Neural Network-Based Controllers for an Electrothermal Furnace System

Wenyu Wei

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

November 2001

0-612-68447-4

ABSTRACT

Neural network schemes are applied in this thesis to a temperature control system problem. The electrothermal furnace is a very popular instrument for applications in material testing area. In this work feedforward neural networks are trained for both identification and control problems of the electrothermal furnace system.

The thesis demonstrates that neural networks can be used effectively for this application problem. which is a highly nonlinear dynamical system. The first emphasis is on the electrothermal furnace model identification and the second emphasis is on the design of neural network based PID and internal model control strategies. Both static and dynamic back-propagation methods are discussed. In the electrothermal furnace models that are introduced. multi-layer feedforward networks are interconnected in novel configurations.

A novel technique based on the internal model control for nonlinear systems using neural networks is proposed. The control structure proposed directly incorporates a model of the plant that was identified by a neural network and its inverse as part of the control strategy. The potential utilizations of the proposed methods are illustrated through experimental and numerical simulations of an electrothermal furnace system.

ACKNOWLEDGEMENTS

I owe very special thanks to my supervisor Dr. Khashayar Khorasani, for guiding my understanding of neural network identification and control as well as for the direction he has provided me during the course of this research.

I would like to thank my sister Mrs. Y. Z. Wei, bother in law Mr. J. H. Wu and father in law Mr. Y. M. Wu for their help and mental support.

I would also like to thanks all my friends and fellow graduate students for the mental support and friendship. I wish to particularly thank my friends Ali and Yonjin for their numerous suggestions and help.

DEDICATION

To my dear wife Ning and our daughter Rosemary who was born on October 13, 2001, with love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

LIST OF ACRONYMS

ANN	artificial neural network
NN	neural network
BPN	backpropagation network
PID	proportional integral and derivative
PC	personal computer
LMS	least mean square error
VLSI	very-large-scale integration
RBF	radial basis function
IMC	internal model control
F	filter
P	plant
C	controller

LIST OF PRINCIPLE SYMBOLS

z^{-1}	unit-delay operator
e	error term
E	global error term
η	learning rate parameter
$\Delta\eta$	change in learning rate
T_0	sampling period
T_i	integral time constant
T_d	derivative time constant
K_P	proportional gain
K_I	integral gain
K_D	derivative gain
$\delta_j(n)$	delta error for neuron *j* at time *n*
ΔW	incremental weight change
λ	regularization parameter
$\in$	"belongs to"

Chapter 1

INTRODUCTION

The field of Artificial Neural Networks (ANN) has been originated from and inspired by the biological neuron networks in human brain. It has brought about fundamental changes in several application areas such as pattern recognition, control, signal processing, to just name a few. It is hard to deny that new and challenging concepts arise constantly in this emerging field.

1.1 A biological neuron

The basic anatomical unit responsible for the processing of information in the nervous system is a cell known as the neuron. A developing neuron is synonymous with a plastic brain: Plasticity permits the developing nervous system to adapt to its surrounding environment. In an adult brain. plasticity may be accounted for by two mechanisms: the creation of new synaptic connections between neurons. and the modification of existing synapses. Axons are the transmission lines and dendrites are the receptive zones. Neurons come in a wide variety of shapes and sizes in different parts of the brain. Fig. 1.1 illustrates the shape of a pyramidal cell, which is one of the most common types of cortical neurons. Like many other types of neurons. it receives most of its inputs through dendritic spines. The neuron itself is imbedded in an aqueous solution of ions. and its selective permeability to these ions establishes a potential gradient responsible for transmitting information. Neurons receive electrochemical input signals from other neurons to which they are connected at sites on their surface. The input signals are combined in various ways. triggering the generation of an output signal by a special region near the cell body.

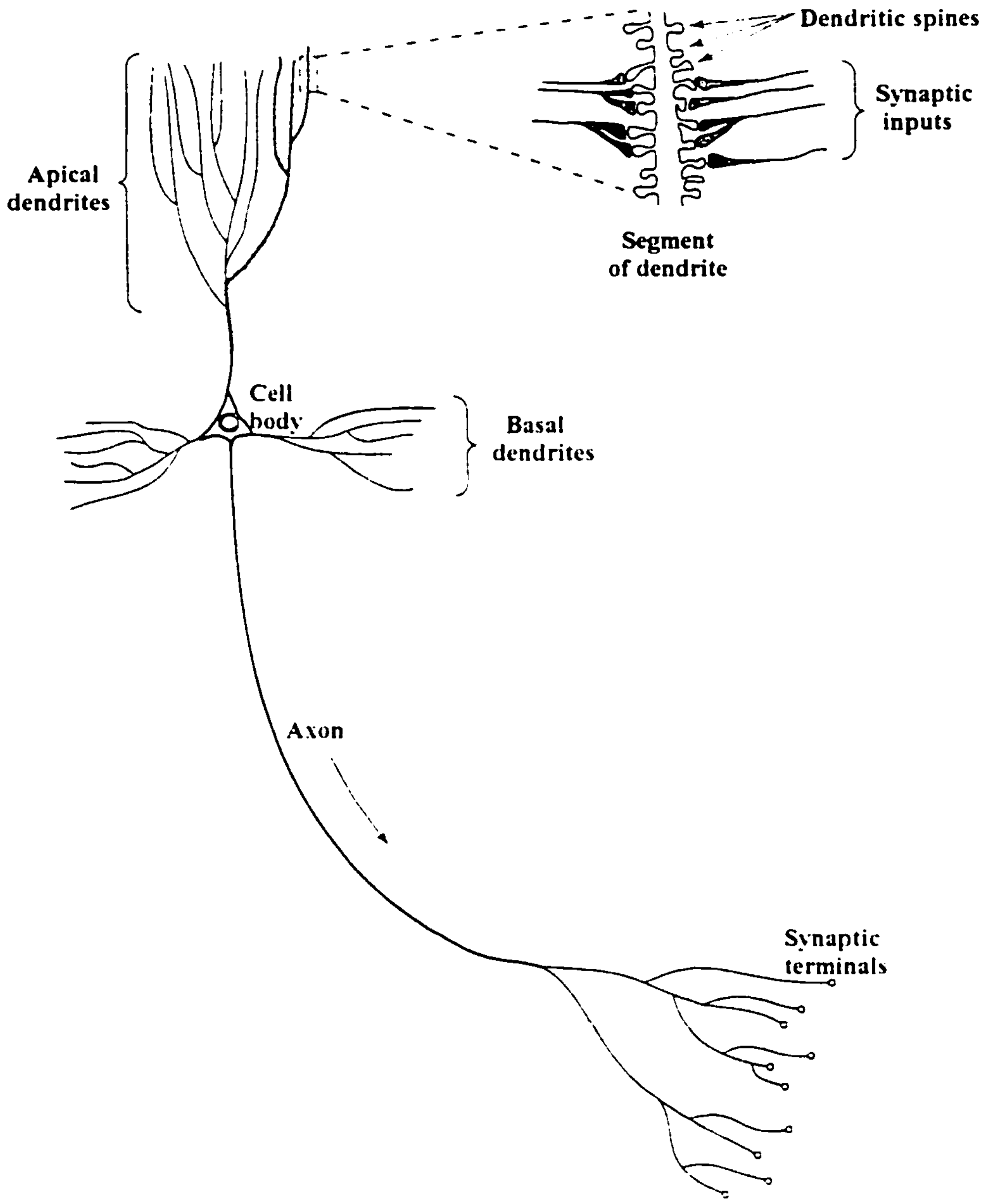


Figure 1.1 The pyramidal cell

1.2 The neural networks

Neural networks (NN). or artificial neural networks to be more precise. represents a technology that has found significant interest in many domains. The networks are endowed with certain unique attributes: universal approximation (input-output mapping). the ability to learn from and adapt to their environment. and the ability to invoke or require weak assumptions about the underlying physical phenomena responsible for the generation of the input-output data.

Application of neural networks offers the following useful properties and capabilities:

1. Nonlinearity. A neuron is basically a nonlinear device. Consequently. a neural network. made up of an interconnection of neurons. itself is a nonlinear operator. Moreover. the nonlinearity introduced is of a special architecture. in the sense that it is distributed throughout the network. Nonlinearity is a highly important property. particularly if the underlying physical mechanism responsible for the generation of an input-output data is inherently nonlinear itself.
2. Input-Output Mapping. A popular paradigm in learning theory is known as supervised learning. which involves the modification of the synaptic weights of the neural network by applying a set of labeled training samples or task examples. Each example consists of a unique input signal and the corresponding desired response. The network is presented by an example picked at random from the set. and the synaptic weights of the network are modified so as to minimize the difference between the desired response and the actual response of the network produced by the input signal in accordance with an approriate statistical criterion.

3. Adaptability. Neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment. In particular. a neural network trained to operate in a specific environment can be easily retrained to deal with minor changes in the operating environmental conditions. Moreover. when it is operating in a nonstationary environment. a neural network can be designed to change its synaptic weights in real-time and on-line. The specific architecture of a neural network for pattern classification. signal processing. and control applications. coupled with the adaptive capability of the network. make it an ideal tool for use in adaptive pattern classification. adaptive signal processing. and adaptive control. As a general rule. it may be said that the more adaptive a system is made in a properly designed fashion. that is by assuming that the adaptive system is guaranteed to be stable. the more robust its performance will likely be when the system is required to operate in a nonstationary environment. It should be emphasized that adaptability does not always automatically lead to robustness: indeed. it may do the very opposite.

4. Contextual Information. Information is represented and stored by the very structure and activation state of a neural network. Every neuron in the network is potentially affected by the global activity of all the other neurons in the network. Consequently. contextual information is dealt with naturally and in a distributed manner by the neural network.

5. VLSI Implementability. The massively parallel nature of a neural network makes it potentially efficient and rapid for certain tasks. This same feature makes a neural network ideally suitable for implementation using very-large-scale-integrated (VLSI) technology. The particular advantages of VLSI are that it provides a means for

capturing truly complex behavior in a highly hierarchical fashion, and makes it possible to use a neural network as a tool for real-time applications involving pattern recognition, signal processing, and control problems, to just name a few.

6. Uniformity of Analysis and Design. Strictly speaking, neural networks enjoy universality as information processors. This is stated in the sense that similar architectures or paradigms are used with some minor modifications in many domains involving the application of neural networks.
7. Neurobiological Analogy. The design of a neural network is motivated by strong correlations with the nervous system and brain operation, which is a living proof that fault-tolerant parallel processing is not only physically possible but also could be fast and computationally efficient and powerful. Neurobiologists look to artificial neural networks as a research tool for the interpretation and alternative means for understanding neurobiological phenomena.

1.3 Historical perspectives on ANN

In 1949, Hebb first proposed a learning rule that has become the starting point for development of artificial neural networks training algorithms. Twenty years later, a group of scientists tried to combine the biological and psychological insights into electronic circuits [2], [3], which were later converted into more flexible medium such as computer simulations. Early successes produced a burst of activity and optimism in this domain.

Marvin Minsky. Frank Rosenblatt. Bernard Widrow and others developed various neural networks that consist of a single layer of artificial neurons.

In 1986 the development of the back-propagation algorithm was reported by Rumelhart. Hinton. and Williams [9]. In that same year. the two-volume books. by Rumelhart and McClelland. was published [1]. This latter book has been a major influence in the adoption and use of back-propagation learning algorithm, which has emerged as one the most popular learning algorithms in neural network area. for the training of multi-layer perceptrons. In fact. back-propagation learning was discovered independently by two other researcheres about the same time [42]. After the development of the back- propagation algorithm in the mid-1980s. it became known that the algorithm had already been described earlier by Werbos in his Ph.D. thesis in 1974 [1]. That thesis was the first documented description of an efficient reverse-mode gradient computation that was applied to a general network model with neural networks being merely a special application case. It is most unfortunate that Werbos's work remained almost unknown in the scientific community for over a decade.

In the 1980s. the neural network field became the centre of extensive research focus and interest. This was partly due to the development of multi-layer learning algorithms that enabled the networks to learn (using a more complex structures) difficult problems that perceptrons could not solve (represent and learn).

Back-propagation (BP) is without a doubt one of the most well-known algorithms that has been used in the neural networks applications [3]. Numerous successful applications of BP have been reported in areas such as pattern recognition [5]. image processing [6]. biomedical engineering [7]. [8]. and control [9]. [10]. to just name a few. However. BP has some limitations. One of the important drawbacks is its fixed network structure. In most practical problems the fixed network structure could either fail to solve a complex problem or become too redundant. Three different approaches can be used to address this problem:

1. Trial and error: Start with a fixed structure. If the learning process of the selected network does not lead to the desired accuracy, then design a new structure by adding or pruning neurons. Clearly, this is not an efficient approach because no knowledge can be gained from the previous design.
2. Choose a large number of neurons in the hidden layer: Two difficulties will arise in using this approach. First, there may be redundant neurons for representing the given function, and thus the computational overhead can become excessive. Furthermore, more neurons in the hidden layer will result in a cost function with additional local minimum points, consequently resulting in a higher probability of getting trapped in a local minimum.
3. Adaptive structure: The network has the ability to adapt its structure according to the statistics of the training set. When the inputs of the network change, the network structure adapts to compensate for these variations. This is accomplished provided that the time used for weight adjustment is much faster than the dynamics of the training set. Thus the network structure adaptation provides more flexibility to the input training set.

This thesis is focused on the use of artificial neural networks for identification and control of a class nonlinear dynamic systems with application to electrothermal furnace system. The capabilities of neural networks to represent complicated nonlinear maps has motivated researchers to use networks directly in a model-based control strategy. The idea proposed here is based on the possibility of training neural networks to learn both the system's input/output relationship and its corresponding inverse relationship. A suitable control strategy that also directly incorporates the plant model is provided by the internal model control (IMC) [15], [16]. The applicability of IMC to control nonlinear systems has been demonstrated by Economou and Morari [17]. The inverse of the nonlinear operator modelling the plant was shown to play a crucial role in the implementation of a nonlinear IMC. The authors have studied analytical and provided numerical methods for determining the necessary construction of the nonlinear inverse operators. In this thesis artificial neural networks are used for the construction of plant models and their inverses, and it is intended that they be used directly within the IMC control structure.

