrates the convergence of the neuron output "$y_o(t)$" to the desired output $y_d(t) = y(t)$. Finally, figure 4.9 shows the behaviour of the neuron's weights.

One way of obtaining an approximate FTO estimation of an unknown linear system is by selecting the neuron input as a set of signals in a neighborhood of the input signal to the unknown plant. That is, the components of the neuron input

vector "$X(t)$" are selected from the set

$$\{u(t) : |u(t)| \leq \Phi\}, \quad \Phi > 0 \tag{4.18}$$

The small value "Φ", which represents a boundary layer thickness around "$u(t)$", is selected to prevent the components of the input vector "$X(t)$" becoming zero simultaneously, thus avoiding, the need for the bias term "b". That is, the product $X^T(t)X(t) \neq 0$ for all "t".

In this case, it is possible to state that the sum of the weight elements of the neuron represents an approximation of the FTO of the unknown system. In other words:

$$0 = y_d(t \geq t_r) - \sum_{i=1}^{n} w_i(t \geq t_r)u_i(t \geq t_r) \tag{4.19}$$

For a small value of "Φ", the following approximation holds true

$$u_1(t \geq t_r) \approx u_2(t \geq t_r) \approx ... \approx u_n(t \geq t_r) \approx u(t \geq t_r), \tag{4.20}$$

and therefore

$$\begin{aligned} 0 &\approx y_d(t \geq t_r) - (\sum_{i-1}^{n} w_i(t \geq t_r))u(t \geq t_r) \\ y_d(t \geq t_r) = y(t \geq t_r) &\approx (\sum_{i=1}^{n} w_i(t \geq t_r))u(t \geq t_r) \\ \frac{y(t \geq t_r)}{u(t \geq t_r)} &\approx \sum_{i=1}^{n} w_i(t \geq t_r) \end{aligned} \tag{4.21}$$

Figures 4.10 and 4.11 show the computer simulation results obtained from estimating the FTO of equation 4.13. Figure 4.10 illustrates the performance of the adaptive neuron in tracking the desired output $y_d(t) = y(t)$, and figure 4.11 sketches the sum of the weight elements of the neuron. In this simulation it was assumed that the input vector to the neuron was the vector $X(t) = (u(t), u(t) + 0.01, u(t) - 0.01)^T$, with "$u(t)$" being the input to the unknown plant. The value of the gain of the learning algorithm was set equal to 10.

Consider now the case when an unknown bounded perturbation "$\xi(t)$" influences the input signal of our dynamic linear system example. Here, it is still possible to estimate the corresponding FTO using an adaptive linear neuron or a more complex neural network structure, like the two layer, or the three layer neural network. Theorem 2, in the previous chapter can be invoked to explain this assertion. The performance of the linear adaptive neuron in estimating the FTO is illustrated by showing the computer simulation results of figures 4.12 and 4.13. In this simulation, the additive input noise was a gaussian distributed sequence, with zero mean, and

standard deviation equal to one. The value of the gain for the learning algorithm however, was selected equal to 100.

4.2.2 CONTINUOUS-TIME NONLINEAR FTO MODELLING

Consider the causal, nonlinear dynamic system described by the equation

$$\dot{X}(t) = F(X(t), U(t)) \tag{4.22}$$

where in general, "$F(.)$" is an $n \times 1$ nonlinear vector function, "$X(t)$" is the $n \times 1$ state vector, and "$U(t)$" is a $p \times 1$ vector control law. A solution "$X(t)$" of equation 4.22 represents a set of curves in the state space, corresponding to a particular control input "$U(t)$", as "t" varies from zero to infinity. Similar to the linear case, it is possible to think of the FTO as a dynamic sensitivity or gain that allows one to obtain a state variable "$x_i(t)$", $i = 1, 2, .., n$, from an input signal "$u_j(t)$", $j = 1, 2, .., p$. Clearly, if the output of this dynamic system, say the $q \times 1$ vector "$Y(t)$", is expressed as

$$Y(t) = G(X(t), U(t)), \tag{4.23}$$

then, the FTO is the dynamic sensitivity or gain that allows one to obtain an output variable "$y_k(t)$", $k = 1, 2, .., q$, from an input variable "$u_j(t)$", $j = 1, 2, .., p$. In order to compare the accuracy of the estimation of the FTO for a nonlinear system like equation 4.22 a single layer, two layer, and three layer neural networks are used. An inverted pendulum mounted on a mobile cart is selected as an illustrative example. This dynamic system may be modelled as follows [65]

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= \frac{1}{\left(\frac{M}{m_p}\right) + (sin(x_3(t)))^2}\left[\frac{u(t)}{m_p} + x_4^2(t) l sin(x_3(t))\right. \\
&- \left. g sin(x_3(t)) cos(x_3(t))\right] \\
\dot{x}_3(t) &= x_4(t) \\
\dot{x}_4(t) &= \frac{1}{l\left(\frac{M}{m_p}\right) + (sin(x_3(t)))^2}\left[-\frac{u(t)}{m_p} cos(x_3(t)) - x_4^2(t) l cos(x_3(t)) sin(x_3(t))\right. \\
&+ \left.\left(1 + \frac{M}{m_p}\right) g sin(x_3(t))\right]
\end{aligned} \tag{4.24}$$

where the components of the state vector $X(t) = (x_1(t), x_2(t), x_3(t), x_4(t))^T$ represent the position and velocity of the cart, and the angular position and angular velocity of the pole, respectively. The control input "$u(t)$" is an external force applied to the cart. Figure 4.14 illustrates the inverted pendulum system. The parameters

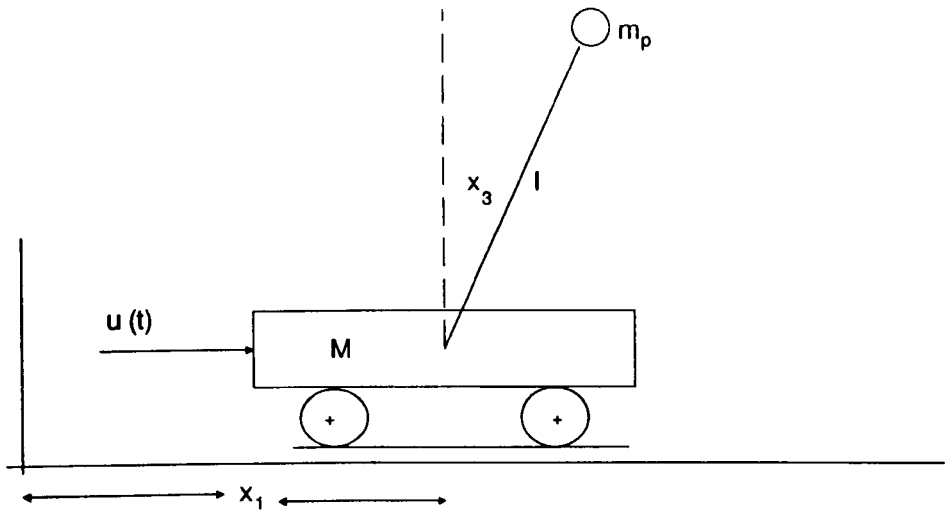


Figure 4.14: Inverted pendulum mounted on a mobile cart system.

"M", "m_p", "l", and "g" represent the mass of the cart, the mass of the pole, the length of the pole, and the acceleration due to gravity, respectively. All dimensions are assumed to be expressed in the metre-kilo-second unit of measurements. In particular, it is considered that the values for "M", "m_p", "l", and "g" are 1.0, 0.1, 0.5, and 9.81, respectively. The following computer simulations are carried out assuming that the inputs to the neural networks involved are taken from a neighborhood of the inverted pendulum applied input force "$u(t)$". That is $X(t) = (u(t), u(t) + 0.05, u(t) - 0.05)^T$.

Figure 4.15 plots the results obtained from estimating an approximate FTO between "$x_1(t)$" and "$u(t)$" when a single adaptive neuron is used. In this case, the value of the adaptation gain "k" of the learning algorithm was set equal to 200. Observe that despite this high adaptation gain, the approximating capability of the single adaptive neuron to track the desired output is not satisfactory. This poor performance is due to the difficulty of the adaptive linear neuron in tracking the nonlinear behaviour of the FTO of the system under study.

Figures 4.16 and 4.17 sketch the results obtained when a two layer neural network is used. This neural network consists of three input nodes with sigmoidal type activation functions, and one linear output node. In this case, the learning algorithm used to train the network corresponds to equations 3-70 and 3-71 of theorem 3, in the previous chapter. As before, the value of the adaptation gains "$k's$" were set equal to 200. Figure 4.18 shows the behaviour of some of the weight elements of the neural network. The weight elements of the network contain information regarding

an approximate FTO between the applied input force "$u(t)$" and the position of the cart "$x_1(t)$".

Consider now figures 4.19 and 4.20 which show the performance of a three layer neural network for tracking the state variables position of the cart, and position of the pole, respectively. Figure 4.21 illustrates the behaviour of some of the weight elements that contain information regarding an approximate FTO between the applied input force "$u(t)$" and the position of the cart "$x_1(t)$". Here, the neural network structure consists of three input nodes, two hidden nodes, and one output node. The input and the hidden nodes have sigmoidal type activation functions, whereas the output node is a linear node. The learning algorithm used to train the network corresponds to equations 3.90, 3.91, and 3.92 of theorem 4, in the previous chapter. The computer simulation was performed considering that the values of the adaptation gains of the algorithm were equal to 200. Observe that, compared to the adaptive linear neuron case, in the two layer and three layer neural network cases, it is possible to obtain good approximations of the nonlinear behaviour of the FTO of the nonlinear system under scrutiny.

Finally, the case when an additive, bounded, stochastic input perturbation affects the behaviour of the dynamic system can also be dealt with using two layer and three layer neural networks. Figures 4.22 and 4.23 show the performance of a two layer neural network with three input nodes. As before, the additive noise signal is assumed to be normally distributed, with zero mean and standard deviation equal to one.

Figures 4.24 and 4.25 show the computer simulation results obtained when a three layer neural network, identical to the one previously used, was implemented to estimate an approximate FTO of the perturbed unknown dynamic system under consideration. It can be verified that under the presence of additive bounded noise, and for the same value of the adaptation gains "k" of the learning algorithms, the three layer neural network outperforms the results obtained from the two layer neural network.

4.3 DISCRETE-TIME DYNAMIC SYSTEMS FTO MODELLING

A large class of discrete-time dynamic systems may be adequately represented by a mathematical model of the form:

$$y(k+1) = f(y(k), y(k-1), ..., y(k-n+1), u(k), u(k-1), ..., u(k-m+1)) + \xi(k) \quad (4.25)$$

where "$y(k)$", "$u(k)$", and "$\xi(k)$" are the system output, system input, and noise variable, respectively; and $f(.) : \Re^{n+m} \rightarrow \Re$ is assumed to be an unknown function.

As in the continuous-time case, the problem of estimating the FTO between the input signal "$u(k)$" and the output signal "$y(k)$" is related to the problem of learning a mapping between known input and output spaces.

A particular case of equation 4.25 is the linear system model structure:

$$y(k+1) = \sum_{i=0}^{n} a_i y(k-i) + \sum_{i=0}^{m} b_i u(k-i) + \xi(k) \tag{4.26}$$

Other model structures that are also particular cases of equation 4.25 can be found in [15].

Usually, the design of a static feedforward neural network to model equation 4.25 takes into consideration the selection of a network structure able to approximate the behaviour of the dynamic system under consideration, and the assignment of the input nodes of the network so that lagged input and output variables are adequately represented in the network. A general series-parallel neural network structure that may be used to model equation 4.25 is the following:

$$y_o(k+1) = \hat{f}(y(k), y(k-1), ..., y(k-n+1), u(k), u(k-1), ..., u(k-m+1)) \tag{4.27}$$

where "$y_o(k)$" represents the neural network output, and $\hat{f}(.) : \Re^{n+m} \rightarrow \Re$ represents the neural network approximation of the function "f" in equation 4.25. In our case however, due to the dynamic behaviour characteristic of the neural network's weights, it is possible to simplify the input node selection by choosing, for example, the input signals to the network in a neighborhood of the input signal to the unknown plant, as suggested by equation 4.18. Figure 4.26 depicts a block diagram of the discrete-time FTO estimation. In order to illustrate our formulation of the FTO problem, the following linear system is presented as a first example:

$$\begin{aligned} y(k+1) &= \left(2 - \frac{Tc}{m}\right) y(k) + \left[\frac{T}{m}(c - T\xi) - 1\right] y(k-1) \\ &+ \frac{T^2}{m} u(k-1) \end{aligned} \tag{4.28}$$

Equation 4.28 is a discrete-time version of the spring-mass-damper system represented by equation 4.13. The sampling time "T" is assumed to be equal to 0.01 seconds, and the values for the parameters "m", "c", and "ξ" are considered to be the same as those used before. Due to the linear characteristic of the proposed dynamic system, a single adaptive neuron with input signal $X(k) = (u(k-1), u(k-1) + 0.05, u(k-1) - 0.05)^T$ is used to estimate an approximation

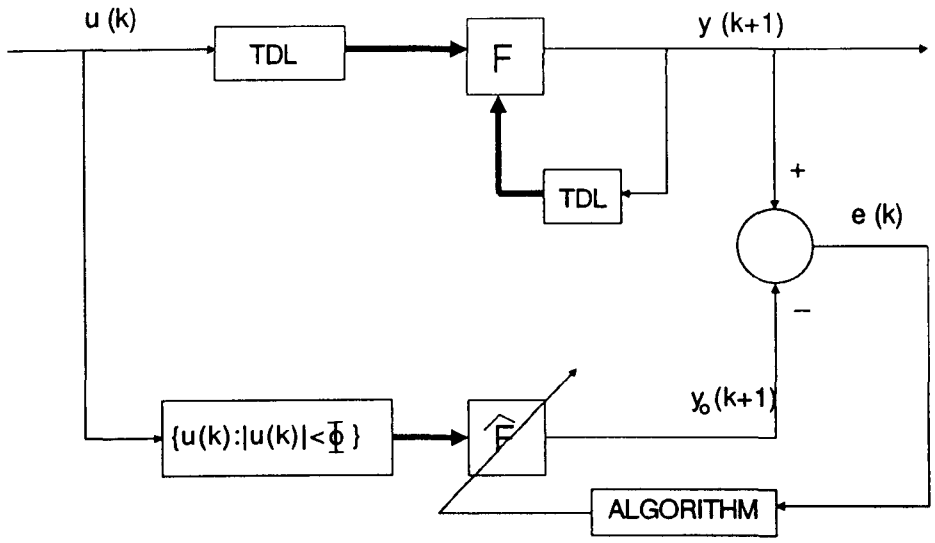


Figure 4.26: Discrete-time FTO estimation.

of the FTO between the variables "$u(k-1)$" and "$y(k)$". In this case, the learning algorithm used to train the weights of the adaptive neuron corresponds to equation 3.111 of theorem 5, in the previous chapter. The value of the adaptation gain "α" of the algorithm is selected equal to 0.1. Figure 4.27 shows the performance of the single adaptive neuron in tracking the desired output $y_d(k) = y(k)$. Figure 4.28 illustrates the convergence of the neuron output to the desired output, and figure 4.29 sketches the behaviour of the neuron weights.

The performance of the adaptive neuron when the input signal to the unknown plant is corrupted by white noise is shown in figure 4.30. Figure 4.31 shows the behaviour of the corresponding weight elements.

A discrete-time version of the inverted pendulum mounted on a cart is presented as a second example of the FTO estimation problem. The mathematical model of this dynamic system may be expressed as:

$$\begin{aligned}
x_1(k+1) &= x_1(k) + Tx_2(k) \\
x_2(k+1) &= x_2(k) + \frac{T}{\left(\frac{M}{m_p}\right) + (sin(x_3(k)))^2}\left[\frac{u(k)}{m_p} + x_3^2(k)lsin(x_3(k))\right. \\
&- \left. gsin(x_3(k))cos(x_3(k))\right] \\
x_3(k+1) &= x_3(k) + Tx_4(k) \\
x_4(k+1) &= x_4(k) + \frac{T}{l\left(\frac{M}{m_p}\right) + (sin(x_3(k)))^2}\left[-\frac{u(k)}{m}cos(x_3(k))\right.
\end{aligned} \tag{4.29}$$

$$- \quad x_4^2(k)lcos(x_3(k))sin(x_3(k)) + \left(1 + \frac{M}{m_p}\right) gsin(x_3(k))\Big]$$

It is assumed that the position of the cart "$x_1(k)$", the angular position of the pole "$x_3(k)$", and the applied input force "$u(k)$" are measurable variables. For the sake of comparing their performance, a two layer, and a three layer neural network with sign type activation functions are designed to obtain FTO approximations of the proposed plant.

The two layer neural network consists of three input nodes, and one output node; whereas the three layer neural network has three input nodes, three hidden nodes, and one output node. In the two layer neural network case, the estimation of the FTO between the applied input force "$u(k)$" and the position of the cart "$x_1(k)$" can be performed using the vector $X(k) = (u(k), u(k) + 0.05, u(k) - 0.05)^T$ as input vector to the network. Here the learning algorithm corresponds to equations 3.136 and 3.137 of theorem 6, in the previous chapter. Figure 4.32 shows the performance of the two layer neural network, and figure 4.33 illustrates the behaviour of some weight elements of the network. Figures 4.34 and 4.35 sketch the computer simulation results obtained from the two layer neural network when the applied input force to the inverted pendulum is corrupted by white noise. Observe that despite the performance of the two layer neural network is satisfactory when the applied input force to the system is not corrupted by noise, (ie. figure 4.32), its performance is not as good when an aleatory noise influences the applied input force, (ie. figure 4.34).

In the three layer neural network case, the input vector to the network was the same used for the two layer neural network case. The learning algorithm used to train the network is represented by the equations 3.149, 3.150, and 3.151 of theorem 7, in the previous chapter. Figure 4.36 shows the performance of the network, whereas figure 4.37 illustrates the behaviour of some weight elements. The performance of the three layer neural network, when the applied input force to the inverted pendulum system is corrupted by white noise is shown in figures 4.38 and 4.39. Both in the two layer and three layer neural network cases, the adaptation gain elements of the diagonal matrix "A", in their corresponding learning algorithms, were selected equal to 0.8. Finally, observe that as expected, the three layer neural network outperforms the results obtained with the two layer neural network.

4.4 INVERSE TRANSFER OPERATOR MODELLING

The previous section showed the approximating capability of neural networks to estimate unknown mappings by using a learning algorithm that requires a set of input signals and a desired output signal to train the network. While in the FTO estimation problem the input and desired output signals are selected from the input and output signals of the unknown plant under study, respectively; in the inverse transfer operator (ITO) problem the goal is to determine the corresponding dynamic sensitivity or gain that enables the time history of the input of the unknown system to be reproduced from its present output. Two important conceptual considerations that must be taken into account when one estimates the ITO are stability and uniqueness of the ITO. That is, if the ITO of the system under scrutiny is unstable, then under our suggested framework, the weight elements of the network would grow without bound, and therefore it would not be possible to estimate the ITO. (ie. non-minimum phase systems). The uniqueness issue is related to the fact that the dynamic system mapping between output and input signals is not a one-to-one mapping, and therefore the estimation of the ITO could give an incorrect result. This problem can be overcome by using the so called specialised inverse learning approach [66].

A common way of training neural networks for reproducing the inverse model of a dynamic system is by using a synthetic training signal to force the system response. Then, the output of the system is connected to the input of the network, and the output of the network is compared to the synthetic training signal to generate an error signal to train the network. This approach is known as generalized inverse learning or direct inverse modelling [61].

Figure 4.40 depicts the ITO estimation scheme used in this work. The main difference between this scheme and the direct inverse modelling scheme is in the on-line training capability of our proposed embedded learning algorithms.

The following two subsections presents continuous-time and discrete-time examples of the ITO estimation problem.

4.4.1 CONTINUOUS-TIME DYNAMIC SYSTEMS ITO MODELLING

Consider a stable, invertible, unknown dynamic system described by the equation

$$Y(t) = H(X(t), U(t)) \tag{4.30}$$

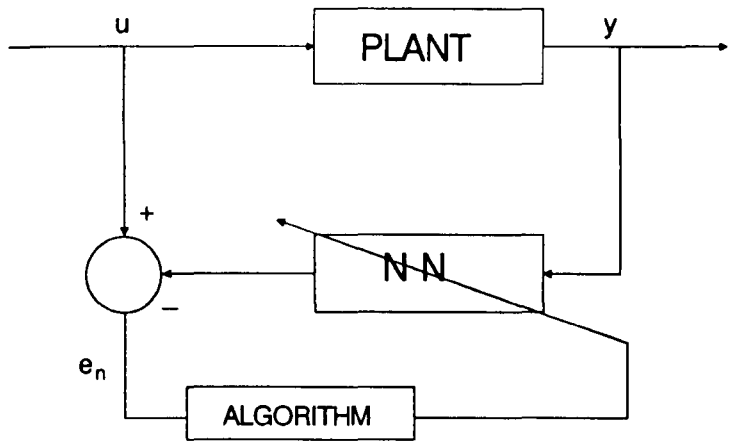


Figure 4.40: ITO estimation scheme.

where $Y(t) \in \Re^q$ is an output vector, $X(t) \in \Re^n$ is a state vector, and $U(t) \in \Re^p$ is a control input vector. It is assumed that both "$Y(t)$" and "$U(t)$" are measurable vector quantities. The ITO is a dynamic sensitivity or gain that enables an input signal variable "$u_j(t)$", $j = 1, 2, .., p$, to be reconstructed from an output variable "$y_i(t)$", $i = 1, 2, .., q$. This reconstruction task is performed by a neural network whose structure depends on the complexity of the inverse dynamic behaviour of the unknown system.

In the case described by the linear model 4.8, the ITO of the system is represented by the equation

$$\frac{u(t)}{y(t)} = \frac{D^{(m)} + a_1 D^{(m-1)} + ... + a_{m-1} D + a_m}{b_0 D^{(m)} + b_1 D^{(m-1)} + ... + b_{m-1} D + b_m} \tag{4.31}$$

The spring-mass-damper system described by equation 4.13 is considered again to illustrate the ITO estimation problem. As a first attempt, a single adaptive neuron is used to accomplish the task at hand. Here, the input signal vector "$X(t)$" for the neuron is taken from a neighborhood of the system output "$y(t)$". In particular the vector "$X(t)$" is selected as $X(t) = (y(t), y(t) + 0.05, y(t) - 0.05)^T$. The performance of the adaptive neuron to reconstruct the time history of the system input "$u(t)$", and the behaviour of the neuron weights are sketched in figures 4.41 and 4.42, respectively. It is important to point out that the algorithm adaptation gain "k" used to obtain these results was equal to 2000. Despite this high gain value, the single adaptive neuron performance is not satisfactory.

Figures 4.43 and 4.44 illustrate the computer simulation results when the desired output for the neuron, (ie. the system input "$u(t)$"), is corrupted by white noise.

Next, a two layer neural network with the same input vector used for the single adaptive neuron, is implemented in the computer to obtain the ITO of the spring-mass-damper system. The algorithm adaptation gains were also selected equal to 2000. Figures 4.45 and 4.46 show the performance of the two layer neural network, and the behaviour of some weight elements, respectively. Similarly, figures 4.47 and 4.48 illustrate the results obtained when the desired neural network output is corrupted by white noise. Observe that, compared to the single adaptive neuron results, the two layer neural network performance is better.

In the nonlinear case, the inverted pendulum mounted on a mobile cart system is used to show the performance of a three layer neural network in reconstructing an approximation of the ITO between the variable position of the cart "$x_1(t)$", and the applied input force "$u(t)$". Figures 4.49 and 4.50 show the performance of the three layer neural network, and the behaviour of some weight elements, respectively. The performance of the three layer neural network when the applied input force to the system, (ie. the network desired output), is influenced by additive white noise is shown in figure 4.51. Figure 4.52 shows some of the corresponding weight elements.

4.4.2 DISCRETE-TIME DYNAMIC SYSTEMS ITO MODELLING

The same general procedure of designing a neural network, and selecting its appropriate input signals and desired output to estimate an approximate FTO of a dynamic system followed in section 4.2.3 will be used in this section. In this case however, the input signals to the neural network are selected from a neighborhood of the output of the unknown dynamic system, and the network desired output is selected as the unknown dynamic system input. Both the discrete-time version of the spring-mass-damper system, and the inverted pendulum mounted on a mobile cart system are presented as illustrative examples.

The approximate ITO for the spring-mass-damper system is estimated using a two layer neural network with input signal $X(k) = (y(k), y(k) + 0.05, y(k) - 0.05)^T$, and desired output $y_d(k) = u(k-1)$. The adaptation gain elements of the diagonal matrix "A", in the learning algorithm, (ie. equations 3.136 and 3.137, in the previous chapter), were selected equal to 0.9. Figure 4.53 shows the performance of the two layer neural network in reconstructing the time history of the unknown

system input. Figure 4.54 illustrates the behaviour of some weight elements of the network. Likewise, figures 4.55 and 4.56 illustrate the performance of the two layer neural network, and the behaviour of some of the corresponding weight elements, respectively, when the input to the unknown system, (ie. the neural network desired output), is corrupted by white noise.

The estimation of the approximate ITO between the output variable position of the cart and the applied input force, in the inverted pendulum mounted on a mobile cart system is performed using a three layer neural network with input signal selected as the vector $X(k) = (x_1(k), x_1(k) + 0.05, x_1(k) - 0.05)^T$, where "$x_1(k)$" represents the variable position of the cart. The structure of the three layer neural network is the same as the one used for estimating the FTO of the inverted pendulum system, with adaptation gain elements, of the diagonal matrix "A" in the learning algorithm equal to 0.9. Figures 4.57 and 4.58 illustrate the performance of the three layer neural network in tracking the desired network output, (ie. the unknown dynamic system input), and the behaviour of some weight elements of the network, respectively. Similarly, figures 4.59 and 4.60 show the results obtained from the three layer neural network when its desired output is corrupted by white noise.

4.5 SUMMARY

In this chapter, the approximation capabilities of dynamically weighted feedforward neural networks have been exploited by constructing the FTO and the ITO of continuous-time and discrete-time unknown dynamic systems.

The FTO of a dynamic system has been defined as a dynamic sensitivity or gain that enables an output variable "y_i" to be obtained from an input variable "u_j". Two examples, both in continuous-time and discrete-time versions, have been proposed to illustrate the FTO estimation problem.

The first example which represented a mathematical model of the spring-mass-damper system has been used to illustrate the FTO estimation for time invariant linear systems. In this case, due to the linear characteristic of the considered system, a linear adaptive neuron was sufficient to perform the FTO estimation.

It has been shown that, in the estimation of an approximate FTO of an n-th order continuous-time system, the input signals to the neural network may be selected from a set of signals in a neighborhood of the unknown dynamic system input. Furthermore, it has been proved that, if the neural network is a single layer network, (ie. a single adaptive neuron), the approximate FTO of the unknown dynamic system under study may be obtained as the sum of the weight elements of

the network.

Additionally,in this chapter, illustrative examples have been used to test the performance of the designed neural networks and their embedded learning algorithms when estimating an approximate FTO of dynamic systems under the presence of additive bounded noise at the system input.

The second example, that represented a mathematical model of the inverted pendulum mounted on a mobile cart system has been used to illustrate the FTO estimation for nonlinear dynamic systems. The computer simulation results obtained for this example have shown that the two layer neural network approximating capabilities are better than those of a single layer network, and that a three layer neural network outperforms the results achieved using a two layer neural network.

The ITO on the other hand, has been defined as a dynamic sensitivity or gain that enables a reconstruction of the input time history of a stable, invertible, unknown dynamic system from the knowledge of its corresponding output. Again, the same examples as mentioned before have been used to illustrate the approximate ITO estimation. The best approximation performance has been obtained using a three layer neural network.

Finally, it has also been shown by the computer simulations performed on the illustrative examples, that the variable structure control-based learning algorithms proposed in chapter 3 operate on an on-line basis, without the requirement of long trial sessions to train the neural networks.

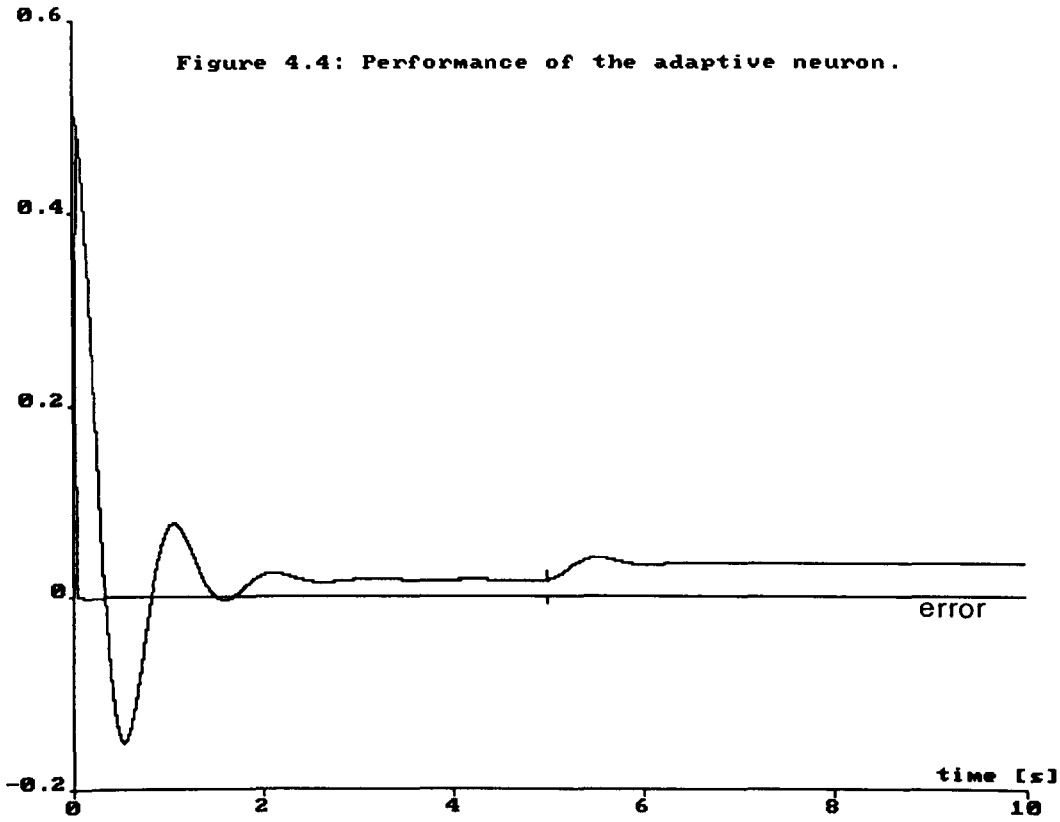


Figure 4.4: Performance of the adaptive neuron.

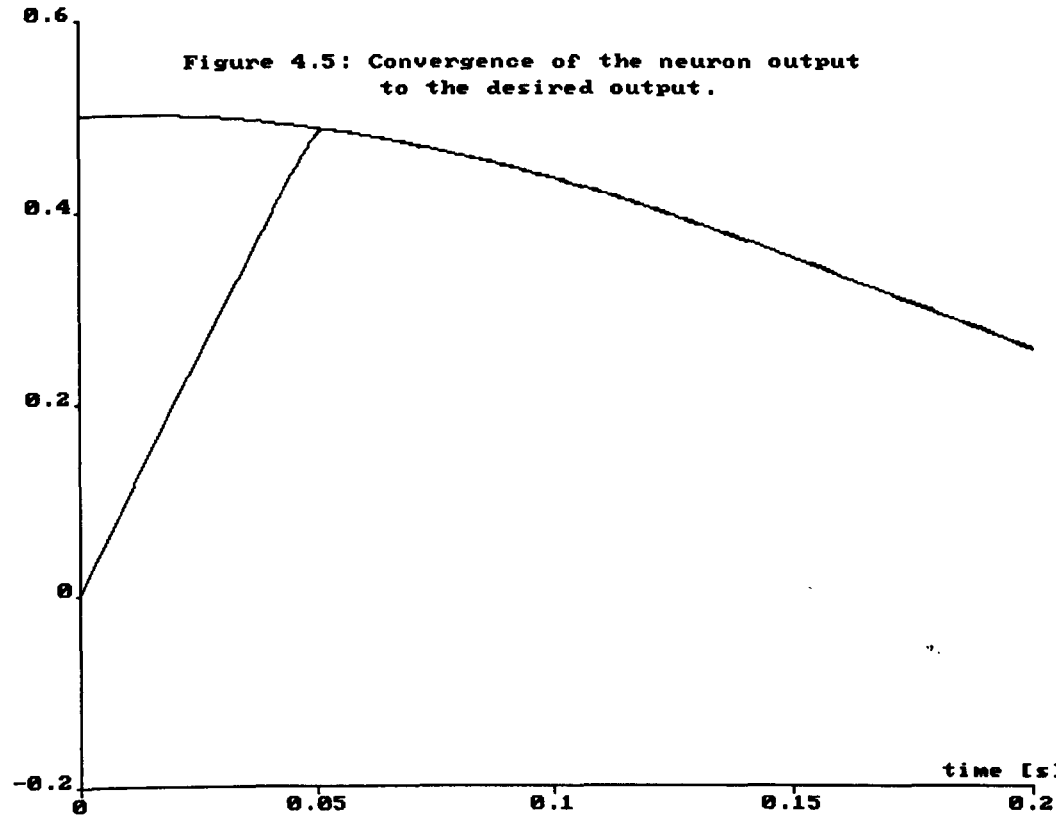


Figure 4.5: Convergence of the neuron output to the desired output.

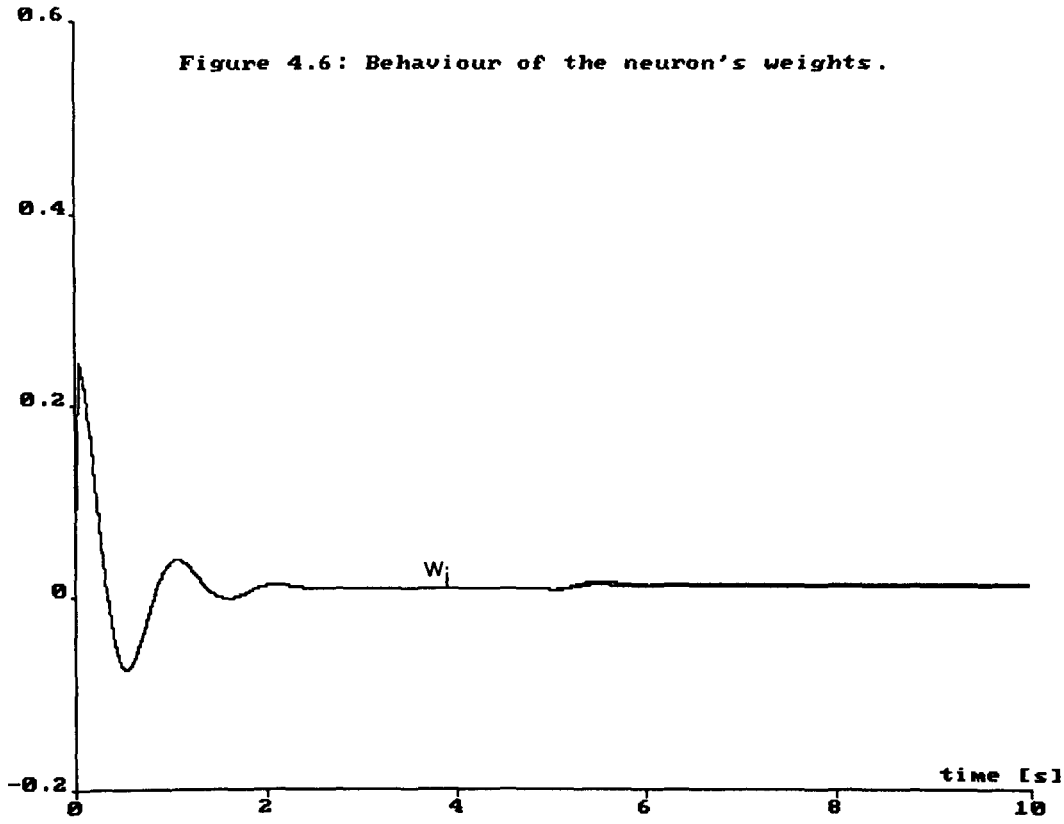


Figure 4.6: Behaviour of the neuron's weights.

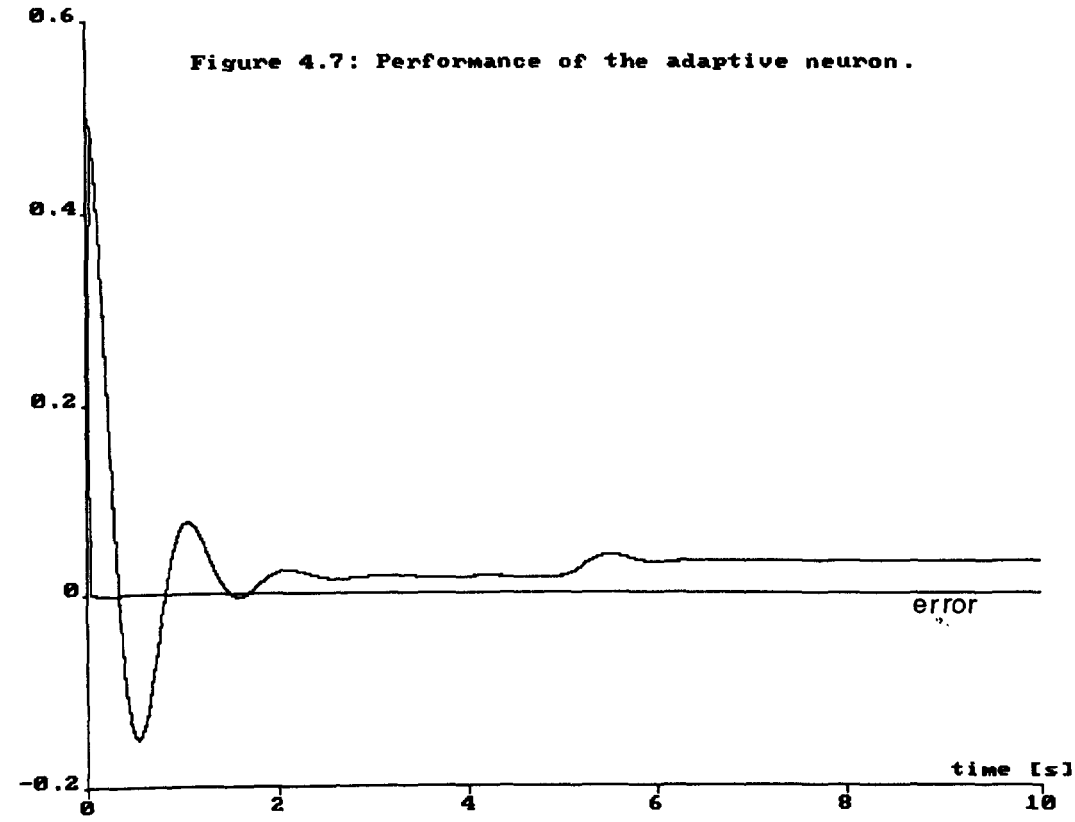


Figure 4.7: Performance of the adaptive neuron.

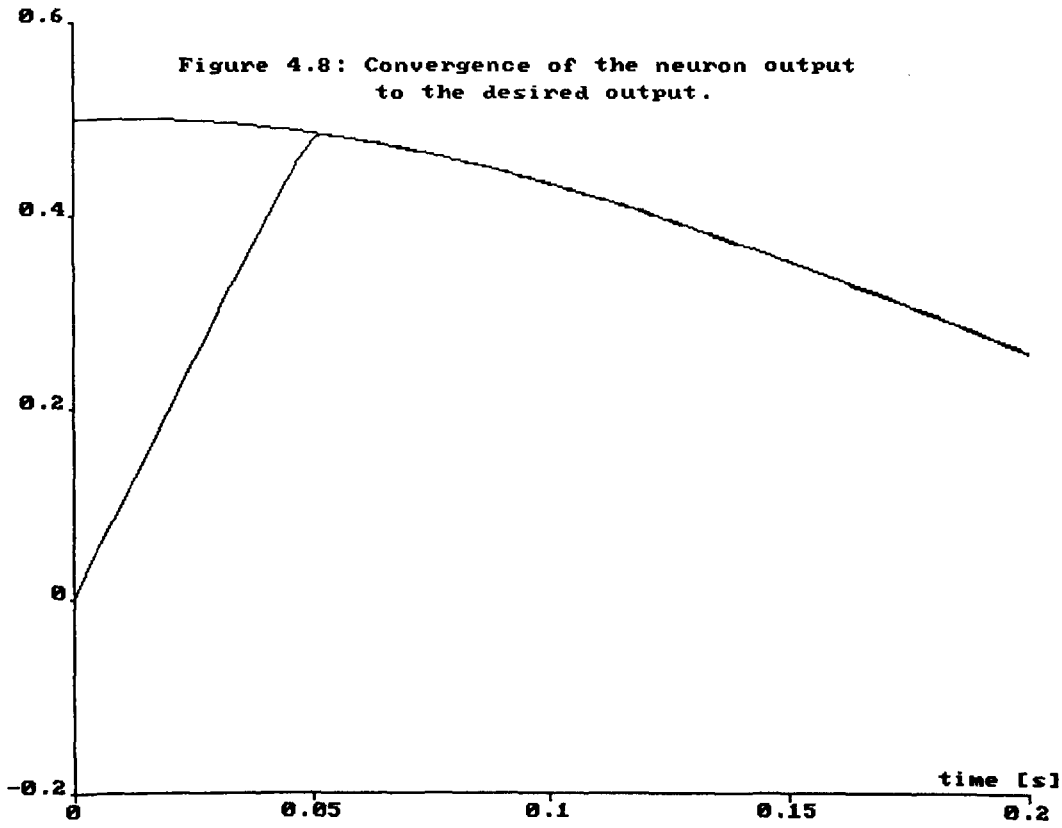


Figure 4.8: Convergence of the neuron output to the desired output.