The idea of using neural networks for nonlinear IMC was also considered by Bhat and McAvoy [22]. However, they did not investigate the invertibility conditions associated with the nonlinear dynamic system and have not proposed learning algorithms for constructing the inverse model. Furthermore, they have not applied and implemented the neural network-based nonlinear IMC controller to our application problem.

1.4 Motivation for this thesis

An application of a neural network-based strategy in temperature control system is presented in this thesis. The electrothermal furnace is a popular instrument widely used in material testing area. A feedforward neural network is trained to identify and control the electrothermal furnace system.

The main objectives here are to demonstrate and illustrate that the identification and control of a neural network-based scheme in an electrothermal furnace temperature control system – a highly nonlinear dynamic system proposed in the literature -- is feasible and its performance is superior to traditional methods such as the standard PID control scheme.

1.5 Figure of merits of the thesis

This thesis focuses on the use of artificial neural networks for the identification and control of nonlinear dynamic systems. Artificial neural networks provide a distinctive computational paradigm and have proved effective for a range of practical problems where conventional model-based and computation techniques have not been quite successful.

An artificial neural network consists of many simple processing elements each having a number of inputs and a single output. The output of each element is determined as a nonlinear function of a weighted sum of its inputs. A large variety of activation functions can be used. however. in this thesis we consider Gaussian activation functions because of their functional representational properties. The basic processing elements are interconnected using adjustable strength links, or weights. The strengths of the individual links determine the overall behaviour of the network [7, 24]. The weights of a particular network are to be selected to achieve a desired input-output relationship. The weights are to be adjusted during a training process when the network is presented with a range of input patterns and its output is compared to the desired output each time. The aim is to successively drive the weights to values, which make the network output equal or very close to the target output.

In this thesis, we are primarily concerned with the use of artificial neural networks for dynamical systems control. Various characteristics of neural networks suggest that they may be useful in certain classes of control problems. In particular, the ability of neural networks to represent arbitrary nonlinear mappings encourages the study of neural networks for complex nonlinear control problems. Relevant features of neural networks in the control context are

(i) the ability to represent arbitrary nonlinear relationships

(ii) adaptation and learning in uncertain systems provided through both off-line and online weight adaptations

(iii) information contained in input-output data is transformed to internal representations allowing data fusion using both quantitative and qualitative signals

(iv) parallel distributed processing architecture allowing fast processing for large-scale dynamical systems

(v) architecture providing a degree of robustness through fault tolerance and graceful degradation

The ability of neural networks to represent nonlinear relationships leads one to the idea of using networks directly in a model-based control strategy. The idea proposed in this thesis is based on the possibility of training networks to learn both the system's input/output relationship and the corresponding inverse relationship. A suitable control strategy, which directly incorporates the plant model, is provided by the internal model control (IMC). The inverse of the nonlinear operator modelling the plant was shown to play a crucial role in the implementation of nonlinear IMC.

1.6 Contributions of the thesis

The contributions of this thesis are in demonstrating that backpropagation neural network can be used effectively in identifying and controlling an electrothermal furnace system. First a three-layer neural network is developed for the identification of the electrothermal furnace system. This network is subsequently used in a neural network-based IMC control strategy. Furthermore, a neural network-based PID control strategy is also proposed and these results are compared to a conventional PID control scheme.

1.7 Outline of the thesis

The thesis is organized as follows: In Chapter 2, a typical neural network topology and its basic principles are introduced. Specifically, the activation functions, supervised learning, back-propagation network and back-propagation training are all described and reviewed in details.

In Chapter 3. a physical electrothermal furnace system is described and its computer control system structure and a neural network-based model for the electrothermal furnace system are introduced.

In Chapter 4. a neural network-based PID and a neural network-based internal model control strategies are proposed. The details regarding the learning algorithms. internal model principle and nonlinear internal model control scheme are also presented.

In Chapter 5. the experimental results of the proposed identification scheme as well as the neural network-based PID and neural network-based nonlinear internal model control strategies applied to our electrothermal furnace system are presented.

In Chapter 6. a summary of the thesis contributions is presented with a brief discussion on the future work in this area.

The contribution of this thesis is in demonstrating that neural networks can be used effectively for identification and control of electrothermal furnace systems – a highly nonlinear dynamical system.

Chapter 2

NEURAL NETWORK TOPOLOGIES AND PRINCIPLES OF OPERATION

2.1 The artificial neuron

The artificial neuron was developed to mimic the first-order characteristics of a biological neuron [1]. As a fundamental building block, a neuron is the basic operating processor in a neural network. Each neuron receives several inputs through its connections, known as synapses. The inputs are the activations of the other neurons multiplied by the weights of the synapses. Each neuron has one output, which is generally related to the state of the neuron and which generally fans out to several other neurons. An abstract model of a neuron is shown in Figure 2.1.

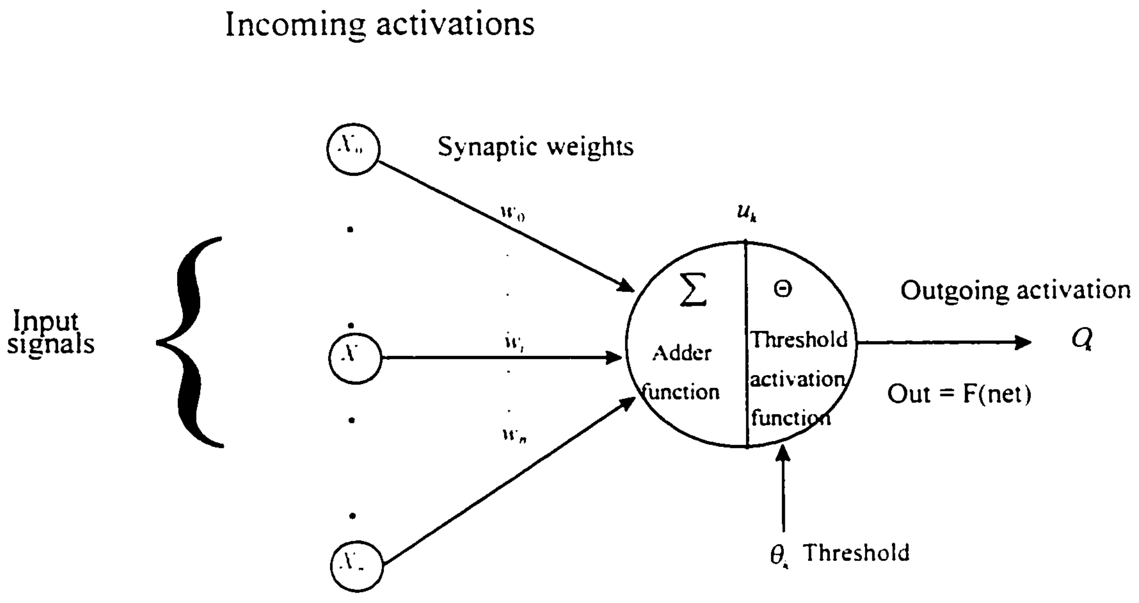


Figure 2.1 Nonlinear model of a neuron

2.1.1 Model of a neuron

A neuron is an information-processing unit that is fundamental to the operation of a neural network. Figure 2.1 shows the model of a neuron. As stated before, one may identify three basic elements associated with a neuron description, as discussed below:

1. A set of synapses or connecting links, each characterized by a weight or strength of its own. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}. It is important to make a note of the manner in which the subscripts of the synaptic weight w_{kj} are written.

2. An adder for summing the input signals, which are weighted by the respective synapses of the neuron. Note that the operations described so far constitute a linear combiner.
3. An activation function for simply limiting the amplitude of the output of a neuron. The activation function is also referred to in the literature as a squashing function in that it squashes (limits) the permissible amplitude range of the output signal to a certain finite value.

The model of the neuron shown in Figure 2.1 also includes an externally applied threshold θ_k that has the effect of modifying the net input to the activation function.

Using mathematical notations, we may describe the processing operation of a neuron k by the following pair of equations:

$$Net_k = u_k = \sum_{j=1}^{p} w_{kj} x_j \qquad (2.1)$$

and

$$O_k = f(Net_k - \theta_k) \qquad (2.2)$$

where $x_1 \ldots\ldots x_p$ are the input signals; $w_{k1} \ldots\ldots w_{kp}$ are the synaptic weights of neuron k; Net_k or u_k is the linear combiner output; θ_k is the neuron threshold; $f(\cdot)$ is the neuron activation function; and O_k is the output signal of the neuron.

In particular, depending on whether the threshold θ_k is positive or negative, the relationship between the effective internal activity level ($v_k = u_k - \theta_k$) [1] or activation potential of neuron k and the linear combiner output u_k is modified in the manner illustrated in Figure 2.2. Note that as a result of this affine transformation, the graph of v_k versus u_k no longer passes through the origin.

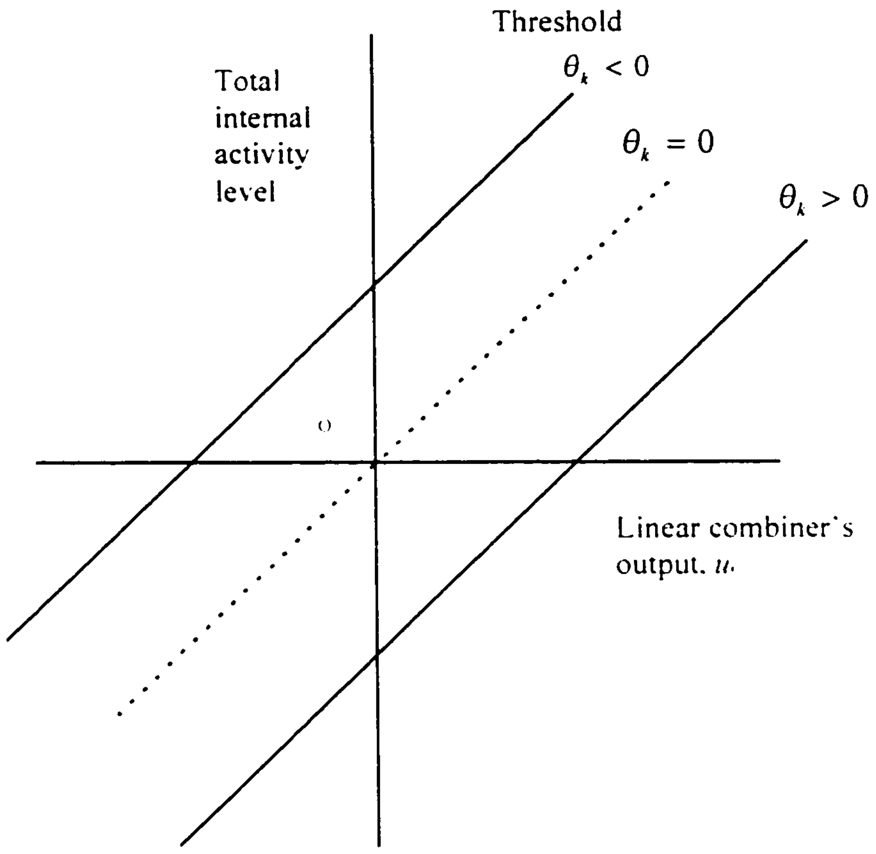


Figure 2.2 Affine transformation produced by a threshold

2.1.2 Types of activation functions

The activation function, denoted by $f(.)$, defines and characterizes the output of a neuron in terms of the activity level at its input. We may identify three basic types of activation functions:

1. Threshold Function. For this type of activation function, depicted in Figure 2.3(a), we have

$$f(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \tag{2.3}$$

Correspondingly, the output of neuron k employing such a threshold function is expressed as

$$O_k = \begin{cases} 1 & \text{if } v_k \geq 0 \\ 0 & \text{if } v_k < 0 \end{cases} \tag{2.4}$$

where v_k is the internal activity level of the neuron; that is,

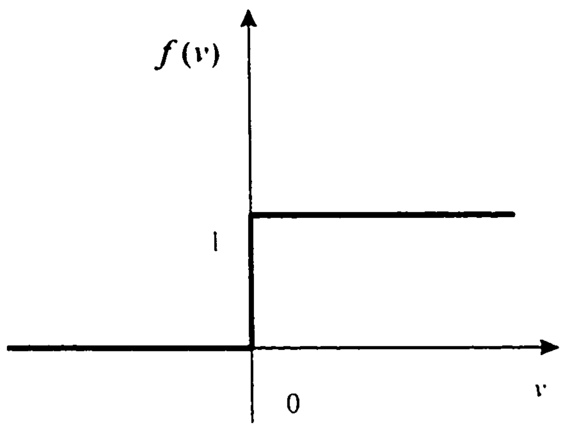
$$v_k = \sum_{j=1}^{p} w_{kj} x_j - \theta_k \tag{2.5}$$

In this model, the output of a neuron takes on the value of 1 if the total internal activity level of that neuron is nonnegative and 0 otherwise.

2. Piecewise-Linear Function. For the piecewise-linear function, shown in Fig. 2.3(b) we have

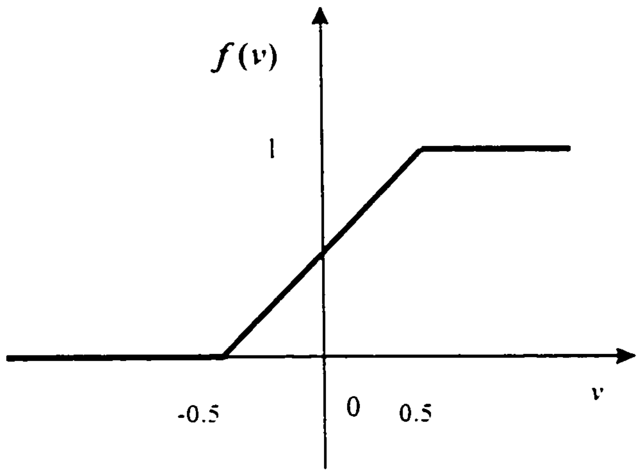
$$f(v) = \begin{cases} 1, & v \geq \frac{1}{2} \\ v, & -\frac{1}{2} > v > \frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \tag{2.6}$$

where the amplification factor inside the linear region of operation is assumed to be unity. This form of an activation function may be viewed as an approximation to a nonlinear amplifier.

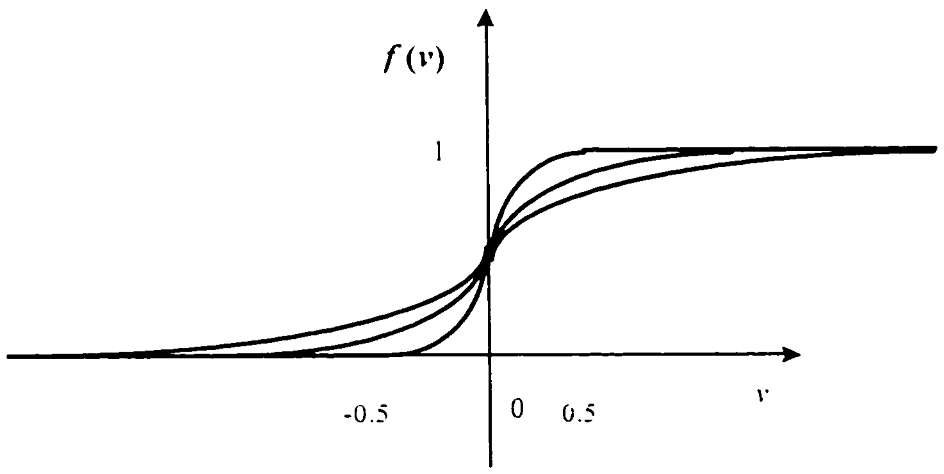


(a)

Figure 2.3 Basic activation functions (a)



(b)



(c)

Figure 2.3 Activation functions (b) and (c)

3. Sigmoidal Function. The sigmoid function is by far the most common form of activation function used in artificial neural networks. It is defined as a

monotonically increasing function that exhibits useful smoothness and asymptotic properties. An example of a sigmoidal function is the logistic function, defined by

$$f(v) = \frac{1}{1 + \exp(-av)} \qquad (2.7)$$

where a is the slope parameter of the sigmoid function. By varying the parameter a, we obtain sigmoid function of different slopes, as illustrated in Figure 2.3(c). In fact, the slope at the origin is a 4. In the limit, as the slope parameter approaches infinity, the sigmoidal function becomes simply a threshold function. Whereas a threshold function assumes the value of 0 or 1, a sigmoidal function assumes a continuous range of values from 0 to 1. Alternatively, the sigmoid may also be selected as the hyperbolic tangent function, defined by

$$f(v) = \tanh(\frac{v}{2}) = \frac{1 - \exp(-v)}{1 + \exp(-v)} \qquad (2.8)$$

2.2.1 Alternative activation functions

As mentioned earlier, the range of an activation function should be appropriate according to the range of target values of a particular problem. The binary sigmoidal function presented in Section 2.1.2, and described by

$$f(v) = \frac{1}{1 + \exp(-av)} \qquad (2.9)$$

with

$$f'(v) = af(v)[1 - f(v)] \tag{2.10}$$

can be modified to cover any desired range, to be centered at any desired value of v, and to have any desired slope at its center.

The binary sigmoidal function can have its range expanded and shifted so that it maps the real numbers into the interval [b, c] for any b and c. To do so, give an interval [b, c], we first define the parameters

$$\gamma = c - b \tag{2.11}$$

$$\eta = -b \tag{2.12}$$

Then the sigmoidal function may be defined as

$$g(v) = \gamma f(v) - \eta = \frac{\gamma}{1 + \exp(-av)} - \eta \tag{2.13}$$

which has now the desired property, namely, its range is (b, c). Furthermore, its derivative also can be expressed conveniently in terms of the original function according to

$$g'(v) = \frac{a}{\gamma}[\eta + g(v)][\gamma - \eta - g(v)] \tag{2.14}$$

The logistic sigmoidal function (or any other function) can be translated to the right or left by the use of an additive constant on the independent variable. However, this is not necessary, since the trainable bias serves the same role [21].

2.1.4 Gaussian Units

Units that employ a Gaussian activation function are used primarily in applications where it is desirable to classify input patterns into one of several predefined classes. Like the other network units that we have already investigated, Gaussian units convert their input activation into an output signal by applying an activation function to the input. Unlike the other types of units that we have studied, Gaussian units are not necessarily output limited.

Let us now consider the form of a typical Gaussian activation function

$$f_j(v_j) = e^{\frac{v_j - 1}{\sigma^2}} \qquad (2.15)$$

where, as before, v_j represents the net input to the unit, and the term σ is a smoothing parameter. The form of this function is shown in Figure 3.4 for values of v_j in the range between -1 and 1.

We can observe from the graph of the Gaussian activation function that there is a very narrow range of input values that will allow the unit to generate an active output. In this manner, the Gaussian unit acts like a filter, allowing only input patterns that produce activations within a very narrow range to pass, while effectively attenuating all other inputs.

Consider the behavior of this unit in a neural network. Since each unit in a given hidden or output layer is typically excited by a number of connections coming from another layer, the pattern contained in the input connections to the unit act as a tuner. When the layer feeding the Gaussian unit produces an output pattern that matches the pattern contained in the input connections to the Gaussian, the Gaussian detects the match and generates a signal indicating that the match occurred. Conversely, when the output pattern from the layer feeding the Gaussian is significantly different from the pattern contained in the input connection weights, the Gaussian produces no output, indicating the input was not successfully matched [41].

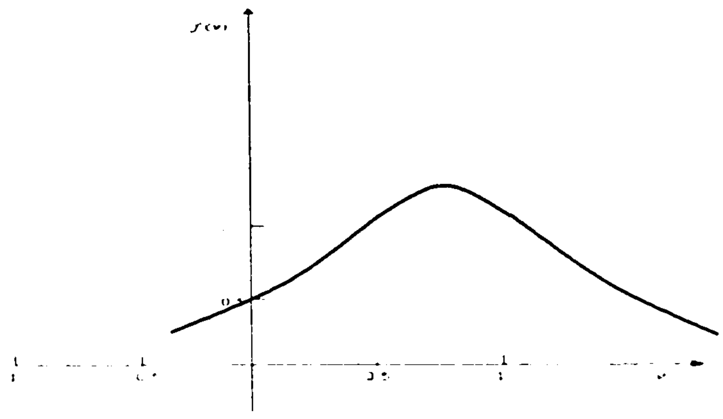


Figure 2.4 A Gaussian activation function

2.1.5 Network architecture

The principle characteristics of any network are defined by both the particular nonlinear activation functions of its processing elements and by the way in which these elements are interconnected.

The simplest basic processing element comprises of weighted linear combination of inputs that are processed through a certain nonlinear function, known as the activation function. Some of the most frequently used activation functions are shown in Fig. 2.5. As briefly indicated earlier each representation of the basic unit has certain advantages and disadvantages. The choice of a specific activation function depends on the particular problem being studied.

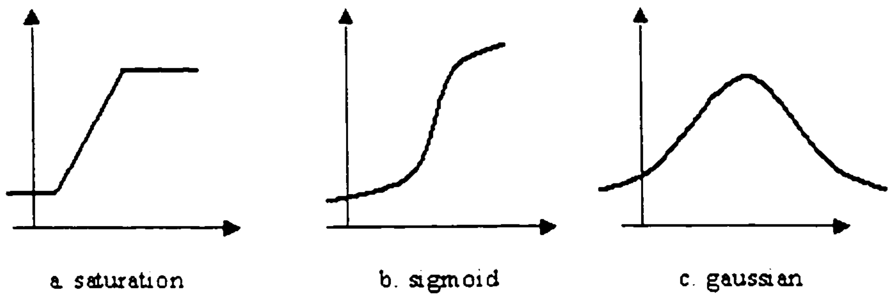


Fig. 2.5 Frequently used activation functions

The basic processing elements by themselves are not very powerful in terms of computational capabilities or representability, but their interconnection allow encoding relationships between the variables, resulting in powerful processing capabilities. For example, the connection of three layers, as shown in Fig. 2.6, provides the possibility of nonlinear mappings between the inputs and the outputs. This capability can be used to represent and learn complex nonlinear mappings among the inputs and outputs of a system.

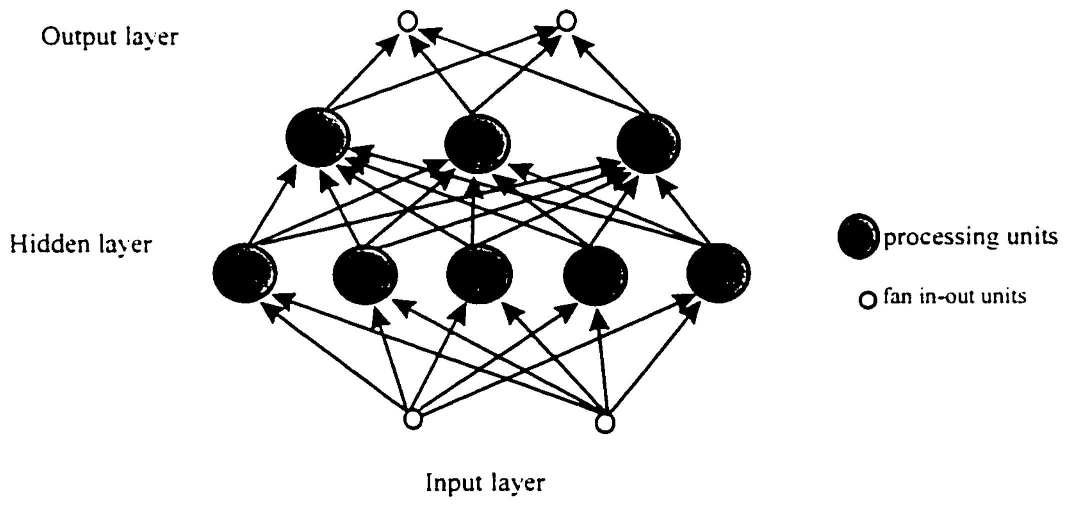


Fig. 2.6 A Multi-layer neural network

The structure depicted in Fig. 2.6 with one hidden layer having sigmoidal activation functions has been shown in the literature to be capable of producing an arbitrary mapping between the input and output variables. The standard approach to training the weights of this type of network is the back-propagation algorithm as described in details in later chapters. However, this method is known to have drawbacks in terms of speed of convergence and in obtaining a unique global solution to the optimization problem. These problems arise mainly due to the nature of the nonlinear optimisation problem having local minima.

The use of Gaussian units, on the other hand, has the advantage of simplifying the problem, because in a natural way it produces a partition of the input space [43]. There is a theoretical support [43] for representation capabilities as well as consistent results achieved with these networks. It has been stated that this network has the best "approximation" property [44] as a nonlinear mapping network.

In this thesis we also consider Gaussian networks consisting of many inputs and outputs. as shown in Fig 2.7. The number of network input units n_i corresponds to the dimension of the input vector. denoted by $x \in \Re^{n_i}$. The linear output unit is fully connected to the hidden units: the network output y (a nonlinear function. g(.), of the input vector x) is a weighted sum of the activation levels of the N hidden units:

$$y = g(x) = \sum_{i=1}^{N} c_i k_i \qquad (2.16)$$

where c_i denotes the connection weight between hidden unit i and the network output. and k_i is the output of the hidden unit i.

The activation level of a hidden unit in a Gaussian network depends only on the distance between the input vector and the centre of the Gaussian function of that unit. The centre of the hidden unit i is denotes by $x_i \in \Re^{n}$. More precisely. the activation level k_i of the hidden unit i is defined as

$$k_i = e^{-d(x,x_i,S)} \qquad (2.17)$$

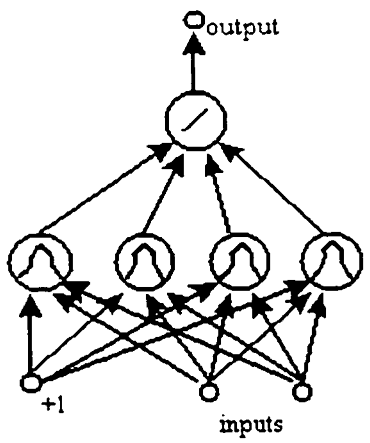


Fig. 2.7 A basic Gaussian neural network

where the distance function d_i for the hidden unit i represents the distance between the centre of the function i for that unit and the input vector x, namely

$$d_i(x, x_i, \Delta) = (x - x_i)^T \Delta (x - x_i) \qquad (2.18)$$

The matrix Δ represents the bandwidth of the hidden unit function. We consider a constant diagonal matrix Δ whose diagonal entries are selected identically. Having such a Δ constrains the representation but simplifies the learning algorithm. We refer to such a network as a regular Gaussian network.

Gaussian network is a kind of a Radial-Basis Function (RBF) networks. The RBF networks and multi-layer perceptrons are examples of nonlinear feedforward neural networks. The linear characteristics of the output layer of the RBF network implies that such a network is more closely related to the Rosenblatt's perceptron than a multi-layer perceptron. However, the RBF network differs from the perceptron in that it is capable of implementing arbitrary nonlinear transformations of the input space [1].

2.1.6 Learning

Learning is the most important property associated with a neural network. It is defined as a change in connection weight values to capture the information contained in the training data. Learning involves adjusting the weights of the network so that the application of a set of inputs produces the correct outputs. The weight adjustment scheme is known as the learning law. All learning methods can be classified into two main categories: supervised learning and unsupervised learning.

1. *Supervised Learning*

Supervised learning involves the association of a target vector representing the desired output values with each input vector. After the output of the network for a given input vector is computed and compared to its target, the difference (error) is fed back so that the network weights are adjusted according to an algorithm that tends to minimize this error. The vectors of the learning (training) set are applied sequentially and the learning procedure is repeated until the error for the entire training set reaches an acceptable low level [20].