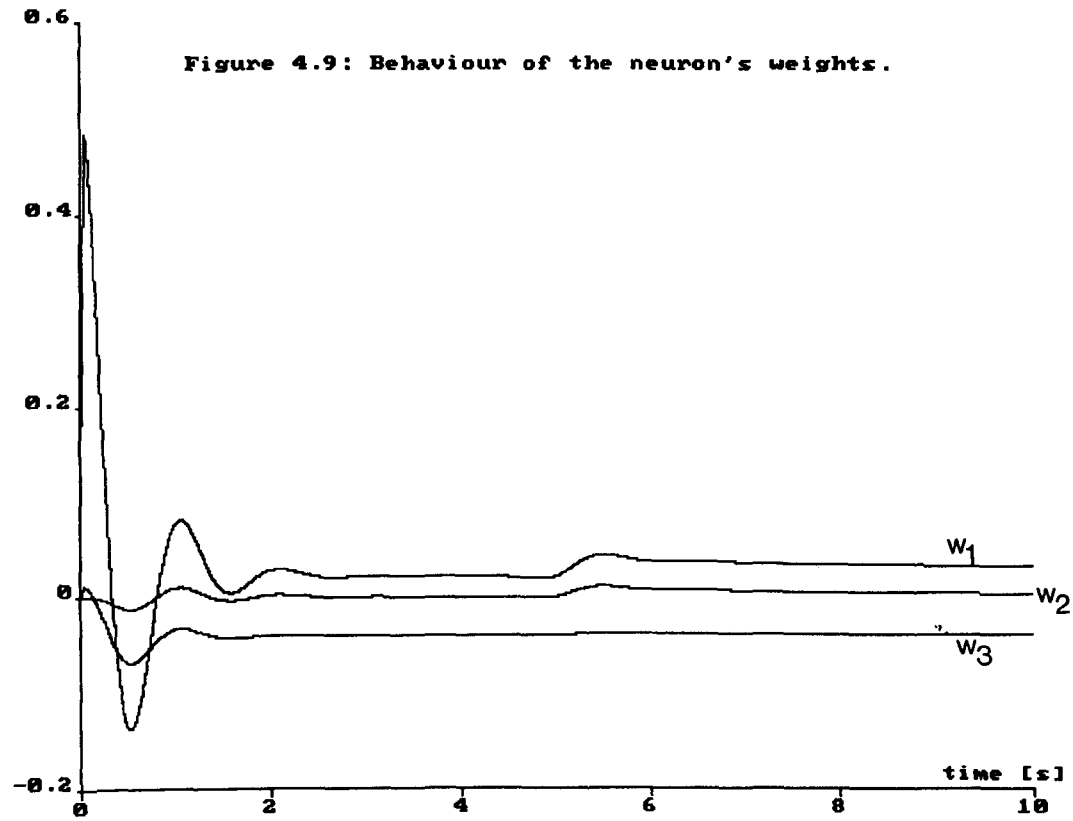


Figure 4.9: Behaviour of the neuron's weights.

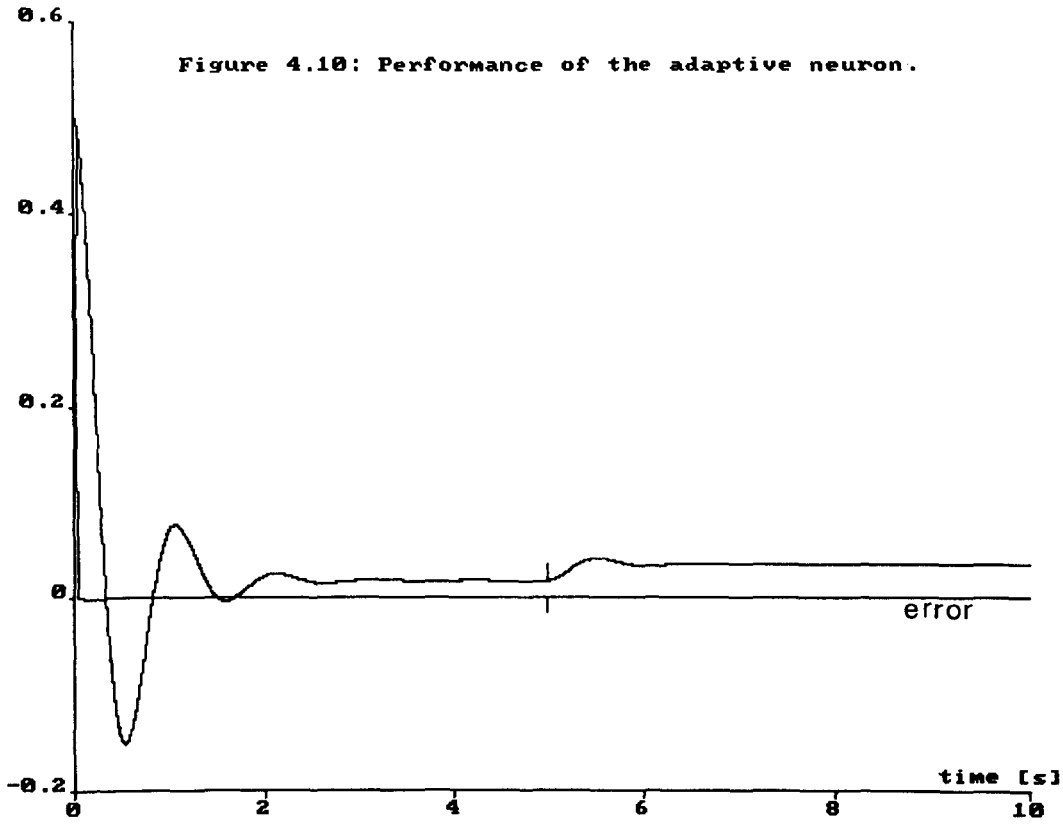


Figure 4.10: Performance of the adaptive neuron.

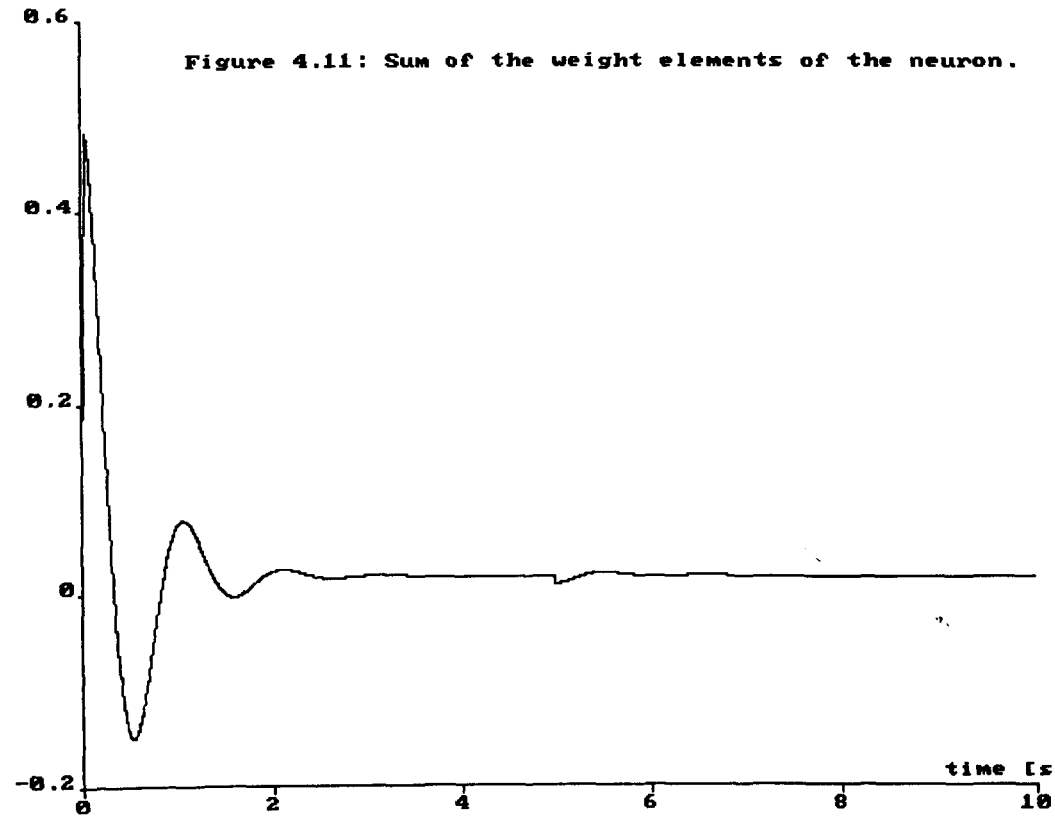


Figure 4.11: Sum of the weight elements of the neuron.

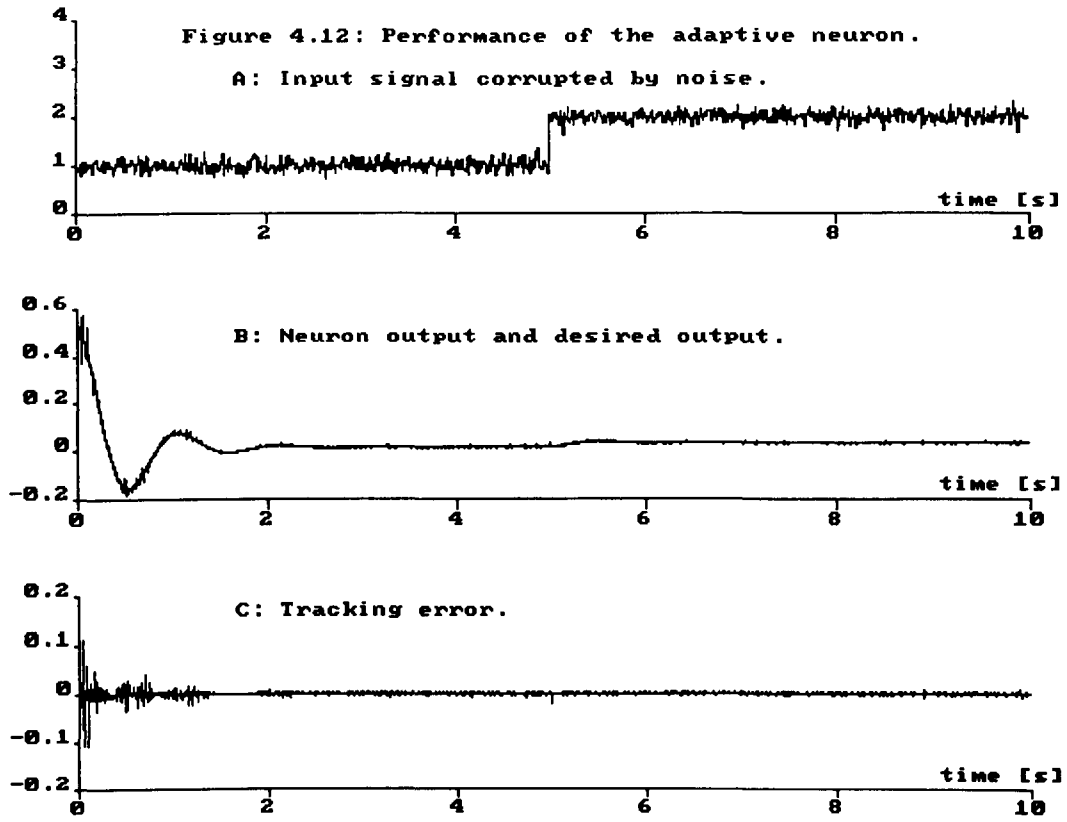


Figure 4.12: Performance of the adaptive neuron.

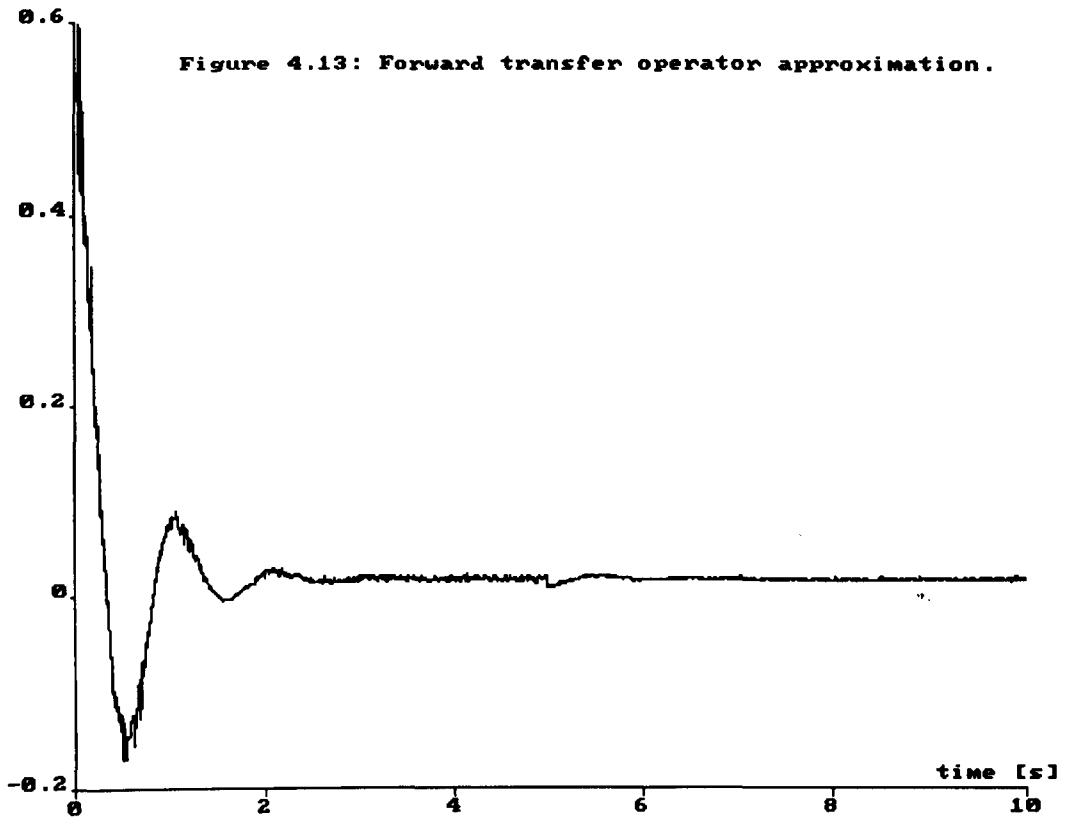


Figure 4.13: Forward transfer operator approximation.

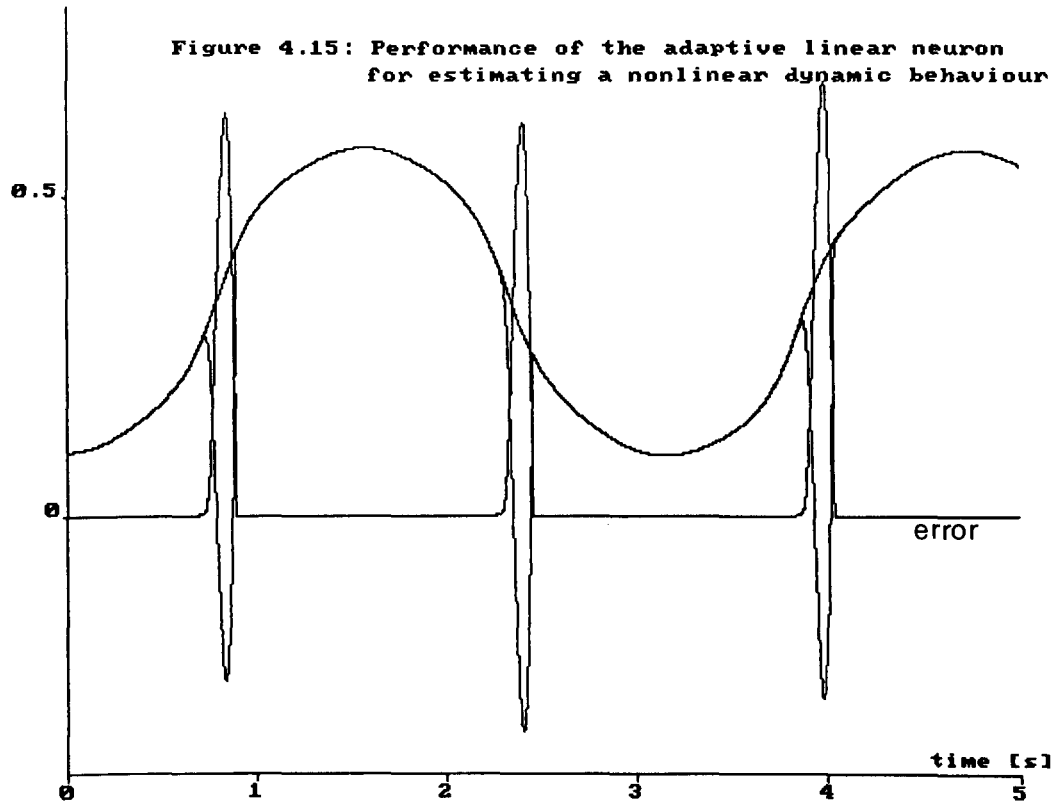


Figure 4.15: Performance of the adaptive linear neuron for estimating a nonlinear dynamic behaviour

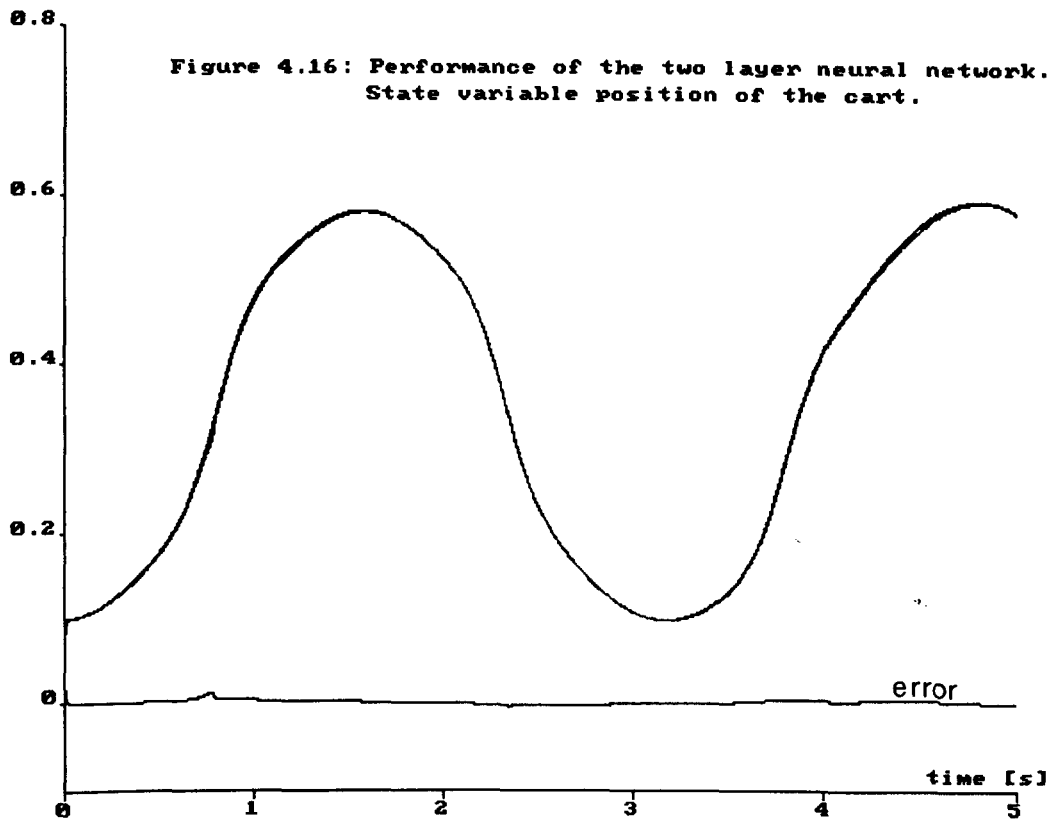


Figure 4.16: Performance of the two layer neural network. State variable position of the cart.

Figure 4.17: Performance of the two layer neural network.
State variable angular position of the pole.

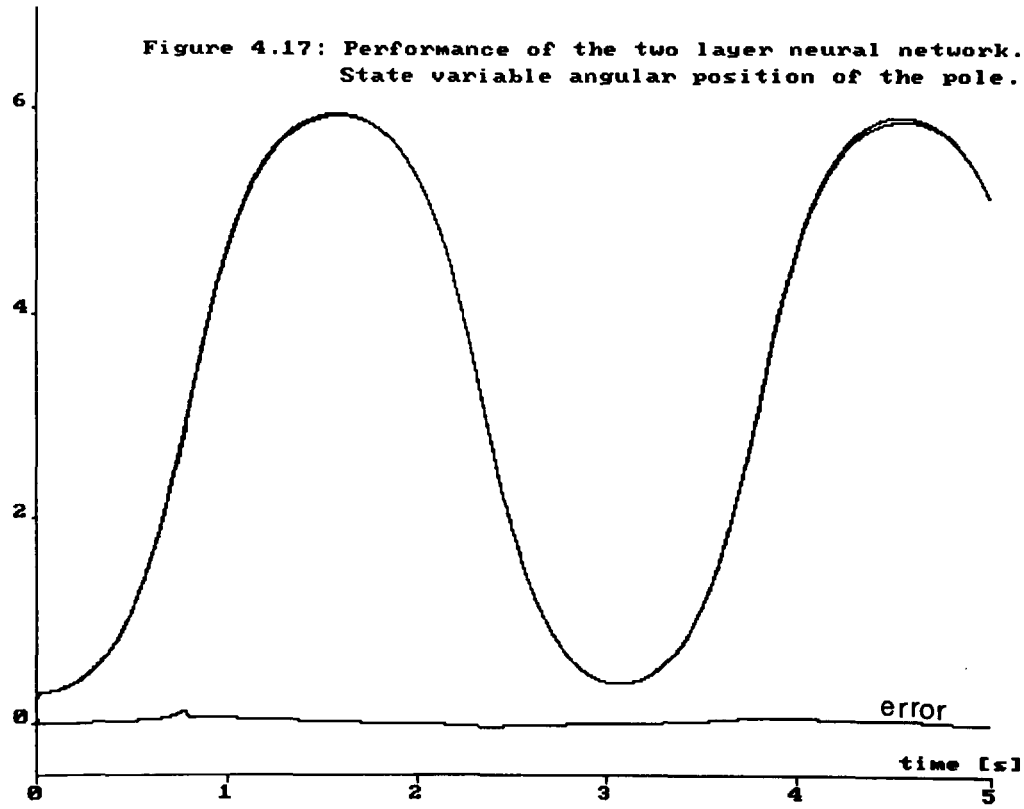


Figure 4.18: Behaviour of some weight elements.

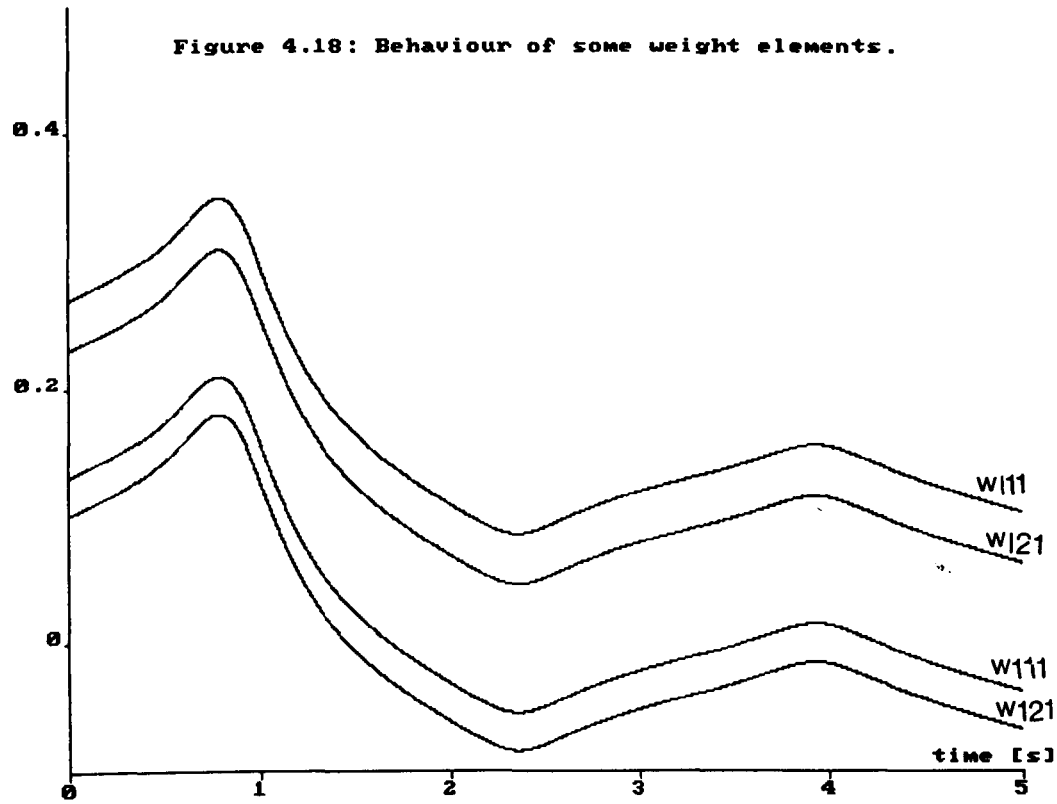


Figure 4 19: Performance of the three layer neural network.
State variable position of the cart.

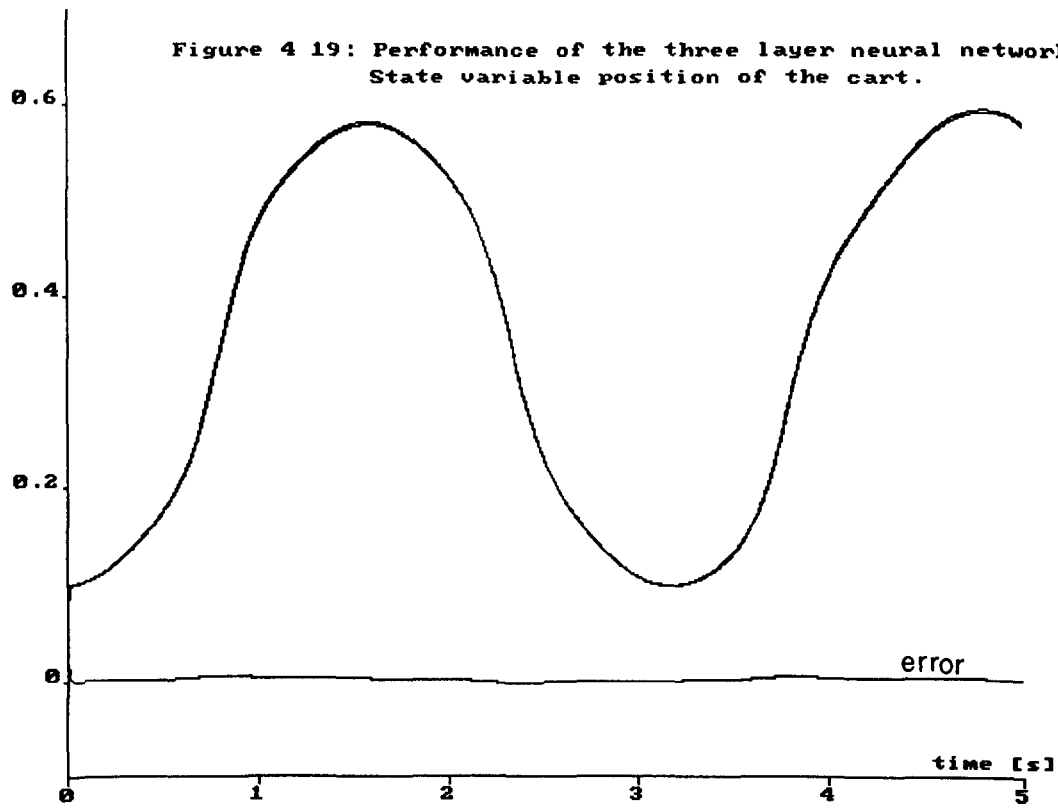
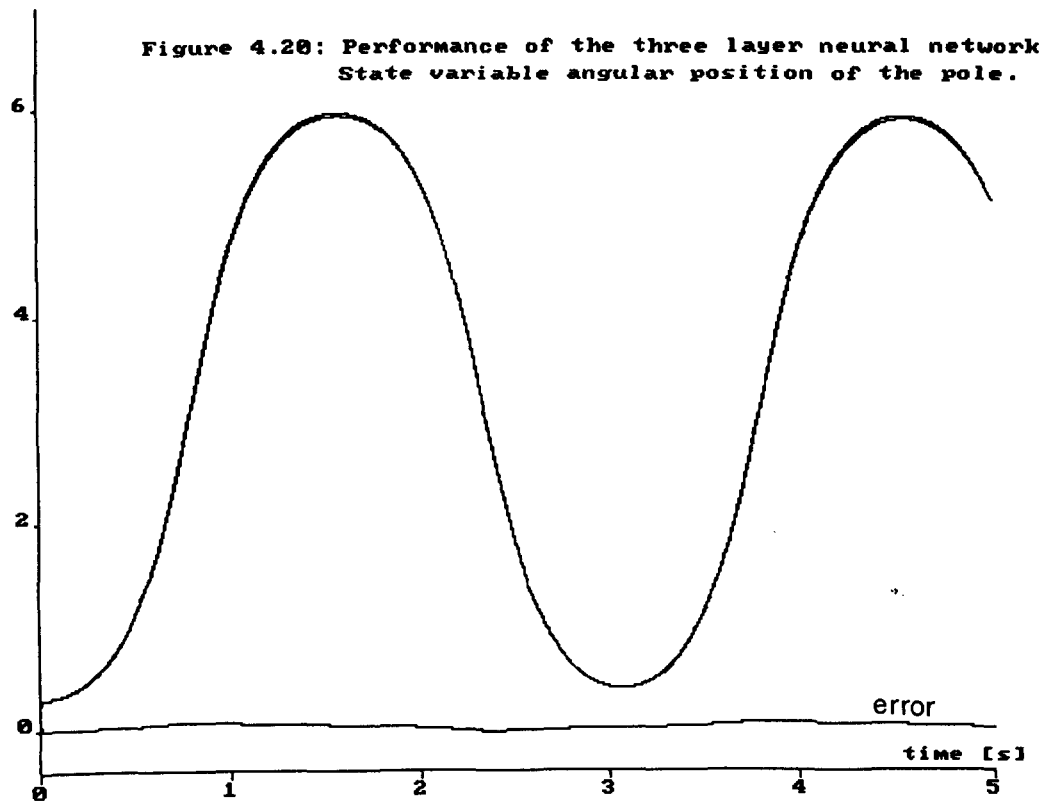


Figure 4.20: Performance of the three layer neural network.
State variable angular position of the pole.



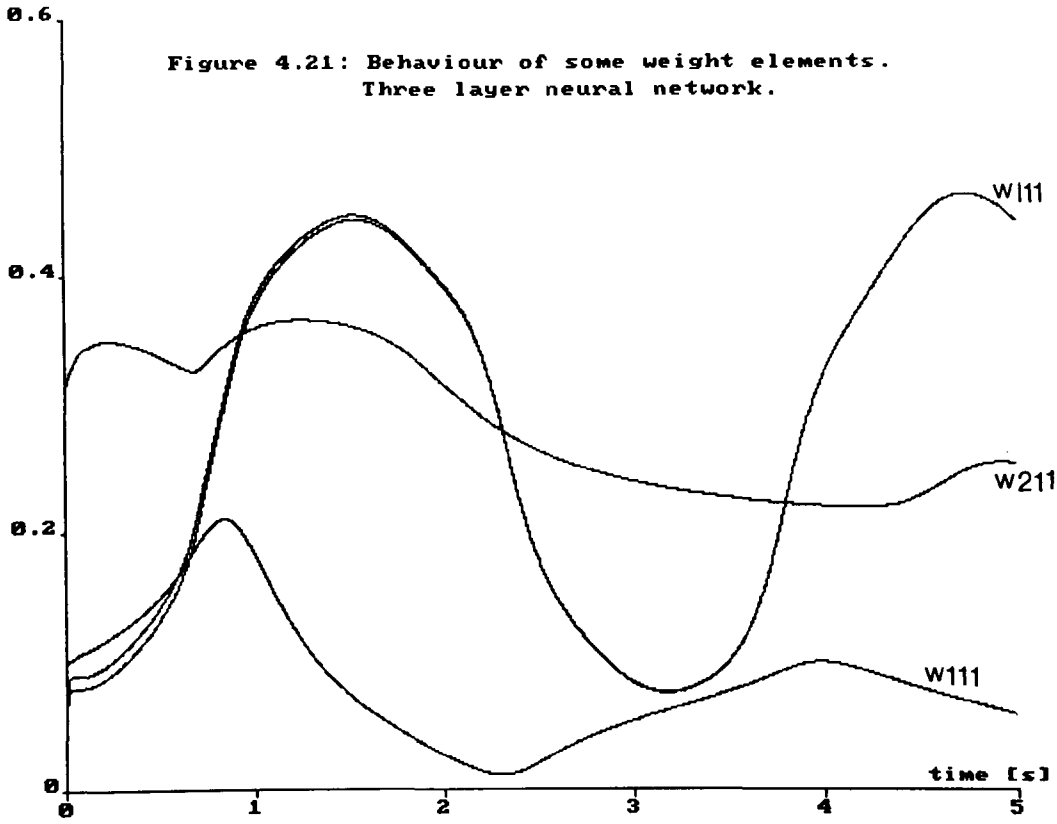


Figure 4.21: Behaviour of some weight elements.
Three layer neural network.

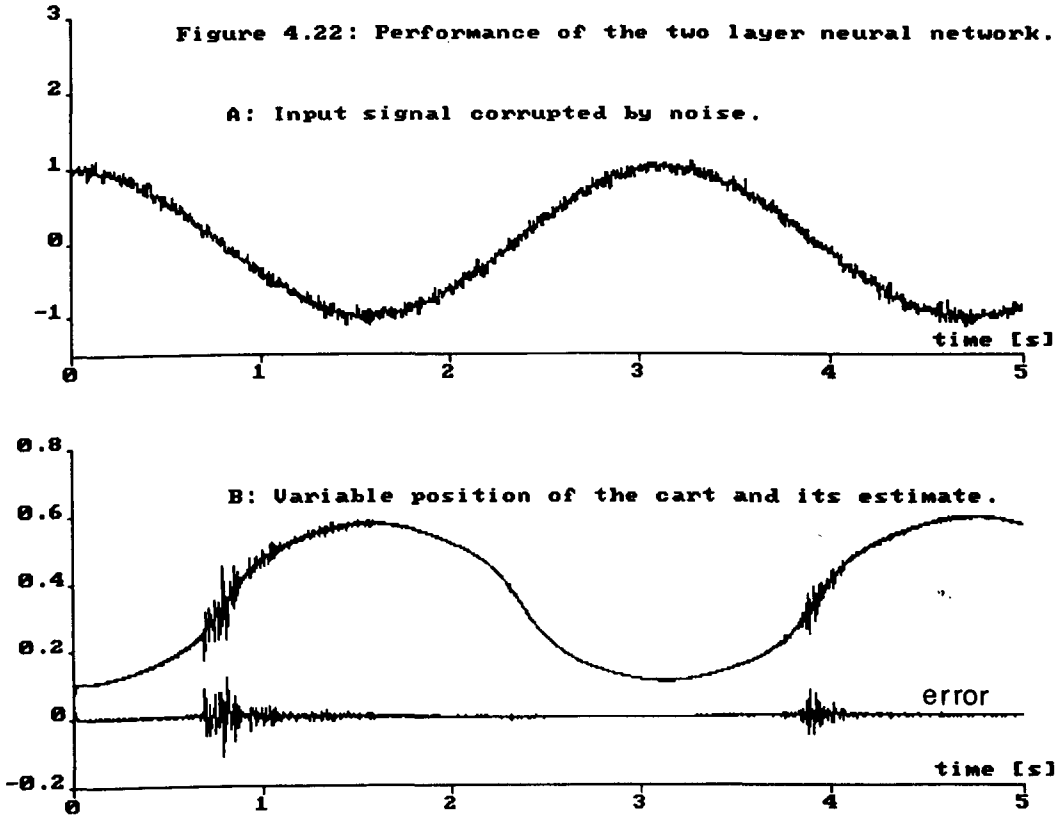


Figure 4.22: Performance of the two layer neural network.

Figure 4.23: Behaviour of some weight elements.
Two layer neural network.

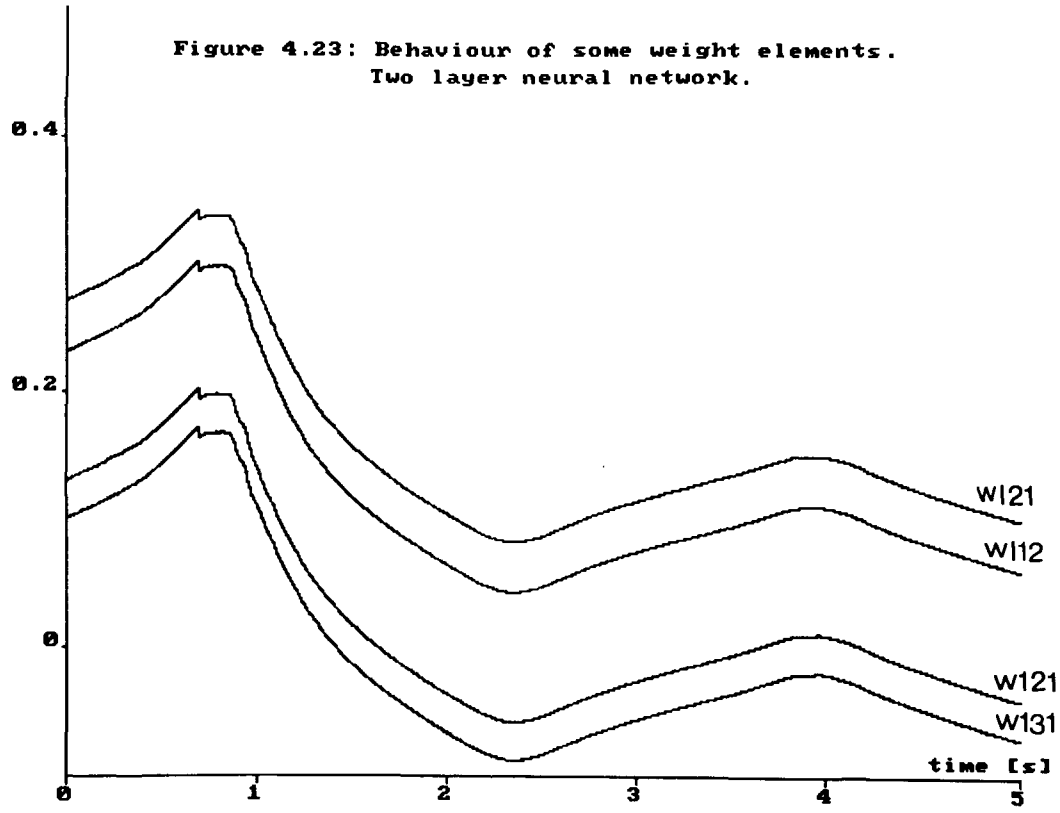
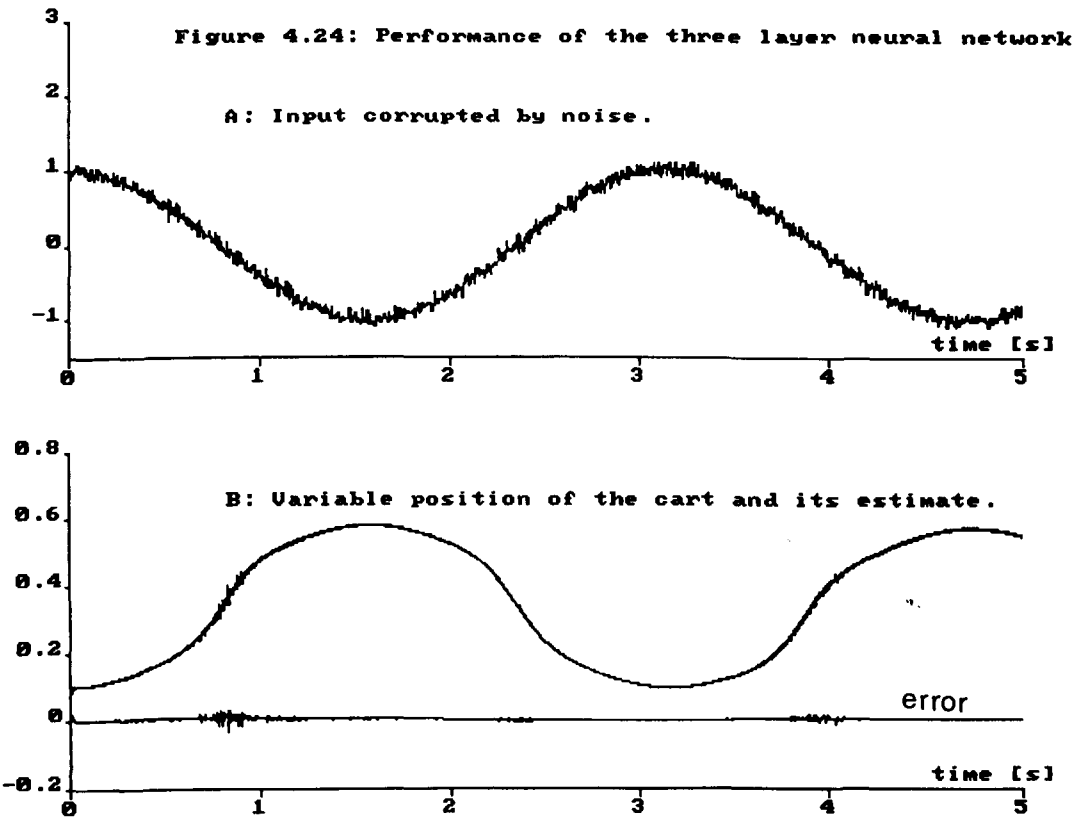


Figure 4.24: Performance of the three layer neural network.