2. *Unsupervised Learning*

Unsupervised learning requires no target vectors for the output, and hence no comparison to a set of predetermined output responses. The learning set consists solely of input vectors, and the learning algorithm modifies network weights so as to produce consistent outputs. The learning process in essence extracts the statistical properties of the learning set and group similar vectors into classes.

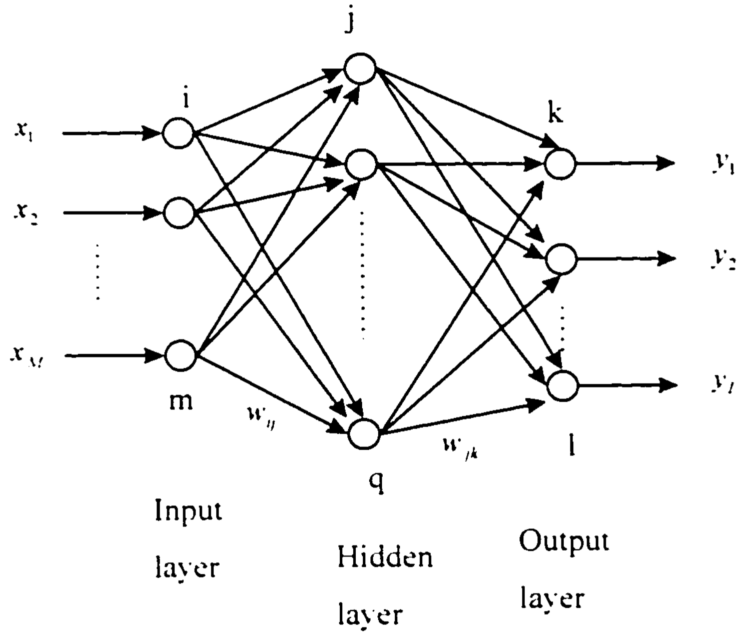
2.2 The backpropagation network

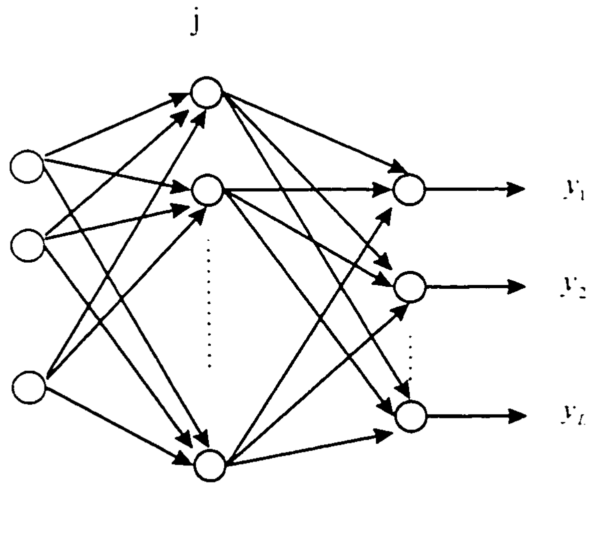
The so-called back-propagation network is simply a multi-layer nonlinear mapping network, with a feedforward configuration. The use of the term "mapping" in describing a network implies that the network is capable of implementing approximations to a variety of functions from an *m-dimensional* space R^m to an *l-dimensional* space R^l. Certain backpropagation networks are used for binary input and output vectors mappings (that is, each of the components of the input or the output vector is either 0 or 1). Backpropagation networks are typically trained using the generalized delta rule, application of which involves the calculation of the network output, a comparison of this output with the desired output, the calculation of an "error" and a backward propagation of this error in order to "correct" future outputs. In this process, each neuron updates the weights of its input connections in such a way that the error associated with its own output activation is decreased.

2.2.1 Multi-layer static neural networks

The main architecture and operation of a multi-layer neural network consists of the following attributes:

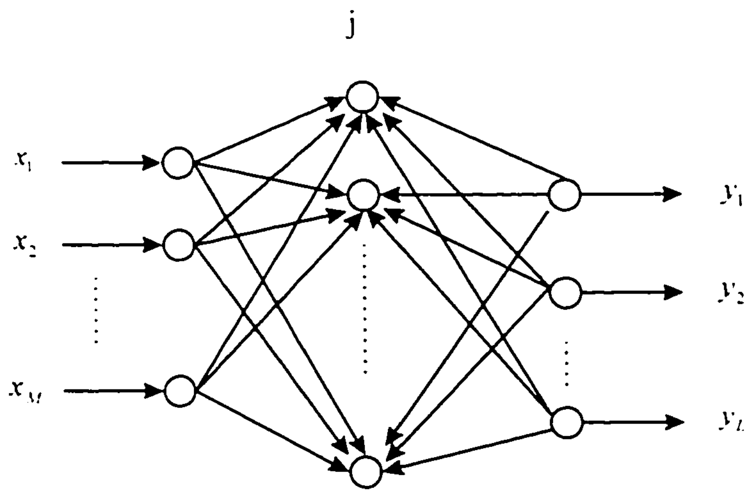
1. A typical backpropagation neural network structure consists of input, hidden, and output layers. Hidden layers may consist of more than one layer. It has not yet been resolved in the neural network literature as to how many hidden layers are necessary for a particular application. The amount of computation taking place at the input layer is quite minimal. The number of neurons in the input layer is equal to the number of input vector components (measured data values). During learning (training), the input layer sends out (fan out) input information to the first hidden layer, as shown in Figure 2.8(a).
2. The last hidden neurons broadcast their results to all the neurons in the output layer. Each output neuron calculates a weighted sum and subtracts its actual results (actual response) from its desired results (targeted output) to produce an error signal (output error). The connections active at this stage of operation and learning are shown with thick arrows in Figure 2.8(b).
3. The output nodes calculate the partial derivatives of the error vector components with respect to the weights, and pass these derivatives back to the hidden layers. The computation during this learning stage is what gives the algorithm its name: the backpropagation. Each hidden neuron calculates the sum of the error derivatives to find its contribution to the output error. The backpropagation of error to a hidden layer is shown in Figure 2.8(c).





(b)

Figure 2.8 (a) and (b) Backpropagation network



(c)

Figure 2.8 (c) Backpropagation network

2.2.2 Backpropagation training procedure

1. Initialize the weights and biases.

   Set all the weights and node biases to small random values in the interval [-1, 1].

2. If the stopping condition specified a priori by the user is not satisfied, then proceed to steps 3 to10, otherwise stop.

3. For each training pair follow steps 4 to 9.

4. Feedforward operation: Present the input vector $x_i = (x_1, x_2 ... x_M)$ and specify the desired output $D = (d_1, d_2 ... d_L)$ (defining the training set). If the net is used as a classifier then all desired outputs are typically set to zero except for that

corresponding to the class the input is from. That desired output is set to 1. The input values could be new on each trial or samples from a training set could be presented cyclically until weights stabilize.

5. Backpropagation of error:

The cost function E_p associated with the network is defined as the output error corresponding to each pattern p and is given by

$$E_p = \frac{1}{2}\sum_{k=1}^{L}(d_{pk} - O_{pk})^2 \tag{2.19}$$

where O_{pk} and d_{pk} are the actual and desired outputs of the kth output neuron for pth pattern, respectively. Each output unit $y_k = (y_1, y_2 \ldots y_L)$ receives a target pattern corresponding to the input training pattern and computes its associated error to be propagated to pervious hidden layers as follows

$$\delta_k = (d_k - O_k)f_k'(Net_k) = O_k(1 - O_k)(d_k - O_k) \tag{2.20}$$

in order to calculate the weight adjustment according to

$$\Delta_p w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \tag{2.21}$$

where η is the learning rate, $\eta > 0$, we have

$$\Delta_p w_{jk} = \eta\delta_{pk}O_{pj} = \eta(d_{pk} - O_{pk})f_k'(Net_{pk})O_{pj} = \eta O_{pk}(1 - O_{pk})(d_k - O_{pk})O_{pj} \tag{2.22}$$

Note that the derivation of the above term is as follow. From

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial Net_k} \cdot \frac{\partial Net_k}{\partial w_{jk}} \tag{2.23}$$

and from the output layer, we have $Net_{pk} = \sum_{j=1}^{q} w_{jk} O_j$

so that,

$$\frac{\partial Net_k}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} (\sum_{j=1}^{q} w_{jk} O_j) O_j$$

Defining δ_k as

$$\delta_k = -\frac{\partial E}{\partial Net_k} = -\frac{\partial E}{\partial O_k} \cdot \frac{\partial O_k}{\partial Net_k} \tag{2.24}$$

with

$$\frac{\partial E}{\partial O_k} = -(d_k - O_k) \qquad \frac{\partial O_k}{\partial Net_k} = f_k'(Net_k)$$

We obtain

$$\delta_k = (d_k - O_k) f_k'(Net_k) = O_k(1 - O_k)(d_k - O_k) \tag{2.25}$$

and, the weight adjustment term for each output layer neuron becomes:

$$\Delta_p w_{jk} = \eta f_k'(Net_{pk})(d_{pk} - O_{pk}) O_{pj} = \eta O_{pk}(1 - O_{pk})(d_{pk} - O_{pk}) O_{pj} \tag{2.26}$$

Associated with each hidden unit $Z_j = (Z_1, Z_2 ... Z_q)$, the previous delta terms are combined as (from units in the output layer above)

$$-\frac{\partial E}{\partial O_j} = \sum_{k=1}^{L} \delta_k w_{jk}$$

The above is then multiplied by the derivative of the corresponding activation function to calculate the "equivalent" tracking error signal

$$\delta_j = f_j'(Net_j) \sum_{k=1}^{L} \delta_k w_{jk} \tag{2.27}$$

The weight adjustment for the hidden layer neurons is then achieved according to

$$\Delta w_{ij} = \eta \delta_j O_i \qquad (2.28)$$

The derivation of the above expressions is as follows.

We have

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial Net_j} \cdot \frac{\partial Net_j}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial Net_j} O_i = \eta(-\frac{\partial E}{\partial O_j} \cdot \frac{\partial O_j}{\partial Net_j}) O_i =$$

$$= \eta(-\frac{\partial E}{\partial O_j}) f'_j(Net_j) O_i = \eta \delta_j O_i \qquad (2.29)$$

where $\partial E / \partial O_j$ may be computed according to

$$-\frac{\partial E}{\partial O_j} = -\sum_{k=1}^{L} \frac{\partial E}{\partial Net_k} \cdot \frac{\partial Net_k}{\partial O_j} = \sum_{k=1}^{L} (-\frac{\partial E}{\partial Net_k}) \frac{\partial}{\partial O_j} (\sum_{j=1}^{q} w_{jk} O_j) =$$

$$= \sum_{k}^{L} (-\frac{\partial E}{\partial Net_k}) w_{jk} = \sum_{k=1}^{L} \delta_k w_{jk}$$

Hence, we have

$$\delta_j = f'_j(Net_j) \sum_{k=1}^{L} \delta_k w_{jk} = O_j(1 - O_j) \sum_{k=1}^{L} \delta_k w_{jk} \qquad (2.30)$$

and consequently the hidden layer neuron weight adjustment becomes

$$\Delta_p w_{ij} = \eta f'_j(Net_{pj})(\sum_{k=1}^{L} \delta_{pk} w_{jk}) O_{pi} = \eta O_{pj}(1 - O_{pj})(\sum_{k=1}^{L} \delta_{pk} w_{jk}) O_{pi} \qquad (2.31)$$

6. Update and adjust weights and biases:

Each unit updates its bias and weights using a recursive algorithm according to

$$w_{ij}(n+1) = w_{ij}(n) + \eta \delta_j O_i \qquad (2.32)$$

where $w_{ij}(n)$ is the weight from hidden node j or from an input to node j at time n. O_i is either the output of node j or is an input. η is the learning gain, and δ_j is an error term for node j. If node j is an output node, then

$$\delta_j = O_j(1 - O_j)(d_j - O_j) \qquad (2.33)$$

Where d_j is the desired output of node j and O_j is the actual output. If node j is an internal hidden node, then

$$\delta_j = O_j(1 - O_j)\sum_{k=1}^{L}\delta_k w_{jk} \qquad (2.34)$$

Where k is over all the nodes in the layer above node j. Internal node thresholds are adapted in a similar manner by assuming that there are connection weights on links from auxiliary constant valued inputs. Convergence is sometime faster if a momentum term is added and the weight adaptation is smoothed according to

$$w_{ij}(n+1) = w_{ij}(n) + \eta\delta_j O_i + \alpha[w_{ij}(n) - w_{ij}(n-1)] \qquad (2.35)$$

where $0 < \alpha < 1$.

7. Stopping criterion:

Iterate the above calculations and operations by presenting new epochs of training examples to the network until the free parameters of the network stabilize their values and the average squared error computed over the entire training set is at a minimum or acceptably small value. The order of presentation of training examples should be randomized from epoch to epoch. The momentum and the learning rate parameters are typically adjusted as the number of training iterations increases.

2.2.3 Conjugate gradient backpropagation

To apply the conjugate gradient method to a backpropagation network, we first view backpropagation learning as an unconstrained nonlinear optimization problem. Thus, we require a vector of independent variables, x, and an objective function, f, defined on the vector space of x which is to be minimized. We must be able to evaluate $f(x)$ and its gradient $g(x) = \nabla f(x)$ at any point x. Recall that the goal of training the network is to modify the weights in such a manner that the network output for each pattern match the desired output. Therefore, the weights will be the vector of independent variables. Although, each weight is associated with a layer in the network and a particular neuron within that layer, for the purposes of the conjugate gradient optimization the weight is considered to be a single one-dimensional vector. The weights are ordered in a vector by layer and then by neuron and number within the neuron. The normalized sum of the output errors over the training set will be the objective function to be optimized. More formally, we define

$$f(x) = \frac{1}{2p}\sum_{p} E_p \tag{2.36}$$

where p is the number of patterns in the training set and E_p is the output error for each pattern p. Specifically, E_p is defined as

$$E_p = \frac{1}{2}\sum_{j}(d_{pj} - O_{pj}(x))^2 \tag{2.37}$$

where $O_{pj}(x)$ and d_{pj} are the actual and desired outputs of the jth output neuron associated with the pth pattern, respectively. We have denoted O_{pj} as a function of x to indicate the dependency of the network outputs on the weight vector x. Thus, the objective function is

$$f(x) = \frac{1}{2p}\sum_{p}\sum_{i}(d_{pi} - O_{pi}(x))^2 \qquad (2.38)$$

A single function evaluation requires that the entire training set be passed through the network, the errors be calculated for each pattern and the results summed and normalized. As the size of the weight vector and the number of patterns in the training set increase, the cost of computing $f(x)$ also increases.

To compute the gradient of the objective function, $\nabla f(x)$, we differentiate (2.36) with respect to x, which gives

$$g(x) = \frac{1}{p}\sum_{p}\nabla E_p(x) \qquad (2.39)$$

The derivation of $\nabla E_p(x)$ is nearly identical to the derivation of $(\partial E_p / \partial w_{ji})$ in Rumelhart [9] and is summarized briefly here. The vector $\nabla E_p(x)$ is composed of elements $(\partial E_p / \partial w_{ji})$, where x_{ji} is the neuron weight connecting the ith to the jth neuron. We write the derivative as a product of two parts

$$\frac{\partial E_p}{\partial x_{ji}} = \frac{\partial E_p}{\partial Net_{pj}} \cdot \frac{\partial Net_{pj}}{\partial x_{ji}} \qquad (2.40)$$

where $(\partial E_p / \partial Net_{pj})$ represents the change in E_p due to a change in the input of the jth neuron and $(\partial Net_{pj} / \partial x_{ji})$ represents the change in the input of the jth neuron due to a change in the weight x_{ji}. The input to the jth neuron, Net_{pj}, is defined as

$$Net_{pj} = \sum_{i} x_{ji} O_{pi}(x) \qquad (2.41)$$

where $O_{pi}(x)$ is the output of the ith neuron ($O_{pi}(x)$ is the ith input when i is an input neuron). Using (2.38), it can be shown that

$$\frac{\partial Net_{pi}}{\partial x_{ji}} = O_{pi}(x) \tag{2.42}$$

We now define

$$\delta_{pi} = \frac{\partial E_p}{\partial Net_{pi}} \tag{2.43}$$

Note that this differs from the δ_{pi} of Rumelhart [9] by a minus sign. Thus,

$$\frac{\partial E_p}{\partial x_{ji}} = \delta_{pj} O_{pi}(x) \tag{2.44}$$

Using this definition of δ_{pi} and following the same arguments as in Rumelhart [9], we arrive at the following relationships: For the output neurons

$$\delta_{pi} = -(d_{pi} - O_{pi}(x)) f_i'(Net_{pi}) \tag{2.45}$$

where $f_i'(Net_{pi})$ is the derivative of the semi-linear activation or squashing function, and for the hidden layer neurons

$$\delta_{pi} = f_i'(Net_{pi}) \sum_k \delta_{pk} x_{ki} \tag{2.46}$$

Therefore, evaluating the gradient requires forward propagation of each pattern in the training set through the network to generate the neuron outputs followed by propagating the values of δ_{pi} backwards through the network. The final gradient is computed by summing $\nabla E_p(x)$ over all the patterns. Since the conjugate gradient method requires evaluating both the function and the gradient values, the calculations should be performed together to maximize efficiency.

2.3 Conclusions

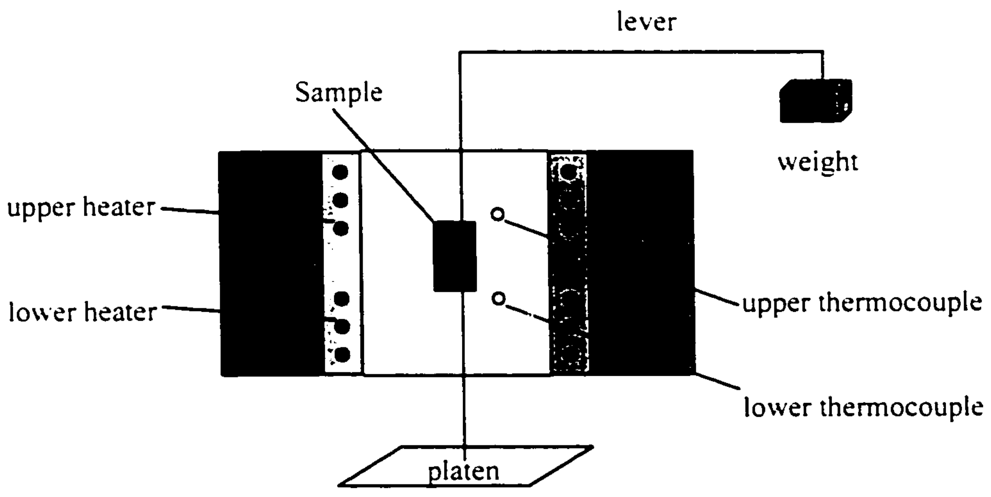
Differences among various topologies and operations of NN exists. but the neurons fundamentally operate in much the same manner. Different activation functions will affect the training results. The backpropagation (BPN) training algorithm allows experimental acquisition of input/output mapping knowledge by using multi-layered networks. Once an input pattern is applied to the input layer nodes of a three layer network. the information is processed and propagated to the hidden layer and on through to the output layer to generate an output pattern. This output pattern is then compared to the desired output pattern in generating an error signal. This error signal is then used to generate appropriately scaled error signal at each layer. The weights on the output layer are adapted using this error information. The error signal is then modified and then employed to adapt the hidden layer weights. This process of comparison of output and target values is continued until all patterns in the data set are learned to within some specified error bounds. The classical algorithm known as the generalized delta rule suffers from the fact that most of the parameters governing it such as learning rate. momentum. slope bias etc. should be fixed initially and thus are not adjusted during the learning process.

Chapter 3

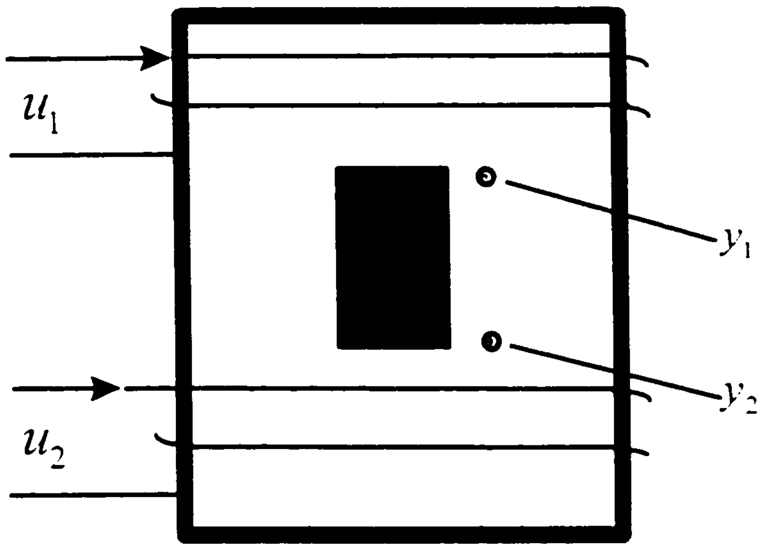
ELECTROTHERMAL FURNACE SYSTEM AND NEURAL NETWORK-BASED IDENTIFICATION & CONTROL

3.1 Description of an electrothermal furnace system and its computer control system

The electrothermal furnace system is widely used for mechanics performance tests on variety of metal samples under high-temperature (200 ^0C ~1200 ^0C). There are two sets of resistors (upper and lower) to heat the furnace to more than one thousand degrees. The block diagram is shown in Figure 3.1.



(a) Section diagram



(b) Block diagram

Figure 3.1 Electrothermal furnace system diagram

The system requirements are as follows:

1. The temperature of the electrothermal furnace varies in the range of 200 ^0C ~1200 ^0C.
2. The temperature of a given sample which is to be tested should be risen from an indoor temperature to the proper temperature setting as quickly as possible.
3. Furnace temperature should be stabilized and regulated to the setting temperature with ± 4 ^0C.
4. The temperature differential on the sample cannot be over ± 3 ^0C between upper and lower sections.

The computer control system for a typical electrothermal furnace system is shown in Figure 3.2 and Table 3.1.

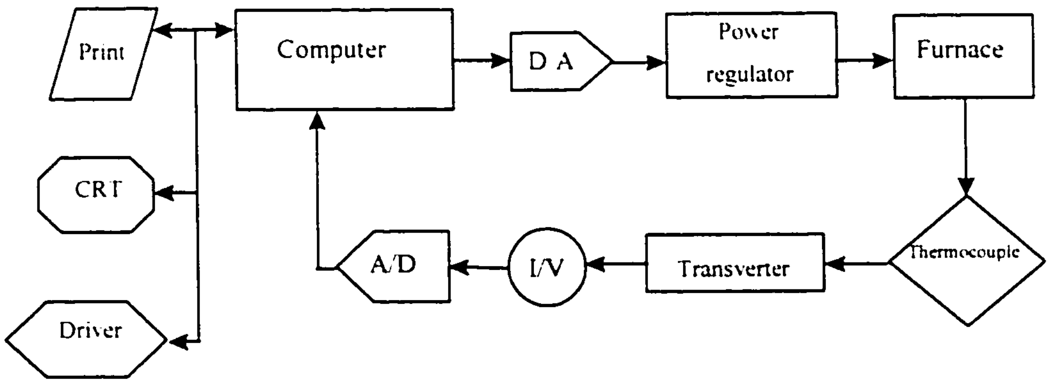


Figure 3.2 The computer control system

Name	Equipment Model
Computer	ICS Advent 7520p-A4 (Pentium)
A/D Card	CS14100
Power Regulator	TKDGB-20
Transverter	PDK-40
D/A Card	CSM14-2
Furnace	FMTB-10-j-72-k *
Control Software	ONSPEC control software for PC **

Table 3.1 A typical components for the electrothermal system

Table 3.1 shows a typical components for the electrothermal system. The system is composed of the following principle items:

* The furnace is made by Shanghai Material Test Equipment Co.

** ONSPEC is an advanced supervisory control and monitoring solution designed to work with plant-floor automation systems to provide real-time data acquisition and control to any plant environment. ONSPEC product offerings provide graphics. data acquisition. trending and alarming. spreadsheet manipulation and other functionalities used to perform command I/O. status. review and analysis. It is made by ONSPEC Automation Solutions. Rancho Cordova. CA.

The schematic of electrothermal furnace system is shown in Fig. 3.3.

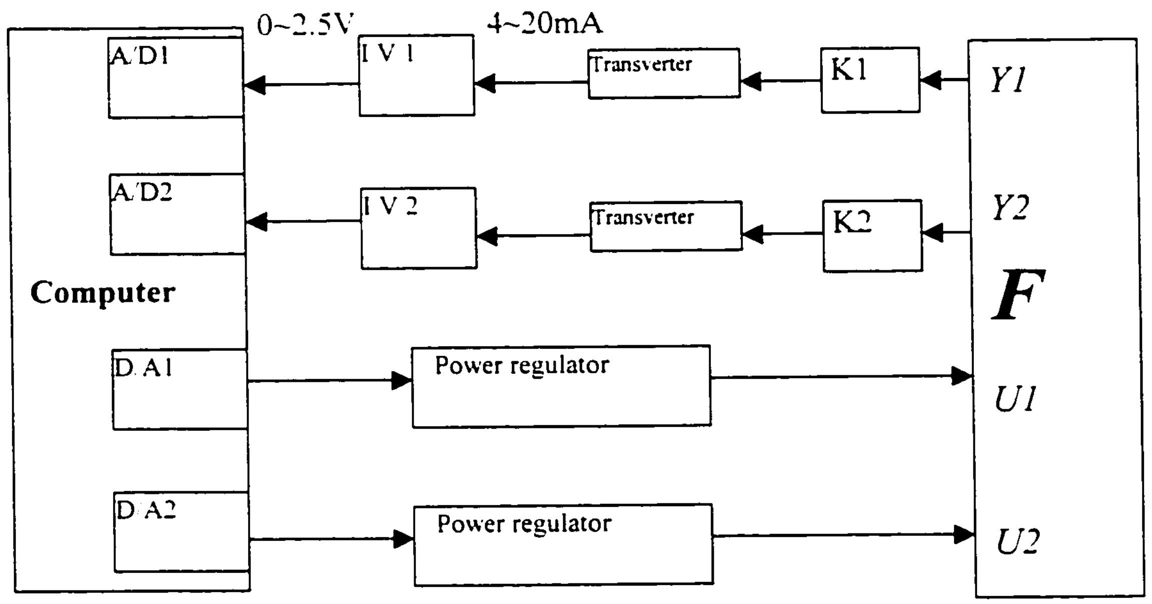


Fig. 3.3 The schematic of the electrothermal furnace system

The thermocouples measure the temperature (y_1 and y_2) of the electrothermal furnace and pass the temperature signals in millivolts to transverter. The transverter transmits the millivolts to a linear 4~20 mA signals. The I/V converters change the 4~20 mA signals to linear 0~2.5 V standard signals and send it to A/D boards. The computer process the data from A/D boards and generate an analog control signals through the D/A boards to the power regulators. The power regulators drive the electrothermal furnace (y_1 and y_1). This process is then repeated again. The Fig 3.4 is a picture of a real electrothermal furnace system. the left picture is the electrothermal furnace and the right picture is its computer control system.