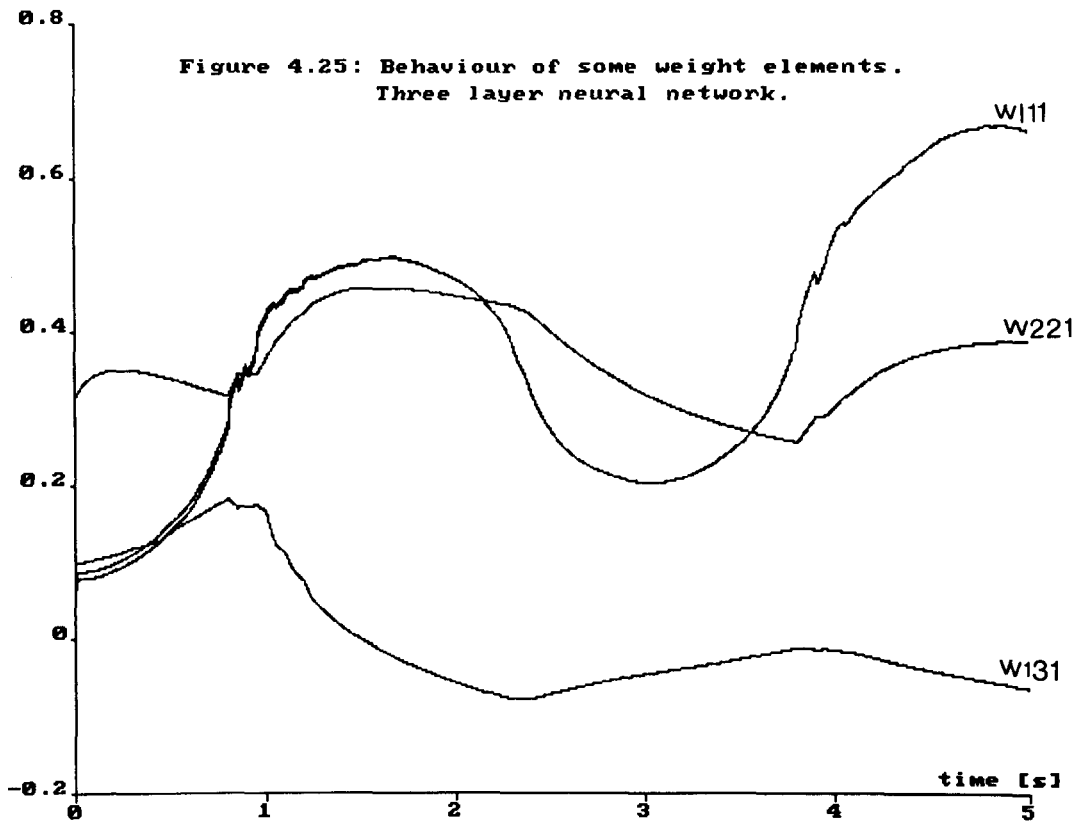


Figure 4.25: Behaviour of some weight elements.
Three layer neural network.

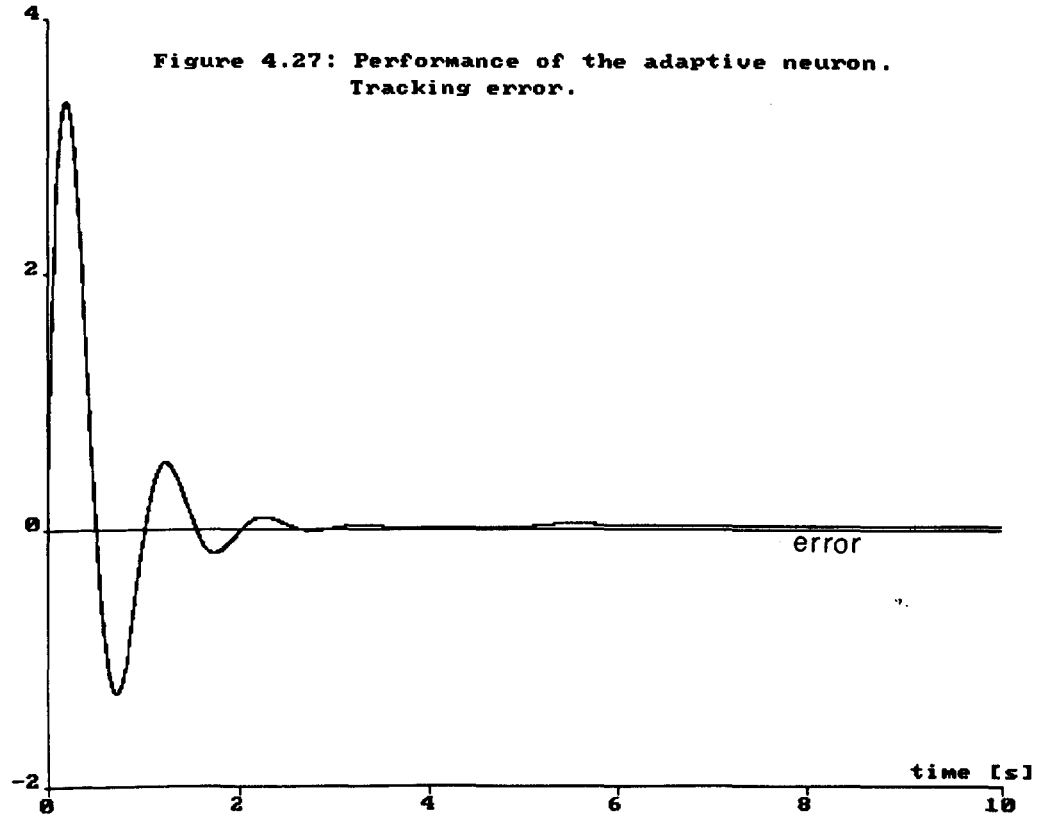


Figure 4.27: Performance of the adaptive neuron.
Tracking error.

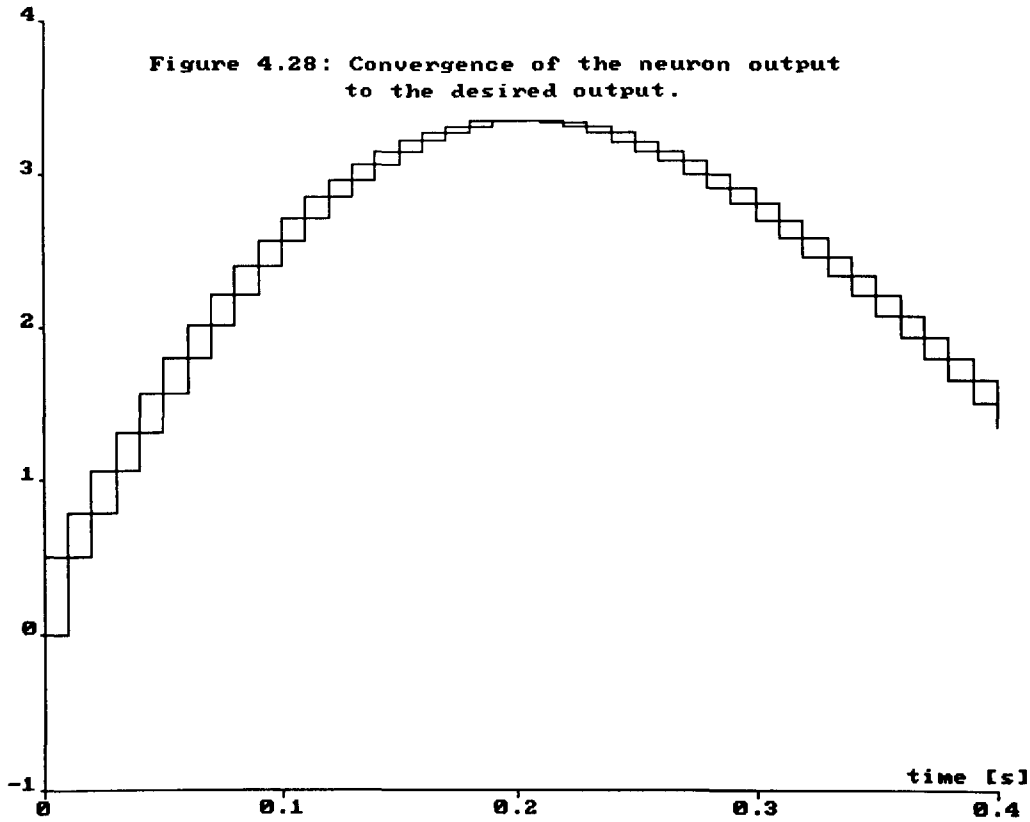


Figure 4.28: Convergence of the neuron output to the desired output.

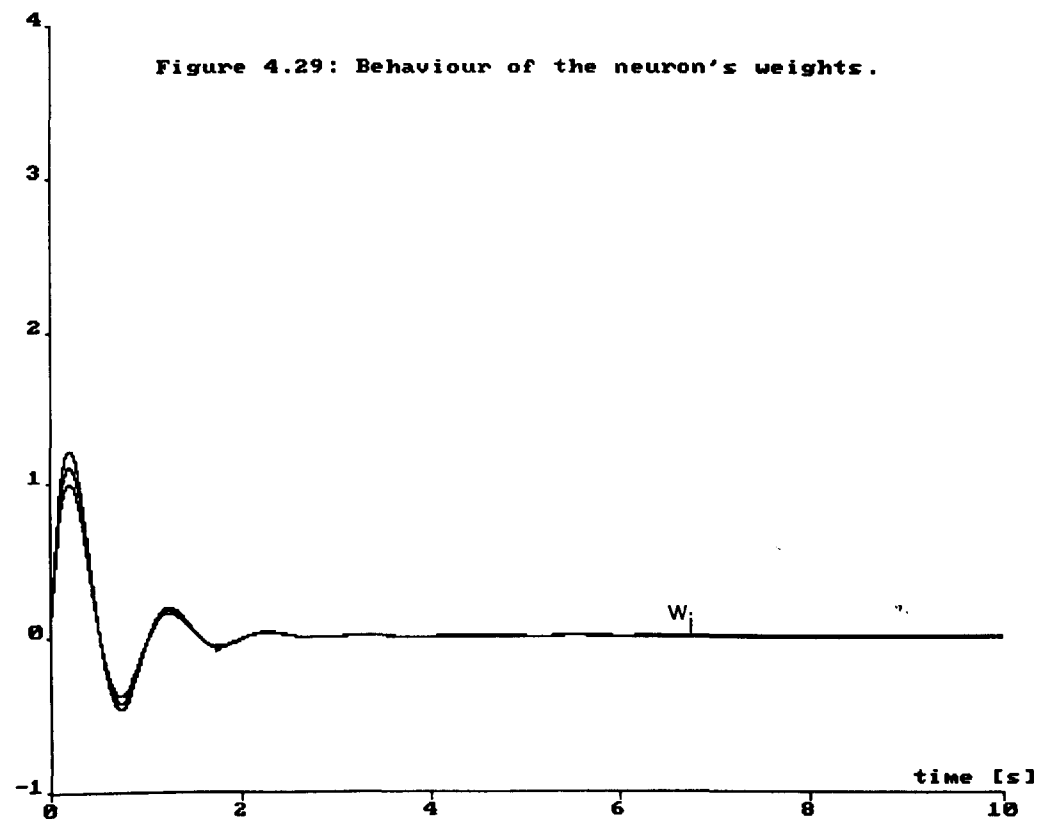


Figure 4.29: Behaviour of the neuron's weights.

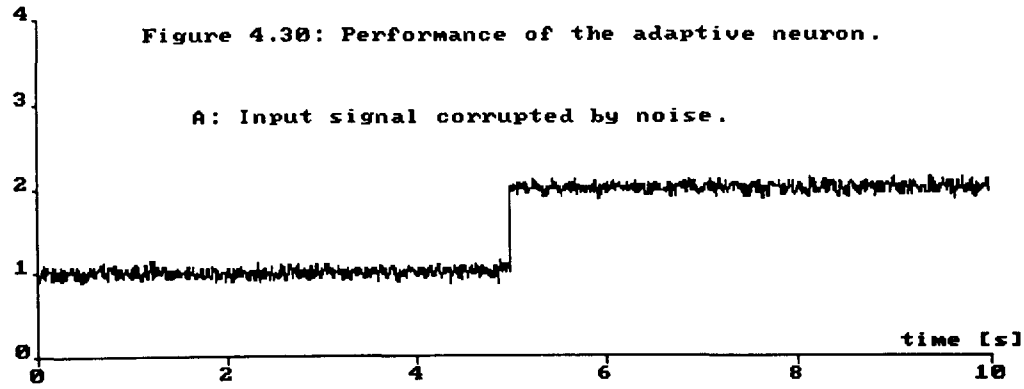
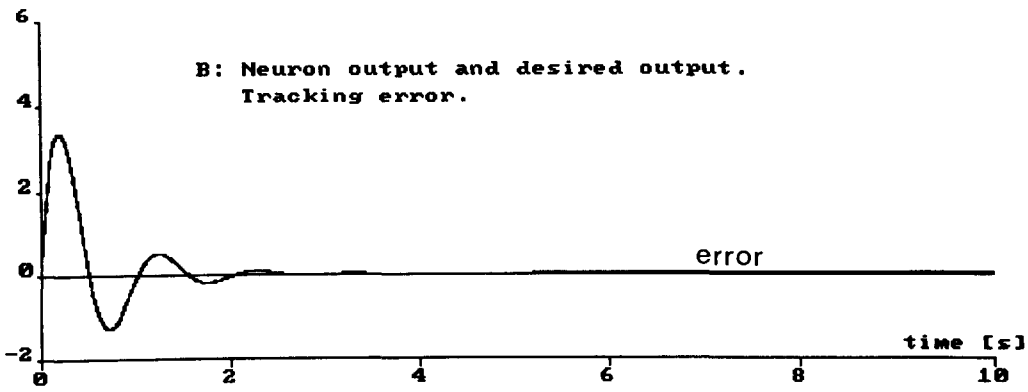


Figure 4.30: Performance of the adaptive neuron.



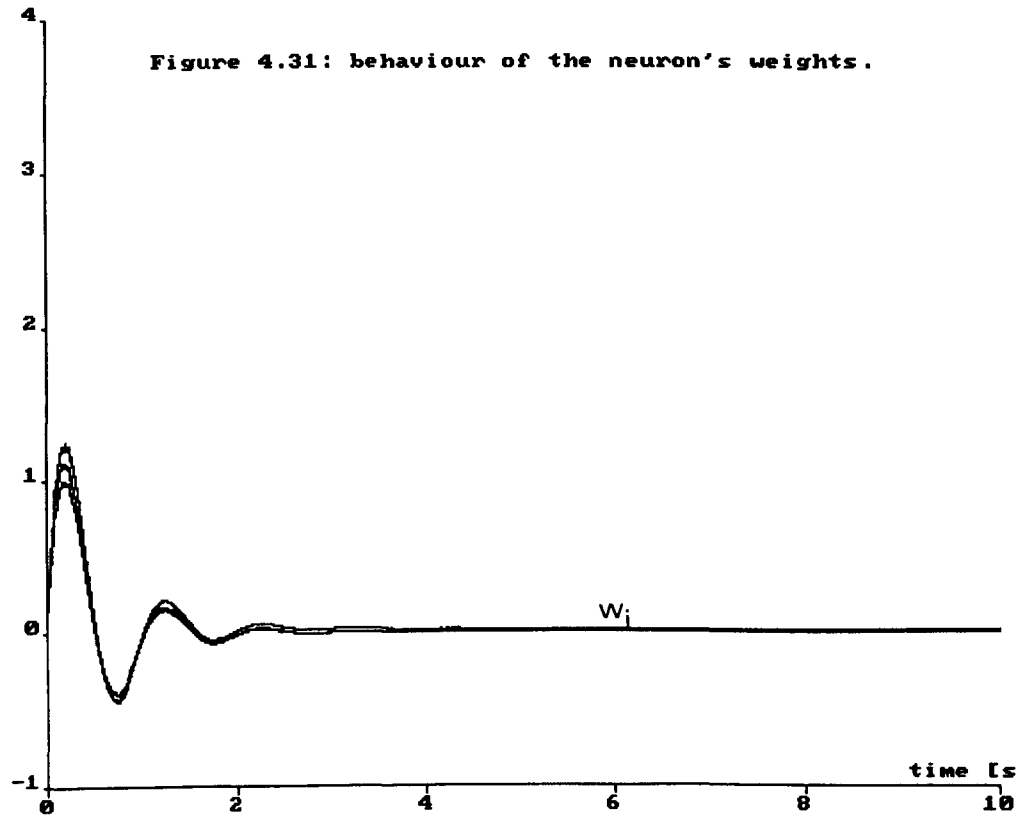
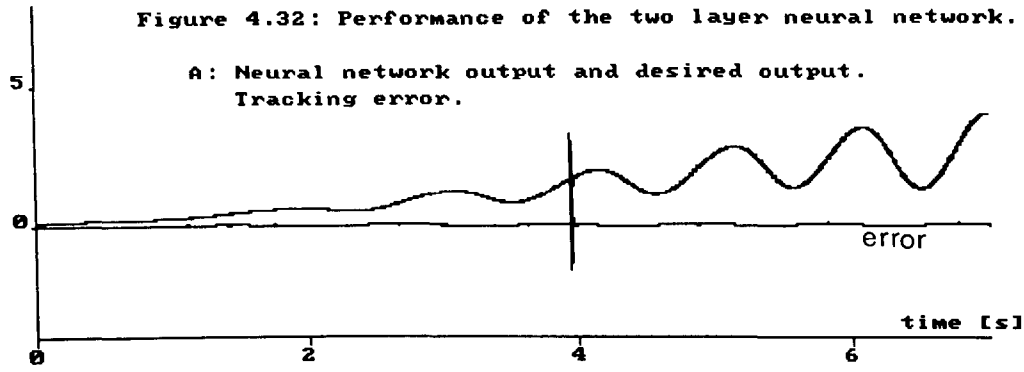


Figure 4.31: behaviour of the neuron's weights.

Figure 4.32: Performance of the two layer neural network.



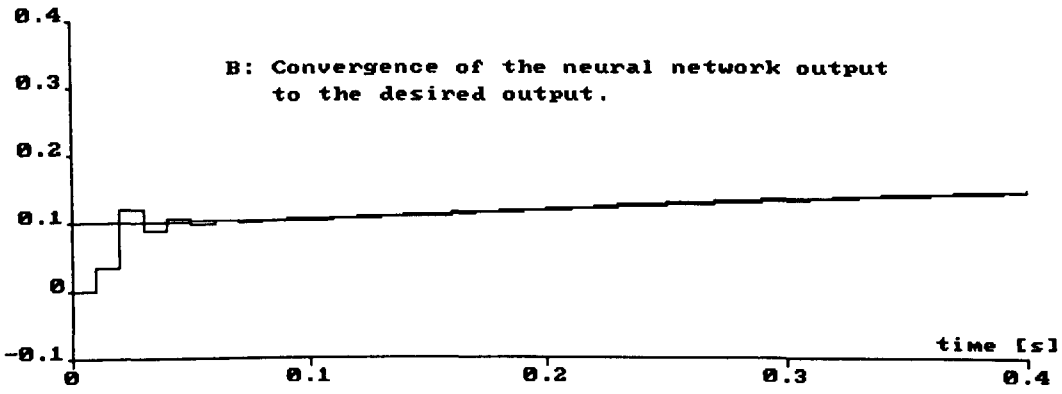
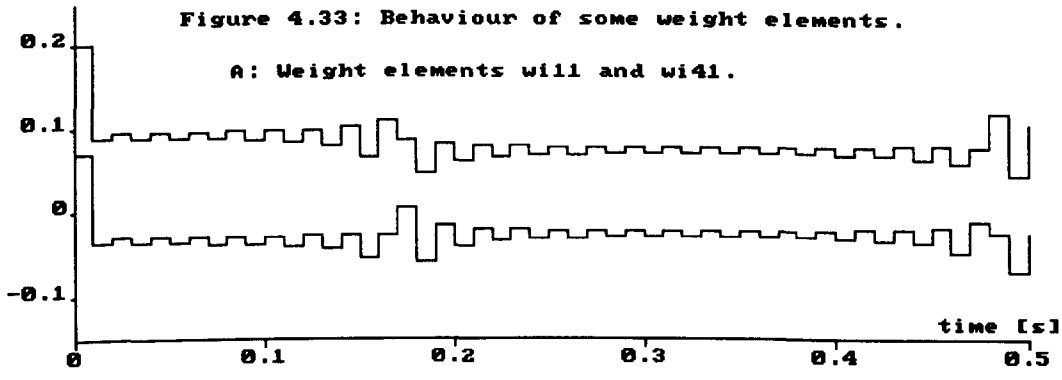
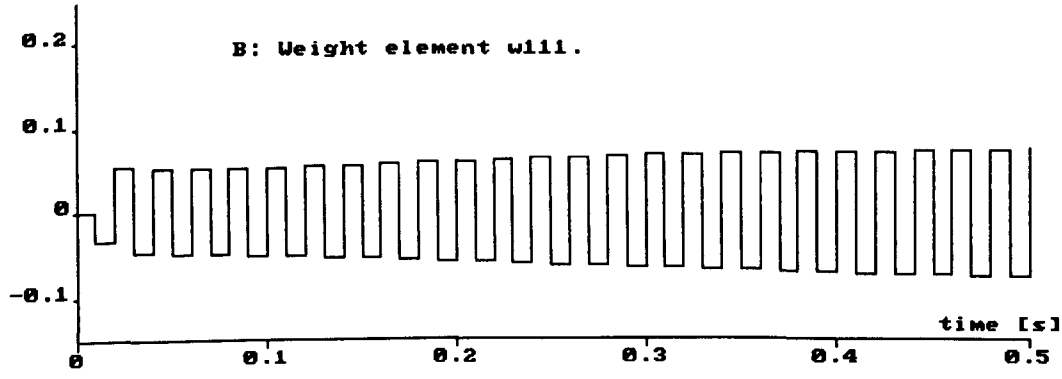


Figure 4.33: Behaviour of some weight elements.





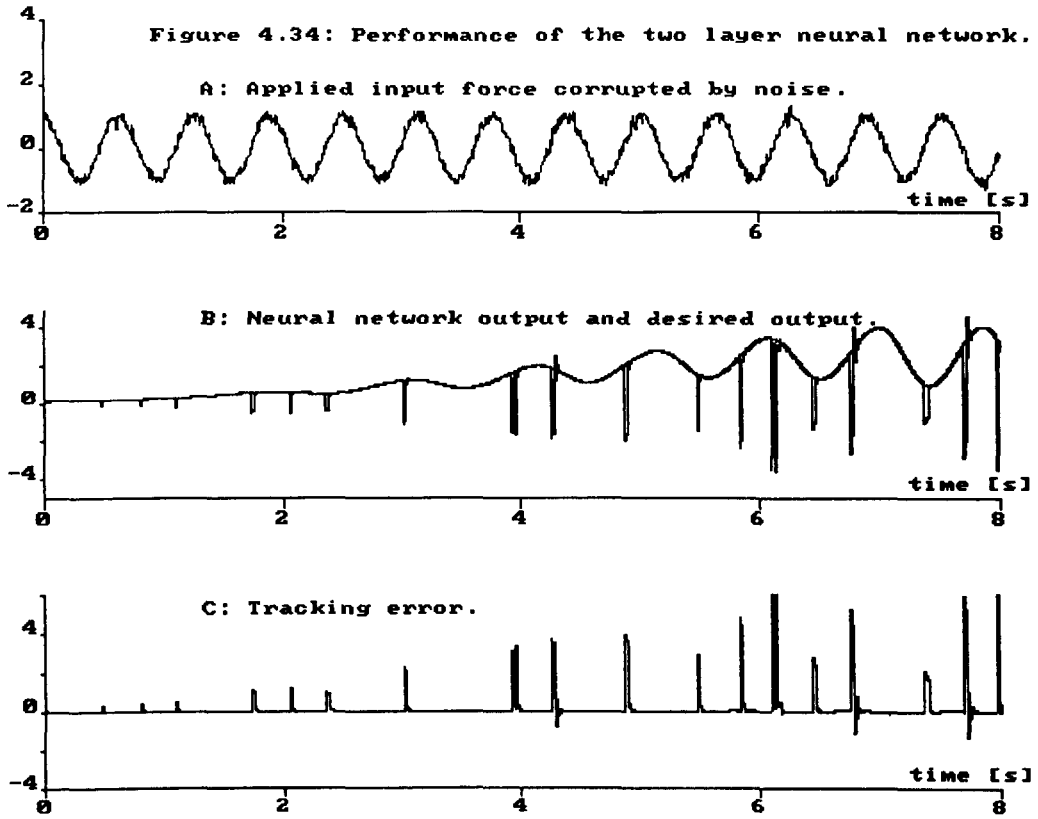


Figure 4.34: Performance of the two layer neural network.

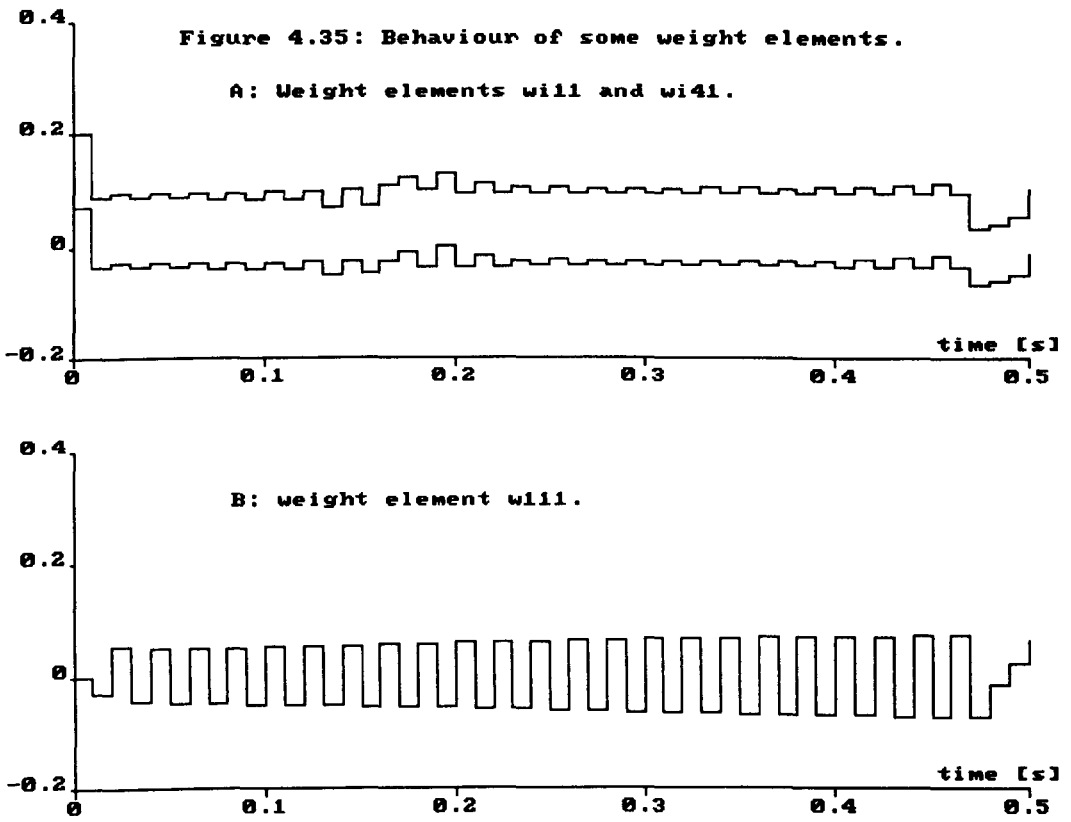


Figure 4.35: Behaviour of some weight elements.

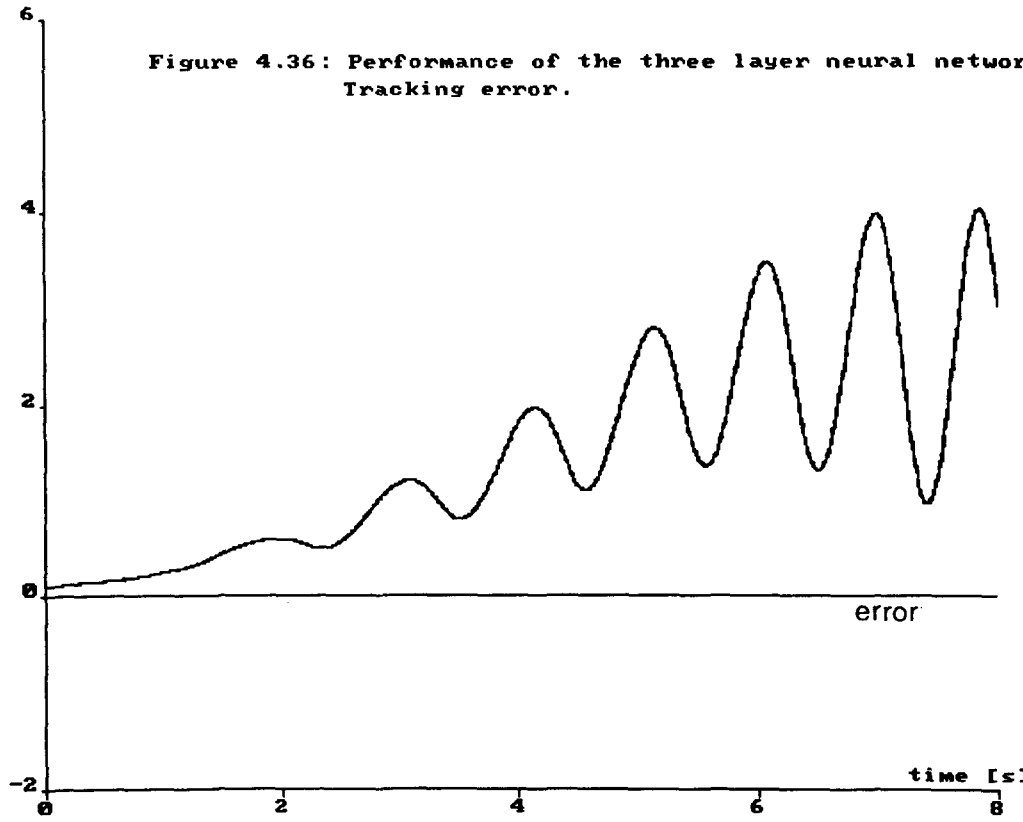


Figure 4.36: Performance of the three layer neural network.
Tracking error.

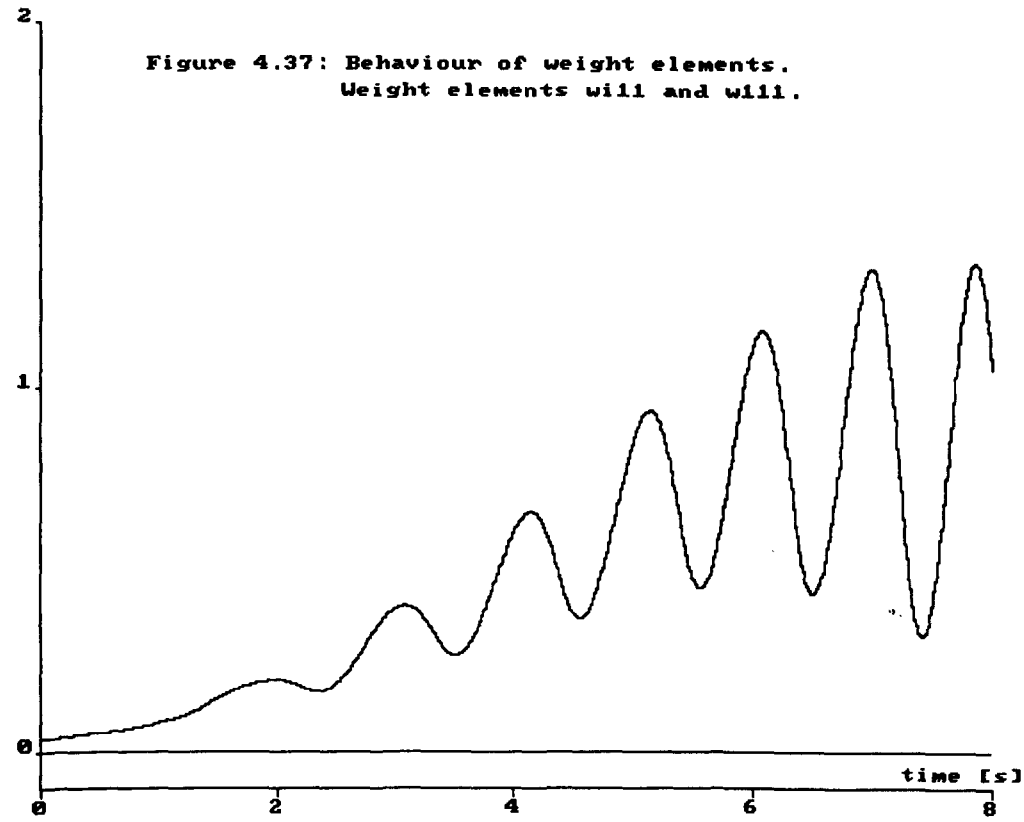


Figure 4.37: Behaviour of weight elements.
Weight elements w11 and w11.

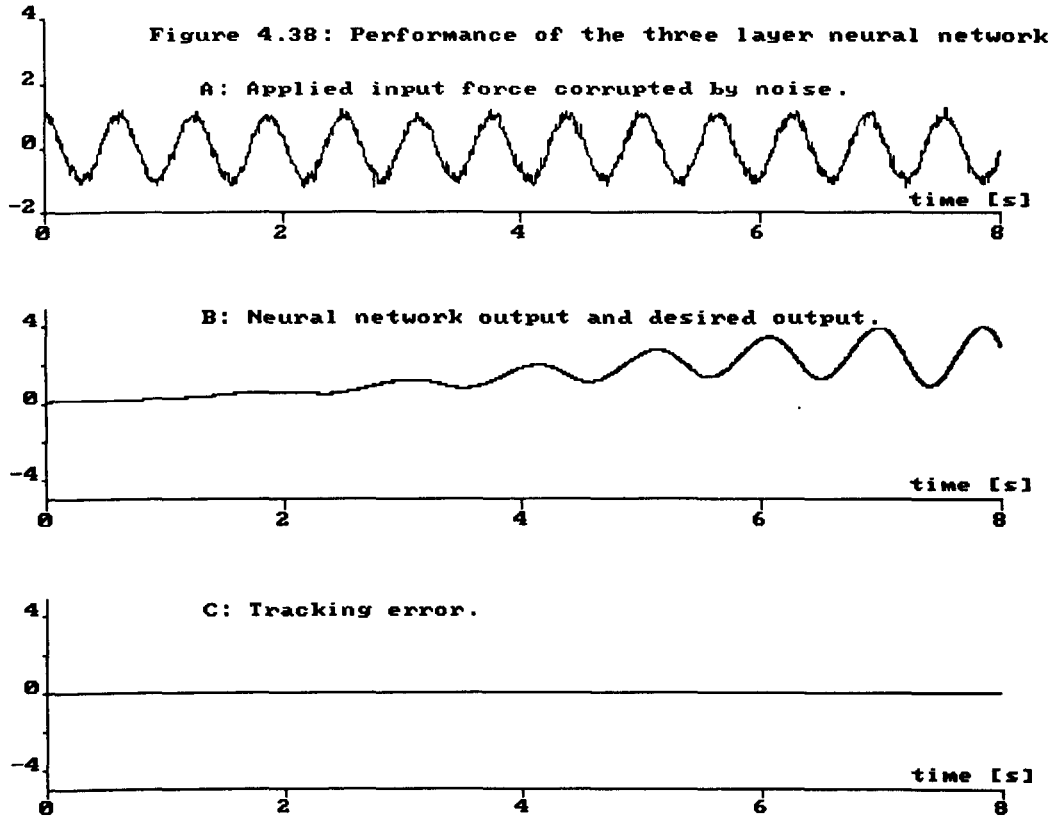


Figure 4.38: Performance of the three layer neural network.

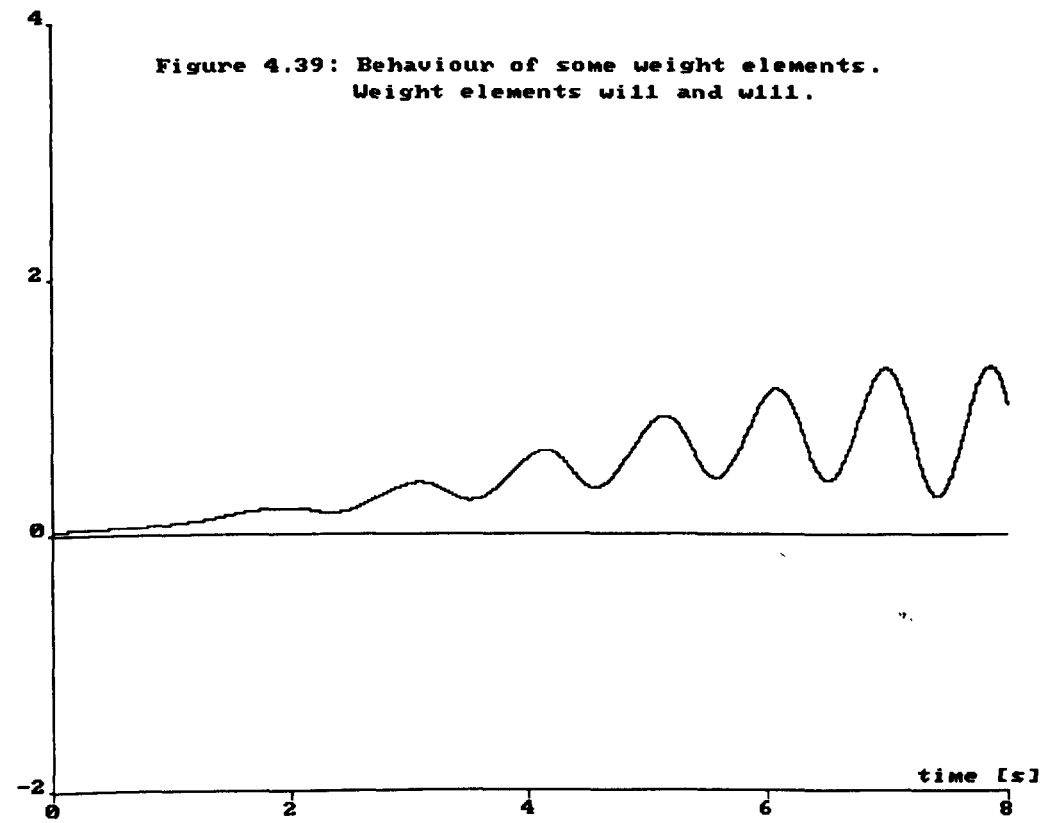


Figure 4.39: Behaviour of some weight elements.
Weight elements w11 and w11.

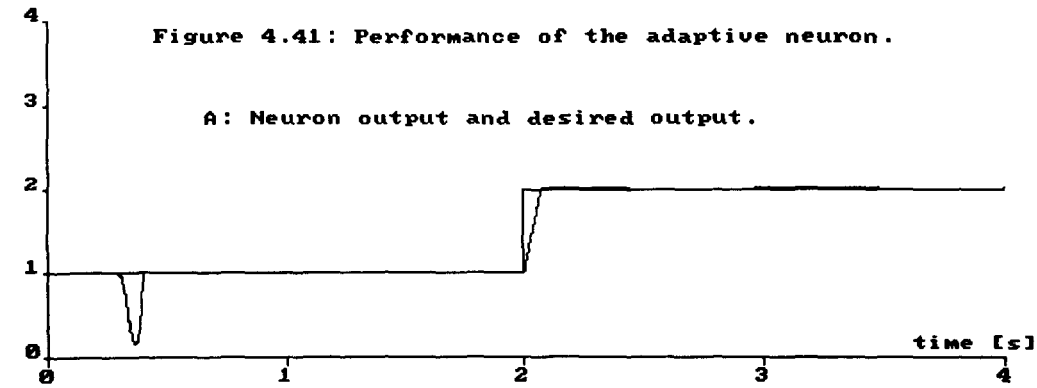
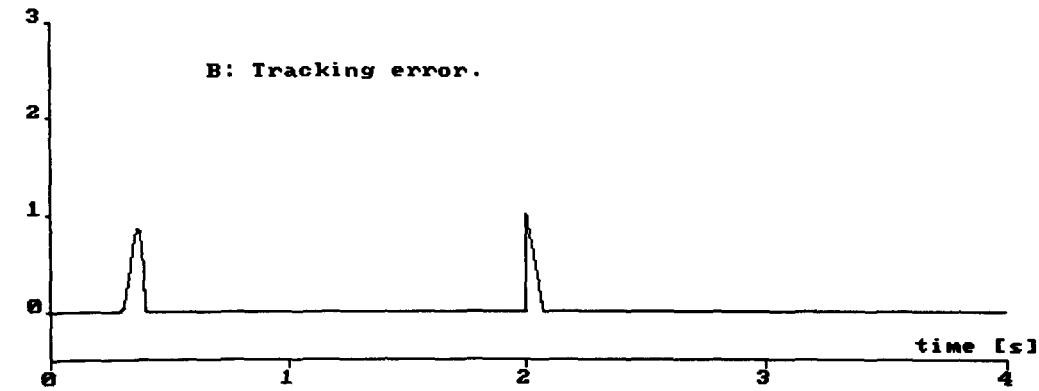


Figure 4.41: Performance of the adaptive neuron.

A: Neuron output and desired output.



B: Tracking error.

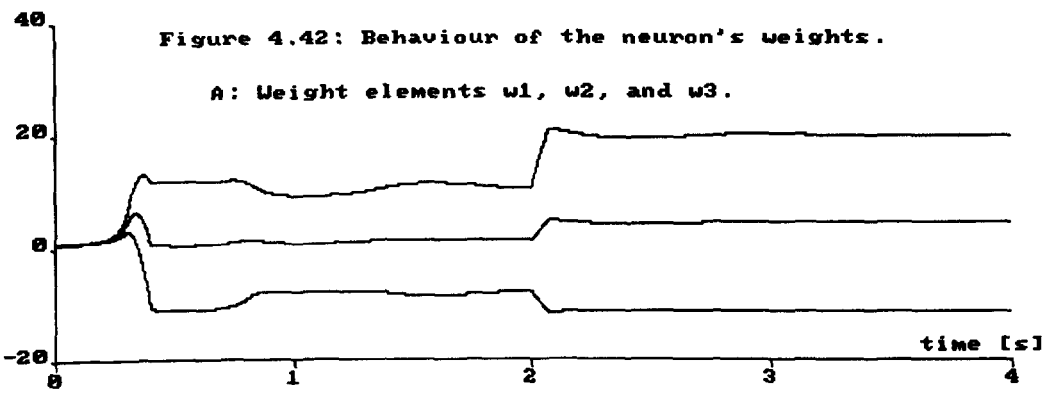
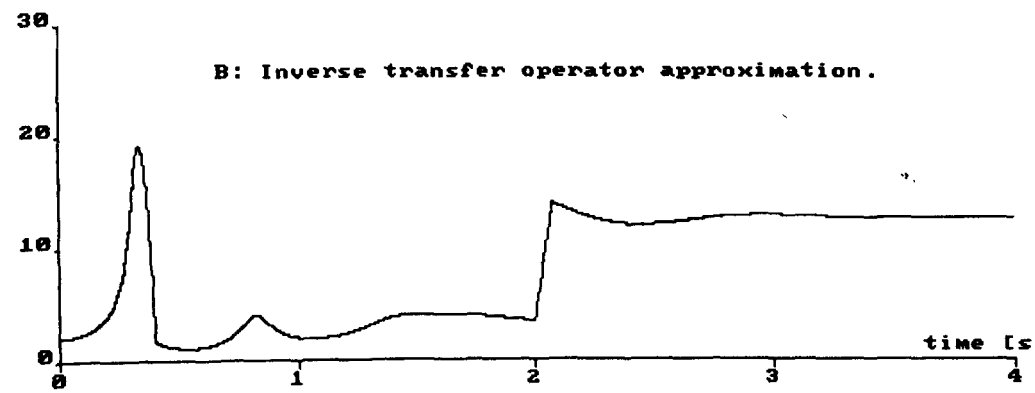


Figure 4.42: Behaviour of the neuron's weights.