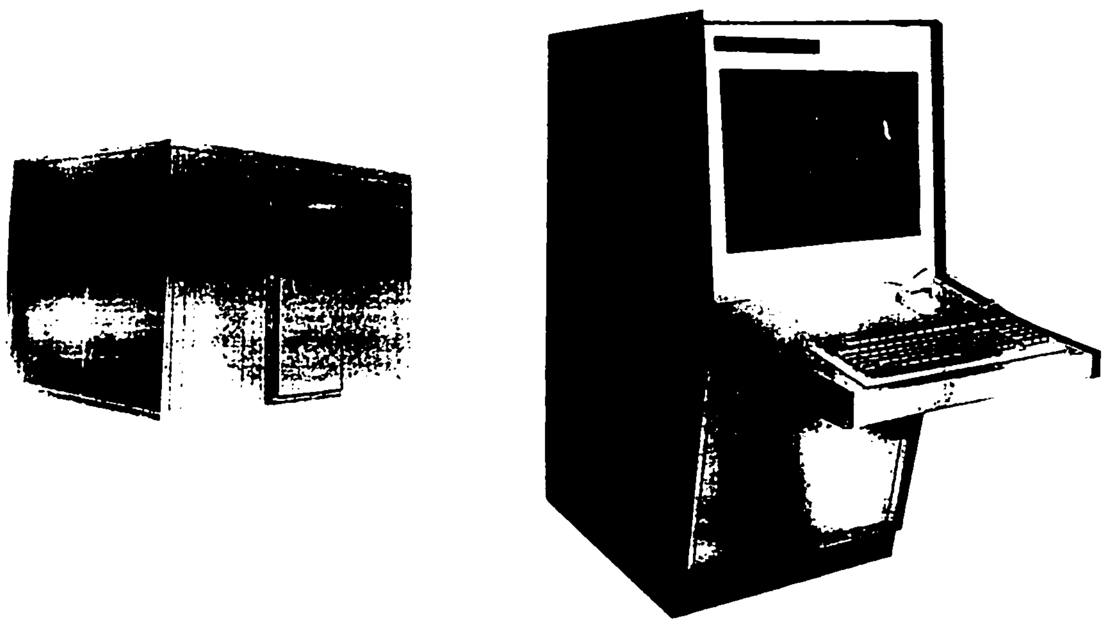


Fig. 3.4 Picture of the real electrothermal furnace system

The temperature of the electrothermal furnace system is controlled by two sets of resistive heaters (upper and lower). The temperature field is distributed and time-variant inside the furnace system. This implies that the temperature is a function of time and space.

Formally, the electrothermal furnace system may be described as a multivariable, non-linear, and a time-variant system. It is generally quite difficult to obtain an accurate model. Based on practical and real-life experience from the electrothemal furnace system, it can be stated that the system can be decomposed into two sub-systems. The mathematical model of each sub-system can be described in two parts. The first part is a pure sluggish inertial system close to an operating point. The other part is a measurable coupling that is interacting between the upper resistive heater set and the lower resistive heater set. The input-output model of the plant may be considered to be described by

$$y_1(k+1) = \alpha_{11}y_1(k) + \alpha_{12}y_1(k-1) + \beta_{11}u_1(k) + \beta_{12}u_1(k-1) + \gamma_{11}\sin\{\zeta_{11}[y_2(k) +$$

$$+ y_2(k-1)]\} + \gamma_{12}\cos\{\zeta_{12}[u_2(k) + u_2(k-1)]\}$$

$$y_2(k+1) = \alpha_{21}y_2(k) + \alpha_{22}y_2(k-1) + \beta_{21}u_2(k) + \beta_{22}u_2(k-1) + \gamma_{21}\sin\{\zeta_{21}[y_1(k) +$$

$$+ y_1(k-1)]\} + \gamma_{22}\cos\{\zeta_{22}[u_1(k) + u_1(k-1)]\} \qquad (3.1)$$

3.2 A neural network-based model for the electro-thermal furnace

The backpropagation neural network is a multi-layer mapping network that has a feedforward architecture. The use of the term "mapping" in describing a network implies that the network is capable of implementing an approximation to a variety of functions from an m-dimensional space R^m to an l-dimensional space R^l [6, 12, 14]. A typical multi-layer backpropagation network with an input layer, an output layer, and two hidden layers is shown in Figure 3.5. For convenience and simplicity we can also depict this in block diagram form as shown in Figure 3.6 with three weight matrices $w^{(1)}, w^{(2)}$, $w^{(3)}$ and a diagonal nonlinear operators $F^{(\cdot)}$ with identical sigmoidal activation functions F following each of the weight matrices. Each layer of the network can then be represented by the expression

$$N_i(Net) = F^{(i)}[w^{(i)}(Net)] \qquad (3.2)$$

where Net is the input vector to the network, so that the input-output mapping of the multi-layer network can then be represented by

$$y = N(x) = F^{(3)}\{w^{(3)}F^{(2)}[w^{(2)}F^{(1)}(w^{(1)}x)]\} = N_3N_2N_1(x) \qquad (3.3)$$

The weights of the network $w^{(1)}, w^{(2)}$ and $w^{(3)}$ are adjusted as described previously so as to mininize a suitable function of the error e between the output y of the network and a desired output y_d. This is then leading to realizing the mapping function $N(x)$ by the network, mapping any input signal into a corresponding output domain.

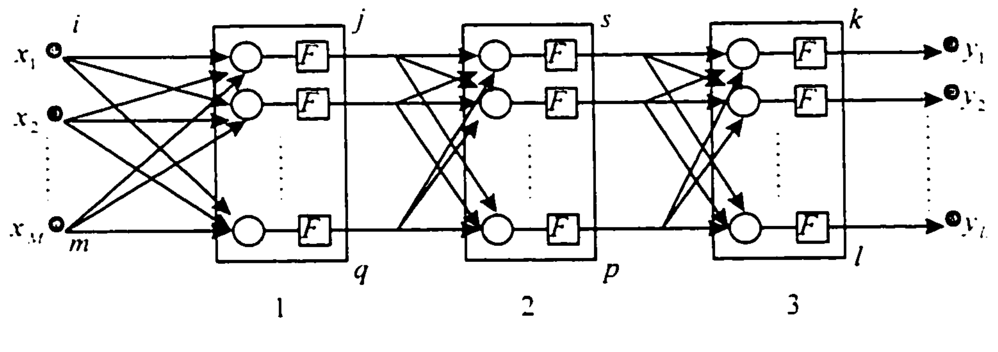


Figure 3.5 A three layer feedforward neural network

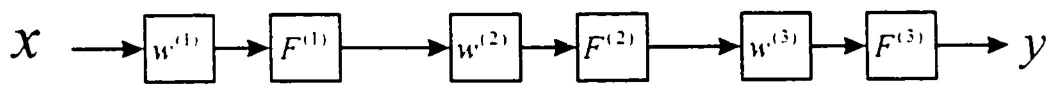


Figure 3.6 An equivalent block diagram representation of a three layer neural network

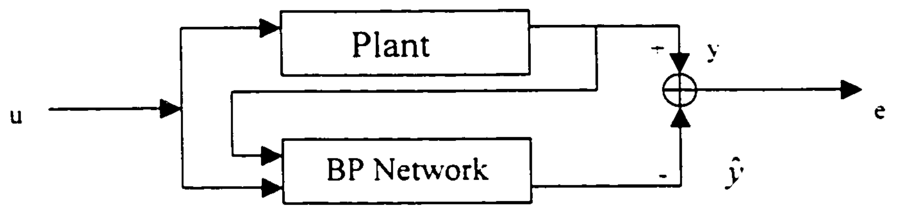
If the activation function (F) of the backpropagation network is taken as a linear operator, then

$$y = F^{(3)}\{w^{(3)}F^{(2)}[w^{(2)}F^{(1)}(w^{(1)}x)]\} = w^{(3)}w^{(2)}w^{(1)}x = wx \tag{3.4}$$

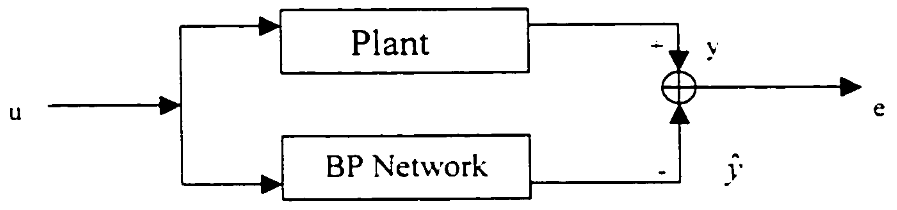
so that the backpropagation network is now equivalent to a linear network representing a linear mapping.

3.3 Dynamic system identification architectures

Conventional identification schemes may be categorized into two models, namely parallel model and the series-parallel model. These paradigms are shown in Figure 3.7. In Figure 3.7, the input is now denoted by u, the plant output by y, and the estimate of the plant output by $\hat{y}$.



(a) Series-parallel



(b) Parallel

Figure 3.7 Two types of model for identification

The backpropagation network in the series-parallel configuration utilizes the system input u and the output signal y from the plant for its inputs and it uses the output error signal e for the network weight adjustments. The backpropagation network in the parallel configuration, on the other hand, utilizes only the system input u for its input. It also uses the output error signal e for the network weight adjustments.

For linear time-invariant systems with unknown parameters, development of identification models are quite well known. Specifically, for a single-input single- output (SISO) controllable and observable plant, the dynamic equation can be written as

$$y_r(k+1) = \sum_{i=0}^{n-1} a_i y_r(k-i) + \sum_{j=0}^{m-1} b_j u(k-j) \tag{3.5}$$

where a_i and b_j are constant but unknown parameters. A similar representation is also possible for the multi-input multi-output (MIMO) system. The above expression implies that the output at time k+1 is a linear combination of the past values of both the input and the output signals. Equation (3.5) motivates the choice of the following auto regressive moving average (ARMA) identification models:

$$\hat{y}_r(k+1) = \sum_{i=0}^{n-1} \hat{a}_i \hat{y}_r(k-i) + \sum_{j=0}^{m-1} \hat{b}_j u(k-j) \tag{3.6}$$

(Parallel model)

$$\hat{y}_r(k+1) = \sum_{i=0}^{n-1} \hat{a}_i y_r(k-i) + \sum_{j=0}^{m-1} \hat{b}_j u(k-j) \tag{3.7}$$

(Series-parallel model)

where $\hat{a}_i(i = 0,1,\ldots,n-1)$ *and* $\hat{b}_j(j = 0,1,\ldots,m-1)$ are adjustable parameters. The output of the parallel identification model (3.6) at time k+1 is $\hat{y}_r(k+1)$ and is a linear combination of its past values as well as those of the input. In the series-parallel model, $\hat{y}_r(k+1)$ is a linear combination of the past values of the input and output of the plant. For developing stable and robust adaptive laws, the series-parallel model is found to be preferable. Further details may be found at [30].

Based on practical experiences and results reported in the literature. the mathematical model of an electrothermal furnace system may be approximated quite reasonably well by a linear system near an operating point. The electrothermal furnace system has two inputs and two outputs. We therefore decompose the system into two sub-systems.

Based on the above discussion. we consider the relationship between the input and the output variables of the sub-systems as a linear model. and represent the coupling interaction and activities between the two sub-systems as nonlinear functions. Consequently. the identification model of each sub-system includes both linear and nonlinear parts. Specifically. the input-output expressions for the electrothermal furnace system can be written as

$$\hat{y}_1(k+1) = a_{11}y_1(k) + a_{12}y_1(k-1) + b_{11}u_1(k) + b_{12}u_1(k-1) + \\ + NN_{21}[y_2(k), y_2(k-1), u_2(k), u_2(k-1)] \tag{3.8}$$

$$\hat{y}_2(k+1) = a_{21}y_2(k) + a_{22}y_2(k-1) + b_{21}u_2(k) + b_{22}u_2(k-1) + \\ + NN_{12}[y_1(k), y_1(k-1), u_1(k), u_1(k-1)] \tag{3.9}$$

The above identification models of the electrothermal furnace system will be used in our experimental results reported later in Chapter 5.

3.4 Conclusions

The electrothermal furnace system is widely used for mechanical performance tests on variety of metal samples under high-temperatures. The electrothermal furnace system is a MIMO nonlinar system. The mathematical representation of the system may be expressed into two sub-systems. The first sub-system is a pure sluggish inertial system when the system is close to its operating point. and the other sub-system is a nonlinear system representing coupling and disturbances effects.

In this chapter. a three-layer feedforword neural network is proposed to be used for the electrothermal furnace system. Two possible identification configurations are presented. It is proposed that the series-parallel configuration will be used subsequently for our electrothermal furnace system.

Chapter 4

NEURAL NETWORK-BASED CONTROLLERS

4.1 A neural network-based PID controller

The conventional PID controller is widely used in practice due to its simple structure and ease of implementation. However, in certain time-variant systems the PID parameters cannot be easily adjusted online. Consequently, it is necessary to design a neural network PID controller strategy to cope with these uncertainties and system parameter variations.

4.1.1 A discrete-time PID control

The expression for a continuous-time PID controller is quite well-known and is given by

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}] \qquad (4.1)$$

where K_p denotes the proportional gain. T_i denotes the integral time constant and T_d denotes the derivative gain.

When the sampling period is sufficiently small. the PID controller in discrete-time representation becomes

$$\Delta u(k) = K_p[\Delta e(k) + \frac{T_0}{T_i}e(k) + \frac{T_d}{T_0}\Delta^2 e(k)] = K_I e(k) + K_p\Delta e(k) + K_D\Delta^2 e(k) \qquad (4.2)$$

where T_0 is the sampling period. K_I is the integral coefficient given by $K_I = \frac{K_p T_0}{T_i}$. K_D is the derivative coefficient given by $K_D = \frac{K_p T_d}{T_0}$.

and Δ^2 is given by $\Delta^2 = 1 - 2z^{-1} + z^{-2}$.

4.1.2 A PID neural controller and learning algorithms

A schematic of a PID neural controller applied to a control system is shown in Figure 4.1. where $y_r(k)$ is reference output setting. $y(k)$ is plant output. and x_1. x_2 and x_3 are neural network state variables defined as $x_1(k) = e(k).x_2(k) = \Delta e(k)$.

$x_3(k) = e(k) - 2e(k-1) + e(k-2)$ *and* $z(k) = y_r(k) - y(k) = e(k)$ is the tracking error.