A: Weight elements w1, w2, and w3.



B: Inverse transfer operator approximation.

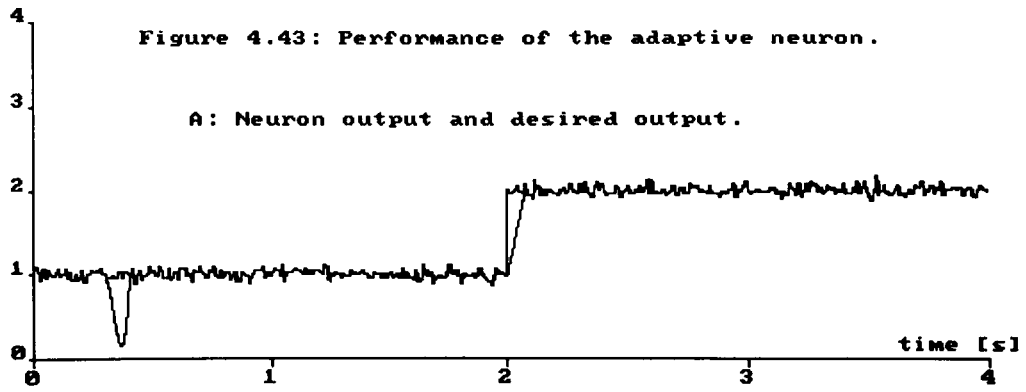
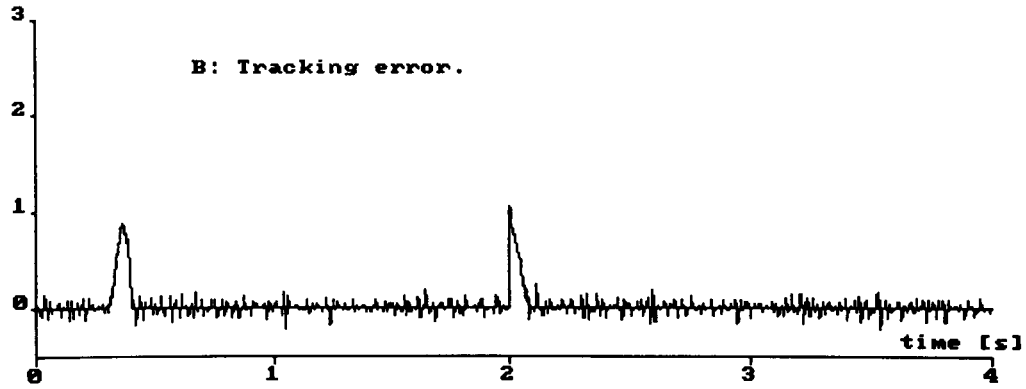


Figure 4.43: Performance of the adaptive neuron.



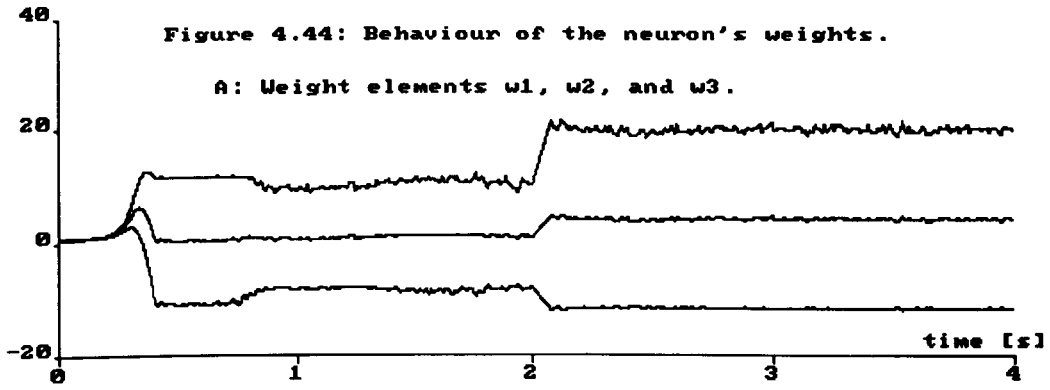
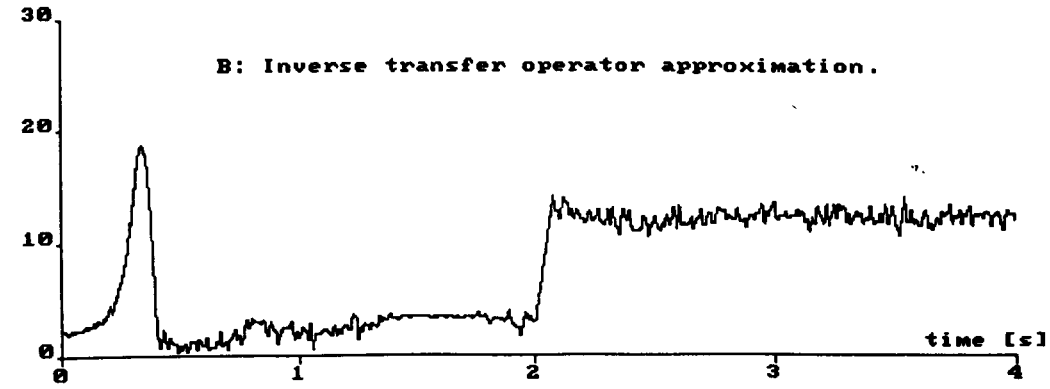


Figure 4.44: Behaviour of the neuron's weights.



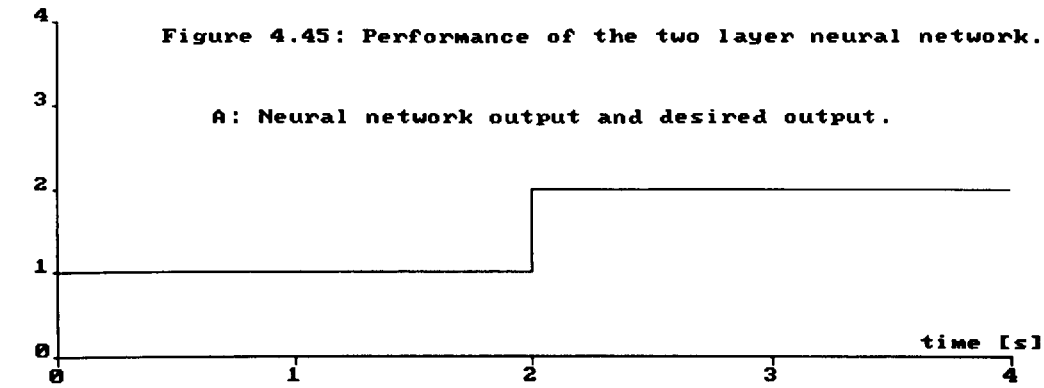
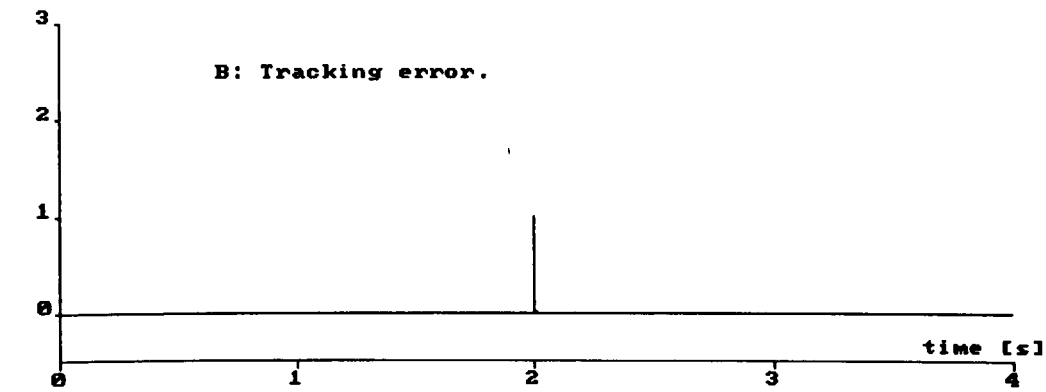


Figure 4.45: Performance of the two layer neural network.

A: Neural network output and desired output.



B: Tracking error.

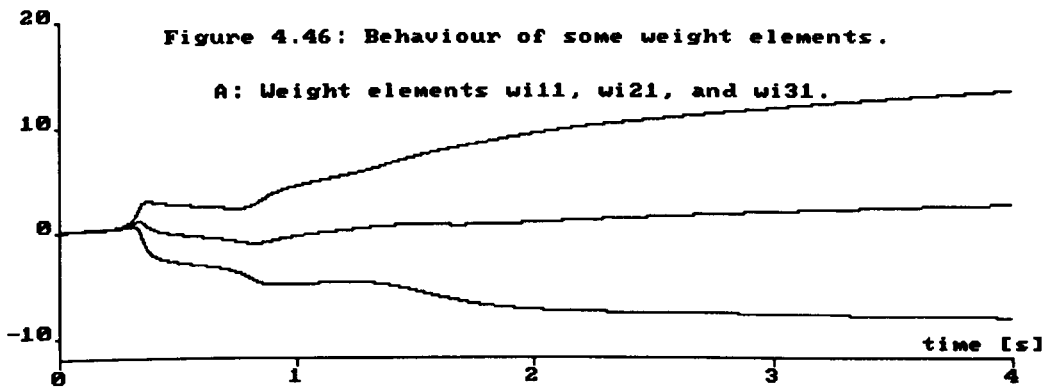
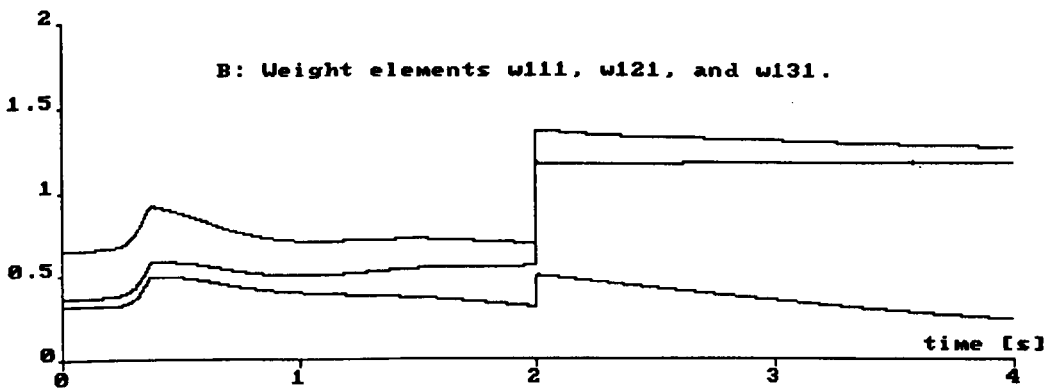


Figure 4.46: Behaviour of some weight elements.

A: Weight elements wi11, wi21, and wi31.



B: Weight elements wi11, wi21, and wi31.

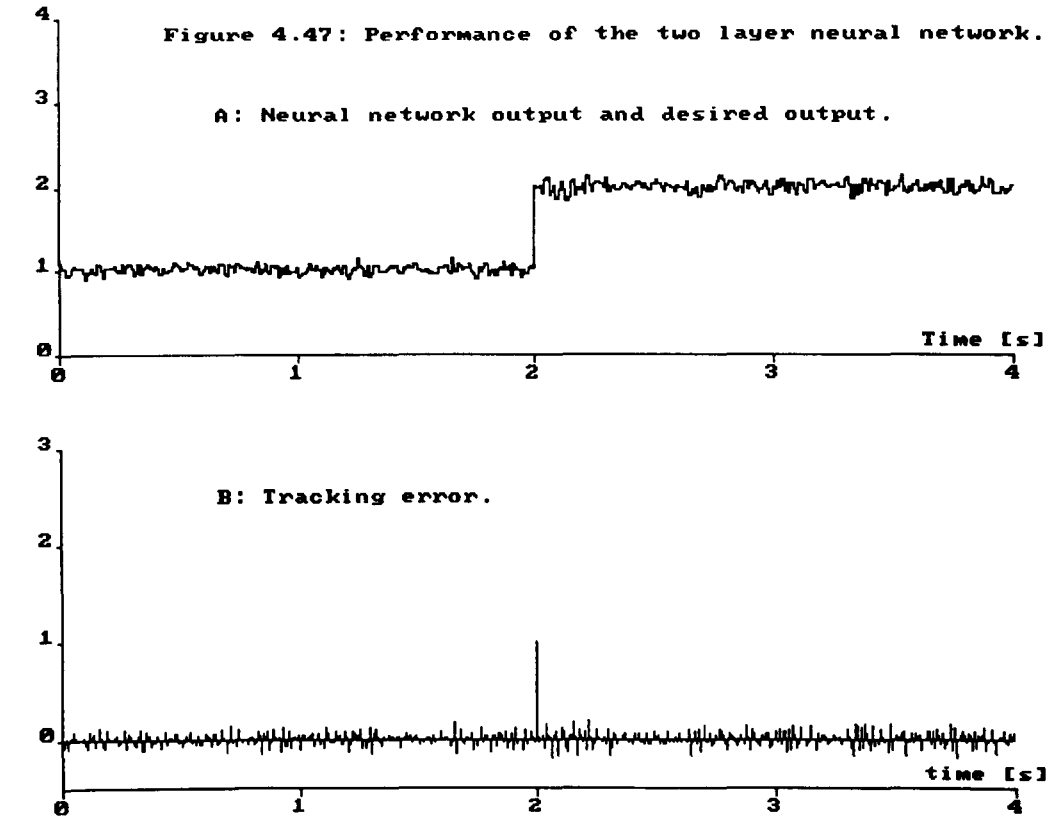


Figure 4.47: Performance of the two layer neural network.

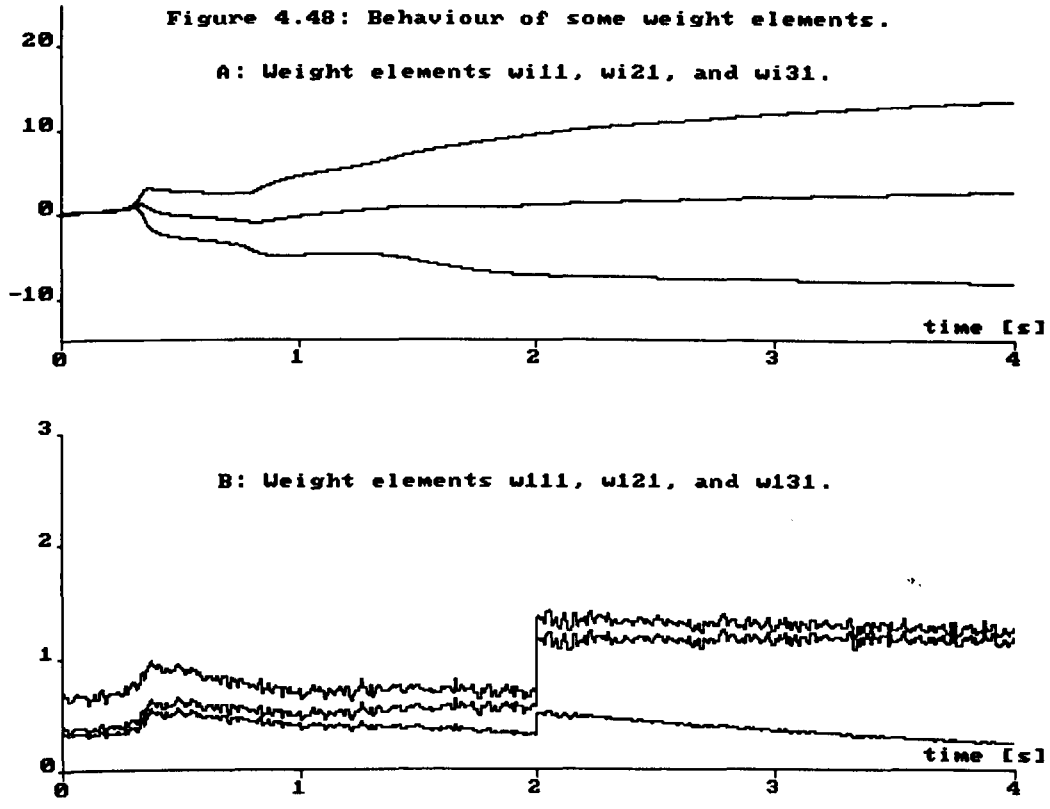


Figure 4.48: Behaviour of some weight elements.

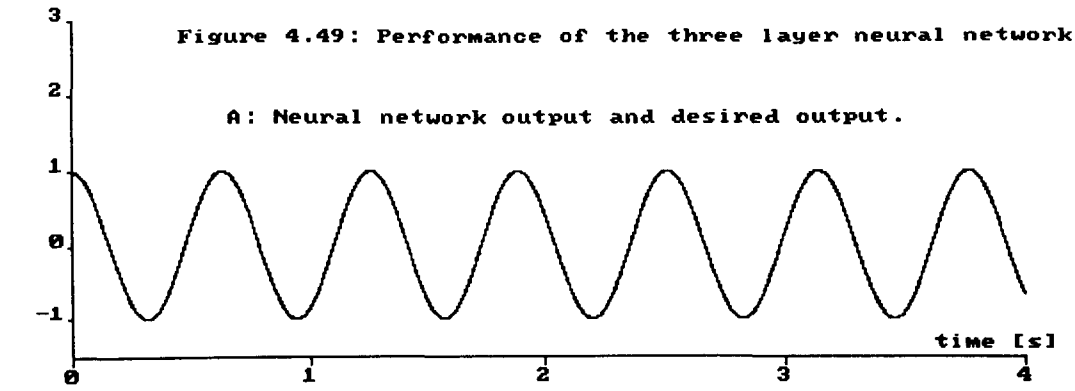
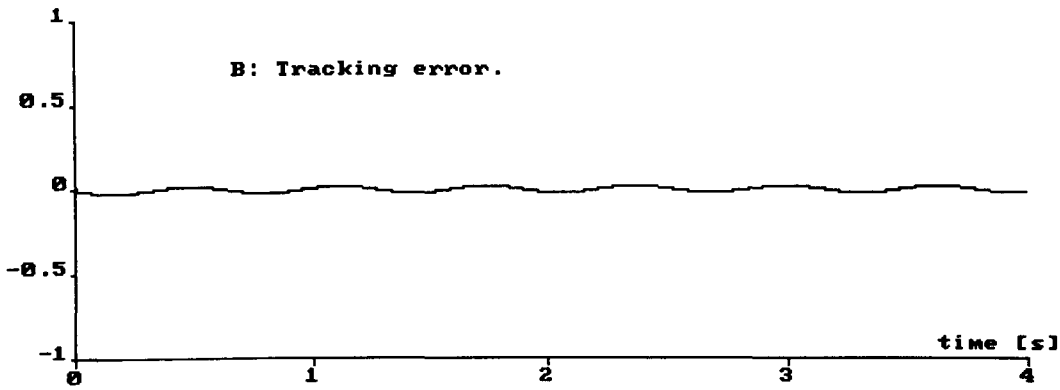
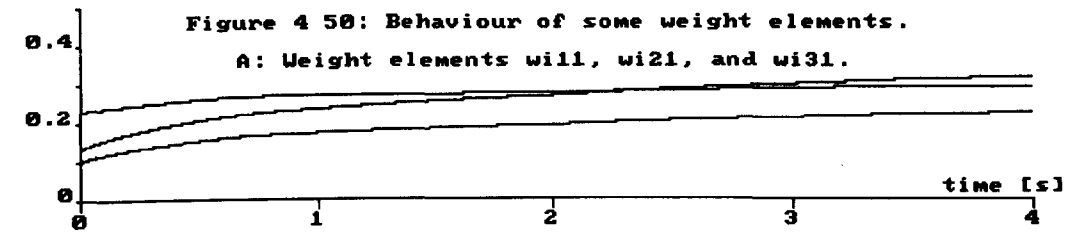
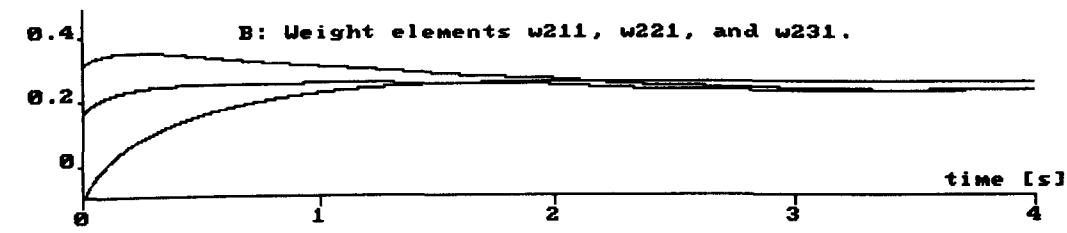


Figure 4.49: Performance of the three layer neural network.







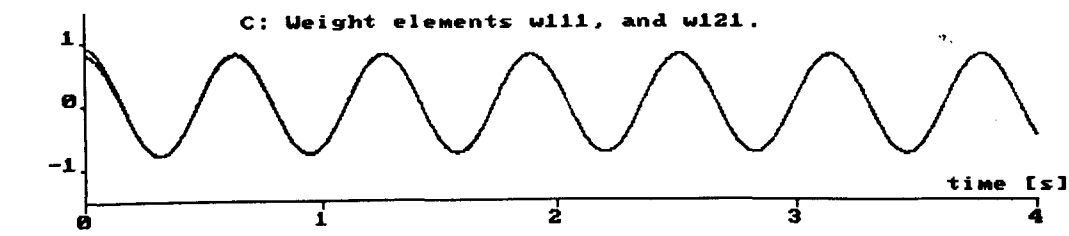


Figure 4 50: Behaviour of some weight elements.

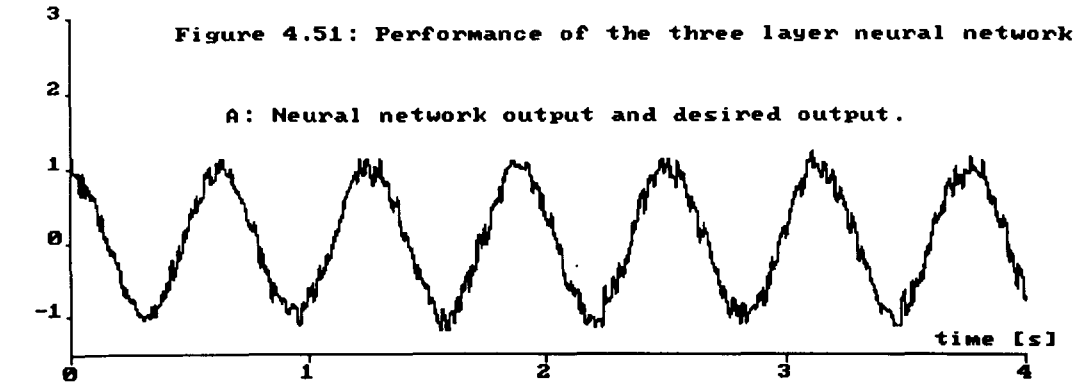
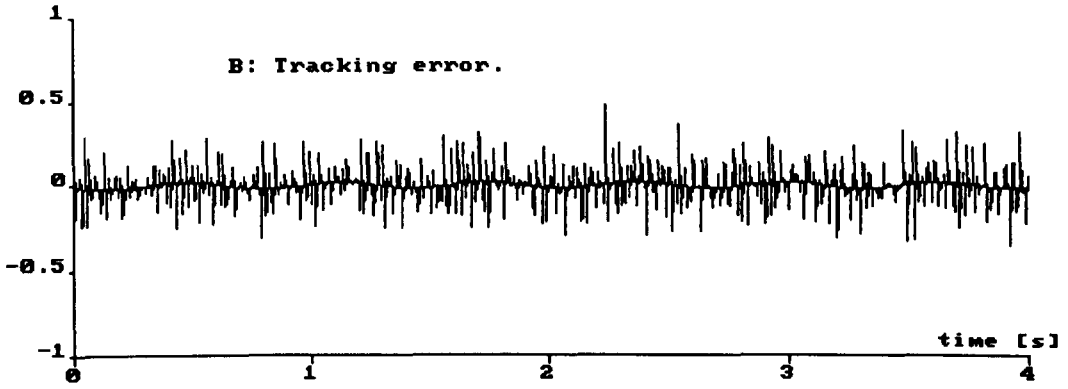


Figure 4.51: Performance of the three layer neural network.



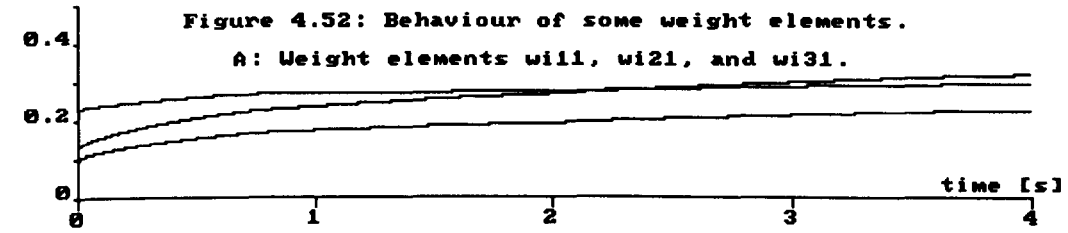
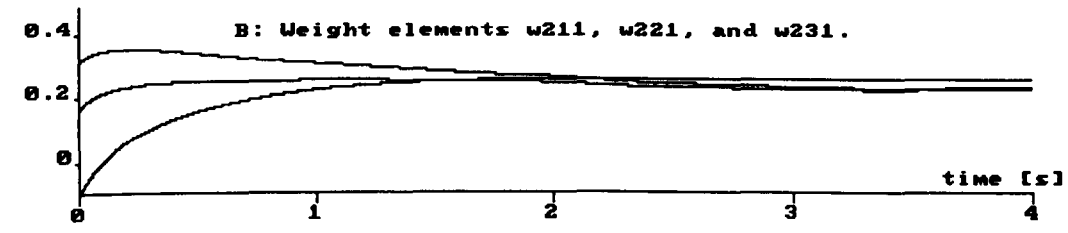
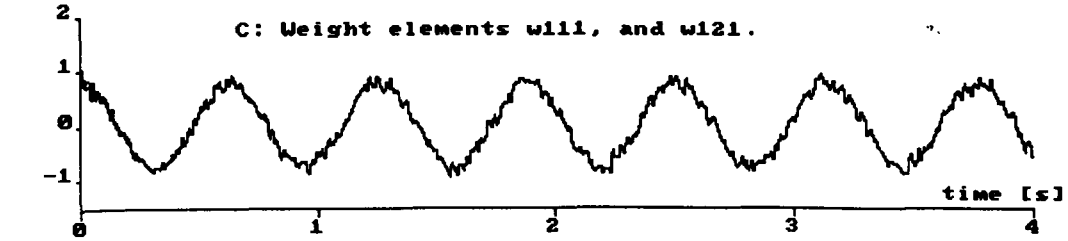


Figure 4.52: Behaviour of some weight elements.





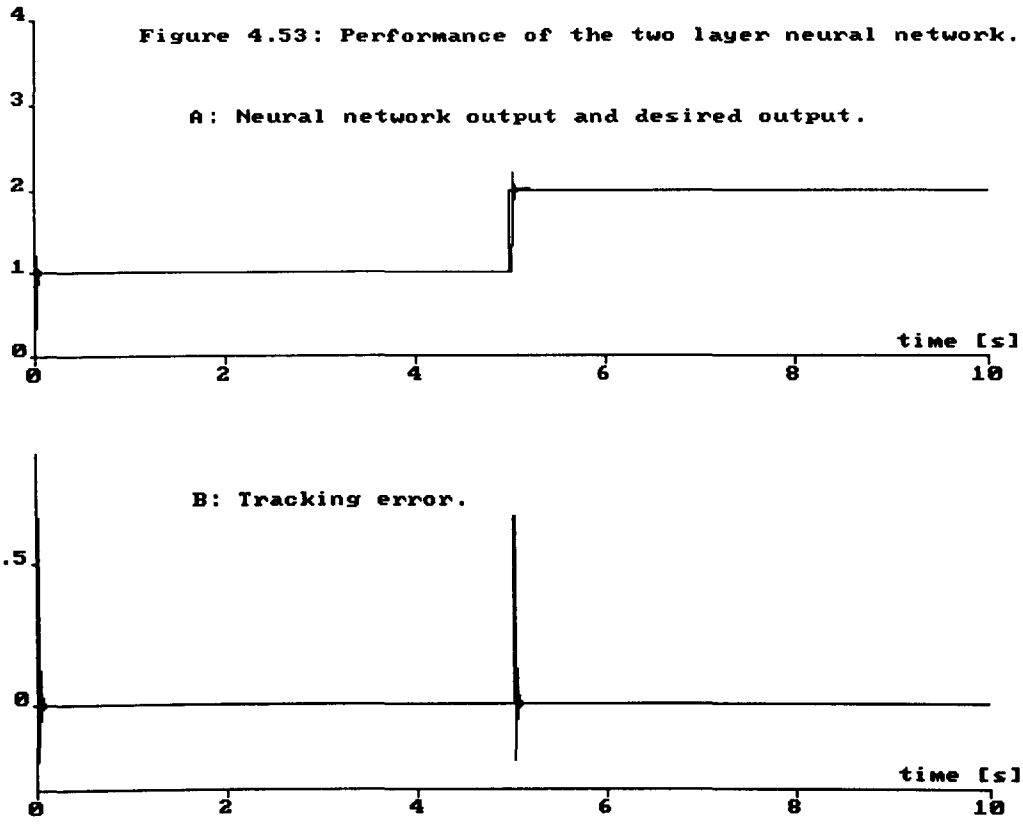


Figure 4.53: Performance of the two layer neural network.

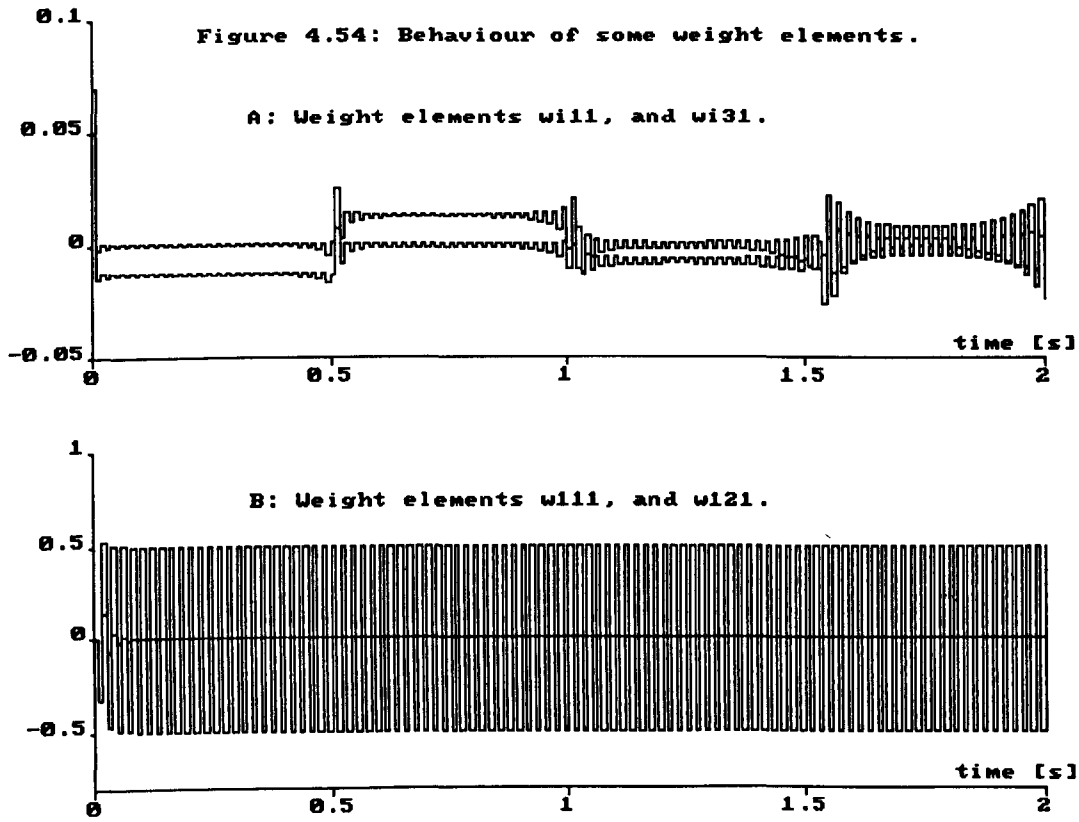


Figure 4.54: Behaviour of some weight elements.

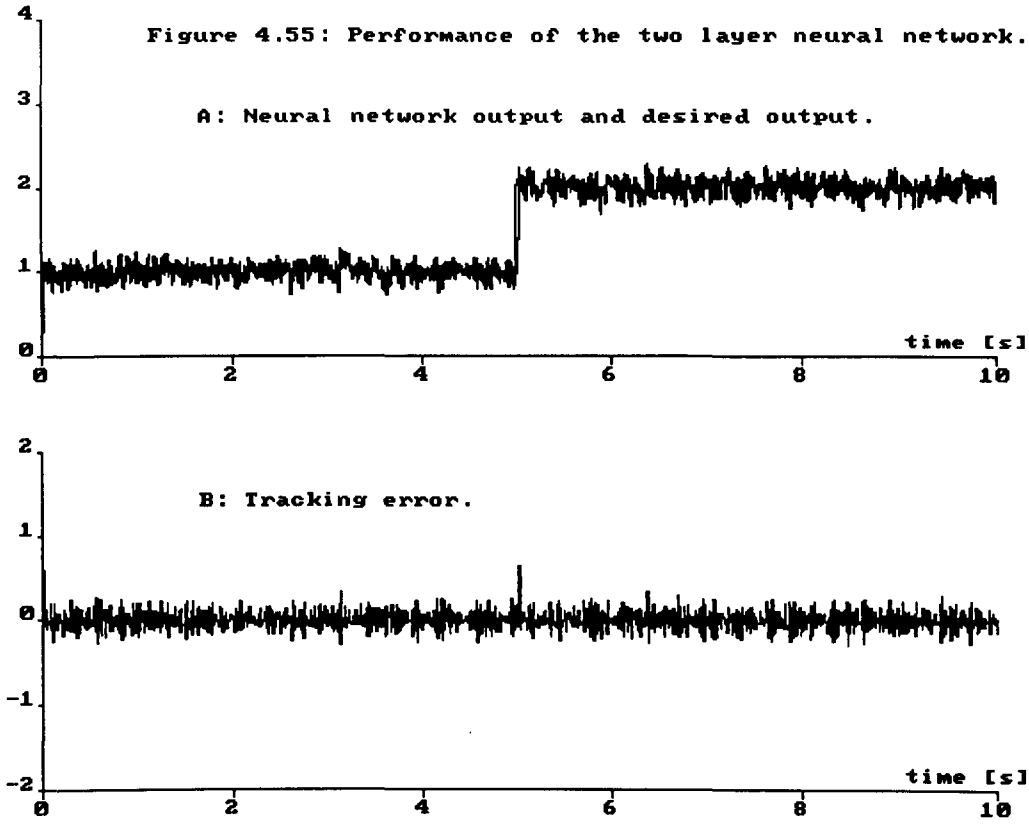


Figure 4.55: Performance of the two layer neural network.

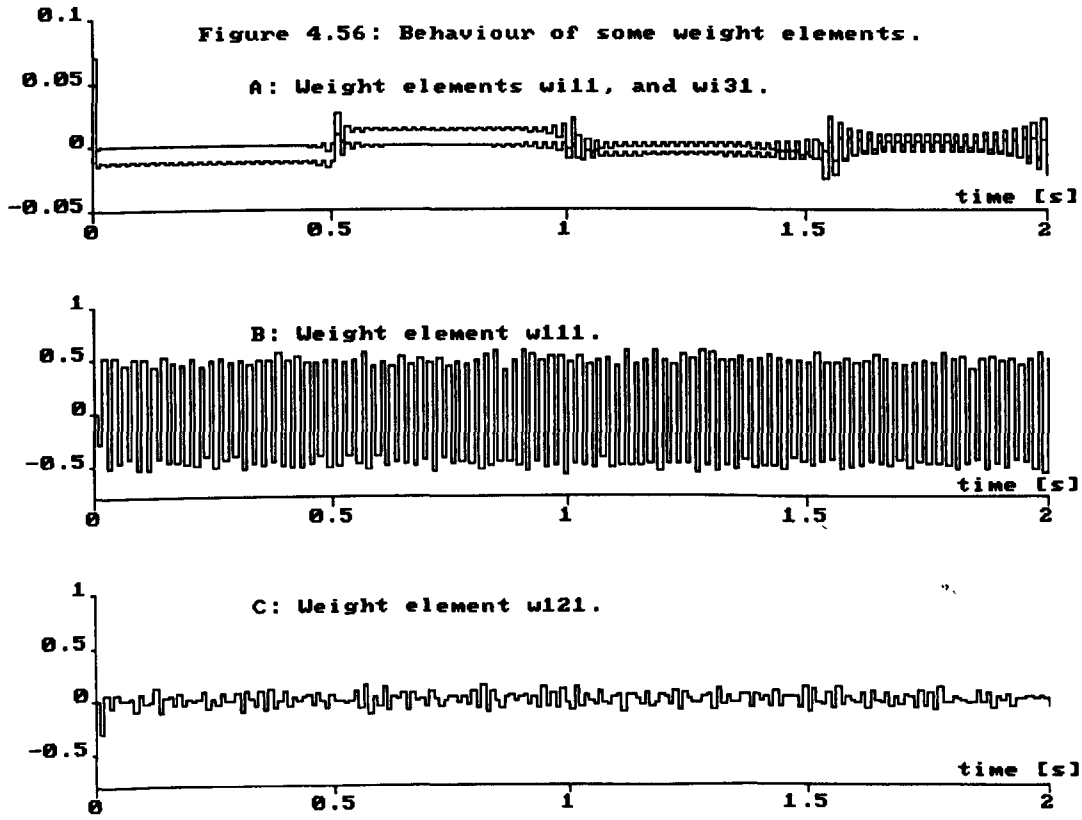


Figure 4.56: Behaviour of some weight elements.

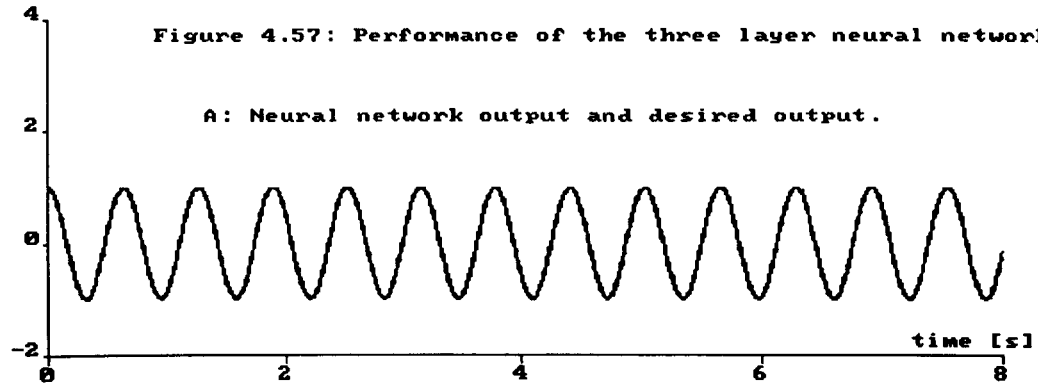
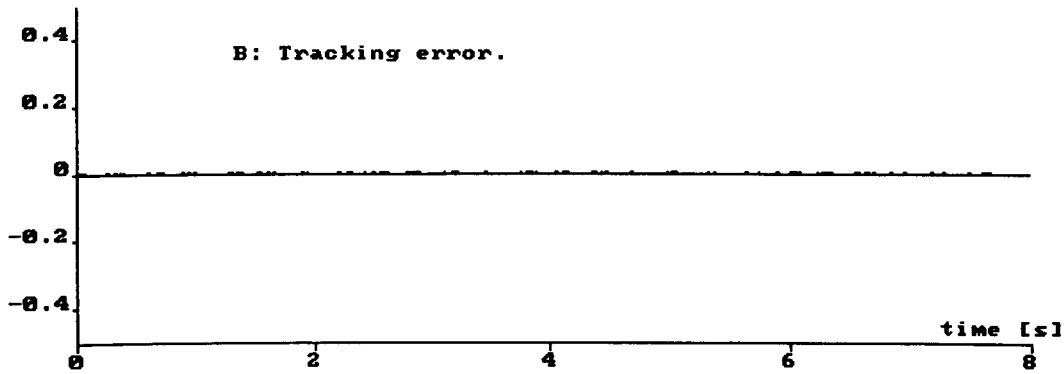


Figure 4.57: Performance of the three layer neural networks.



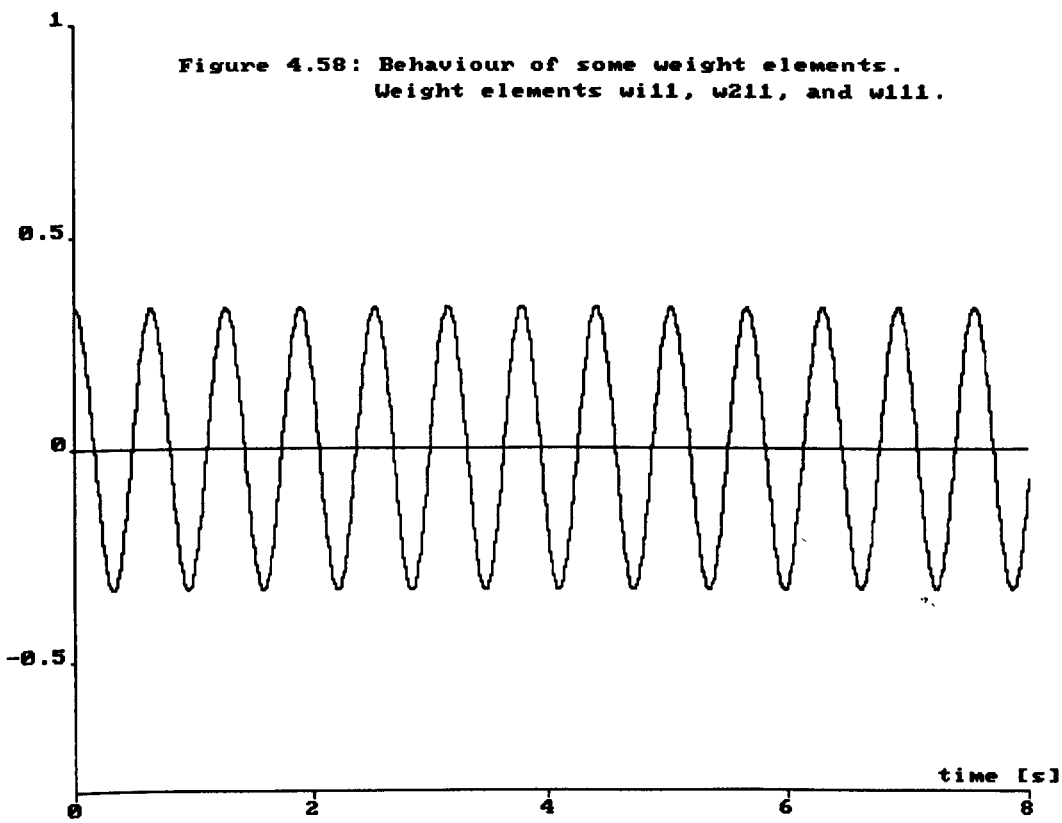


Figure 4.58: Behaviour of some weight elements.
Weight elements w111, w211, and w111.

Figure 4.59: Performance of the three layer neural network.

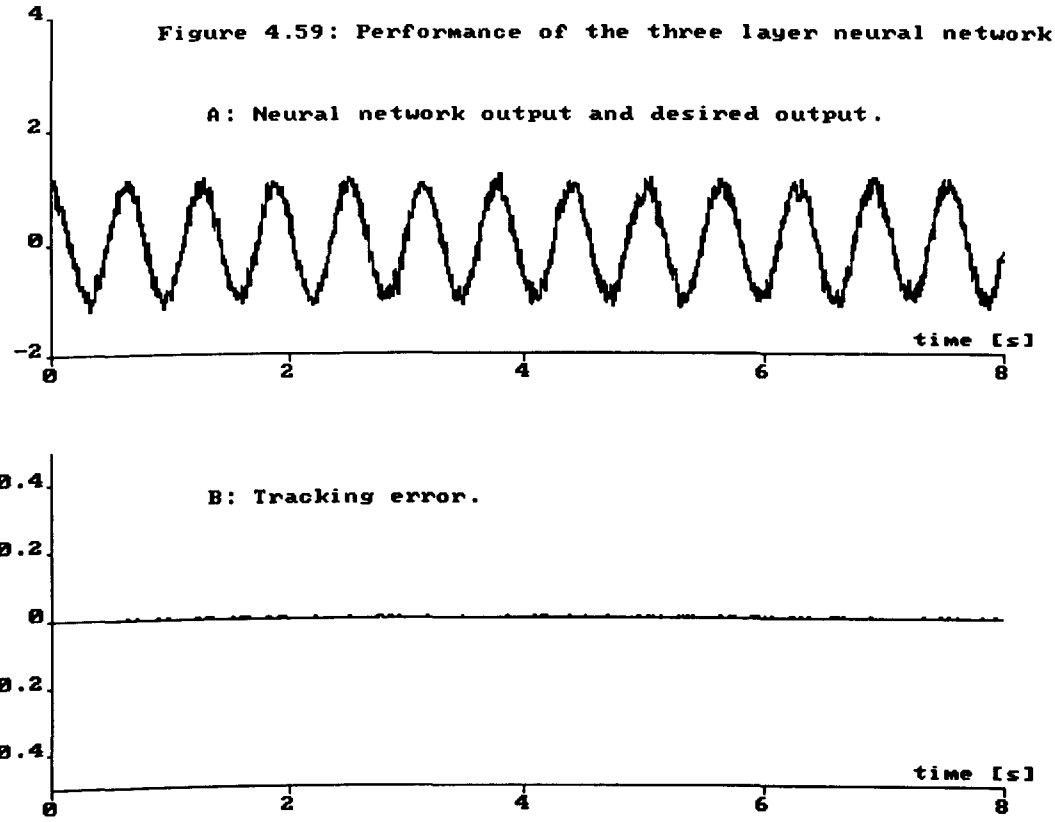
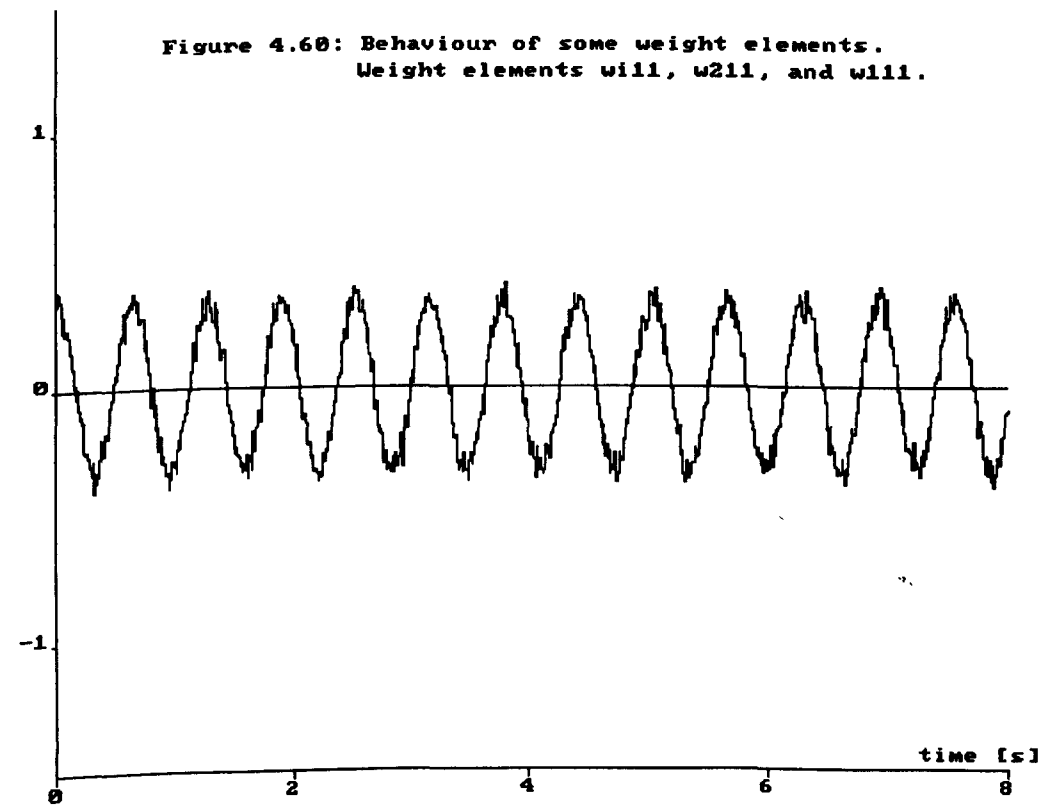


Figure 4.60: Behaviour of some weight elements.
Weight elements wi11, w211, and wi11.



Chapter 5

INVERSE MODEL NEURAL NETWORK-BASED CONTROL OF DYNAMIC SYSTEMS

5.1 INTRODUCTION

Commonly, the first stage for the neural network-based control of an unknown dynamic system is the development of an accurate neural network representation of the plant under consideration. Obtaining such a network representation or model involves a training phase where the neural network is presented with a set of previously collected input-output data of the system operation. In some special cases however, the training phase can be accomplished on-line by connecting the neural network model in an open-loop configuration with the unknown plant which is excited with a set of selected inputs in order to measure the corresponding outputs [66, 67]. Once the quality of the network representation of the unknown dynamic system is guaranteed, a neural network controller may be trained using data provided by the neural network representation. In some cases, the controller is not a neural network, but its design incorporates information provided by a neural network representation of the system. It is after the neural network representation and the controller are placed together with the unknown plant, when on-line control actions obtained from the controller can be applied to drive the plant. A major difficulty of this indirect off-line neural network-based control method for dynamic systems is to guarantee convergence of the network outputs to values of the response of the dynamic system for which the network was not previously trained. This drawback, known in the neural network literature as the generalization problem [68] may have adverse effects on

the quality of the control system. It is possible however, to continue slowly training the neural network representation and the controller after the off-line training phase has been completed. In this case, the neural network-based control system is capable of providing excellent results [15].Here however, the difficulty arises in determining when to stop the off-line training phase, and how slowly adaptation must proceed in order to guarantee stability [69].

In contrast to the indirect off-line neural network-based control methods just described, there are direct on-line neural network-based control methods for dynamic systems where identification and control are simultaneously carried out [60, 70, 71]. In these on-line methods, since learning takes place based upon the current input-ouput values of the process operation, the generalization issue mentioned before is no longer a problem. Here, the learning algorithms used to adapt the weights of the network must be able to cope with time-varying behaviour of the dynamic system being controlled, and the control system must possess an interaction mechanism between the estimated value of the unknown parameters or dynamics and the generated control action used to stabilize the system. Obviously, this simultaneous interaction between estimation and control produces a high dimensional closed-loop system for which the stability analysis is far more complex than that used for the learning algorithms and the process by themselves.

In this chapter, some of the variable structure control-based learning algorithms proposed in chapter 3, and the scheme for estimating the ITO for dynamic systems shown in figure 5.1 are incorporated into a control scheme for dealing with tracking problems for some linear and a class of nonlinear dynamic systems. The incorporation of approximated estimation of the FTO's or of the ITO's into a control scheme design raises important considerations regarding the effects that these approximated estimations may have on the quality of the behaviour of the control system. If the estimation error produced by the neural network on-line determination of the FTO's or of the ITO's of the dynamic system is small, it should be expected that the control scheme is able to keep controlled the behaviour of the plant under scrutiny. A similar situation, in the context of indirect off-line neural network-based control methods has been studied by Levin and Narendra in a recently published paper [72]. In their paper, they used the concepts of stability under perturbation and strong stability under perturbation of dynamic systems, [73], to clarify the effects of such errors of estimation.

Other important considerations to be taken into account in the design of our neural network-based control scheme are related to the time it takes for the network output to converge to the desired network output, and to provide suitable values of

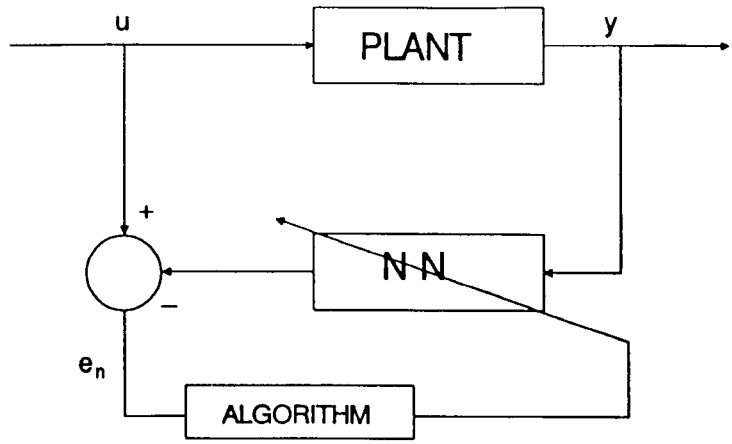


Figure 5.1: Scheme for ITO identification.

the approximated ITO of the dynamic system being controlled. This convergence time, although small, as shown by the examples in the previous chapter, may introduce and undesirable transient behaviour in the controlled system output. The following section presents a neural network-based control scheme that takes into account the before mentioned consideration. This scheme represents an implementation of the direct inverse dynamics control method found in [7, 16].

5.2 DIRECT ITO CONTROL SYSTEM

It is commonly acknowledged that the lack of robustness of the neural network-based direct inverse dynamics control schemes is due to the absence of feedback signals from the output or state variables of the process [61]. In such a case, the control system cannot cope with variation of the neural network estimation of the ITO when the signals used to drive the network on-line are different from those used to train the network off-line, (ie. the generalization problem). This lack of robustness can be overcome by using a neural network with on-line learning capabilities of the ITO of the unknown system, and a feedback path of the output of the system.

Figure 5.2 shows a neural network-based direct inverse dynamic control scheme that incorporates an on-line neural network estimator "NN" of the ITO of the unknown system, and a feedback path of the system output. In this control scheme, the estimator is a single layer neural network as the one shown in figure 5.3, and the

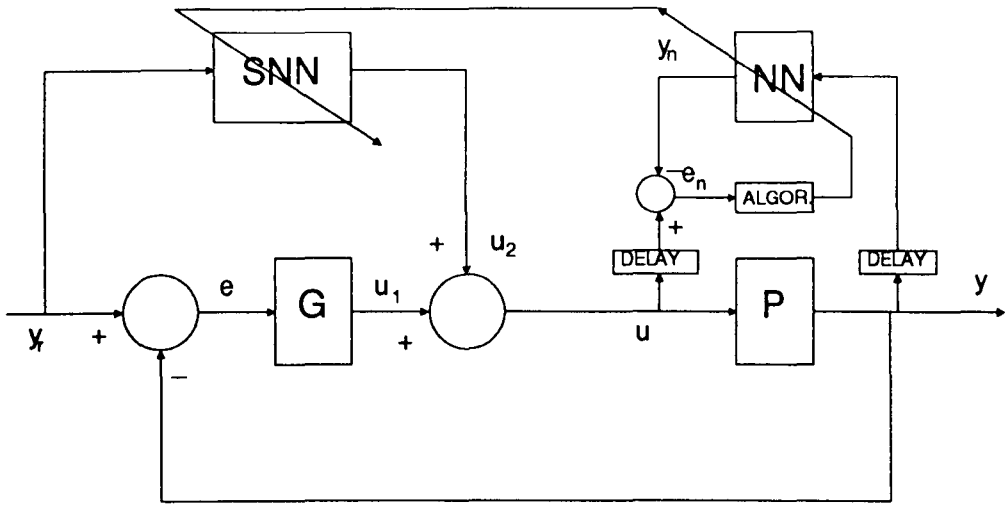


Figure 5.2: Direct inverse dynamics control scheme.

overall controller is composed of the two following elements:

- A feedforward controller designed with a "slave neural network", "SNN", with input signal equal to the desired plant output or setpoint, and output equal to the product of the approximated on-line estimation of the ITO and the desired plant output or setpoint. Notice from figure 5.2 that the weights of the slave neural network "SNN" are adapted by the same algorithm used to adapt the weights of the on-line neural network estimator "NN".

- A feedback path that includes a filter "G" whose design will be explained later.

The task of the overall controller is to produce an identity map between the desired plant output or setpoint and the actual plant output, and to generate an appropriate control action that enables the system output to be driven according to a reference signal or setpoint. Notice that in the proposed control scheme both the values of the generated system input and system output are used as the desired on-line neural network output and neural network input, respectively. The following section presents a mathematical formulation that explains the operation of the proposed control scheme for some continuous-time unknown linear dynamic systems.

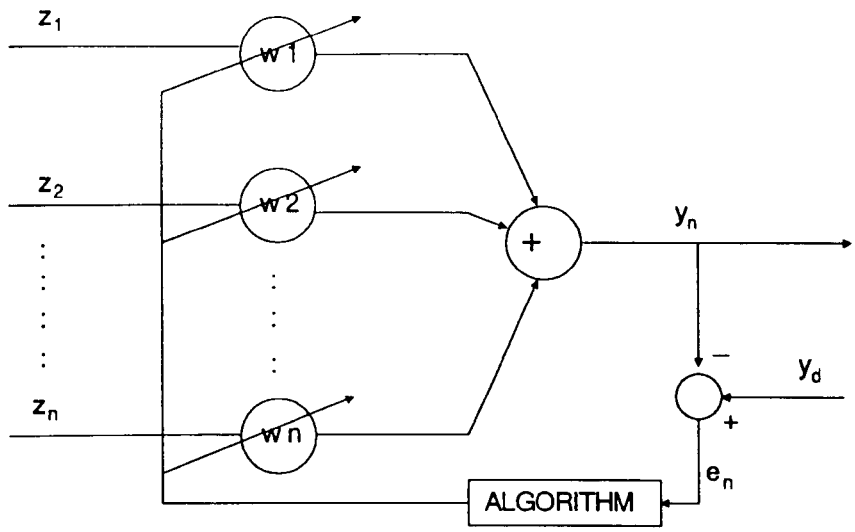


Figure 5.3: Single layer neural network.

5.3 CONTINUOUS-TIME LINEAR SYSTEMS CASE

Consider a controllable unknown linear system represented by the following mathematical model,

$$(D^n + a_1 D^{n-1} + ... + a_{n-1} D + a_n) y(t) = (b_0 D^m + b_1 D^{m-1} + ... + b_{m-1} D + b_m) u(t) \quad (5.1)$$

where $n \geq m$, $D = \frac{d}{dt}$ is the operator differentiation, $u(t) \in \Re$ and $y(t) \in \Re$ are the input and output of the unknown system, respectively; and "a_i" and "b_j", i=1,2,...,n; j=0,1,...,m, are real valued unknown parameters.

The ITO between the signals "$u(t)$" and "$y(t)$" may be symbolically expressed by the equation,

$$\frac{u(t)}{y(t)} = \frac{D^n + a_1 D^{n-1} + ... + a_{n-1} D + a_n}{b_0 D^m + b_1 D^{m-1} + ... + b_{m-1} D + b_m} \quad (5.2)$$

Notice that in order for the weights of "NN" to represent an approximation of the ITO of the unknown linear system, the behaviour of equation 5.2 must be bounded. Failing to satisfy this condition would imply that when the approximation of the ITO estimation is being performed, the weights of "NN" would grow without bound and therefore, it would not be possible to achieve any suitable approximation of the ITO of the unknown system. In general terms, this condition may be interpreted as the existence of an inverse plant model from the output of the unknown system

to its input [75, 76]. For time-invariant linear systems, the existence of an inverse plant model is related to the concept of output controllability of the system [77]. That is, if the unknown linear system is output controllable, then there exists an inverse system model that enables the control input "$u(t)$" to be obtained from the desired and actual plant output. This assertion can be easily verified by obtaining a state-space representation of equation 5.1 and by using the definition of output controllability for time-invariant linear systems. In other words, the model

$$\dot{X}(t) = A\,X(t) + b\,u(t), \tag{5.3}$$

$$y(t) = c\,X(t), \tag{5.4}$$

with initial condition $X(t_0) = X_0$, constitutes a state space representation of equation 5.1; where $X(t) \in \Re^n$ is a state vector, $u(t) \in \Re$ is the control input, $y(t) \in \Re$ is the system output, and the factors "A", "b", and "c" are matrices of appropriate dimension. The system output can be expressed in terms of the transition matrix "$\Phi(.,.)$" as

$$y(t) = c[\Phi(t,t_0)X_0 + \int_{t_0}^{t} \Phi(t,\tau)bu(\tau)d\tau] \tag{5.5}$$

Let the variable "$\Delta y(t)$" be defined as

$$\Delta y(t) = y(t) - c\Phi(t,t_0)X_0 \tag{5.6}$$

The problem of carrying over the system output "y(t)" from a given initial value to a desired value or setpoint "$y_r(t)$" is similar to the problem of carrying over the variable "$\Delta y(t)$" from a zero initial value into a final value "$\Delta y(t_f)$", where

$$\Delta y(t_f) = \int_{t_0}^{t_f} c\Phi(t_f,\tau)bu(\tau)d\tau, \tag{5.7}$$

this is possible however, if and only if the scalar product of "$\Delta y(t_f)$" with some nonzero constant value "η" can be made arbitrarily large by a suitable choice of "u(t)". That is, assuming $k > 0$, let the control input be selected as

$$u(t) = k\,b^T\,\Phi^T(t_f,t)\,c^T\,\eta \tag{5.8}$$

in which case, the scalar product "$\eta\,\Delta y(t_f)$" is given by

$$\eta\,\Delta y(t_f) = k\int_{t_0}^{t_f} \eta\,c\,\Phi(t_f,\tau)\,b\,b^T\,\Phi^T(t_f,\tau)\,c^T\,\eta\,d\tau \tag{5.9}$$

Notice that selecting the control input "u(t)" as expressed in equation 5.8 implies that the integral equation 5.9 is positive definite which in turns is equivalent to satisfying the condition for output controllability of a linear system. In this case,

equation 5.8 would contain information regarding the inverse dynamics of the linear system.

Having clarified the connection between plant invertibility and controllability in linear systems, let us resume the operation of the suggested control scheme. Observe that from figure 5.2, the following equation holds true

$$y(t) = P\left(u_1(t) + u_2(t)\right) \tag{5.10}$$

where "P" represents the unknown forward transfer operator of the linear system. The signals "$u_1(t)$" and "$u_2(t)$" represent the feedback and feedforward components of the control action, respectively.

The signal "$u_1(t)$" may be written as

$$u_1(t) = G\left(y_r(t) - y(t)\right) \tag{5.11}$$

where "G" constitutes a filter, and "$y_r(t)$" represents the desired plant output or setpoint. It must be clarified that the tasks of the filter "G" are to compensate the effects that an unstable pole has on the system response, and to prevent from having a not proper ITO as a result of trying to control a strictly proper unknown dynamic system.

The signal "$u_2(t)$" is the output of the slave neural network "SNN" that provides information regarding the approximated estimation of the ITO of the unknown system. Its task is to produce an identity map between the actual plant output and the desired plant output or setpoint.

If the on-line neural network estimator "NN" involved in the control scheme is a single layer neural network with input vector "Z(t)" selected from a neighbourhood of the unknown plant output "$y(t)$", (ie. $Z(t) \in \{y(t) :\mid y(t) \mid \leq \Theta\}$, with "$\Theta$" a small constant value), then by using the continuous-time version of the learning algorithm represented by equation 3.32, in theorem 1 of chapter 3, the error signal "$e_n(t)$" in the control scheme tends to zero in a finite time, and the ITO of the unknown linear plant may be approximated by the sum of the weight elements of the network. That is, suppose the neural network input vector "Z(t)" is selected as

$$Z(t) = (z_1(t), z_2(t), ..., z_n(t))^T = (y(t), y(t) + \theta_1, ..., y(t) + \theta_{n-1})^T \tag{5.12}$$

with $\theta_i \leq \Theta$, i=1,2,...,n-1. Observe that the error signal "$e_n(t)$" in the control scheme is equal to

$$e_n(t) = u(t) - y_n(t) \tag{5.13}$$

where "$y_n(t)$", the output of "NN", is given by

$$y_n(t) = \sum_{i=1}^{n} w_i(t)\, z_i(t) \tag{5.14}$$

If the learning algorithm represented by equation 3.32 of chapter 3 is used to adapt the weights of the network then, after a short convergence time "t_r", the error equation 5.13 equals zero. That is,

$$0 = u(t \geq t_r) - \sum_{i=1}^{n} w_i(t \geq t_r) z_i(t \geq t_r) \tag{5.15}$$

Now, since "θ_i", i=1,2,...,n-1 in equation 5.12 is a small value for each "i", the following approximation is valid

$$z_1(t) \approx z_2(t) \approx \ldots \approx z_n(t) \approx y(t) \tag{5.16}$$

and therefore

$$0 \approx u(t \geq t_r) - \left(\sum_{i=1}^{n} w_i(t \geq t_r) \right) y(t \geq t_r) \tag{5.17}$$

Clearly, if

$$\hat{P}^{-1} = \sum_{i=1}^{n} w_i(t \geq t_r) \tag{5.18}$$

then

$$\frac{u(t \geq t_r)}{y(t \geq t_r)} \approx \hat{P}^{-1} \tag{5.19}$$

where "$\hat{P}^{-1}$" represents an approximation of the ITO of the unknown system. Under these circumstances, observe that the output of "SNN" in the control scheme is equal to

$$u_2(t) = \sum_{i=1}^{n} w_i(t)\, y_r(t) \approx \hat{P}^{-1} y_r(t) \tag{5.20}$$

Substituting equations 5.11 and 5.20 into equation 5.10 yields,

$$y(t) \approx \frac{P\,\hat{P}^{-1} + G\,P}{1 + G\,P} y_r(t) \tag{5.21}$$

Equation 5.21 is a dynamic equation that represents the closed-loop response of the unknown linear plant in the proposed control scheme. Notice that, since it takes a short convergence time for the term "$\hat{P}^{-1}$" in equation 5.21 to represent an approximation of the ITO of the unknown plant, the product "$P\hat{P}^{-1}$" is not exactly equal to the identity function for all time "t" and therefore, a transient behaviour component of the closed-loop response of the system will be present. The design of the filter "G" in the proposed scheme, influences the duration of the transient behaviour component, and in the case of unstable unknown plants, it may be used to compensate the effects of unstable poles of the system.

Based on empirical evidence, if the dynamic order of the unknown system under study is known to be at least equal to 2, the filter "G" may be designed as follows

$$\frac{u_1(t)}{e(t)} = \frac{K_G\,(\alpha_1 + D)(\alpha_2 + D)}{(\beta_1 + D)(\beta_2 + D)} \tag{5.22}$$

with $K_G, \alpha_1, \alpha_2, \beta_1, \beta_2$ suitably chosen.

In order to show the performance of the proposed control scheme, the problem of controlling the position of the spring-mass-damper system is presented as an illustrative example. The mathematical model of this system which is assumed to be unknown is represented by the state equations

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -\frac{R}{m}\,x_1(t) - \frac{C}{m}\,x_2(t) + \frac{1}{m}\,u(t) \end{aligned} \tag{5.23}$$

with output variable given by the equation

$$y(t) = x_1(t) \tag{5.24}$$

where "$x_1(t)$" and "$x_2(t)$" represent the vertical position and velocity of the mass, respectively; and the dimensions of the unknown parameters "R", "C", and "m" are expresed in the metre-kilogram-second system of units, and for the sake of the computer simulation are assumed to be equal to 58.37, 5.84, and 1.46, respectively [85]. The acting input force "u(t)" and the vertical position of the mass "$x_1(t)$" are considered to be available for measurements at all time. The control objective is satisfied by making the system output "y(t)" follow a reference signal "$y_r(t)$". In particular, it is assumed that the reference signal is the function

$$y_r(t) = \begin{cases} cos(3t) & \text{if } t \leq t_0 \\ 2 & \text{otherwise} \end{cases} \tag{5.25}$$

where "t_0" is a positive value.

The filter "G" for this second order dynamic system is designed as follows

$$\frac{u_1(t)}{e(t)} = \frac{K_G\,(\alpha_1 + D)}{(\beta_1 + D)} \tag{5.26}$$

Figures 5.4 and 5.5 illustrate the performance of the proposed control scheme and the approximated ITO estimation provided by the on-line neural network estimator, respectively. Observe that, after a relatively small transient behaviour the steady-state component of the system output is off-set free, and that despite the abrupt change in the reference signal, the system output signal tracks it with great

accuracy. In this computer simulation, the initial values of the weight elements of the on-line neural network estimator were selected randomly between -1 and 1. Also, the value of the adaptation gain "k" of "NN", and the parameters "K_G", "α_1" and "β_1" of the filter "G" were selected equal to 8, 150, 1.33, and, 1 respectively.

It is important to clarify the effects that the approximated ITO estimation has on the response of the unknown system. Figure 5.6 illustrates these effects. Here, the signal "$u_2(t)$" that contains information about the approximated ITO estimation has been suppressed from the control action "u(t)", and as a result, the system output does not follows the changes of the reference signal. In this computer simulation the value of the parameters of the filter "G" and the adaptation gain "k" of "NN" remained the same as before.

Figure 5.7 illustrates the performance of the control scheme when the measurement of the unknown plant delayed output and delayed input signals were corrupted by white noise, (ie.the input and desired output for "NN"). In this case, despite the fact of the presence of noise in the neural network input and desired output signals, the generated control action is able to control the unknown plant output by forcing it to track the prescribed reference input. The corresponding approximated ITO estimation, a typical input signal of "NN" corrupted by measurement noise, and the desired output signal of "NN" corrupted by noise are sketched in figure 5.8.

The following section discusses the applicability of the proposed control scheme to control a certain class of continuous-time unknown nonlinear dynamic systems.

5.4 CONTINUOUS-TIME NONLINEAR SYSTEMS CASE

Consider an unknown nonlinear dynamic system represented by the following mathematical model

$$\dot{X}(t) = f(X(t)) + g(X)\,u(t) \tag{5.27}$$

$$y(t) = h(X(t)) \tag{5.28}$$

where "f(.)", "g(.)", and "h(.)" are unknown bounded, smooth functions; $X(t) \in \Re^n$ is the state vector, $u(t) \in \Re$ is the control input, and $y(t) \in \Re$ is the system output.

The control problem is solved by generating a control input "u(t)" such that the system output "y(t)" tracks a prescribed reference signal or setpoint.

The existence of an ITO for the nonlinear system represented by equations 5.27 and 5.28 can be loosely interpreted as the satisfaction of a local reachability

condition. That is to say, finding a function

$$u(t) = l(y(t), y_r(t)) \tag{5.29}$$

that represents the ITO of the unknown system and enables the plant output "y(t)" to be brought to the reference signal "$y_r(t)$".

In terms of our proposed control scheme, if the ITO of the unknown system exists, then the function described by equation 5.29 can be written as

$$u(t) = \hat{P}^{-1}\, y_r(t) + G\, e(t) \tag{5.30}$$

where "$\hat{P}^{-1}$" represents an approximation of the ITO estimation of the unknown system, "G" is a filter whose design is similar to the one of equation 5.22, and the variables"$y_r(t)$" and "e(t)" represent the desired system output and feedback error, respectively.

In order to illustrate the performance of the proposed control scheme, the problem of controlling the angular position of an inverted pendulum connected to a dc motor via a gear train, [78], is presented as an example. A mathematical model for this inverted pendulum system, which is schematically depicted in figure 5.9 is the following

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= \frac{g}{l} sin(x_1(t)) + \frac{N\, K_m}{m\, l^2} x_3(t) \qquad (5.31) \\ \dot{x}_3(t) &= -\frac{k_b\, N}{L_a} x_2(t) - \frac{R_a}{L_a} x_3(t) + \frac{1}{L_a} u(t) \\ y(t) &= x_1(t) \qquad (5.32) \end{aligned}$$

where for the sake of the computer simulation, the unknown parameters "g", "l", "m", "N", "K_m", "K_b", "R_a", and "L_a" are assumed to be 9.8 m/s^2, 1 m, 1 kg, 10, 0.1 N.m/A, 0.1 v.s/rad, 1 Ω, and 0.1 H, respectively. The state variables "$x_1(t)$", "$x_2(t)$", and "$x_3(t)$" represent the angular position of the pole, the angular velocity of the pole, and the armature current in the motor, respectively.

Again, a single layer neural network will be used as the on-line ITO estimator, and the desired system response will be the signal

$$y_r(t) = \begin{cases} 0.1 & \text{if } t \leq 15 \\ -0.1 & \text{otherwise} \end{cases} \tag{5.33}$$

Figures 5.10 and 5.11 illustrate the inverted pendulum controlled state variables and the performance of the control scheme in generating the control action when

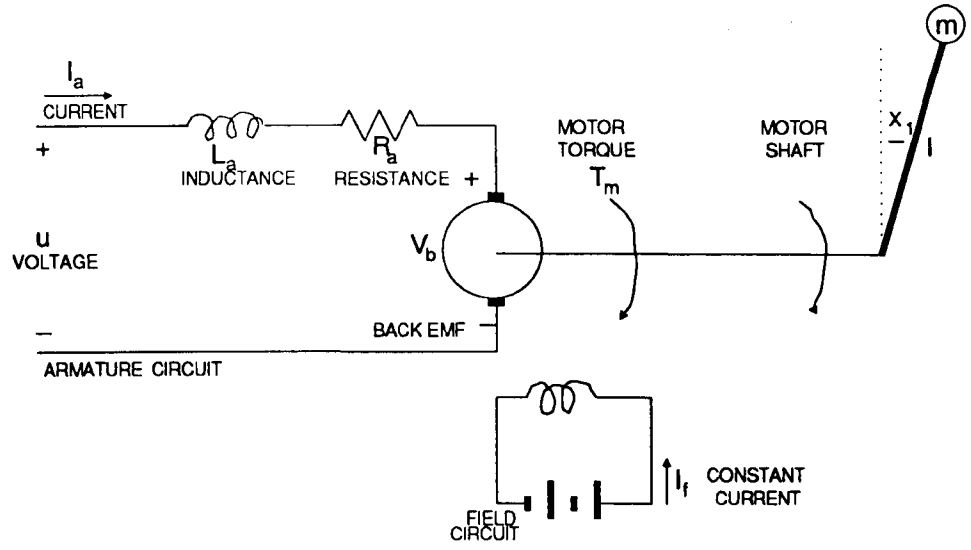


Figure 5.9: Schematic representation of the inverted pendulum.

the filter "G" was selected as

$$G(D) = \frac{K_G(\alpha_1 + D)}{(\beta_1 + D)} \tag{5.34}$$

with $K_G = 80300$, $\alpha_1 = 3.86$, and $\beta_1 = 1000$.

Notice that after a short transient behaviour, the controlled system output follows the desired output accurately. In this computer simulation, the value of the adaptation gain "k" of "NN" was chosen equals to 10, and the initial conditions of its weight elements were selected randomly between -1 and 1.

The computer simulation results, when the input signals and desired signal to "NN" were corrupted by measurement noise are sketched in figures 5.12, 5.13, and 5.14. Figure 5.12 illustrate the inverted pendulum controlled state variables and the desired output response. Figure 5.13 sketches the generated control action and the approximated ITO estimation of the system; and figure 5.14 depicts a typical "NN" input signal corrupted by noise and the desired "NN" output signal also influenced by noise.

The following section presents the examples of the spring-mass-damper system and the dc motor driven inverted pendulum system when the on-line neural network estimator "NN" is trained using the discrete-time version of the learning algorithms to generate discrete-time control actions.

5.5 DISCRETE-TIME ALGORITHMS

This section presents the design of discrete-time controllers for continuous-time unknown dynamic systens using our proposed control scheme. The basic difference with respect to the control design of the previous section is in the use of the discrete-time version of the learning algorithms for training the on-line neural network estimator. Under these circumstances, it must be emphasized that different from the continuous-time learning algorithm case, in the discrete-time case the convergence of the neural network estimator output to its desired output is asymptotic. That is, say a single layer neural network estimator is used in the control scheme, then

$$e_n(k) = u(k) - y_n(k) \tag{5.35}$$

$$y_n(k) = W^T(k)\, Z(k) \tag{5.36}$$

where "Z(k)" is the neural network input which is selected from the set $\{y(k) :\mid y(k) \mid \leq \Theta\}$ with Θ a small value. If the learning algorithm presented in theorem 5 of chapter 3 is used to adapt the weight elements then, by appropriate selection of the adaptation gain of the network the following equation holds valid

$$\lim_{k\to\infty} e_n(k) = 0 \tag{5.37}$$

and therefore

$$\lim_{k\to\infty} y_n(k) = u(k) \tag{5.38}$$

On the other hand, since "Θ" is a small value then, the components of the vector "Z(k)" may be assumed to be approximately equal to "y(k)" and therefore,

$$y_n(k) = \sum_{i=1}^{n} w_i(k) y(k) \tag{5.39}$$

Thus $\sum_{i=1}^{n} w_i(k) \longrightarrow \frac{u(k)}{y(k)}$ asymptotically.

Now, the generated control action may be written as

$$u(k) = G\, e(k) + \left(\sum_{i=1}^{n} w_i(k)\right) y_r(k) \tag{5.40}$$

where "G" is selected analogously as in equation 5.22.

The performance of the control scheme with a discrete-time generated control action will be tested using the spring-mass-damper system , and the dc motor driven inverted pendulum system presented in the previous sections. In both examples, an ideal zero order hold is inserted between the generated feedforward control action and the plant.

Figures 5.15 and 5.16 show the performance of the control scheme in controlling the state variables of the spring-mass-damper system, and the generated control action and approximated ITO estimation of the plant, respectively. It should be pointed out that the parameters of "G" used in this computer simulation were the same as the ones used in section 5.3. Also, both the sampling time for the on-line neural network estimator and the controller, and the adaptation gain of "NN" were selected equal to 0.03 seconds and 0.01, respectively.

Figure 5.17 illustrates the performance of the control scheme when the input signal and desired output signal of "NN" were corrupted by measurement white noise.

On the other hand, figure 5.18 shows the controlled state variables of the dc motor driven inverted pendulum system. Again, the parameters of "G" were selected the same as the ones used in section 5.4. In this computer simulation, the sampling time for the neural network estimator "NN" and the controller, as well as the adaptation gain of "NN" were selected equal to 0.01 seconds and 0.01, respectively. It is worth noticing the off-set free response of the unknown plant. The generated control action and the approximated ITO estimation are shown in figure 5.19.

Finally, figure 5.20 shows the state variables position of the pole and angular velocity of the pole, and a typical "NN" input signal and desired output corrupted by measurement white noise. Despite the presence of measurement white noise in the "NN" input and desired output signal, the control scheme is robust enough to keep the output of the unknown plant tracking the desired plant output.

A point of importance is to realize that in the proposed control scheme a zero setpoint or reference signal is not admissible since it would yield a zero output from the feedforward portion of the generated control action.

5.6 SUMMARY

In this chapter, a direct inverse dynamics control scheme for controlling unknown linear or nonlinear dynamic systems has been presented, and its performance has been tested by computer simulations on a spring-mass-damper system and a dc motor driven inverted pendulum system.

In the proposed control scheme, the control action has been generated as the sum of the output of a feedforward controller plus the output of a feedback filter. The feedforward control action has included an approximation of the ITO of the unknown system; whereas the feedback control action, although its structure was

known, its parameters have been selected by trial an error.

Section 5.3 has shown the connection between system invertibility and controllability for continuous-time linear systems. In this section, a continuous-time control action for controlling unknown linear plants based upon a continuous-time variable structure-type of learning algorithm for training an on-line neural network estimator has been obtained. The spring-mass-damper system has been presented as an illustrative example of the performance of the control system. The robustness of the control scheme with respect to measurement noise in the neural network signals has also been tested by computer simulation results. These results have shown that one of the effects of the feedforward component of the control action was to eliminate the off-set from the system response with respect to the reference signal.

The performance of the control scheme in controlling a certain class of nonlinear systems via a continuous-time generated control action has been presented in section 5.4. The dc motor driven inverted pendulum system has been presented as illustrative example. The robustness of the control scheme with respect to measurement noise in the neural network signals has also been tested in the studied example.

Section 5.5 has shown the feasibility of obtaining a discrete-time control action for controlling continuous-time unknown dynamic systems within the proposed control scheme. Here, the discrete-time version of the learning algorithm has been used to adapt the weights of the on-line neural network ITO estimator. Both, the linear example of the spring-mass-damper system, and the nonlinear example of the dc motor driven inverted pendulum system have been presented to illustrate the performance of the control scheme.

It is important to mention the relevance of an adequate selection of the adaptation gain of the learning algorithm, both in the continuous-time as in the discrete-time version of the algorithms. Indeed, the selection of the adaptation gain influences the ability of the neural network to track its desired output accurately, and therefore it may have an adverse effect on the quality of the response of the control system. In this work, the selection of the adaptation gain has been done by trial and error.

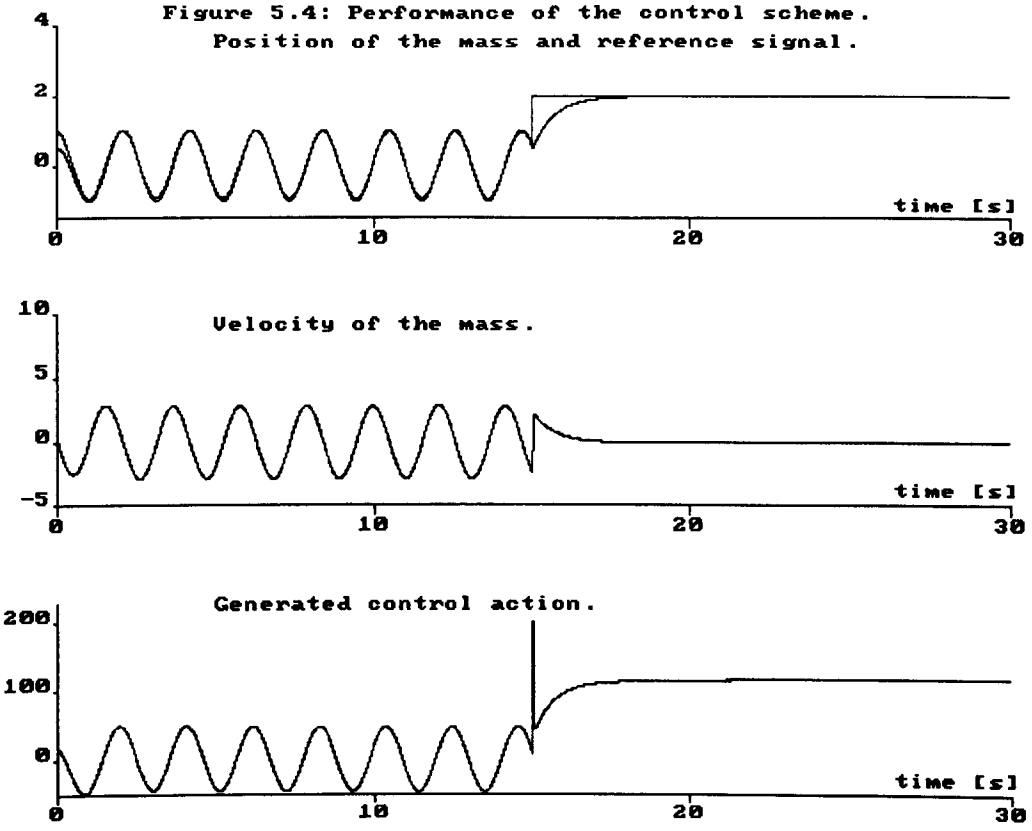


Figure 5.4: Performance of the control scheme.