Specifically, we have

$$u(k) = u(k-1) + K\sum_{i=1}^{3} w_i(k)x_i(k) \qquad (4.3)$$

where $w_i(k)$ is the weight coefficient associated with $x_i(k)$, K is the proportional coefficient of the neuron, with K >0.

The neural network PID controller achieves process control through adjustment of the weight coefficients. The adjustment of these weights is accomplished according to the cost or performance function (2.13), which depends on the tracking error of the neural network.

In other words, the adjustment of the weights are governed by

$$w_i(k+1) = (1-c)w_i(k) + \eta\gamma_i(k) \qquad (4.4)$$

where

$$\gamma_i(k) = z(k)u(k)x_i(k) \qquad (4.5)$$

With $\gamma_i(k)$ denotes the incremental training signal, η denotes the learning rate, η >0;

$z(k)$ denotes the output error tracking signal, $z(k) = y_r(k) - y(k)$;

c denotes the weight adiustment constant, $c \geq 0$.

Consequently,

$$\Delta w_i(k) = -c[w_i(k) - \frac{\eta}{c} z(k)u(k)x_i(k)] \qquad (4.6)$$

where

$$\Delta w_i(k) = w_i(k+1) - w_i(k)$$

If we define a function $f_i(w_i(k), z(k), u(k), x_i(k))$, such that

$$\frac{\partial f_i}{\partial w_i} = w_i(k) - \frac{\eta}{c}\gamma_i(z(k), u(k), x_i(k)) \qquad (4.7)$$

then (4.6) can be rewritten as

$$\Delta w_i(k) = -c\frac{\partial f_i(\cdot)}{\partial w_i(k)} \qquad (4.8)$$

When c is sufficiently small, then $w_i(k)$ can converge to its static equilibrium value w_i^{*}.

To summarize the above derivations, the neural network-based PID controller is expressed as follow:

$$u(k) = u(k-1) + K\sum_{i=1}^{3} w_i(k)x_i(k)$$

where

$$w_i(k) = \frac{w_i(k)}{\sum_{i=1}^{3}|w_i(k)|}$$

$$w_1(k+1) = w_1(k) + \eta_I z(k)u(k)x_1(k)$$

$$w_2(k+1) = w_2(k) + \eta_P z(k)u(k)x_2(k)$$

$$w_3(k+1) = w_3(k) + \eta_D z(k)u(k)x_3(k)$$

with

$x_1(k) = e(k);$

$x_2(k) = \Delta e(k);$

$x_3(k) = \Delta^2 e(k) = e(k) - 2e(k-1) + e(k-2)$

η_I, η_P, η_D denote the integral, proportion and derivative learning rates.

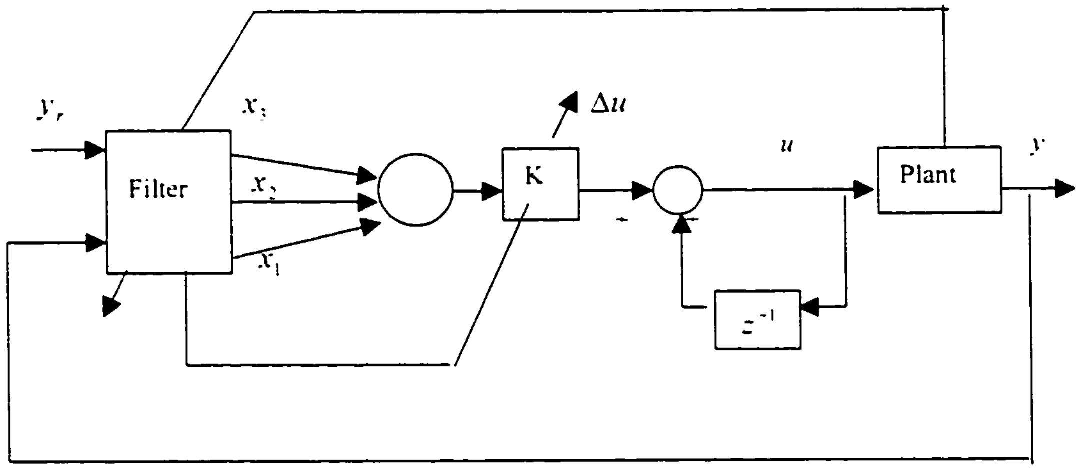


Figure 4.1 A single neuron PID controller

4.2 A neural network-based IMC controller architecture

The internal model control (IMC) structure provides a direct method for the design of nonlinear feedback controllers [16], [17]. According to the properties described below, if a good model of the plant is available, the closed-loop system will achieve satisfactory set-point tracking despite uncertainties and disturbances acting on the plant [22].

4.2.1 The internal model principle

We shall consider an error-driven control system as shown in Fig. 4.2. When an output disturbance d(t) is acting on the system, one is then led to the structure shown in Fig. 4.3.

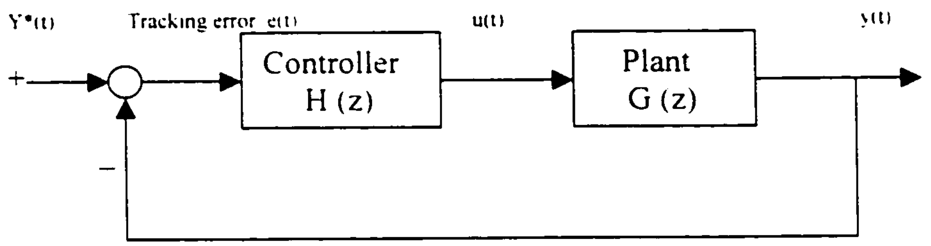


Fig.4.2 Error-driven architecture of a linear control system

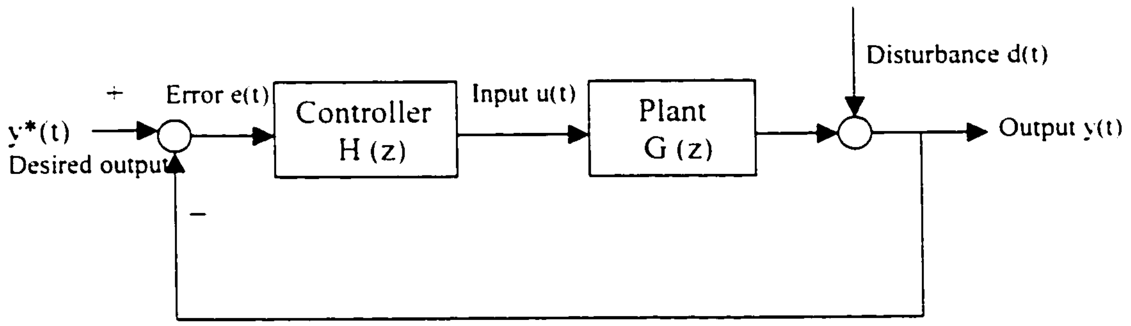


Fig.4.3 Error-drive architecture of a linear system with external disturbance

For a system expressed in the discrete-time domain and using the z-transform, the output Y(z) can be expressed in terms of the transform of the set-point sequence, Y*(z), and of the disturbance, D(z), as follows (refer to Figure 4.3)

$$Y(z) = \frac{G(z)H(z)}{1+G(z)H(z)}Y^*(z) + \frac{1}{1+G(z)H(z)}D(z)$$

The corresponding frequency response of the output is obtained by setting $z = e^{jw}$, that is,

$$Y(e^{jw}) = \frac{G(e^{jw})H(e^{jw})}{1+G(e^{jw})H(e^{jw})}Y^*(e^{jw}) + \frac{1}{1+G(e^{jw})H(e^{jw})}D(e^{jw})$$

Thus, we see that the output will faithfully reproduce the desired output, provided that the loop gain, $[G(e^{jw})H(e^{jw})]$, is sufficiently large over the range of frequencies dominant in both $Y^*(e^{jw})$ and $D(e^{jw})$.

In fact, the loop gain can be made infinite at a set of frequencies by placing the corresponding modes in the characteristic polynomial of the controller, H(z). In the case of a constant disturbance or a constant desired output, the frequency of interest is zero, and thus one should include $(1-q^{-1})$ in the controller. This clearly gives integral action as commonly employed in practical control system design [15].

The IMC structure is quite well-know and has been extensively studied in the literature. Furthermore, it has been shown to cover and characterize a number of other control design techniques of apparently different origin, in addition to having a number of desirable properties.

The nonlinear IMC structure is shown in Fig. 4.4.

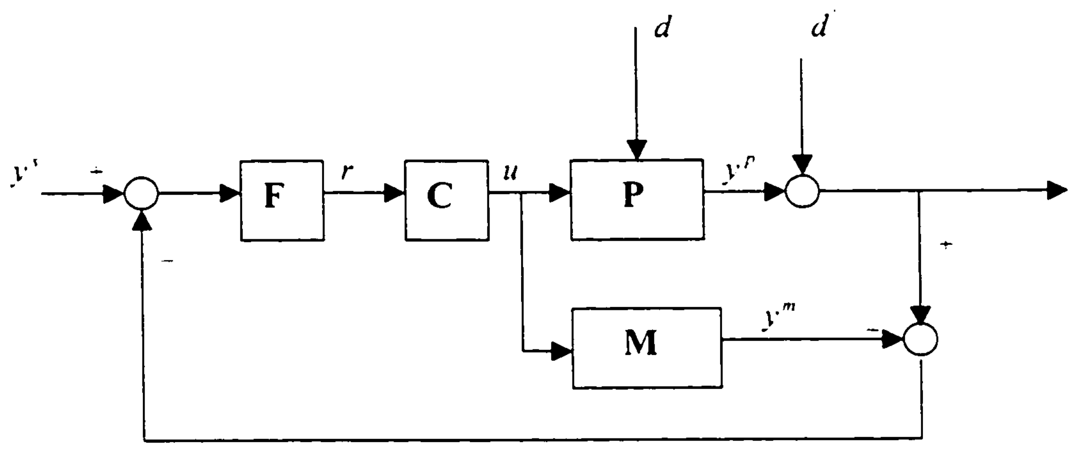


Fig. 4.4 Nonlinear IMC architecture

The nonlinear operators denoted by P, M and C represent the plant, the plant model, and the controller, respectively. The operator F denotes a filter, which is to be discussed subsequently.

The important characteristics of the above IMC architecture are summarized as following [36]:

Property 1: Assume that the plant and the controller are input-output stable and that the model is an ideal representation of the plant. Then, the closed-loop system is input-output stable.

Property 2: Assume that (i) the inverse of the operator describing the plant model exists. (ii) that this inverse is used as the controller. and (iii) that the closed-loop system is input-output stable with this controller. Then. the design performance specification can be satisfied with proper selection of the system parameters.

Property 3: Assume that (i) the inverse of the steady state model of the operator exists. (ii) that the steady state controller operator is equal to this. and (iii) that the closed-loop system is input-output stable with this controller. Then, offset free control performance is attained for asymptotically constant inputs.

The IMC structure above provides a direct method for the design of nonlinear feedback controllers. According to the above properties. if a good model of the plant is available. the closed-loop system will achieve exact set-point tracking despite unmeasured disturbances acting on the plant.

Discussion so far has considered only the idealized case of a perfect model. leading to a so-called perfect control. In practice. however. a perfect model can never be obtained. In addition. the infinite gain required by the control law would lead to sensitivity problems under model uncertainty. To overcome and address some of these issues the filter F is introduced in the forward path. By suitable design. the filter can be selected to reduce the feedback gain system. thereby deviating from a perfect controller. This introduces robustness into the IMC structure. A full treatment on robustness and filter design issues for IMC are given by Morari and Zafiriou [16]. The second role of the filter F is to project the error signal e into an appropriate input space for the controller.

The IMC architecture presented above is critical because the stability and robustness properties of the structure can be analyzed and manipulated in a transparent manner. even for a nonlinear system. Thus IMC provides a general framework for nonlinear system control. Such generality is not apparent in other approaches available to nonlinear control.

4.2.2 Nonlinear IMC using neural networks

We propose a two steps procedure for using neural networks directly within the IMC structure. The first step involves training a network to represent the plant model. This network is then used as the plant model operator M in the control structure shown in Fig. 4.4. The schematic shown in Fig. 4.5 provides the methodology for training the network to represent or model the plant. Here, the error signal used to adjust the network weights is the difference between the plant output and the network output. Thus, the network is forced, in a sense, to copy the plant dynamics.

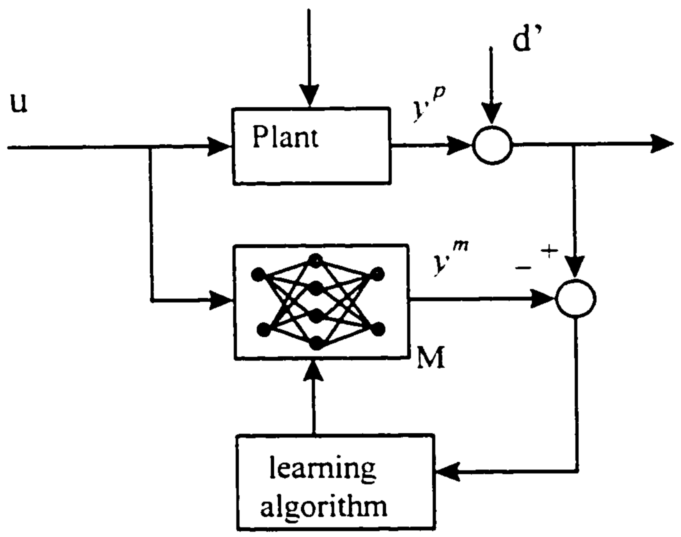


Fig. 4.5 Neural network-based modelling of the plant

Following standard IMC practice we select the controller as the plant inverse model. The next step in the procedure is to train a second network to represent the inverse of the plant. To do this we use the architecture shown in Fig. 4.6. For reasons to be explained later, we employ the plant model output y^m in inverse learning architecture rather than the plant output y^r itself. For inverse modelling, the error signal used to adjust the network weights is defined as the difference between the network input and the plant model output. This tends to force the map between the reference input r and the plant model output y^r into a unity map. Once the inverse model is obtained, the network is used as the controller block C in the control structure of Fig. 4.4. The final IMC architecture incorporating the trained networks is shown in Fig. 4.7.