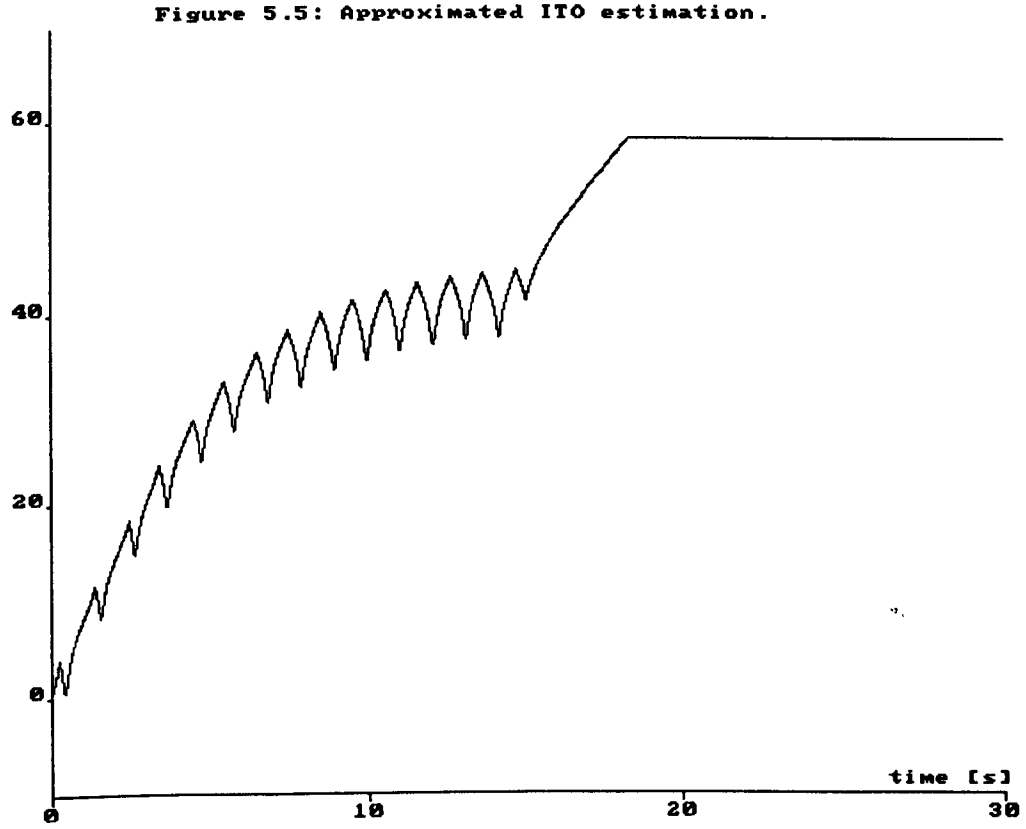


Figure 5.5: Approximated ITO estimation.

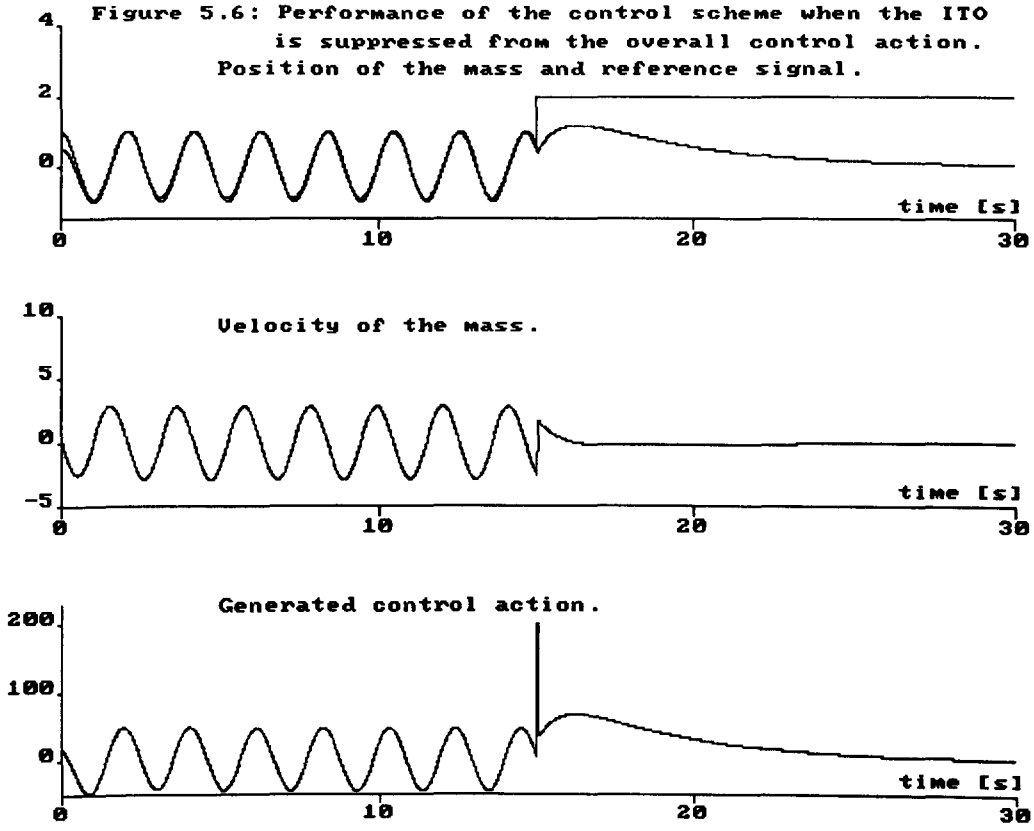


Figure 5.6: Performance of the control scheme when the ITO is suppressed from the overall control action.

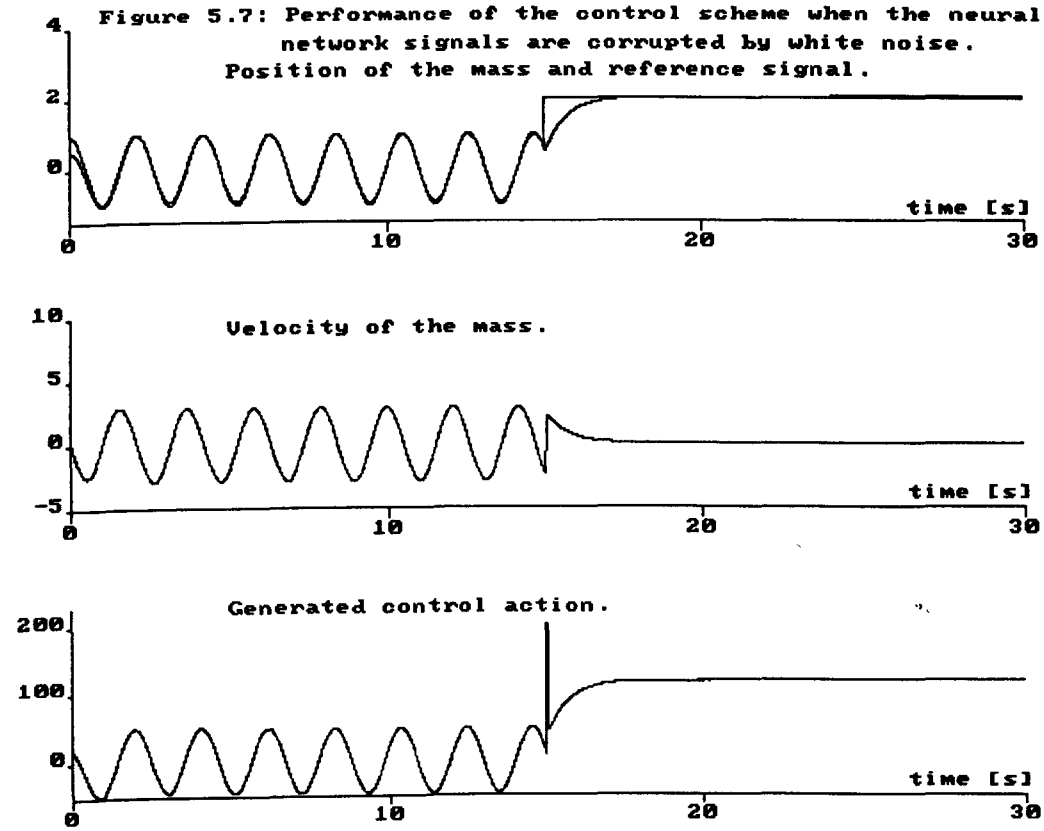


Figure 5.7: Performance of the control scheme when the neural network signals are corrupted by white noise.

Figure 5.8: Neural network signals corrupted by noise.

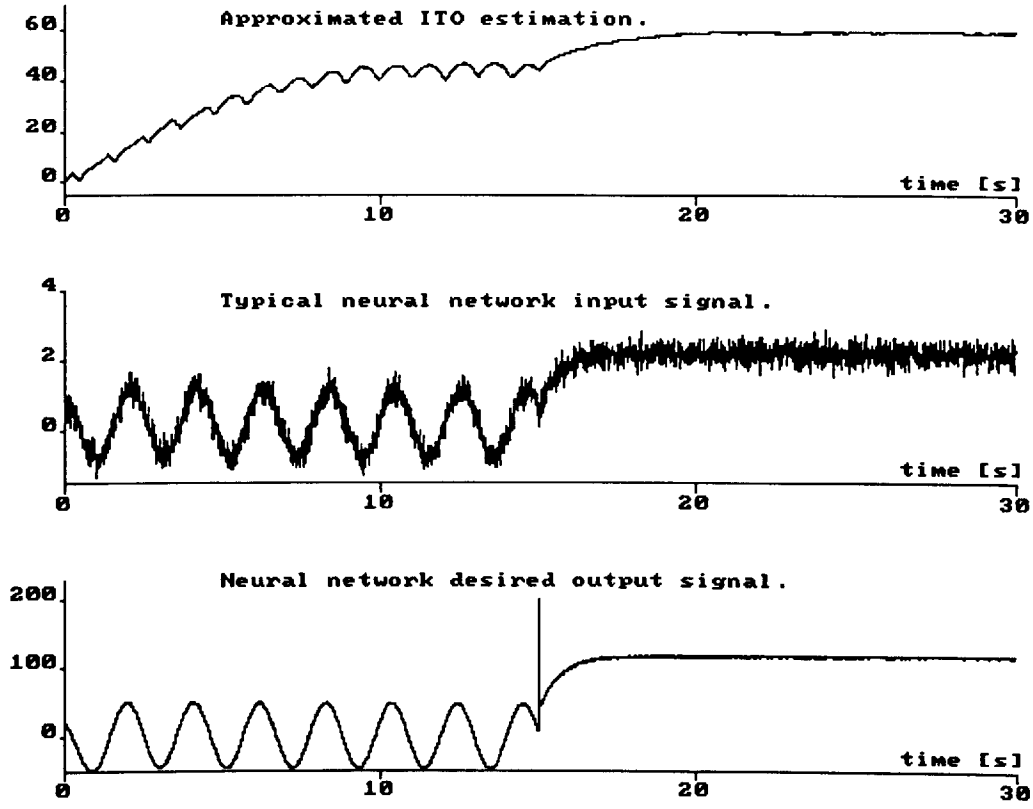


Figure 5.10: Inverted pendulum controlled state variables.

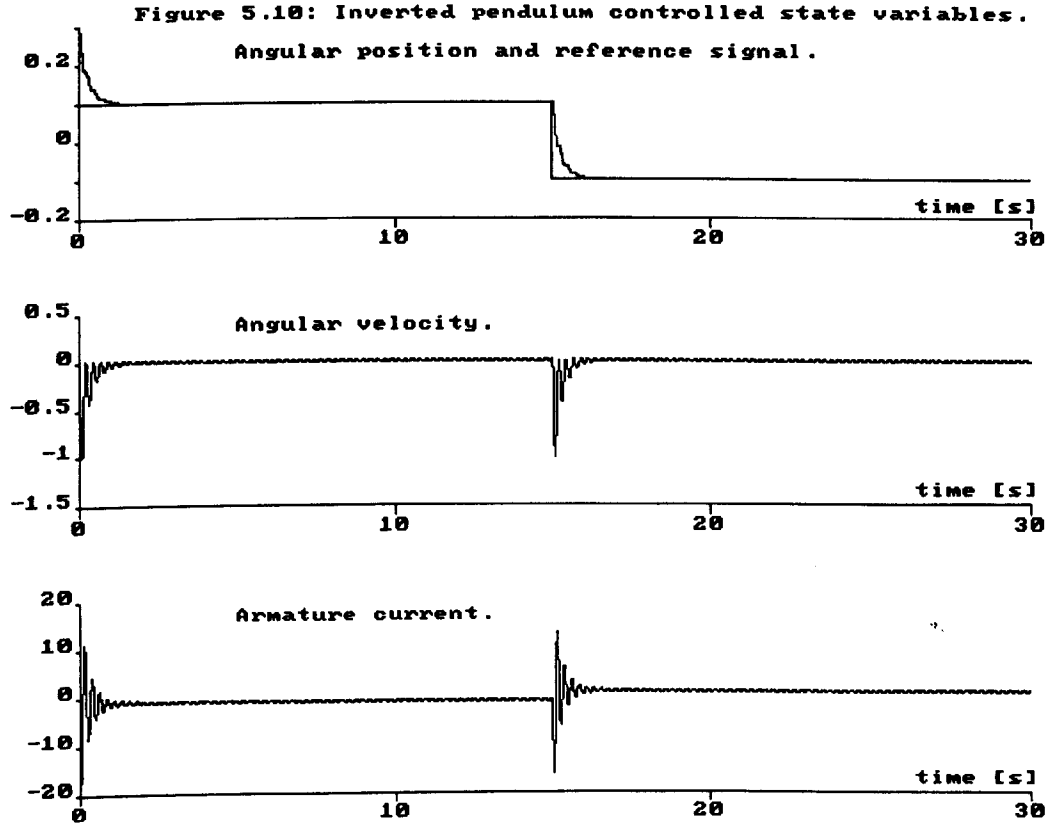


Figure 5.11: Performance of the control scheme.

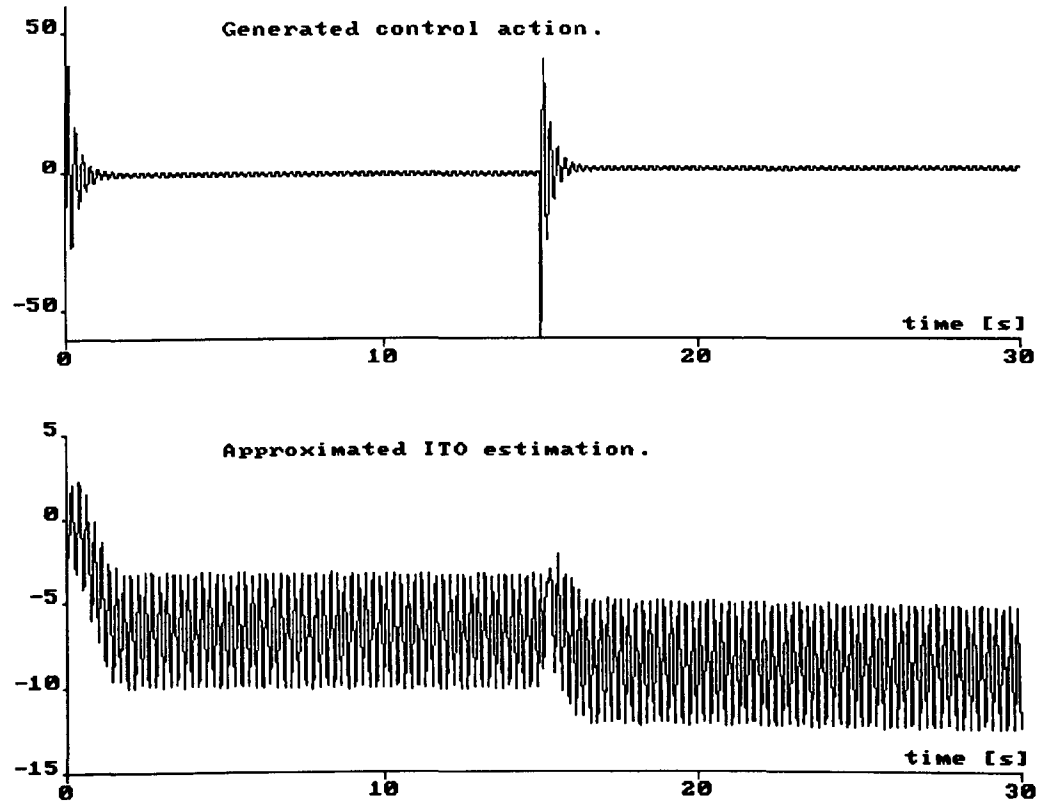


Figure 5.12: Inverted pendulum controlled state variables.

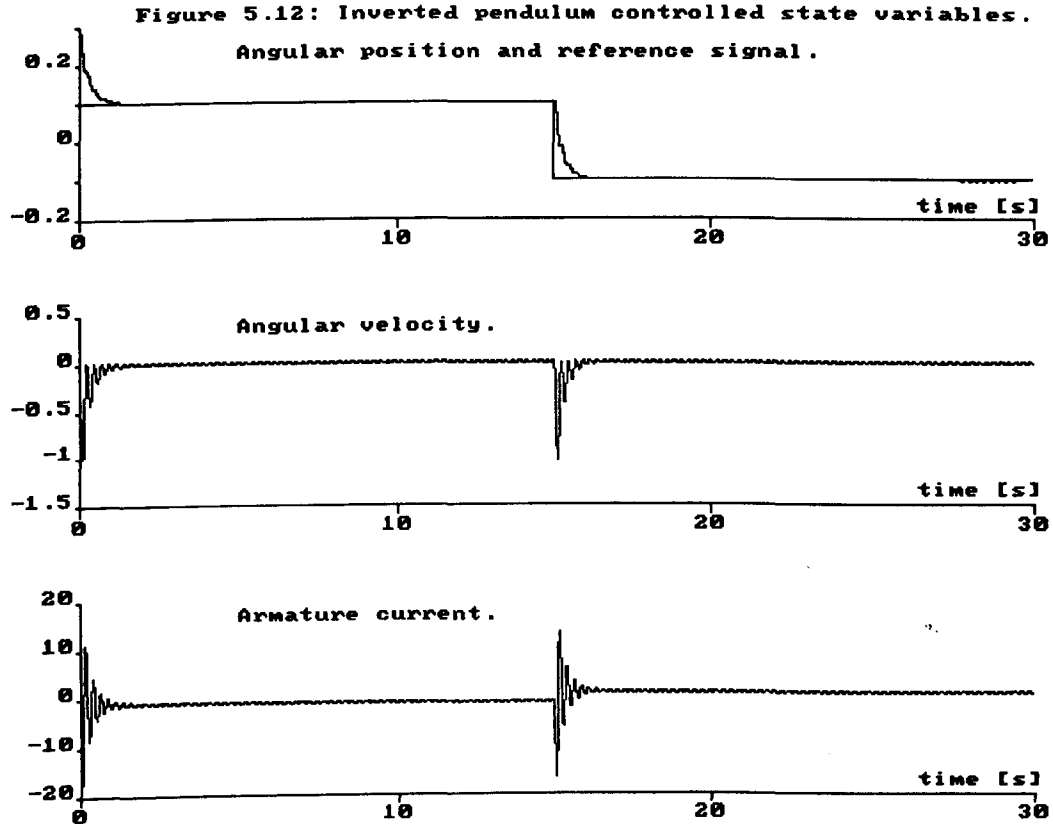


Figure 5.13: Performance of the control scheme.

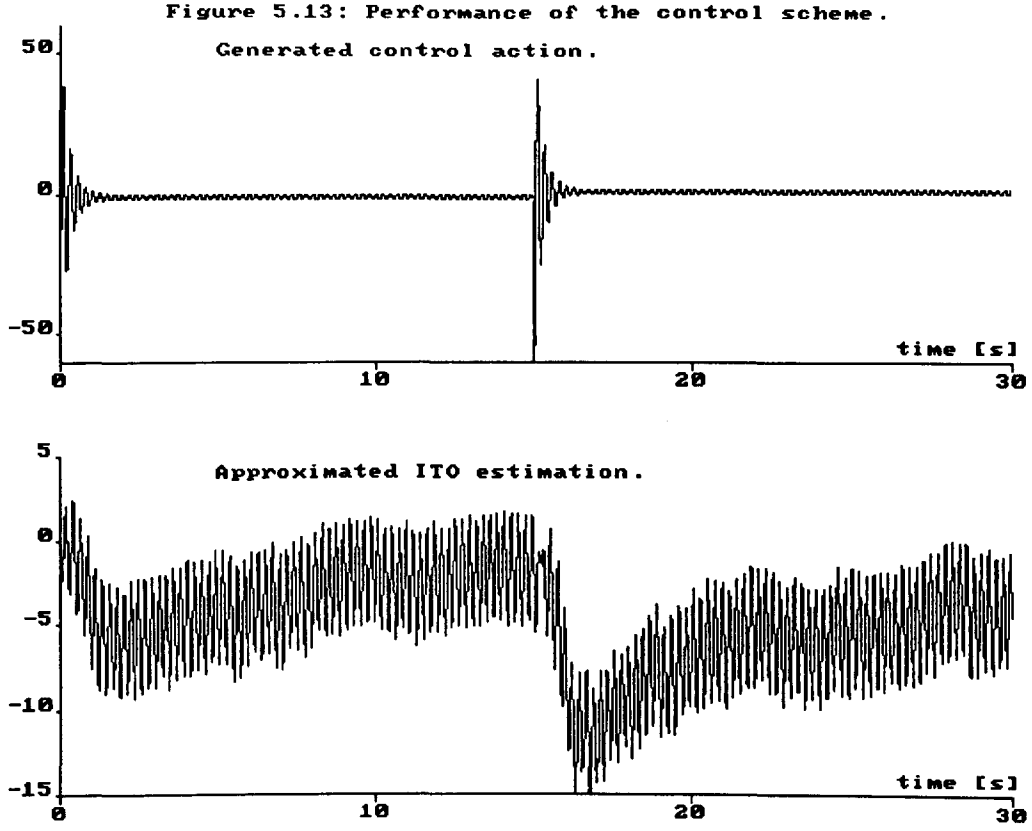
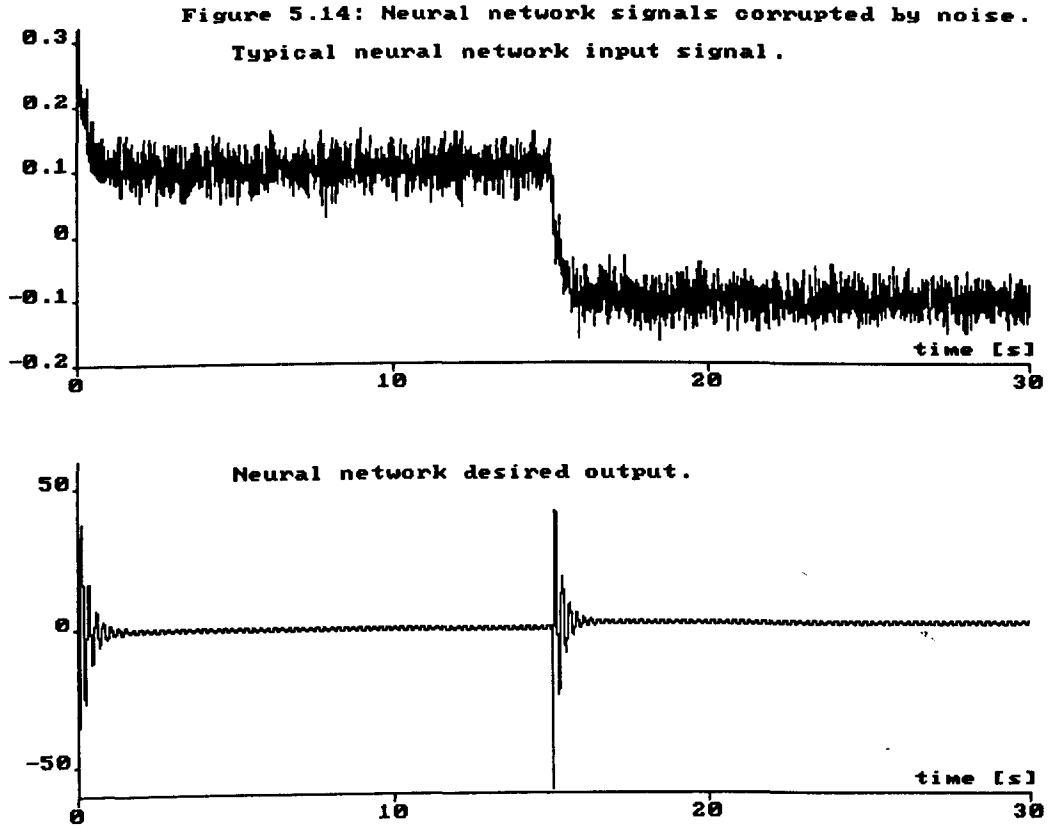


Figure 5.14: Neural network signals corrupted by noise.



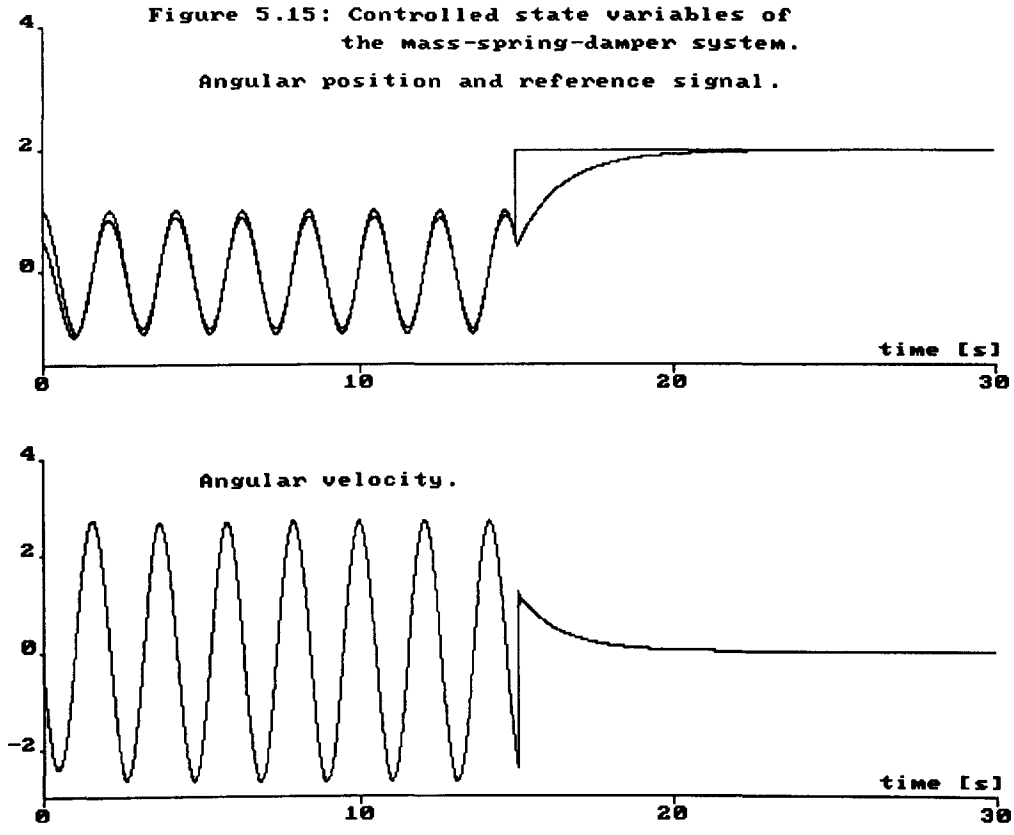


Figure 5.15: Controlled state variables of the mass-spring-damper system.

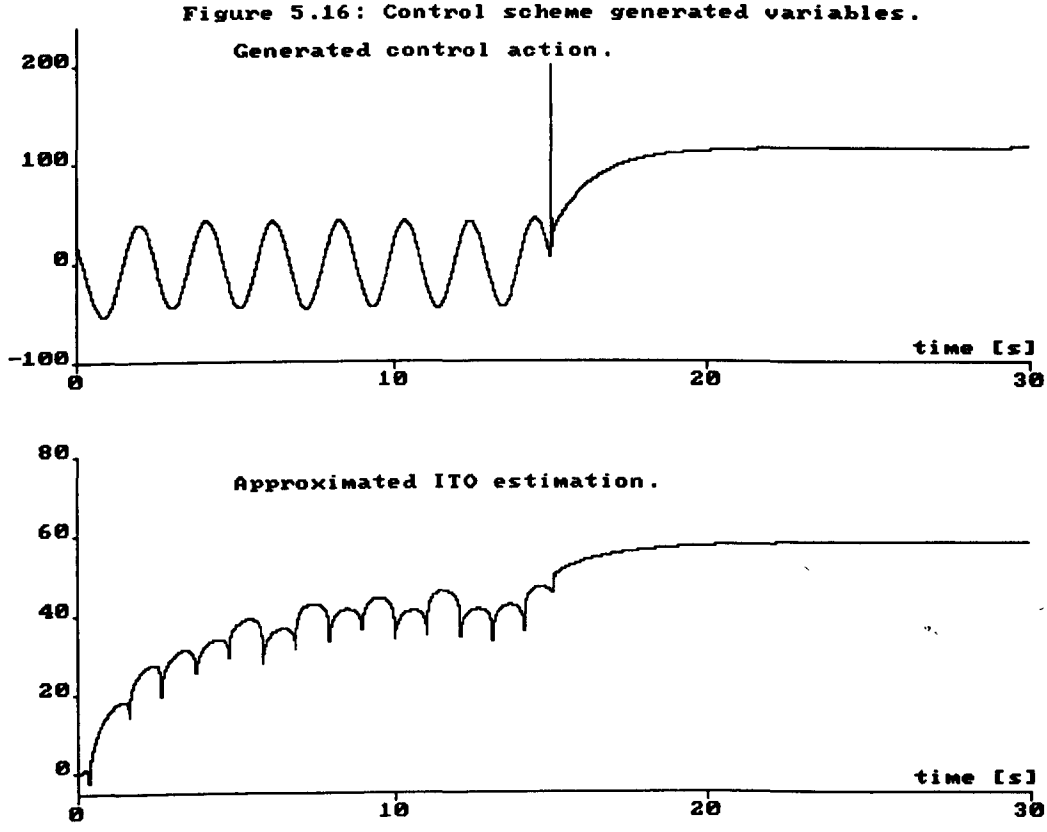


Figure 5.16: Control scheme generated variables.

Figure 5.17: Performance of the control scheme.

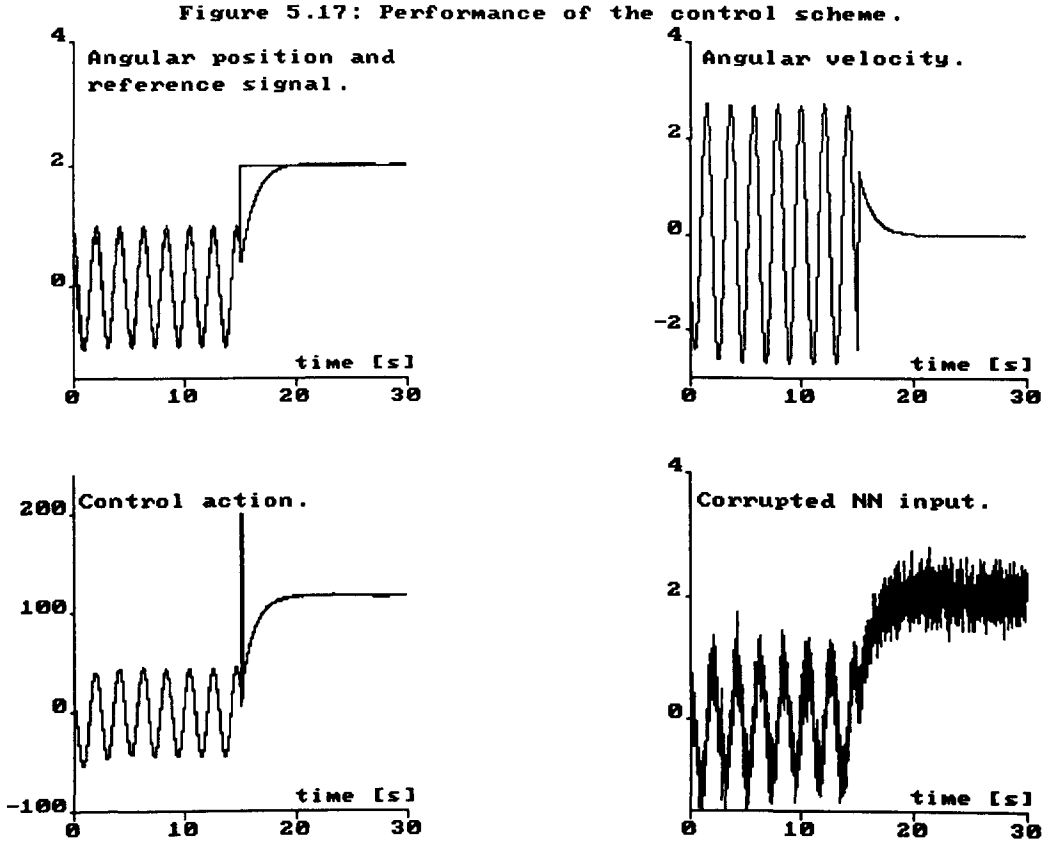


Figure 5.18: Inverted pendulum controlled state variables.

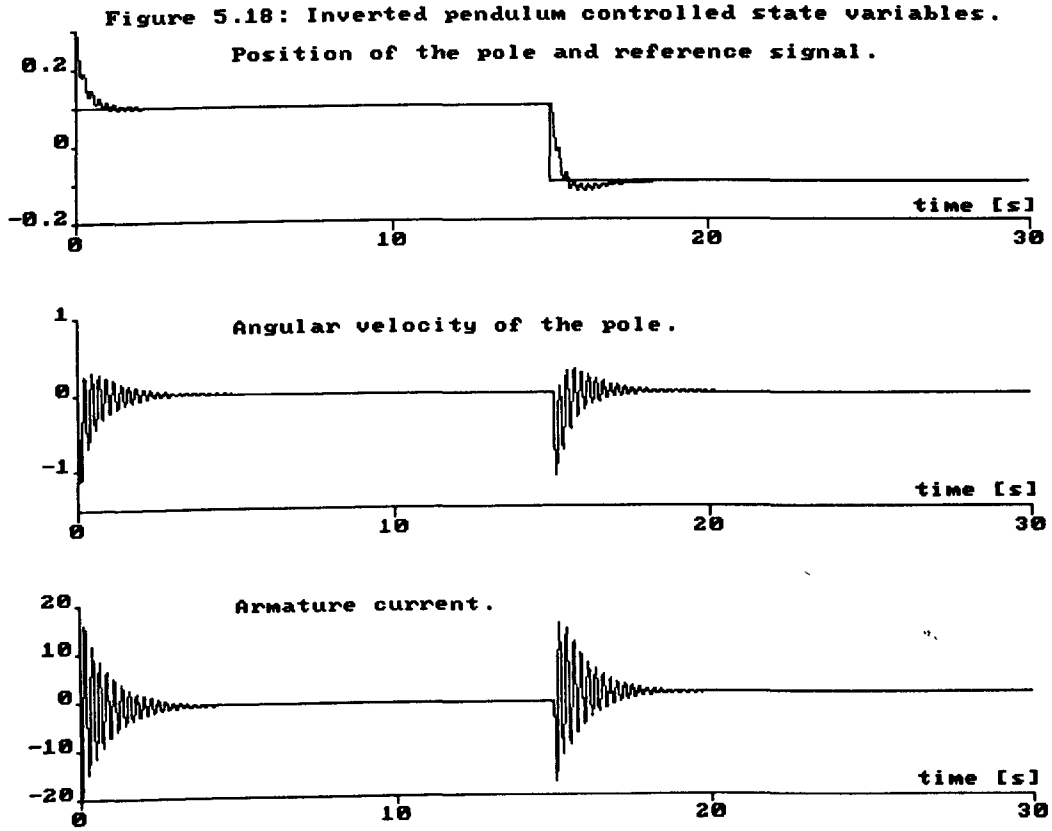


Figure 5.19: Control scheme generated signals.

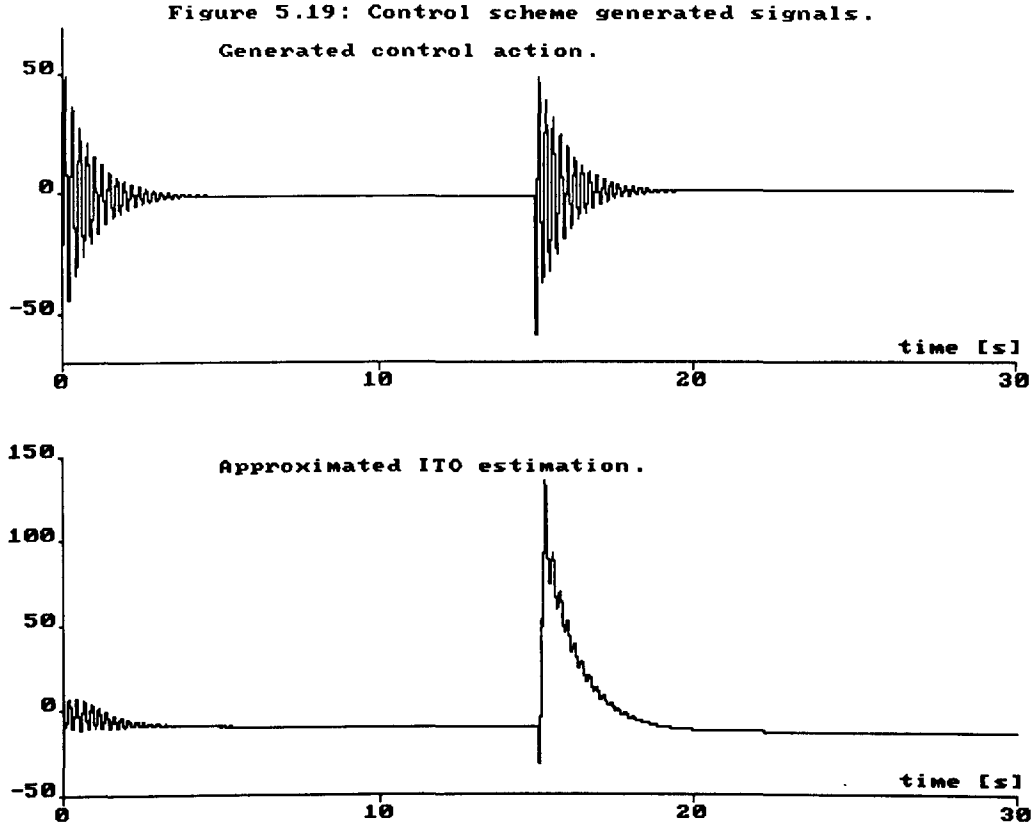
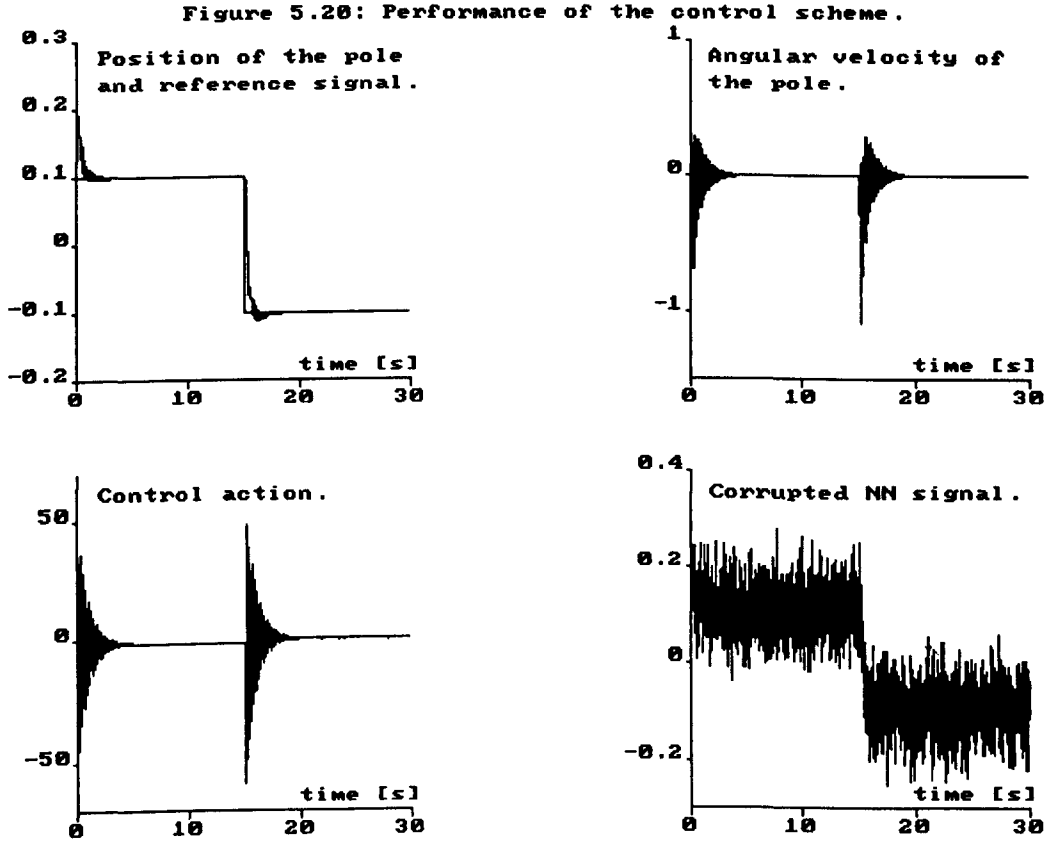


Figure 5.20: Performance of the control scheme.



Chapter 6

OTHER NEURAL NETWORK CONTROL APPLICATIONS

6.1 INTRODUCTION

The main objective of this chapter is to present an exploration of other potential applications of on-line trained neural networks. In particular, the chapter contains computer simulation results obtained from implementing an internal model neural network-based control scheme, and a model reference neural network-based adaptive control scheme. It must be clarified that these are preliminary results and more reseach work is necessary in order to assess global stability conditions for the operation of the suggested control schemes.

6.2 INTERNAL MODEL CONTROL SCHEME

Two important elements for the design of an internal model control scheme (IMCS) as the one depicted in figure 6.1 are the availability of the FTO and ITO of the dynamic system. In this control scheme, if the FTO represents a perfect copy of the forward dynamics of the system then the feedback path would only carry the influence of the disturbance affecting the system output, and the control system would operate in an open-loop configuration the stability of which would depend on the stability of the interconnection between the ITO controller and the actual dynamic system. In most practical situations, however, the FTO is not a perfect model of the dynamic system and therefore, the feedback signal in the control scheme contains the effects of this model mismatch as well as the influence of disturbances affecting the system output. As a result of this FTO mismatch, the stability and

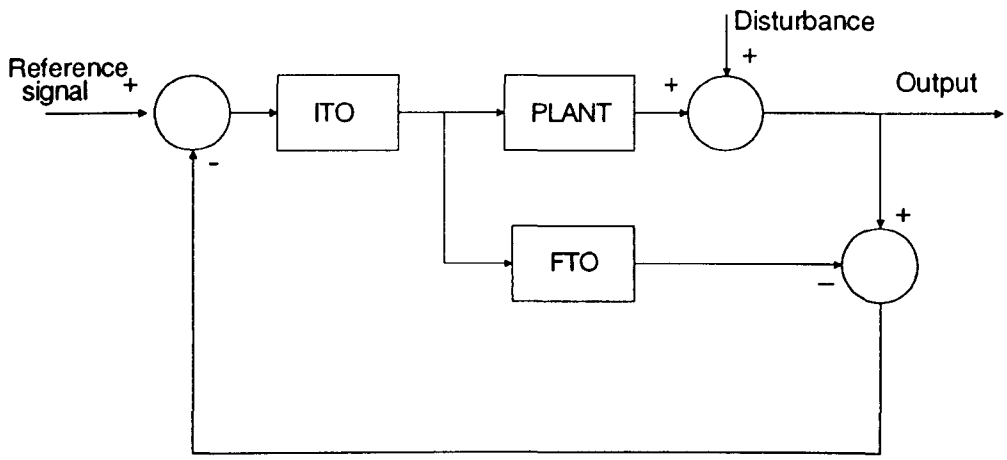


Figure 6.1: Internal model control scheme

robustness characteristics of the resulting closed-loop configuration may only be attained by detuning the ITO controller and augmenting it by a low-pass filter that may be adjusted on-line [74]. For a stable open-loop system, the selection of the low-pass filter depends on the system type. This filter selection enables the asymptotic tracking properties present in the open-loop system to be retained by the closed-loop system.

A comprehensive study of the internal model control scheme that covers unstable linear systems may be found in [74]. The extension to nonlinear plants is analysed in [79].

From the neural networks applications point of view, the implementation of an IMCS may be carried out by designing neural networks to approximate the FTO and ITO of the dynamic system under consideration. The work by Hunt et al [75] presents examples of the use of neural networks for nonlinear plants in an IMSC. In their work, they explore conditions for the invertibility of discrete-time nonlinear systems, and present neural network-based architectures for off-line identification of the FTO's and ITO's of nonlinear plants. In the mentioned work, the IMCS may be implemented once the off-line training phase of the neural network representations of the FTO and ITO have been completed.

In the next section, computer simulation results obtained from the implementation of an IMCS with on-line training of the neural network representations of the FTO and ITO of stable unknown linear systems are presented.

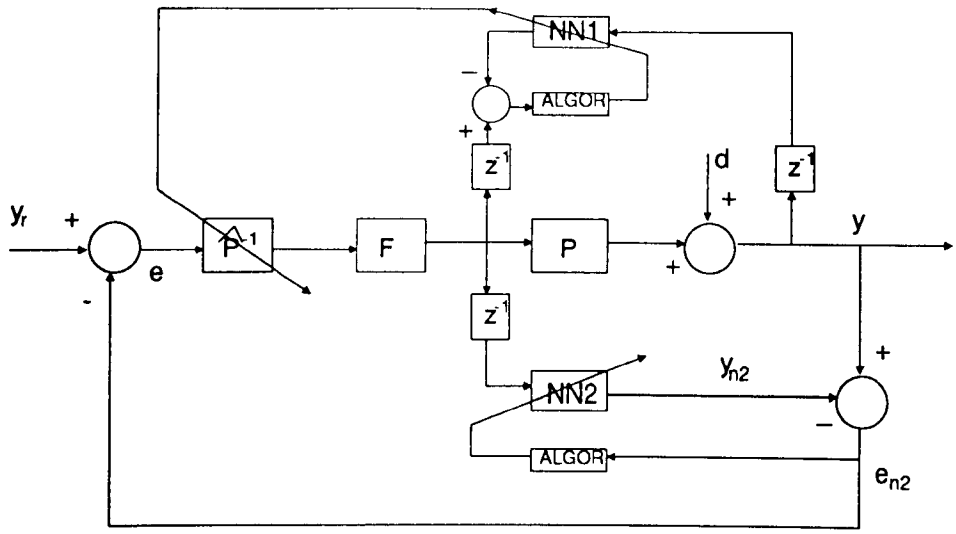


Figure 6.2: Neural network-based IMCS

6.2.1 NEURAL NETWORK-BASED IMCS

Figure 6.2 illustrates a neural network-based internal model control scheme. Observe that the task of the on-line neural network estimator in the parallel path to the unknown plant is to emulate its behaviour and therefore, to produce a feedback signal that contains the influence of the perturbation affecting the output of the system. On the other hand, the task of the on-line neural network ITO estimator is to generate an on-line approximation of the ITO of the unknown system that includes the effects of the output disturbances upon the system output. This ITO approximation is then multiplied by the difference between the reference signal and the FTO estimator error, and the resulting signal is processed by a filter to generate the actual control action.

From a mere qualitative viewpoint, the overall system stability depends on both the stability of the process and the stability of the FTO and ITO estimation processes.

Mathematically speaking, the following formulation may be used to clarify the system operation for invertible, stable unknown dynamic linear systems.

Observe that from figure 6.2

$$y(t) = P(D)u(t) + d(t) \tag{6.1}$$

$$u(t) = \hat{P}^{-1}(D)\,F(D)\,e(t) \tag{6.2}$$

where the factor "$\hat{P}^{-1}(D)$" comes from the on-line estimation of the ITO performed

by the neural network NN1 (see chapter 4 for mathematical details on the ITO estimation process). On the other hand, the feedback error signal "e(t)" may be written as

$$e(t) = y_r(t) - e_{n2}(t) \tag{6.3}$$

where the signal "$e_{n2}(t)$" that represents the FTO estimation error is given by

$$e_{n2}(t) = P(D)u(t) + d(t) - y_{n2}(t) \tag{6.4}$$

The on-line neural network FTO estimator output signal "$y_{n2}(t)$" may be represented by the equation

$$y_{n2}(t) = \tilde{P}(D)u(t-\tau) \tag{6.5}$$

where "$\tilde{P}(D)$" symbolizes an approximation of the FTO of the unknown system, and "$u(t-\tau)$" is the delayed control input. Here it will be assumed that $u(t-\tau) \approx u(t)$ for "τ" a small positive value and therefore, equation 6.5 may be rewritten as

$$y_{n2}(t) = \tilde{P}(D)u(t) \tag{6.6}$$

Substituting equation 6.6 into 6.4, and the resulting equation into equation 6.3 yields,

$$e(t) = y_r(t) - P(D)u(t) - d(t) + \tilde{P}(D)u(t) \tag{6.7}$$

Replacing equation 6.7 into 6.2 and the result into equation 6.1 yields

$$\begin{aligned} y(t) &= \frac{P(D)\hat{P}^{-1}(D)F(D)}{1+\hat{P}^{-1}(D)F(D)(P(D)-\tilde{P}(D))}\, y_r(t) \\ &+ \frac{1-\tilde{P}(D)\hat{P}^{-1}(D)F(D)}{1+\hat{P}^{-1}(D)F(D)(P(D)-\tilde{P}(D))}\, d(t) \end{aligned} \tag{6.8}$$

In order to show the performance of the IMCS with on-line ITO and FTO neural network estimators, the spring-mass-damper system studied in the previous chapter will be used as an illustrative example. In this case, both the ITO and the FTO neural network estimators will be single layer neural networks with on-line training using a continuous-time learning algorithm. Appendix D contains the simnon programs used to perform the computer simulations.

For the sake of the computer simulations, the same mathematical model and parameters of the spring-mass-damper system used in chapter 5 are used here. That is to say:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -40\,x_1(t) - 4\,x_2(t) + 0.68\,u(t) \\ y(t) &= x_1(t) \end{aligned} \tag{6.9}$$

The design of the low-pass filter "F" illustrated in the forward path of figure 6.2 is as follows

$$F(D) = \frac{K_F(\tau_1 D + 1)}{(\tau_2 D + 1)^n} \tag{6.10}$$

with the index "n" equal to the dynamic order of the unknown system. In particular, the low-pass filter for the example at hand is designed as

$$F(D) = \frac{K_F(\tau_1 D + 1)}{(\tau_2 D + 1)^2} \tag{6.11}$$

with $\tau_1 = 0.2$, $\tau_2 = 0.2$, and $K_F = 1$.

Figure 6.3 sketches the bahaviour of the controlled variables and the generated control action when the output of the system is disturbed by a white noise perturbation. In this computer simulation the reference signal was selected equals to

$$y_r(t) = \begin{cases} 1 & \text{if } t \leq 15 \\ 2 & \text{otherwise} \end{cases} \tag{6.12}$$

and the adaptation gains for the single layer neural networks NN1 and NN2 were selected by trial and error equal to 5 and 40, respectively.

Similarly, figure 6.4 shows the estimated ITO and FTO approximations, as well as the FTO estimation error. Note that these estimation processes were influenced by the white noise perturbation that affected the system output.

6.3 MODEL REFERENCE CONTROL

In general terms, a model reference adaptive control system as the one depicted in figure 6.5 operates by adjusting the controller parameters based on the error between the output of the system and the output of a known reference model. The performance of the system is specified by the reference model. Mathematically speaking, the model reference adaptive control problem may be formulated as follows [15]:

Given the input-output measurements "u(k)", "y(k)" obtained from a dynamic system "f", and given a desired plant output "$y_m(k)$" obtained from a known reference model "f_m" with input signal equal to "$u_m(k)$", we want to determine a control input "u(k)", for all $k \geq k_0$, such that

$$\lim_{k \to \infty} |y(k) - y_m(k)| \leq \psi \tag{6.13}$$

for some specified constant value $\psi \geq 0$.

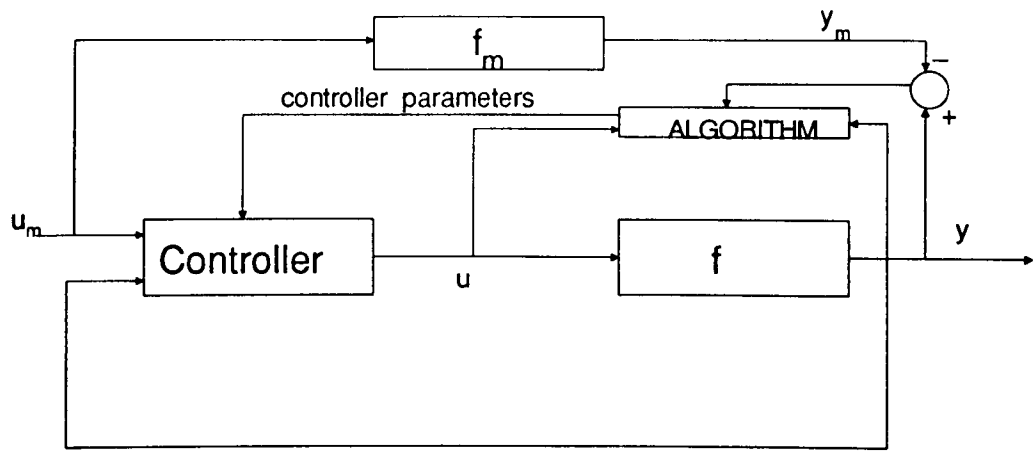


Figure 6.5: Model reference adaptive control system.

An issue of great importance in solving the above adaptive control problem is related to the determination of the adaptive law for the adjustment of the control parameters that would enable the overall system configuration to be stable.

There are basically three approaches for designing the adaptive law [80]:

- The gradient approach. Here, it is assumed that the controller parameters change more slowly than the other variables in the system. The resulting closed-loop system is not necessarily stable.

- The Lyapunov approach. In this case, the Lyapunov stability theory is used to design an adaptation law that results in a stable closed-loop system. The main difficulty here is to find an accurate model for the plant, and an appropriate lyapunov function for the given dynamic system.

- The passivity theory approach, that enables the adaptation law to be designed in such a way that the overall system is BIBO stable [81]. As before, this approach relies on a good knowledge of the system.

If the dynamic system under consideration is a linear system, then it is possible to design a stable adaptive law that results in a stable closed-loop system provided that some prior information about the plant forward transfer operator is available [82]. In the nonlinear system case however, a definitive conclusion about the global adaptive control problem is not yet known.

6.3.1 NEURAL NETWORK-BASED ADAPTIVE CONTROL SCHEME

The class of dynamic systems to be considered here may be described by the following mathematical model

$$y(k+1) = f(y(k), y(k-1), ..., y(k-n+1)) + \sum_{j=0}^{m} \beta_j u(k-j), \quad m < n \tag{6.14}$$

where "f(.,.)" is a smooth unknown linear or nonlinear function of its arguments, and "β_j", $j = 0, ..., m$ is a set of known parameters with "β_0" different from zero.

The prescribed performance for the given plant is determined by the output of a stable reference model described by the equation

$$y_m(k+1) = f_m(y_m(k), y_m(k-1), ..., y_m(k-n+1)) + u_m(k) \tag{6.15}$$

where "$f_m(.,.)$" is a known function, and "$u_m(k)$" is a known bounded external reference input. If the function "f(.,.)" were known, it would be possible to control the system 6.14 by selecting a control law represented by the equation

$$\begin{aligned} u(k) &= \frac{1}{\beta_0}(-f(y(k), y(k-1), ..., y(k-n+1)) \\ &- \sum_{j=1}^{m} \beta_j u(k-j) + \hat{y}_m(k+1)\Big) \end{aligned} \tag{6.16}$$

with "$\hat{y}_m(k+1)$" given by

$$\hat{y}_m(k+1) = f_m(y(k), y(k-1), ..., y(k-n+1)) + u_m(k) \tag{6.17}$$

Observe that under these circumstances the output error "$e_o(k)$" defined as the difference between the plant output and the reference model output may be written as

$$e_o(k+1) = f_m(e_o(k), e_o(k-1), ..., e_o(k-n+1)) \tag{6.18}$$

that represents a stable equation.

In our problem formulation however, in order to be able to implement equation 6.16, the unknown function "f(.,.)" must be previously estimated. In the neural network-based adaptive control scheme illustrated in figure 6.6, the task of the neural network is to perform an on-line estimation "$\hat{f}_k$" of the unknown function "f(.,.)", and to pass these estimated values to the controller in order to generate the actual control action. In other words, equation 6.16 may be rewritten in terms of "$\hat{f}_k$" as follows

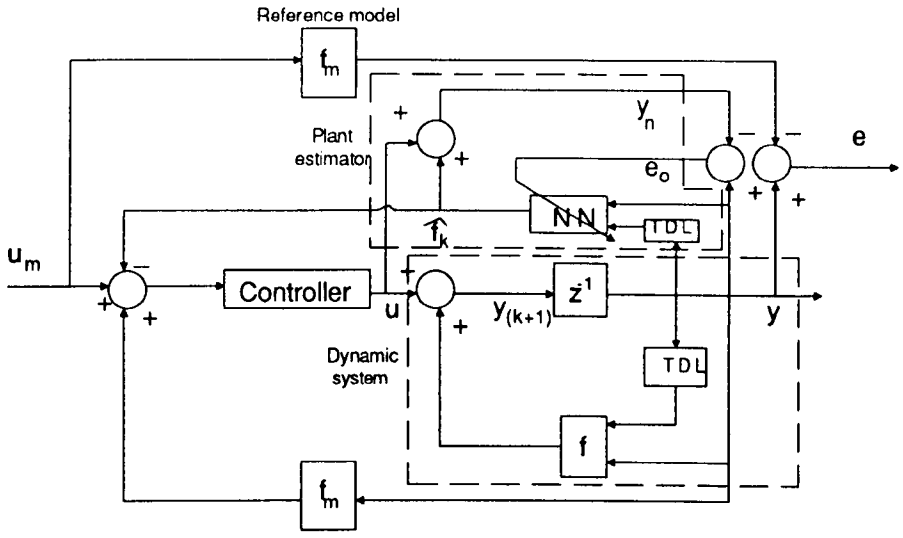


Figure 6.6: Neural network-based adaptive control scheme.

$$u(k) = \frac{1}{\beta_0}\left(-\hat{f}_k - \sum_{j=1}^{m}\beta_j u(k-j) + \hat{y}_m(k+1)\right) \tag{6.19}$$

where "$\hat{f}_k$" is the output of the neural network estimator. As a result, the closed-loop system dynamics may be represented by the equation

$$y(k+1) = f(y(k), y(k-1), ..., y(k-n+1)) - \hat{f}_k + \hat{y}_m(k+1) \tag{6.20}$$

Note that from figure 6.6, the error "$e_o(k+1)$" may be written as

$$\begin{aligned} e_o(k+1) &= y(k+1) - y_n(k+1) \\ &= f(y(k), y(k-1), ..., y(k-n+1)) - \hat{f}_k \end{aligned} \tag{6.21}$$

and therefore, the dynamics of this error may be represented by

$$\begin{aligned} e_o(k+1) - e_o(k) &= f(y(k), ..., y(k-n+1)) - f(y(k-1), ..., y(k-n)) \\ &- (\hat{f}_k - \hat{f}_{k-1}) \end{aligned} \tag{6.22}$$

where "$\hat{f}_k$" and "$\hat{f}_{k-1}$" are the present and the previous estimates of "f(.,.)".

Then, if a discrete-time learning algorithm, say the one described in section 3.3.2.1 of chapter 3, is used to update the weights of a single layer neural network estimator, equation 6.22 becomes

$$\begin{aligned} e_o(k+1) - e_o(k) &= (1-\alpha)e_o(k) + f(y(k), y(k-1), ..., y(k-n+1)) \\ &- f(y(k-1), y(k-2), ..., y(k-n)) \end{aligned} \tag{6.23}$$

It is obvious that the stability of equation 6.23 depends upon the behaviour of the difference "$f(y(k), y(k-1), ..., y(k-n+1)) - f(y(k-1), y(k-2), ..., y(k-n))$" and on the selection of the parameter "α". The following computer simulation results obtained from a linear and a nonlinear system, illustrate the performance of the proposed control scheme when the unknown functions "f(.,.)" are smooth functions, and the corresponding parameter "α" is selected in the interval (0, 2). Appendix E contains the simnon computer programs used for implementing the adaptive control scheme.

Firstly, consider the linear system described by the equation

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= par_1\, x_1(k) + par_2\, x_2(k) + par_3\, u(k) \\ y(k) &= x_1(k) \end{aligned} \tag{6.24}$$

where "par_1" and "par_2" are unknown, and "par_3" is assumed to be equals to 1. Obviously, the unknown function "f(.,.)" is the function

$$f(x_1(k), x_2(k)) = par_1\, x_1(k) + par_2\, x_2(k) \tag{6.25}$$

For the sake of the computer simulation the unknown parameters "par_1" and "par_2" are selected equal to -0.4 and 0.1, respectively.

The known reference model will be the following

$$\begin{aligned} x_{m_1}(k+1) &= x_{m_2}(k) \\ x_{m_2}(k+1) &= 0.6\, x_{m_1}(k) + 0.2\, x_{m_2}(k) + u_m(k) \\ y_m(k) &= x_{m_1}(k) \end{aligned} \tag{6.26}$$

where the input signal "$u_m(k)$" is defined by

$$u_m(k) = \begin{cases} 2 & \text{if } kT < 15 \\ Cos(3kT) & \text{otherwise} \end{cases} \tag{6.27}$$

A single layer neural network will be used for the on-line estimation of the unknown function "f(.,.)". Figure 6.7 shows the performance of the neural network-based adaptive control scheme. In this computer simulation, the output of the system was perturbed by a white noise disturbance and yet, the control system was able to maintain the system variables regulated according to the reference model output. The adaptation gain for the neural network was selected by trial and error equals to 0.02. Figure 6.8 illustrates the on-line estimated unknown function, and the output error used to train the neural network estimator.

Finally, we shall consider a nonlinear system described by the equation

$$y(k+1) = f(y(k), y(k-1)) + u(k) \tag{6.28}$$

where the unknown function "f(y(k),y(k-1))" is given by

$$f(y(k), y(k-1)) = \frac{y(k)\,y(k-1)(y(k)+2.5)}{1+y^2(k)+y^2(k-1)} \tag{6.29}$$

This nonlinear plant has been reported by Narendra et al [15] in the context of off-line trained neural network-based adaptive control and by Colina-Morles et al [70] in the context of on-line trained neural network-based adaptive control scheme.

In this case, the reference model is the same described by equation 6.26, with reference input given by the following equation

$$u_m(k) = \begin{cases} 2 & \text{if } kT < 15 \\ 1 & \text{otherwise} \end{cases} \tag{6.30}$$

Similar to the previous example, a single layer neural network with adaptation gain selected by trial and error equals to 0.01 is used for the on-line estimation of the unknown function. Figure 6.9 illustrates the performance of the adaptive control scheme when a white noise perturbation affects the system output. Note that despite the perturbation, the control system keeps the plant output variable regulated. Figure 6.10 sketches the on-line estimation of the unknown function, and the output error between the plant and the reference model.

6.4 SUMMARY

This chapter has included various aspects on the potential applications of on-line trained neural networks to deal with control problems. In particular, a neural network-based internal model control scheme, and a neural network-based adaptive control scheme have been considered and computer simulation results obtained from using these proposed control scheme on unknown dynamic systems have been presented.

The main aspect in the internal model control scheme that has been formulated here is the possibility of obtaining simultaneous on-line estimations of the forward transfer operator and the inverse transfer operator of an unknown dynamic plant that enables the output variable of the plant to be regulated according to the specifications provided by a reference input signal. Both the selection of the neural network adaptation gain, and the parameters of the low-pass filter involved in the

scheme have been selected by trial and error. Only stable open-loop systems have been studied. It has been verified that the stability of the configuration depends on the stability of the plant and the stability of the estimation process used to obtain the FTO and the ITO of the system. Further research work is necessary in order to assess global stability conditions for the operation of the scheme.

On the other hand, an important characteristic of the neural network-based adaptive control scheme that has been presented here is the capability of generating an adaptive control law based on a simultaneous on-line estimation of the parameters of the controller. In other words, we have proposed a direct adaptive method for adjusting the controller parameters based on the error between the plant output and a neural network-based model for the plant . The stability of such method strongly depends on the behaviour of the unknown function. In this study, we have only concentrated in analysing unknown systems that have a smooth behaviour corresponding to the considered reference signals.

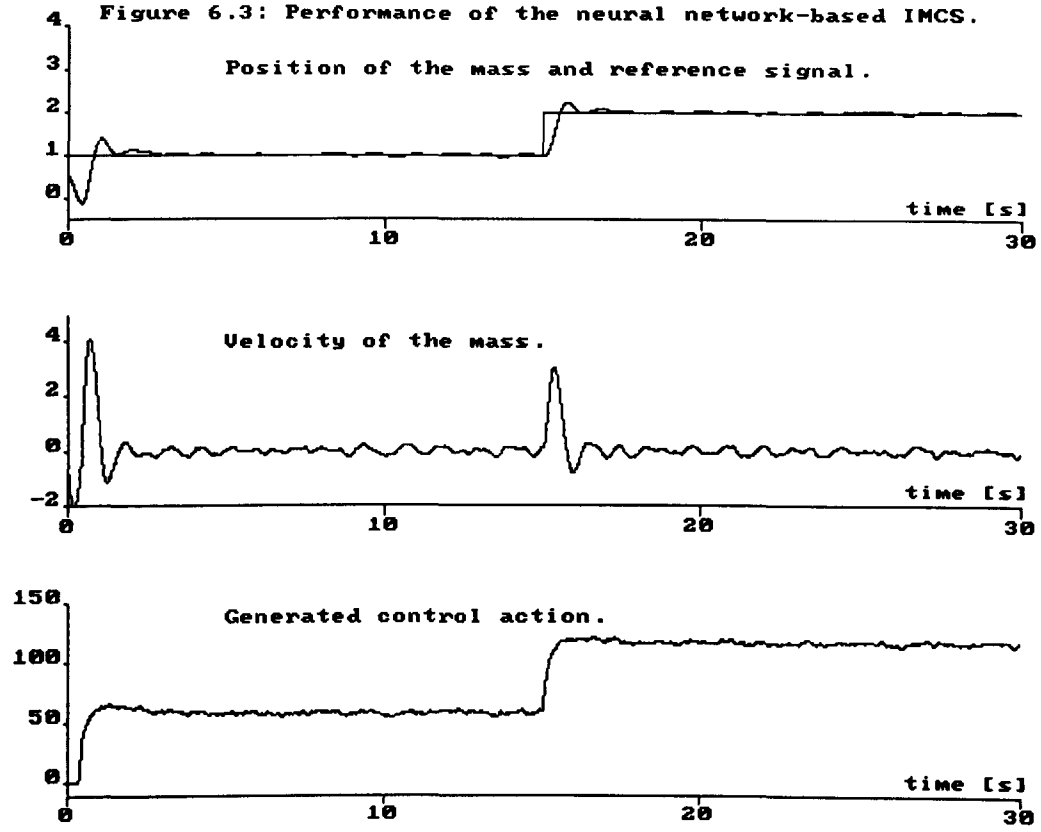


Figure 6.3: Performance of the neural network-based IMCS.

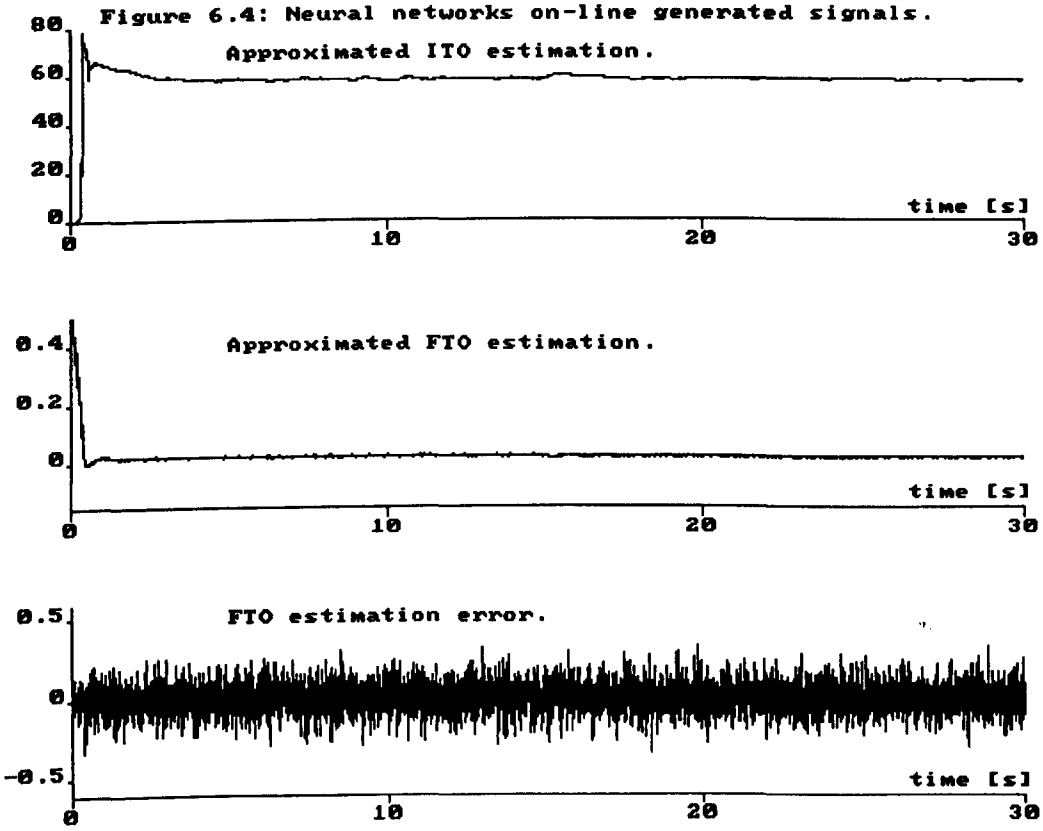


Figure 6.4: Neural networks on-line generated signals.

Figure 6.7: Performance of the NN-based adaptive scheme.

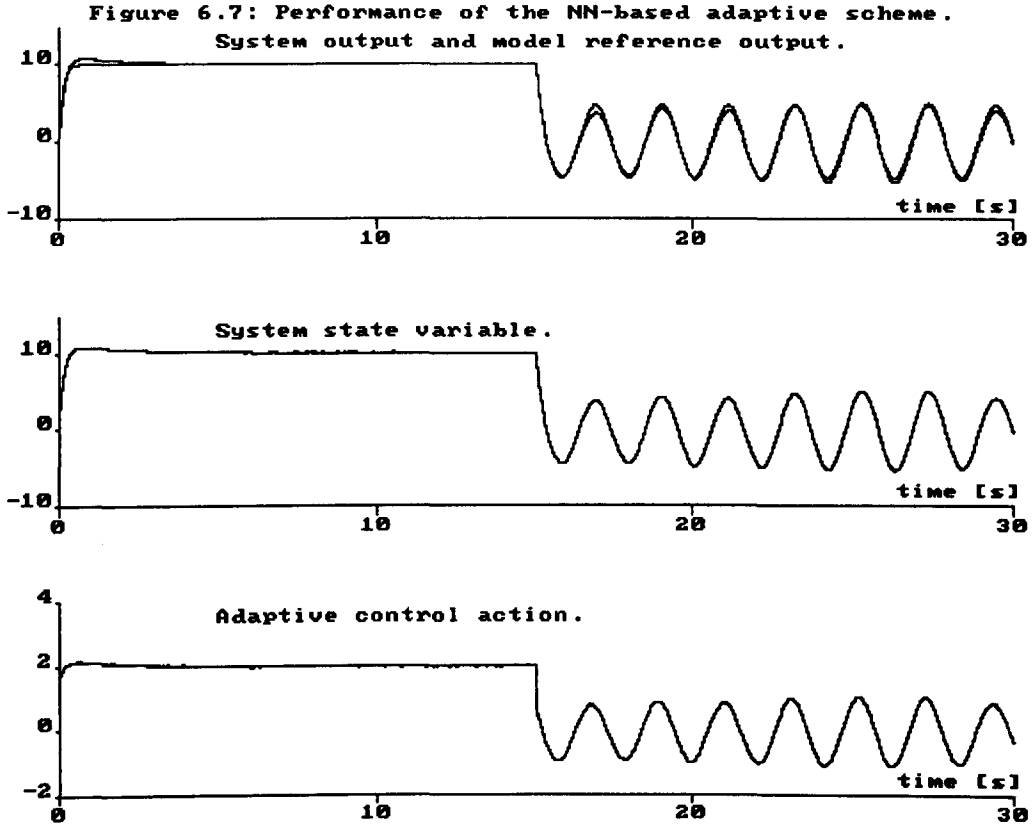
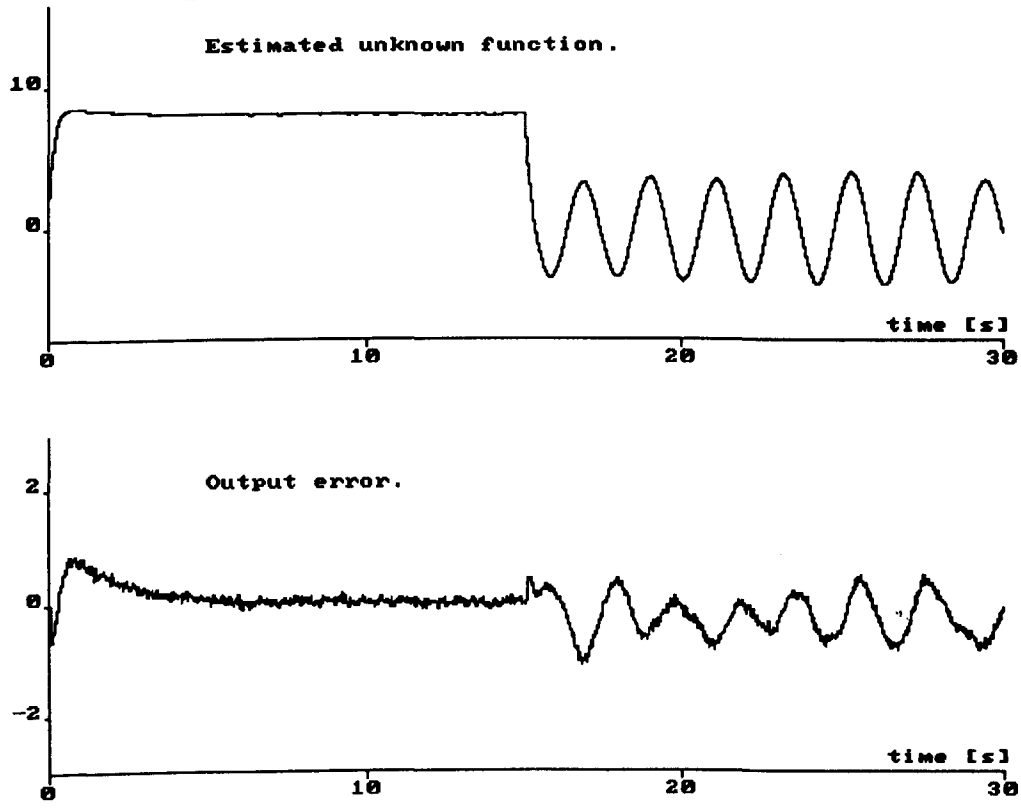


Figure 6.8: Neural network signals.



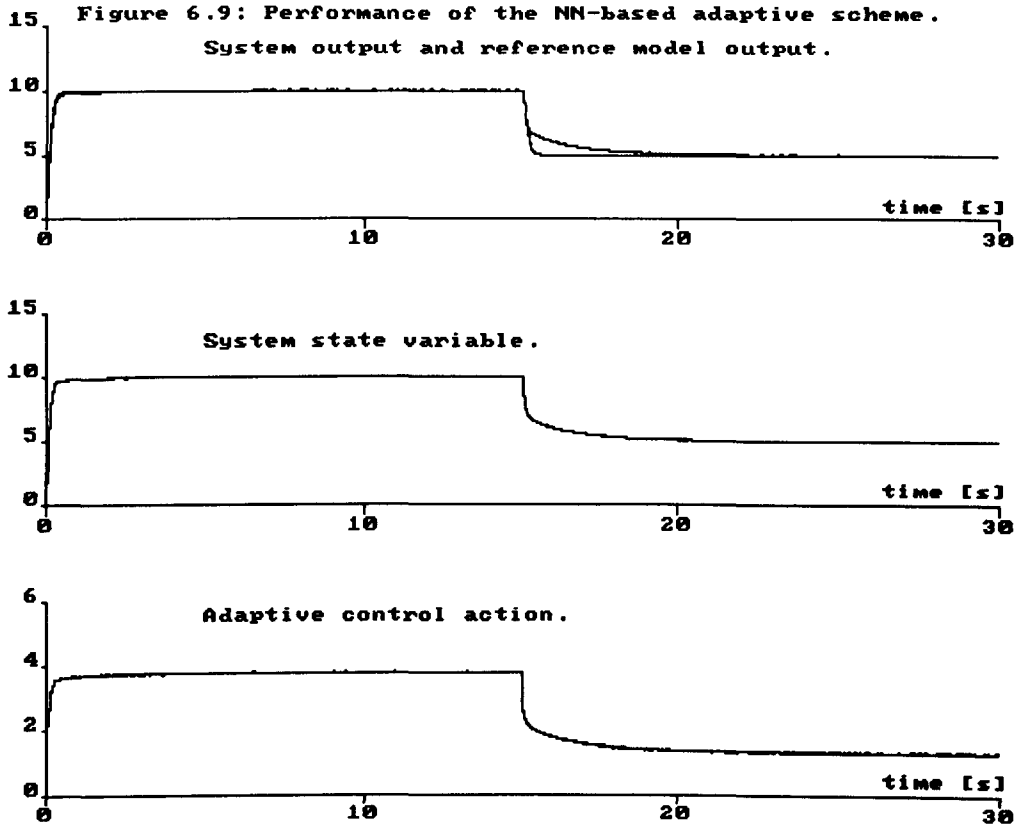


Figure 6.9: Performance of the NN-based adaptive scheme.

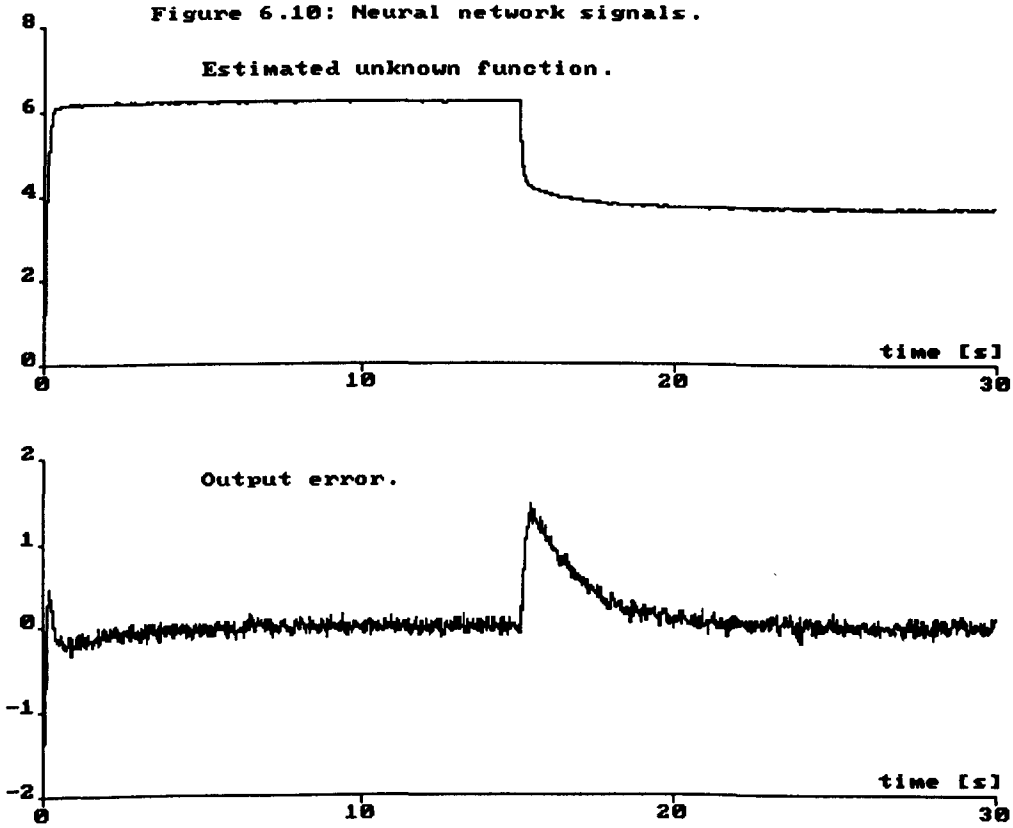


Figure 6.10: Neural network signals.

Chapter 7

CONCLUSIONS

This thesis has presented new results regarding the applicability of artificial feedforward neural networks for the on-line identification and control of unknown dynamic systems.

By way of an introduction, an overview of important features of artificial neural networks has been presented. This overview has included a taxonomy of neural networks in terms of their topological configurations and the type of learning algorithms supported; as well as a categorization of learning algorithms with a description of the Widrow-Hoff delta rule, and the μ-least square algorithm that constitute the basis for the celebrated back-propagation learning algorithm.

The problem of on-line identification of the forward transfer operator of an unknown dynamic system has been addressed in terms of an approximation problem where the approximating function is represented by a neural network with an embedded adaptive rule or learning algorithm that influences the accuracy of the on-line identification process. In this sense, both discrete-time and continuous-time variable structure control-based on-line learning algorithms have been introduced, and the performances of single layer, two layer, and three layer feedforward neural networks in identifying unknown linear and nonlinear dynamic systems have been tested.

The main idea in the learning algorithms that have been presented here is to adapt the weights of the neural network in such a way that a sliding surface along the error function defined by the difference between the teaching signal and the actual output signal of the network is created. This idea, taken from the variable structure control design literature enables the teaching signal to be emulated by the neural network output, and results in a neural network structure that exhibits robustness characteristics with respect to external disturbances affecting the network's signals.

In the continuous-time versions of the learning algorithms that have been pre-

sented here, the output of the neural network is able to track its teaching signal in a finite time. This convergence characteristic, together with the robustness features, and the boundedness of solutions provided by the on-line trained neural networks has been mathematically proved; and computer programs written for the Simnon computer language have been included.

The discrete-time versions of the learning algorithms that have been formulated in this thesis, included a generalization of the learning algorithm proposed by Sira-Ramirez et al [59] that took into consideration time-varying neural networks' input and teaching signals. Similar as in the continuous-time cases, the discrete-time version of the algorithms have been developed for single layer, two layer, and three layer neural networks; and mathematical proofs on the asymptotic convergence characteristics, boundedness of solutions, and robustness features with respect to external perturbations on the networks' signals have been reported.

The on-line forward transfer operator identification problems for continuous-time and discrete-time dynamic systems that have been considered in this thesis have been solved by proposing a neural network-based identification scheme where the input signals to the neural network have been taken from a neighborhood of the input signal to the unknown system, and the teaching signal for the neural network has been selected equal to the output signal of the unknown system. The on-line variable structure control-based learning algorithms have been used to adapt the weights of the neural networks involved in the scheme. In particular, the approximation capabilities of one layer, two layer, and three layer feedforward neural networks to give on-line estimations of the forward transfer operator of unknonw dynamic systems have been tested by means of computer simulations. In these computer simulations, it has been contemplated the effects of external bounded perturbations (white noise) influencing the signals of the neural networks. The best performance has been obtained using three layer neural networks adapted either with the continuous-time or the discrete-time learning algorithm.

In the thesis, the on-line inverse transfer operator identification problems for continuous-time and discrete-time unknown dynamic systems have also been studied. In this case, the teaching and input signals to the neural networks have been selected equal to the input signal and from a neighborhood of the output signal of the unknown plant, respectively. The best performance has been attained using three layer neural networks.

In the direct inverse dynamic control problem that has been solved here, the applied control action to the unknown plant has been generated from the output of a feedforward controller plus the filtered output of a feedback controller. The task of

the feedforward controller has been to provide an approximation of the inverse transfer operator of the system that enables an identity mapping between the unknown dynamic system and the approximation to be created. The feedback controller task on the other hand, has been to compensate for unstable poles of the system and to enable the inverse transfer operator approximation to be implemented when the unknown dynamic system forward transfer operator was strictly proper.

Different from other neural network-based direct inverse dynamic control schemes [7, 76], in the scheme that has been proposed here the neural network estimator has provided on-line approximations of the inverse transfer operator of the unknown plant, and has included a feedback loop. The incorporation of these features has resulted in a control scheme that exhibits robustness characteristics with respect to bounded external perturbations affecting the output of the unknown system.

For continuous-time linear systems, it has been mathematically shown the connection between system invertibility and controllability and, in the nonlinear system case the condition for plant invertibility has been loosely interpreted as satisfying a local reachability condition.

The performance of the control scheme has been tested by computer simulations using a spring-mass-damper system, and a dc motor driven inverted pendulum system. In both cases, the selection of the adaptation gain of the algorithms that have been used to adjust the weights of the neural networks, as well as the parameters of the filter involved in the feedback component of the control action have been done on a trial and error basis.

The main aspect in the neural network-based internal model control scheme that has been formulated in this work is the possibility of obtaining simultaneous on-line estimations of the FTO and the ITO of unknown dynamic systems that enables the output variable of the plant to be regulated according to the specifications provided by a reference input signal. The performance of the proposed control scheme has been tested by computer simulations using a stable open-loop unknown plant which output signal has been corrupted by white noise. Despite the output disturbance, the control system has been able to keep the unknown system output regulated. It has been verified that the stability of the configuration depends on the stability of the unknown plant and the stability of the estimation processes of the FTO and ITO.

An important feature of the neural network-based adaptive control scheme that has been presented in this thesis is the capability of generating an adaptive control law based on a simultaneous on-line estimation of the parameters of the controller. In this control scheme, identification and control have been performed

simultaneously, and it has been empirically proved that the configuration is stable when the unknown system being considered has a smooth behaviour.

There is significant scope for further development along the ideas presented in this work. The identification schemes and the direct inverse dynamic control scheme presented in the thesis have only been used on single-input single-output dynamic systems. The extension of these schemes to consider multiple-input multiple-output systems is a natural way of action for further research work. In particular, knowing the fact that the variable structure control-based learning algorithms that have been proposed are general enough to be used for on-line adaptation of multiple-output neural networks.

Also an open area for further research work is the study of the applicability and conditions for the stability of the neural network-based IMCS to consider unstable open-loop systems, and nonlinear dynamic systems in general. Similarly, it is important to establish general conditions for the global stability of the proposed neural network-based adaptive control scheme.

Appendix A

Single Layer Neural Networks

A.1 Continuous-Time Computer Program

Computer program written for the "Simnon" computer language used to implement the continuous-time variable structure control-based learning algorithm for a single layer neural network with 4 input nodes. Note that the function "SIGN(E)" is implemented in terms of $\frac{E}{(ABS(E)+DELTA)}$, where "DELTA" is a very small positive value.

```
CONTINUOUS SYSTEM NN1
"LEARNING ALGORITHM FOR A SINGLE LAYER NEURAL NETWORK US-
ING VSC
INPUT YD U UD1 UD2 UD3
OUTPUT EN1
STATE W1 W2 W3 W4
DER DW1 DW2 DW3 DW4
DW1=V1
DW2=V2
DW3=V3
DW4=V4
E=YD-YN
YN=W1*U+W2*UD1+W3*UD2+W4*UD3
MO=1/MU
V1=MO*U*KN1*(E/(ABS(E)+ DELTA))
V2=MO*UD1*KN1*(E/(ABS(E)+ DELTA))
V3=MO*UD2*KN1*(E/(ABS(E)+ DELTA))
```

```
V4=MO*UD3*KN1*(E/(ABS(E)+ DELTA))
MU=U*U+UD1*UD1+UD2*UD2+UD3*UD3
WT1=W1+W2+W3+W4
EN1=E
DELTA: 0.005
KN1:1
END
```

A.2 Discrete-Time Computer Program

Computer program written for the "Simnon" computer language used to implement the discrete-time variable structure control-based learning algorithm for a single layer neural network 3 input nodes.

```
DISCRETE SYSTEM SINEURON
"LEARNING ALGORITHM FOR A SINGLE LAYER NEURAL NETWORK
INPUT Y YD1 UD U11 U12 U1D U11D U12D
OUTPUT YO
STATE W11 W12 W13 W1B
NEW NW11 NW12 NW13 NW1B
TIME T
TSAMP TS
MO=UD*UD+U11*U11+U12*U12+B*B
MOI=1/MO
TER=(UD-U1D)*W11+(U11-U11D)*W12+(U12-U12D)*W13
NW11=W11+MOI*UD*(A*E+Y-YD1-TER)
NW12=W12+MOI*U11*(A*E+Y-YD1-TER)
NW13=W13+MOI*U12*(A*E+Y-YD1-TER)
NW1B=W1B+MOI*B*(A*E+Y-YD1-TER)
E=YD1-YO
YO=W11*U1D+W12*U11D+W13*U12D+W1B*B
WT=W11+W12+W13
TS=T+TT
TT:0.01
A:0.6
B:1.        END
```

Appendix B

Two Layer Neural Networks

B.1 Continuous-time Computer Program

Computer program written for the "Simnon" computer language to implement the continuous-time variable structure control-based learning algorithm for a two layer neural network with 3 input nodes, and 1 output node.

```
CONTINUOUS SYSTEM TWOLAYER
"CONTINUOUS-TIME TWO LAYER NEURAL NETWORK
INPUT U1 U2 U3 U4 YD
OUTPUT WT E
STATE WI11 WI12 WI13 WIB1
STATE WI21 WI22 WI23 WIB2
STATE WI31 WI32 WI33 WIB3
STATE WI41 WI42 WI43
STATE W111 W121 W131
DER DWI11 DWI12 DWI13 DWIB1
DER DWI21 DWI22 DWI23 DWIB2
DER DWI31 DWI32 DWI33 DWIB3
DER DWI41 DWI42 DWI43
DER DW111 DW121 DW131
DWI11=UI11
DWI12=UI12
DWI13=UI13
DWIB1=UIB1
DWI21=UI21
```

```
DWI22=UI22
DWI23=UI23
DWIB2=UIB2
DWI31=UI31
DWI32=UI32
DWI33=UI33
DWIB3=UIB3
DWI41=UI41
DWI42=UI42
DWI43=UI43
DW111=U111
DW121=U121
DW131=U131
MO=U1*U1+U2*U2+U3*U3+U4*U4+B1*B1
MIO=1/MO
Y11=WI11*U1+WI21*U2+WI31*U3+WI41*U4+WIB1*B1
Y12=WI12*U1+WI22*U2+WI32*U3+WI42*U4+WIB2*B1
Y13=WI13*U1+WI23*U2+WI33*U3+WI43*U4+WIB3*B1
GAMMA1=1/(1+EXP(-Y11))
GAMMA2=1/(1+EXP(-Y12))
GAMMA3=1/(1+EXP(-Y13))
UI11=MIO*U1/Y11*EXP(-Y11)*GAMMA1
UI12=MIO*U1/Y12*EXP(-Y12)*GAMMA2
UI13=MIO*U1/Y13*EXP(-Y13)*GAMMA3
UI21=MIO*U2/Y11*EXP(-Y11)*GAMMA1
UI22=MIO*U2/Y12*EXP(-Y12)*GAMMA2
UI23=MIO*U2/Y13*EXP(-Y13)*GAMMA3
UI31=MIO*U3/Y11*EXP(-Y11)*GAMMA1
UI32=MIO*U3/Y12*EXP(-Y12)*GAMMA2
UI33=MIO*U3/Y13*EXP(-Y13)*GAMMA3
UI41=MIO*U4/Y11*EXP(-Y11)*GAMMA1
UI42=MIO*U4/Y12*EXP(-Y12)*GAMMA2
UI43=MIO*U4/Y13*EXP(-Y13)*GAMMA3
UIB1=MIO*B1/Y11*EXP(-Y11)*GAMMA1
UIB2=MIO*B1/Y12*EXP(-Y12)*GAMMA2
UIB3=MIO*B1/Y13*EXP(-Y13)*GAMMA3
M1=GAMMA1*GAMMA1+GAMMA2*GAMMA2+GAMMA3*GAMMA3
```

```
M11=1/M1
YO=W111*GAMMA1+W121*GAMMA2+W131*GAMMA3
E=YD-YO
U111=-W111+M11*GAMMA1*K*E
U121=-W121+M11*GAMMA2*K*E
U131=-W131+M11*GAMMA3*K*E
K:10.    B1:1.    END
```

B.2 Discrete-Time Computer Program

Computer program written for the "Simnon" computer language used to implement the discrete-time variable structure control-based learning algorithm for a two layer neural network with 3 input nodes, and 1 output node.

```
DISCRETE SYSTEM LAYER2
"DISCRETE-TIME TWO LAYER NEURAL NETWORK
INPUT Y1 Y2 UD UDD
OUTPUT YN
TIME T
TSAMP TS
STATE WI11 WI12 WI13
STATE WI21 WI22 WI23
STATE W111 W121 W131
NEW NWI11 NWI12 NWI13
NEW NWI21 NWI22 NWI23
NEW NW111 NW121 NW131
NWI11=WI11+UI11
NWI12=WI12+UI12
NWI13=WI13+UI13
NWI21=WI14+UI14
NWI22=WI15+UI15
NWI23=WI16+UI16
NW111=W111+U111
NW121=W121+U121
NW131=W131+U131
Y11=WI11*UD+WI21*UDD
```

```
Y12=WI12*UD+WI22*UDD
Y13=WI13*UD+WI23*UDD
A=UD*SIGN(UD)+UDD*SIGN(UDD)
UI11=-2*(SIGN(UD)*Y11)/A
UI12=-2*(SIGN(UD)*Y12)/A
UI13=-2*(SIGN(UD)*Y13)/A
UI21=-2*(SIGN(UDD)*Y11)/A
UI22=-2*(SIGN(UDD)*Y12)/A
UI23=-2*(SIGN(UDD)*Y13)/A
Z11=SIGN(Y11)
Z12=SIGN(Y12)
Z13=SIGN(Y13)
U111=-2*W111-(A1*EN+P1-P2)*Z11/N1
U121=-2*W121-(A1*EN+P1-P2)*Z12/N1
U131=-2*W131-(A1*EN+P1-P2)*Z13/N1
YN=W111*Z11+W121*Z12+W131*Z13
EN=P1-YNN
P1=Y2-UD
P2=Y1-UDD
TS=T+TN
A1:1
TN:0.02
N1:3
END
```

Appendix C

Three Layer Neural Networks

C.1 Continuous-time Computer Program

Computer program written for the "Simnon" computer language used to implement the continuous-time variable structure control-based learning algorithm for a three layer neural network with 3 input nodes, 3 hidden nodes, and 1 output node.

```
CONTINUOUS SYSTEM TRILAYER
"CONTINUOUS-TIME THREE LAYER NEURAL NETWORK
INPUT U1 U2 U3 U4 YD
STATE WI11 WI12 WI13 WIB1
STATE WI21 WI22 WI23 WIB2
STATE WI31 WI32 WI33 WIB3
STATE WI41 WI42 WI43
STATE W211 W212 W221 W222 W231 W232
STATE W111 W121
DER DWI11 DWI12 DWI13 DWIB1
DER DWI21 DWI22 DWI23 DWIB2
DER DWI31 DWI32 DWI33 DWIB3
DER DWI41 DWI42 DWI43
DER DW211 DW212 DW221 DW222 DW231 DW232
DER DW111 DW121
DWI11=UI11
DWI12=UI12
DWI13=UI13
DWIB1=UIB1
```

```
DWI21=UI21
DWI22=UI22
DWI23=UI23
DWIB2=UIB2
DWI31=UI31
DWI32=UI32
DWI33=UI33
DWIB3=UIB3
DWI41=UI41
DWI42=UI42
DWI43=UI43
DW211=U211
DW212=U212
DW221=U221
DW222=U222
DW231=U231
DW232=U232
DW111=U111
DW121=U121
MO=U1*U1+U2*U2+U3*U3+U4*U4+B1*B1
MIO=1/MO
Y11=WI11*U1+WI21*U2+WI31*U3+WI41*U4+WIB1*B1
Y12=WI12*U1+WI22*U2+WI32*U3+WI42*U4+WIB2*B1
Y13=WI13*U1+WI23*U2+WI33*U3+WI43*U4+WIB3*B1
GAMMA1=1/(1+EXP(-Y11))
GAMMA2=1/(1+EXP(-Y12))
GAMMA3=1/(1+EXP(-Y13))
UI11=MIO*U1/Y11*EXP(-Y11)*GAMMA1
UI12=MIO*U1/Y12*EXP(-Y12)*GAMMA2
UI13=MIO*U1/Y13*EXP(-Y13)*GAMMA3
UI21=MIO*U2/Y11*EXP(-Y11)*GAMMA1
UI22=MIO*U2/Y12*EXP(-Y12)*GAMMA2
UI23=MIO*U2/Y13*EXP(-Y13)*GAMMA3
UI31=MIO*U3/Y11*EXP(-Y11)*GAMMA1
UI32=MIO*U3/Y12*EXP(-Y12)*GAMMA2
UI33=MIO*U3/Y13*EXP(-Y13)*GAMMA3
UI41=MIO*U4/Y11*EXP(-Y11)*GAMMA1
```

```
UI42=MIO*U4/Y12*EXP(-Y12)*GAMMA2
UI43=MIO*U4/Y13*EXP(-Y13)*GAMMA3
UIB1=MIO*B1/Y11*EXP(-Y11)*GAMMA1
UIB2=MIO*B1/Y12*EXP(-Y12)*GAMMA2
UIB3=MIO*B1/Y13*EXP(-Y13)*GAMMA3
M1=GAMMA1*GAMMA1+GAMMA2*GAMMA2+GAMMA3*GAMMA3
M11=1/M1
Y21=W211*GAMMA1+W221*GAMMA2+W231*GAMMA3
Y22=W212*GAMMA1+W222*GAMMA2+W232*GAMMA3
GAMMA4=1/(1+EXP(-Y21))
GAMMA5=1/(1+EXP(-Y22))
U211=-W211+M11*GAMMA1/Y21*EXP(-Y21)*GAMMA4
U212=-W212+M11*GAMMA1/Y22*EXP(-Y22)*GAMMA5
U221=-W221+M11*GAMMA2/Y21*EXP(-Y21)*GAMMA4
U222=-W222+M11*GAMMA2/Y22*EXP(-Y22)*GAMMA5
U231=-W231+M11*GAMMA3/Y21*EXP(-Y21)*GAMMA4
U232=-W232+M11*GAMMA3/Y22*EXP(-Y22)*GAMMA5
M2=GAMMA4*GAMMA4+GAMMA5*GAMMA5
M22=1/M2
E=YD-YO
YO=W111*GAMMA4+W121*GAMMA5
U111=-W111+M22*GAMMA4*K*E
U121=-W121+M22*GAMMA5*K*E
K:10
B1:1
END
```

C.2 Discrete-Time Neural Network

Computer program written for the "Simnon" computer language used to implement the discrete-time variable structure control-based learning algorithm for a three layer neural network with 5 inputs nodes, 3 hidden nodes, and 1 output node.,,

```
DISCRETE SYSTEM LAYER3
"DISCRETE-TIME THREE LAYER NEURAL NETWORK
INPUT Y1 Y2 UD UDD R
OUTPUT YN
```

```
TIME T
TSAMP TS
STATE WI11 WI12 WI13 WI14 WI15 WI16
STATE W211 W212 W213 W221 W222 W223
STATE W231 W232 W233 W241 W242 W243
STATE W251 W252 W253 W261 W262 W263
STATE W111 W121 W131
NEW NWI11 NWI12 NWI13 NWI14 NWI15 NWI16
NEW NW211 NW212 NW213 NW221 NW222 NW223
NEW NW231 NW232 NW233 NW241 NW242 NW243
NEW NW251 NW252 NW253 NW261 NW262 NW263
NEW NW111 NW121 NW131
NWI11=WI11+UI11
NWI12=WI12+UI12
NWI13=WI13+UI13
NWI14=WI14+UI14
NWI15=WI15+UI15
NWI16=WI16+UI16
NW211=W211+U211
NW212=W212+U212
NW213=W213+U213
NW221=W221+U221
NW222=W222+U222
NW223=W223+U223
NW231=W231+U231
NW232=W232+U232
NW233=W233+U233
NW241=W241+U241
NW242=W242+U242
NW243=W243+U243
NW251=W251+U251
NW252=W252+U252
NW253=W253+U253
NW261=W261+U261
NW262=W262+U262
NW263=W263+U263
NW111=W111+U111
```

```
NW121=W121+U121
NW131=W131+U131
X=R
Y21=WI11*X
Y22=WI12*X
Y23=WI13*X
Y24=WI14*X
Y25=WI15*X
Y26=WI16*BIAS
AB=BIAS*SIGN(BIAS)
A11B=SIGN(BIAS)
A=X*SIGN(X)
A11=SIGN(X)
UI11=-2*(A11*Y21)/A
UI12=-2*(A11*Y22)/A
UI13=-2*(A11*Y23)/A
UI14=-2*(A11*Y24)/A
UI15=-2*(A11*Y25)/A
UI16=-2*(A11B*Y26)/AB
Z21=SIGN(Y21)
Z22=SIGN(Y22)
Z23=SIGN(Y23)
Z24=SIGN(Y24)
Z25=SIGN(Y25)
Z26=SIGN(Y26)
U211=-2*(Z21*Y11)/N2
U212=-2*(Z21*Y12)/N2
U213=-2*(Z21*Y13)/N2
U221=-2*(Z22*Y11)/N2
U222=-2*(Z22*Y12)/N2
U223=-2*(Z22*Y13)/N2
U231=-2*(Z23*Y11)/N2
U232=-2*(Z23*Y12)/N2
U233=-2*(Z23*Y13)/N2
U241=-2*(Z24*Y11)/N2
U242=-2*(Z24*Y12)/N2
U243=-2*(Z24*Y13)/N2
```

```
U251=-2*(Z25*Y11)/N2
U252=-2*(Z25*Y12)/N2
U253=-2*(Z25*Y13)/N2
U261=-2*(Z26*Y11)/N2
U262=-2*(Z26*Y12)/N2
U263=-2*(Z26*Y13)/N2
Y11=W211*Z21+W221*Z22+W231*Z23+W241*Z24+W251*Z25+W261*Z26
Y12=W212*Z21+W222*Z22+W232*Z23+W242*Z24+W252*Z25+W262*Z26
Y13=W213*Z21+W223*Z22+W233*Z23+W243*Z24+W253*Z25+W263*Z26
Z11=SIGN(Y11)
Z12=SIGN(Y12)
Z13=SIGN(Y13)
YNN=W111*Z11+W121*Z12+W131*Z13
U111=(A1*EN+P1-P2)*Z11/N1
U121=(A1*EN+P1-P2)*Z12/N1
U131=(A1*EN+P1-P2)*Z13/N1
EN=P1-YNN
P1=Y2-UD
P2=Y1-UDD
YN=YNN+GAIN*EN
TS=T+TN
A1:1
GAIN:0.831
TN:0.02
N2:6
N1:3
BIAS:1
END
```

Appendix D

Computer Programs For Implementing The IMCS

The following computer programs may be used to implement the neral network-based IMCS in a digital computer.

```
CONTINUOUS SYSTEM FTF
"UNKNOWN LINEAR SYSTEM. (SPRING-MASS-DAMPER)
INPUT U RN
OUTPUT Y1 Y2
STATE X1 X2
DER DX1 DX2
DX1=X2
DX2=-(R/M)*X1-(C/M)*X2+(1/M)*U
Y1=X1+SW*RN
Y2=X2
M:1.459
R:58.37
C:5.837
SW:0
x1:0.5
x2:0.2
END

CONTINUOUS SYSTEM CONFTF
"IMC CONTROLLER FOR FTF
INPUT WITO YR EE DV
```

```
OUTPUT U V
STATE Z1 Z2
DER DZ1 DZ2
DZ1=Z2
DZ2=-P1*Z1-P2*Z2+P3*V+P4*DV
ER=YR-EE
V=K1*WITO*ER
P1=1/(TAO2*TAO2)
P2=2/TAO2
P3=KF*P1
P4=TAO1*P3
U=Z1
K1:1
TAO1:0.2
TAO2:0.2
KF:1
END

     DISCRETE SYSTEM RETARDO
"DELAY FOR U AND Y1
INPUT U Y1
OUTPUT U1 U2 U3 U4 Y11 Y12 Y13 Y14
TIME T
TSAMP TS
STATE UD1 UD2 UD3 UD4 YD1 YD2 YD3 YD4
NEW NUD1 NUD2 NUD3 NUD4 NYD1 NYD2 NYD3 NYD4
NUD1=U
NUD2=UD1
NUD3=UD2
NUD4=UD3
NYD1=Y1
NYD2=YD1
NYD3=YD2
NYD4=YD3
U1=UD1
U2=UD2
U3=UD3
```

```
U4=UD4
Y11=YD1
Y12=YD2
Y13=YD3
Y14=YD4
TS=T+TR
TR:0.01
END

      CONTINUOUS SYSTEM SLITO
"LEARNING ALGORITHM USING VSC
INPUT YD U UD1 UD2 UD3
OUTPUT WITO
STATE W1 W2 W3 W4 W5
DER DW1 DW2 DW3 DW4 DW5
DW1=V1
DW2=V2
DW3=V3
DW4=V4
DW5=V5
E=YD-YN
YN=W1*U+W2*UD1+W3*UD2+W4*UD3+W5*B
MO=1/(B2+MU)
V1=MO*U*KI*(E/(ABS(E)+ DELTA))
V2=MO*UD1*KI*(E/(ABS(E)+ DELTA))
V3=MO*UD2*KI*(E/(ABS(E)+ DELTA))
V4=MO*UD3*KI*(E/(ABS(E)+ DELTA))
V5=MO*B*KI*(E/(ABS(E)+ DELTA))
MU=U*U+UD1*UD1+UD2*UD2+UD3*UD3
B2=B*B
WITO=W1+W2+W3+W4+W5
B:0
DELTA:0.005
KI:5
W1:0.1
W2:0.1
W3:0.1
```

```
END

     CONTINUOUS SYSTEM SLFTO
"LEARNING ALGORITHM USING VSC
INPUT YD U UD1 UD2 UD3 Y1
OUTPUT EE
STATE W1 W2 W3 W4 W5
DER DW1 DW2 DW3 DW4 DW5
DW1=V1
DW2=V2
DW3=V3
DW4=V4
DW5=V5
E=YD-YN
YN=W1*U+W2*UD1+W3*UD2+W4*UD3+W5*B
MO=1/(B2+MU)
V1=MO*U*KF*(E/(ABS(E)+ DELTA))
V2=MO*UD1*KF*(E/(ABS(E)+ DELTA))
V3=MO*UD2*KF*(E/(ABS(E)+ DELTA))
V4=MO*UD3*KF*(E/(ABS(E)+ DELTA))
V5=MO*B*KF*(E/(ABS(E)+ DELTA))
MU=U*U+UD1*UD1+UD2*UD2+UD3*UD3
B2=B*B
WFTO=W1+W2+W3+W4+W5
EE=E "Y1-WFTO*U
B:0
DELTA:0.005
KF:40
W1:0.1
W2:0.1
W3:0.1
END

     DISCRETE SYSTEM RANDOM
"PSEUDO RANDOM NUMBER GENERATOR
OUTPUT RN
TIME T
```

```
TSAMP TS
RN=C*NORM(T)+MEAN
TS=T+TR
TR:0.01
MEAN:0
C:0.05
END

     DISCRETE SYSTEM DERV
"DETERMINATION OF THE DERIVATIVE OF V
INPUT V
OUPUT DV
TIME T
TSAMP TS
STATE V1
NEW NV1
NV1=V
DV=(V-V1)/TDV
TS=T+TDV
TDV:0.02
END

     CONNECTING SYSTEM IMCFTF
"CONNECTION FOR FTF, CONFTF, RETARDO, SLITO, SLFTO, RANDOM,
DERV.
TIME T
U[FTF]=U[CONFTF]
RN[FTF]=RN[RANDOM]
Y1[RETARDO]=Y1[FTF]
U[RETARDO]=U[CONFTF]
YD[SLITO]=U1[RETARDO]
U[SLITO]=Y11[RETARDO]+OFF1
UD1[SLITO]=Y11[RETARDO]-OFF2
UD2[SLITO]=Y11[RETARDO]+OFF3
UD3[SLITO]=Y11[RETARDO]-OFF4
YD[SLFTO]=Y1[FTF]
U[SLFTO]=U1[RETARDO]+OF1
```

UD1[SLFTO]=U2[RETARDO]+OF1
UD2[SLFTO]=U3[RETARDO]-OF1
UD3[SLFTO]=U4[RETARDO]+OF1
Y1[SLFTO]=Y1[FTF]
YR[CONFTF]=REF
WITO[CONFTF]=WITO[SLITO]
EE[CONFTF]=EE[SLFTO]
REF=IF T¡T1 THEN R1 ELSE R2
DV[CONFTF]=DV[DERV]
V[DERV]=V[CONFTF]
T1:15
R1:1
R2:2
OFF1:0.1E-4
OFF2:0.1E-4
OFF3:0.1E-4
OFF4:0.1E-4
OF1:1
END

Appendix E

Computer Programs For Implementing The Adaptive Control Scheme

The following computer programs may be used to implement the neural network-based adaptive control scheme in a digital computer.

```
DISCRETE SYSTEM LINEAR
"PLANTA LINEAL EN TIEMPO DISCRETO
INPUT U RN
OUTPUT Y1 Y2
TIME T
TSAMP TS
STATE X1 X2
NEW NX1 NX2
NX1=X2
NX2=F+U
F=PAR1*X1+PAR2*X2
Y1=X1+CT1*RN
Y2=X2
TS=T+TP
TP:0.02
PAR1:0.2
PAR2:0.6
CT1:0
END
```

```
      DISCRETE SYSTEM NONLIN
"PLANTA NOLINEAL EN TIEMPO DISCRETO
INPUT U RN
OUTPUT Y1 Y2
TIME T
TSAMP TS
STATE X1 X2
NEW NX1 NX2
NX1=X2
NX2=F+U
F=(X1*X2)*(X2+2.5)/(1+X2*X2+X1*X1)
Y1=X1+CT1*RN
Y2=X2
TS=T+TP
TP:0.02
CT1:0
END

      DISCRETE SYSTEM CONTRO
"ADAPTIVE CONTROL FOR UNKNOWN PLANT.
INPUT YN Y1 Y2 R
OUTPUT U
TIME T
TSAMP TS
U=-YN+YM
YM=0.6*Y2+0.2*Y1+R
TS=T+TC
TC:0.02
END

      DISCRETE SYSTEM ONELAYER
"SINGLE LAYER NEURAL NETWORK
INPUT YD U UU UU1 UU2
OUTPUT YO WT
TIME T
TSAMP TS
```

```
STATE W11 W12 W13 XN
NEW NW11 NW12 NW13 NXN
NW11=W11+MOI*SIGN(UU)*(A1*E)
NW12=W12+MOI*SIGN(UU1)*(A1*E)
NW13=W13+MOI*SIGN(UU2)*(A1*E)
NXN=YO+U+CT*E
MO=UU*SIGN(UU)+UU1*SIGN(UU1)+UU2*SIGN(UU2)
MOI=1/MO
E=YD-XN
YO=W11*UU+W12*UU1+W13*UU2
wt=w11+w12+w13
TS=T+TT
CT:0
TT:0.01
A1:0.6
w11:0.5
w12:0.3
w13:0.1
END

     DISCRETE SYSTEM MODELO
"REFERENCE MODEL FOR ADAPTIVE CONTROL
INPUT R Y1
TIME T
TSAMP TS
STATE X1M X2M
NEW NX1M NX2M
NX1M=X2M
NX2M=0.2*X1M+0.6*X2M+R
EOUT=Y1-X1M
TS=T+TM
TM:0.02
END

     DISCRETE SYSTEM RANDOM
"PSEUDO RANDOM NUMBER GENERATOR
OUTPUT RN
```

```
TIME T
TSAMP TS
RN=C*NORM(T)+MEAN
TS=T+TR
TR:0.01
MEAN:0
C:0.05
END

     CONNECTING SYSTEM ENLACE
"CONNECTING SYSTEM FOR NONLIN, CONTRO, MODELO, ONELAYER,
RANDOM
TIME T
U[NONLIN]=U[CONTRO]
RN[NONLIN]=RN[RANDOM]
YN[CONTRO]=YO[ONELAYER]
Y1[CONTRO]=Y1[NONLIN]
Y2[CONTRO]=Y2[NONLIN]
R[CONTRO]=R[MODELO]
YD[ONELAYER]=Y2[NONLIN]
U[ONELAYER]=U[CONTRO]
UU[ONELAYER]=Y2[NONLIN]+OF1
UU1[ONELAYER]=Y2[NONLIN]+OF2
UU2[ONELAYER]=Y2[NONLIN]+OF3
Y1[MODELO]=Y1[NONLIN]
R[MODELO]=IF T¿T0 THEN R1 ELSE R2
T0:15
R1:1
R2:2
OF1:0.05
OF2:0.1
OF3:0.13
END
```

Bibliography

[1] E. Davalo; P. Naïm, "Neural Networks".
MacMillan Education LTD, London, 1991.

[2] W. S. McCulloch; W. Pitts, "A Logical Calculus of the Ideas Imminent in Nervous Activity".
Bulletin of Mathematical Biophysics, Number 5, pp 115-133, 1943.

[3] F. Rosenblatt, "The Perceptron, a Perceiving and Recognizing Automation".
Cornell Aeronautical Laboratory Report Number 85-460-1, Jan. 1957.

[4] G. Nagy, "Neural Networks - Then and Now".
IEEE Trans. on Neural Networks, Vol. 2, Number 2, pp 316-318, March 1991.

[5] M. L. Minsky; S. A. Papert, "Perceptrons: An Introduction to Computational Geometry".
The MIT Press, Cambridge, MA., 1969. An expanded edition was published by the MIT Press in 1988.

[6] N. Wiener, "Cybernetics: Or Control and Communication in the Animal and the Machine".
The Mit Press, Cambridge, MA., 1948.

[7] W. T. Miller; R. S. Sutton; P. W. Werbos, "Neural Networks for Control".
Chapter 1, By A. G. Barto, pp 5-58, Mit Press, Cambridge, MA., 1990.

[8] S. Grossberg, "Adaptive Pattern Classification and Universal Recording: Parallel Development and Coding of Neural Feature Detectors".
Biological Cybernetics, Number 23, pp 121-134, 1976.

[9] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities".
Proceedings of the National Academy of Sciences, Vol. 79, pp 2554-2558, USA, 1982.

[10] P. J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavior Sciences".
Ph.D. Thesis, Harvard University, Committee on Applied Mathematics, USA., 1974.

[11] Y. LeCum, "Une Procedure D'apprentissage Pour Reseau a Sequil Assymetrique".
Proceedings of Cognitiva, Number 85, pp 599-604, Paris, 1985.

[12] D. B. Parker, "Learning Logic".
Technical Report TR-47, MIT, Cambridge, 1985.

[13] D. E. Rumelhart; G. E. Hinton; R. J. Williams, "Learning Internal Representations by Error Propagation".
Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations, Mit Press, Cambridge, MA., 1986.

[14] J. A. Anderson; E. Rosenfeld, "Neurocomputing: Foundations of Research".
Mit Press, Cambridge, MA., 1988.

[15] K. S. Narendra; K. Parthasarathy, "Identification and Control for Dynamic Systems Using Neural Networks".
IEEE Trans. on Neural Networks, number 1, pp 4-27, 1990.

[16] K. J. Hunt; D. Sbarbaro; R. Zbikowski; P. J. Gawthrop, "Neural Networks for Control Systems - A Survey".
Automatica, Vol. 28, Number 6, pp 1083-1112, 1992.

[17] R. P. Lippmann, "An Introduction to Computing with Neural Nets".
IEEE ASSP Magazine, pp 4-22, April 1987.

[18] T. Kohonen, "Self Organization and Associative Memory".
Springer-Verlag, Berlin, 1984.

[19] P. Simpson, "Foundations of Neural Networks".
In Artificial Neural Networks: Paradigms, Applications, and Hardware Implementation, Edited by E. Sanchez-Sinencio and C. Lau.
IEEE Press, N. J. 1992.

[20] B. Widrow; M. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation".
Proc. IEEE, Vol. 78, Number 9, pp 1415-1442, Sept. 1990.

[21] F. Rosenblatt, "On the Convergence of Reinforcement Procedures in Simple Perceptrons".
Cornell Aeronautical Laboratory Report, VG-1196-G-4, Buffalo, NY., Feb. 1960.

[22] D. Andes; B. Widrow; M. Lehr; E. Wan, "MRIII: A Robust Algorithm for Training Analog Neural Networks".
Proc. Intl. Joint Conf. on Neural Networks, Vol. 1, pp 533-536, Wash. DC., Jan. 1990.

[23] G. G. Lorentz, "The 13th Problem of Hilbert".
In F. E. Browder Editor, Mathematical Developments Arising from Hilbert Problems, American Mathematical Society, Providence, R. I., 1976.

[24] K. Hornik; M. Stinchcombe; H. White, "Multilayer Feedforward Networks are Universal Approximators".
Neural Networks, Vol. 2, pp 359-366, Pergamon Press, 1989.

[25] D. O. Hebb, "The Organisation of Behaviour".
Wiley, New York, 1949.

[26] J. J. Hopfield; D. I. Feinstein; R. G. Palmer, "Unlearning Has a Stabilizing Effect in Collective Memories".
Nature, Vol. 304, pp 158-159, USA., 1983.

[27] S. Kirkpatrick; C. D. Gellatt; M. P. Vecchi, "Optimisation by Simulated Annealing".
Science, Vol. 220, pp 671-680, USA., 1983.

[28] D. H. Ackley; G. E. Hinton; T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines".
Cognitive Science, Vol. 9, pp 147-169, USA., 1985.

[29] T. Kohonen, "Self-Organization and Associative Memory".
Springer-Verlag, Berlin, 1984.

[30] K. Fukushima, "Cognitron: A Self-Organizing Multilayered Neural Network".
Biological Cybernetics, Vol. 20, pp 121-136, 1975.

[31] K. Fukushima, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position".
Biological Cybernetics, Vol. 36, pp 193-202, 1980.

[32] K. Fukushima, "Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition".
Neural Network, Vol. 1, pp 119-130, Pergamon Press, 1988.

[33] G. A. Carpenter; S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine".
Computer Vision, Graphics and Processing, Vol. 37, pp 54-115, USA., 1987.

[34] G. A. Carpenter; S. Grossberg, "ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns".
Applied Optics, Vol. 26, pp 4919-4930, USA., 1987.

[35] T. Poggio; F. Girosi, "Networks for Approximation and Learning".
Proceedings of The IEEE, Number 78:(9), pp 1481-1497, September 1990.

[36] J. S. Albus, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)".
Journal of Dynamic Systems, Meas, Control, pp 220-227, 1975.

[37] E. Walach; B. Widrow, "The Least Mean Fourth (LMF) Adaptive Algorithm and its Family".
IEEE Trans. Infor. Theory, Vol. IT-30, pp 275-283, March 1984.

[38] D. B. Parker, "Optimal Algorithms for Adaptive Neural Networks: Second Order Back Propagation, Second Order Direct Propagation, and Second order Hebbian Learning".
Proc. First IEEE Intl. Conf. on Neural Networks, Vol. 2, pp 593-600, San Diego, CA, June 1987.

[39] S. Omohundro, "Efficient Algorithms with Neural Network Behaviour".
Complex Systems, Vol. 1, pp 273, 1987.

[40] T. Poggio, "On Optimal Nonlinear Associative Recall".
Biological Cybernetics, Vol. 19, pp 201-209, 1975.

[41] D. S. Broomhead; D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks".
Complex Systems, Vol. 2, pp 321-355, 1988.

[42] S. Chen; S. A. Billings; C. F. N. Cowan; P. M. Grant, "Practical Identification of Narmax Models Using Radial Basis Functions".
Int. Journal Control, Vol. 52, pp 1327-1350, 1990.

[43] H. Sira-Ramirez, "Non-Linear Discrete Variable Structure Systems in Quasi-Sliding Mode".
Int. Journal Control, Vol. 54, Number 5, pp 1171-1187, 1991.

[44] V. I. Utkin, "Variable Structure Systems With Sliding Mode: A Survey".
IEEE Trans. Automatic Control, Number 22, pp 212-222, 1977.

[45] V. I. Utkin, "Discontinuous Control Systems: State of the Art in Theory and Applications".
Proc. IFAC. 10th World Congress, Munich, 1987.

[46] H. Sira-Ramirez, "Variable Structure Control of Non-Linear Systems".
Int. Journal Systems SCI, Vol. 18, Number 9, pp 1673-1689, 1987.

[47] H. Sira-Ramirez, "Differential Geometric Methods in Variable-Structure Control".
Int. Journal Control, Vol. 48, Number 4, pp 1359-1390, October 1988.

[48] W. Itkis, "Control Systems of Variable Structure".
Wiley, New York, 1976.

[49] V. I. Utkin, "Sliding Modes and their Applications in Variable Structure Systems".
MIR, Moscow, 1978.

[50] J. J. Slotine; W. Li, "Applied Nonlinear Control".
Prentice-Hall Inc., New Jersey, 1991.

[51] C. Miloslavjevic, "General Conditions for the Existence of a Quasisliding Mode on the Switching Hyperplane in Discrete Variable Systems".
Plenun Publishing Corporation, pp 307-314, 1985. Translated from Avtomatika I Telemekhanika, Number 3, pp 36-44, March 1985.

[52] H. P. Opitz, "Robustness Properties of Discrete-Variable Structure Controllers".
Int. Journal Control, Vol. 43, Number 3, pp 1003-1014, 1986.

[53] M. E. Magaña; S. H. Zak, "The Control of Discrete-Time Uncertain Systems".
Technical Report TR-EE 87-32, School of Electrical Engineering, Purdue University, Indiana, 1987.

[54] S. Z. Sarpturk; Y. Istefanopulos; O. Kaynak, "On the Stability of Discrete-Time Sliding Mode Control Systems".

IEEE Trans. on Automatic Control, Vol. AC-32, Number 10, pp 930-932, October 1987.

[55] S. V. Drakunov; V. I. Utkin, "On Discrete-Time Sliding Modes".
Proceedings of the IFAC Symposium on Nonlinear Control Systems, pp 484-489, Capri, Italy, 1989.

[56] K. Furuta, "Sliding Mode Control of a Discrete System".
System and Control Letter, Vol. 14, Number 2, pp 145-152, February 1990.

[57] R. W. Brockett, "Finite Dimensional Linear Systems".
Academic Press, 1970.

[58] S. Hui; S. Zak, "Robust Stability Analysis of Adaptation Algorithms for Single Perceptron".
IEEE Trans. on Neural Network, Vol. 2, Number 2, pp 325-328, March 1991.

[59] H. Sira-Ramirez; S. Zak, "The Adaptation of Perceptrons with Applications to Inverse Dynamic Identification of Unknown Dynamic Systems".
IEEE Trans. on Systems, Man, and Cybernetics, Vol. 21, Number 3, pp 634-643, May-June 1991.

[60] J. G. Kuschewski; S. Hui; S. H. Zak, "Application of Feed-Forward Networks to Dynamical Systems Identification and Control".
IEEE Trans. on Control Systems Technology, Vol. 1, Number 1, pp 37-49, March 1993.

[61] K. J. Hunt; D. Sbarbaro, "Studies in Neural Network Based Control".
In Neural Network for Control and Systems, Edited by K. Warwick; G. W. Irwin; and K. J. Hunt; Chapter 6, pp 94-122, IEE Control Engineering Series 46, 1992.

[62] D. T. Pham; X. Liu, "Neural Networks for Discrete Dynamic System Identification".
Journal of Systems Engineering, Springer-Verlag, Vol. 1, Number 1, pp 51-60, London 1992.

[63] K. I. Funahashi, "On the Approximate Realization of Continuous Mappings by Neural Networks".
Neural Networks, Vol.2, pp 183-192, 1989.

[64] J. Golten; A. Verwer, "Control Systems Design and Simulation".
McGraw-Hill Book Company, London 1991.

[65] R. Middleton; G. Goodwin, "Digital Control and Estimation A Unified Approach".
Prentice-Hall International Editions, USA, 1990.

[66] D. Psaltis; A. Sideris; A. Yamamura, "A Multilayered Neural Network Controller".
IEEE Control Systems Magazine, Vol.8, pp 17-21, 1988.

[67] M. I. Jordan; D. E. Rumelhart, "Forward Models: Supervised Learning with a Distal Teacher".
Cognitive Science, Vol.16, pp 307-354, USA, 1992.

[68] C. Hall; R. Smith, "Pitfalls in the Application of Neural Networks for Process Control".
In "Neural Networks for Control and Systems", Edited by K. Warwick, G. W. Irwin, and K. J. Hunt, Chapter 12, pp 243-256, IEE Control Engineering Series 46, 1992.

[69] J. J. Slotine; R. M. Sanner, "Neural networks for Adaptive Control and Recursive Identification: A Theoretical Framework".
In "Essays on Control", Edited by H. L. Trentelman, and J. C. Willems, Chapter 11, pp 381-436, Birkhauser, Boston, 1993.

[70] E. Colina-Morles; N. Mort, "Neural Network-Based Adaptive Control Design". Journal of Systems Engineering, Vol.2, Number 1, pp 9-14, Springer-Verlag, London, 1993.

[71] R. M. Sanner; J. J. Slotine, "Gaussian Networks for Direct Adaptive Control". IEEE Trans. on Neural Networks, Vol.3, Number 6, pp 837-863, November 1982.

[72] A. U. Levin; K. Narendra, "Control of Nonlinear Dynamical Systems Using Neural Networks: Controllability and Stabilization".
IEEE Trans. on Neural Networks, Vol.4, Number 2, pp 192-206, March 1993.

[73] P. Seibert, "Stability under Perturbation in Generalized Dynamical Systems".
In "Nonlinear Differential Equations and Nonlinear Mechanics", Edited by J. P. LaSalle, and S. Lefschetz, Academic Press, pp 463-473, New York 1963.

[74] M. Morari; E. Zafiriou, "Robust Process Control" Prentice Hall, Englewood Cliffs, New Jersey 1989.

[75] K. J. Hunt; D. Sbarbaro, "Neural Networks for Nonlinear Internal Model Control".
IEE Prodeeding-D, Vol.138, Number 5, pp 431-438, Sept. 1991.

[76] W. Li; J. J. Slotine, "Neural Network Control of Unknown Nonlinear Systems".
Proceedings of the 1989 American Control Conference, pp 1136-1141, USA, 1989.

[77] F. Csáki, "State Space Methods for Control Systems".
Akadémiai Kiadó, Budapest, 1977.

[78] S. H. Zák; C. A. MacCarley, "State Feeback Control of Nonlinear Systems".
Int. Journal of Control, Vol.43, Number 5, pp 1497-1514, 1986.

[79] C. G. Economou; M. Morari; B. O. Palsson, "Internal Mode Control. Extension to Nonlinear Systems".
Ind. Eng. Chem. Process Des. Dev., Vol. 25, pp 403-411, 1986.

[80] K. J. Åström; B. Wittenmark, "Adaptive Control".
Addison-Wesley Publishing Company, USA, 1989.

[81] D. J. Hill; P. J. Moylan, "Dissipativeness and Stability of Nonlinear systems".
MIT Press, Cambridge, Mass., 1988.

[82] K. S. Narendra; Y. H. Lin, "Stable Discrete Adaptive Control".
IEEE Trans. on Automatic Control, Vol. 25, pp 456-461, June 1980.

[83] W. T. Miller; F. H. Glanz; L. G. Kraft, "Application of a General Learning Algorithm to the Control of Robotic Manipulators".
The Int. Journal of Robotic Research, Number 6, pp 84-98, 1987.

[84] H. Gomi; M. Kawato, "Neural Network Control for a Closed-Loop System Using Feedback-Error-Learning".
Neural Networks, Vol. 6, pp 933-946, 1993.

[85] K. Ogata, "Systems Dynamics".
Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1978.