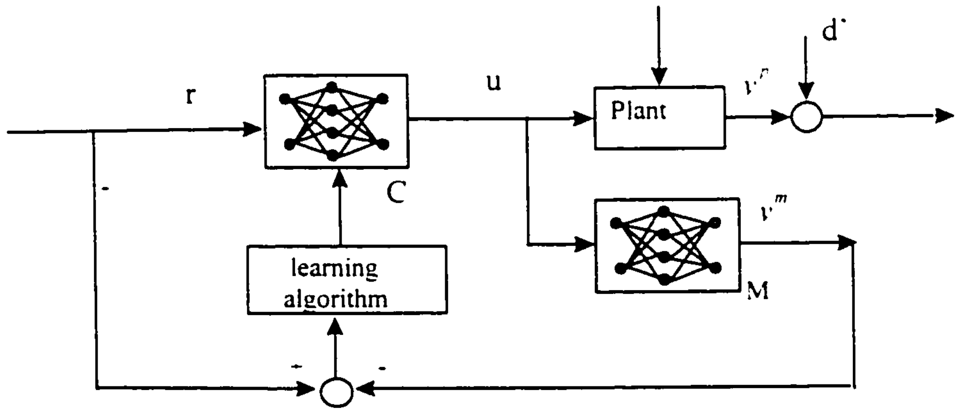


Fig. 4.6 Specialized IMC learning structure

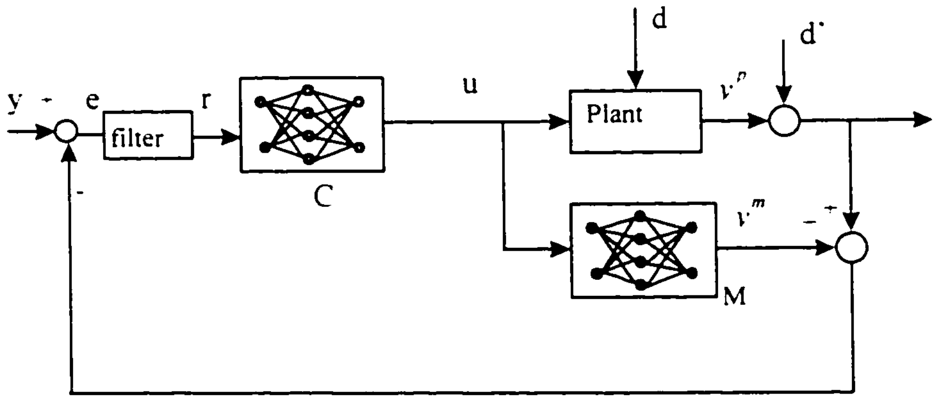


Fig. 4.7 General IMC neural network-based architecture

4.2.3 Invertibility of discrete-time nonlinear system

The inversion of nonlinear operators plays a central role in the development of nonlinear IMC. In this subsection we briefly explore and study the invertibility of nonlinear dynamical systems. Consider a general analytic system Σ governed by the following nonlinear difference equation

$$\sum y^r(k+1) = f(y^r(k),....,y^r(k-n),\ u(k),....,u(k-m)), \qquad y^r \in \Re, u \in \Re \qquad (4.9)$$

where n is the order of the system. with $m \leq n$. For simplicity we consider single-input single-output systems. although the approach can also be generalized to multivariable systems.

The system Σ is defined to be invertible at $[y^P(k),\ldots,y^P(k-n),u(k-1),\ldots,u(k-m)]^T$ if there is a subset A of $\Re^{m+n+1}$, such that for $[y^P(k),\ldots,y^P(k-n),u(k-1),\ldots,u(k-m)]^T \in A$,

$f(y^P(k),\ldots,y^P(k-n);u^1(k),\ldots,u(k-m)) \neq f(y^P(k),\ldots,y^P(k-n);u^2(k),\ldots,u(k-m))$

for any distinct inputs $u^1(k),u^2(k) \in R$.

The system Σ is singular if for any $[y^P(k),\ldots,y^P(k-n),u(k-1),\ldots,u(k-m)]^T \in A$ and for any distinct inputs $u^1(k),u^2(k)$, the resulting outputs are equal:

$$f(y^P(k),\ldots,y^P(k-n);u^1(k),\ldots,u(k-m)) = f(y^P(k),\ldots,y^P(k-n);u^2(k),\ldots,u(k-m))$$

This follow from the observation that if $f(y^P(k),\ldots,y^P(k-n);u(k),\ldots,u(k-m))$ is monotonic and $u^1(k) \succ u^2(k)$ then for the same point

$[y^P(k),\ldots,y^P(k-n),u(k-1),\ldots,u(k-m)]^T$,

$$\begin{aligned} f(y^P(k),\ldots,y^P(k-n);u^1(k),u(k-1),\ldots,u(k-m)) \succ \\ f(y^P(k),\ldots,y^P(k-n);u^2(k),u(k-1),\ldots,u(k-m)) \end{aligned}$$

Similarly if $u^1(k) \prec u^2(k)$ then

$$\begin{aligned} f(y^P(k),\ldots,y^P(k-n);u^1(k),u(k-1),\ldots,u(k-m)) \prec \\ f(y^P(k),\ldots,y^P(k-n);u^2(k),u(k-1),\ldots,u(k-m)) \end{aligned}$$

and the system is therefore invertible.

Therefore, if $f(y^P(k),\ldots,y^P(k-n);u(k),\ldots,u(k-m))$ is monotonic with respect to $u(k)$ then the system is invertible at $[y^P(k),\ldots,y^P(k-n),u(k-1),\ldots,u(k-m)]^T$.

4.2.4 Learning algorithms

The architectures used for plant modelling and plant inverse modelling were described in previous sections. The learning laws used for training these networks are now considered below.

The plant is to be modelled using a network that may be described by the following input-output representations

$$y^m(k+1) = \sum_{i=1}^{N_m} c_i^m k_i^m \tag{4.10}$$

where

$$K_i^m = \exp(-d_i^m(x^m(k), x_i^m, \Delta^m)) \tag{4.11}$$

Here, the m superscript designates a variable that is related with the plant model. The structure of the plant model is chosen to be the same as that of the plant, where the model output is a nonlinear function of the present and past plant outputs $y^p(k)......y^p(k-n)$, and the present and past plant inputs $u(k)......u(k-m)$. The neural network model input vector $x^m(k)$ is thus given as

$$x^m(k) = [y^p(k),...., y^p(k-n), u(k),....u(k-m)]^T$$

We denote the centre of the Gaussian function of the hidden unit i as

$$x_i^m = [y_{i,0},...., y_{i,n}, u_{i,0},...., u_{i,m}]^T$$

The parameters x_i^m and Δ^m are fixed to meet the representation (interpolation) conditions. The parameters c_i^m is adjusted to minimize the mean square error between the real plant and the model according to the following rule

$$c_i^m(k+1) = c_i^m(k) + \alpha\, k_i (y^r(k+1) - y^m(k+1)) \qquad (4.12)$$

where $0 \leq \alpha \prec 1$ is a gain parameter. Fig. 4.5 depicts the structure to be used. Using standard systems theory it can be shown [45] that if the plant can be modelled by (4.10). the least mean square solution can be found that satisfies (4.12) [45].

If the plant (or equivalently its model) is invertible then the inverse of the plant can be approximated by using the similar approach as that used for the plant. This inverse model is now used as the controller. For robustners reasons we choose to use the plant model inverse rather than the inverse of the actual plant. We utilize the second network responsible for inverse modelling of the plant dynamics according to the following expression

$$u(k) = \sum_{i=1}^{N^c} c_i^c k_i^c \qquad (4.13)$$

where

$$k_i^c = \exp(-d_i^c(x^c(k), x_i^c, \Delta^c)) \qquad (4.14)$$

where the C superscript designates a variable that is related with the controller. The inverse of the function f in (4.9) depends on the future plant output value $y^r(k+1)$. In order to obtain an implementable approximation, we replace this value by the controller input value r. Finally, since we need to approximate the inverse of the plant model, we select the controller network input vector $x^c(k)$ as

$$x^c(k) = [y^m(k), \ldots, y^m(k-n), r(k+1), u(k-1), \ldots, u(k-m)]^T$$

where r(k+1) is determined at the time k by a suitable definition of the IMC filter F. The centre of the Gaussian function of the hidden unit i is given by

$$x_i^c = [y_{i,0},\ldots,y_{i,n},r_i,u_{i,1},\ldots,u_{i,m}]^T$$

The detailed descriptions for the learning rules are descried below. Both iterative and non-iterative methods are considered.

(i) Non-iterative method: To adjust c_i^c in (4.13), the architecture shown in Fig. 4.6 is used. This architecture is similar to the specialized learning architecture presented by Psaltis [24]. The parameters c_i^c are adjusted to minimize the mean square error between the output of the model and the input of the controller. This leads to the following learning algorithm [24]:

$$c_i^c(k+1) = c_i^c(k) + \alpha k_i^c\,(r(k+1) - y^m(k+1))\frac{\partial y^m(k+1)}{\partial u(k)} \qquad (4.15)$$

Note that if the actual plant output was used in the control architecture, then in the learning algorithm $\partial y(k+1)/\partial u(k)$ would have been required to be calculated. This can be achieved approximately by using a first-order difference changes of each input to the plant and by measuring the resulting changes at the output. On the hand, by using the plant model output, the corresponding derivative $\partial y^m/\partial u$ can be calculated explicitly. Specifically, from (4.10) one obtains

$$\frac{\partial y^m(k+1)}{\partial u(k)} = -2\Delta^m \sum_{i=1}^{N^m} c_i^m k_i^m (u(k) - u_{i,0}) \qquad (4.16)$$

Another possible approach involves the use of a synthetic signal [47]. This lead to the so-called general learning architecture as shown in Fig. 4.8. In this case the adaptation law for the weight adjustments does not depend explicitly on the derivative of the plant output, namely we have

$$c_i^c(k+1) = c_i^c(k) + \alpha\, k_i^c(s_i - u(k)) \qquad (4.17)$$

where s_s is the synthetic signal [47].

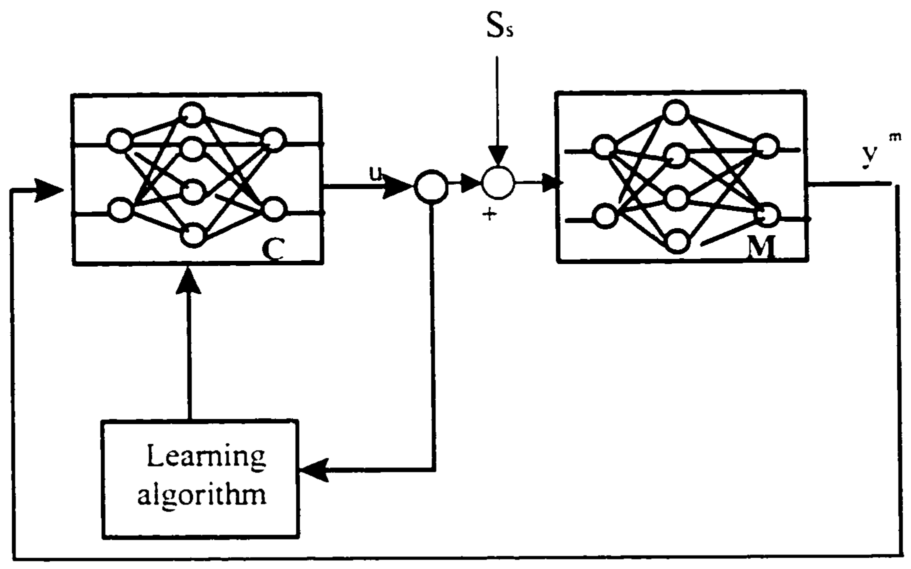


Fig. 4.8 Use of the synthetic signal leading to the general learning structure

(ii) Iterative method: This method uses the plant model to obtain its inverse. A recursive method is used to find the inverse of the model at each operating point. This method can also be used for singular systems which only satisfy the invertibility conditions described earlier locally (that is invertibility conditions are not satisfied in the whole operating space). This approach can also be used when it is necessary to have "small" networks due to computational memory or processing limitations and constraints. In this case the restricted accuracy of the trained network can be enhanced by using the network to

provide stored initial values for the iterative method, establishing a compromise between speed of convergence and storage capacities.

At time k, the objective is to find an input u which will produce a model output $y^m(k+1)$ equal to r(k+1). It can be shown that is possible to use the method of successive substitution [27] as described below

$$u^{n+1}(k) = u^n(k) + \gamma\,(r(k+1) - y^m(k+1))$$

where γ is a weight to be chosen by trial and error or other heuristic methods .

4.3 Conclusions

In this chapter, two neural network-based controllers (PID and IMC) are proposed. Both of these architectures will be used in experimental and simulation results obtained for the electrothermal furnace system.

It is feasible in theory to use neural network learning laws to adjust the PID parameters online. The thesis instead proposed a neural network-based PID controller architecture.

The use of artificial neural networks for nonlinear IMC has also been explored. We studied the invertibility of a class of nonlinear dynamical systems and derived an invertibility characterization. We proposed architectures and algorithms for training networks to represent nonlinear dynamic maps along with the inverse of these systems.

Chapter 5

IDENTIFICATION AND CONTROL RESULTS

The electrothermal furnace system is now used as an application system for our study. As discussed in previous chapters the electrothermal furnace is used for mechanical performance tests of variety of metal samples under high temperatures ($200^{\circ}c \sim 1200^{\circ}c$). Based on practical and field experiences, the mathematical model of an electrothermal furnace system is believed to be approximated quite reasonably accurately as a linear system near an operating point. The electrothermal furnace has two inputs and two outputs, representing a multivariable system. We separate the process into two sub-systems. According to the previous discussions and results in earlier chapters, we consider the relationship between the input and output variables of each sub-system as a linear model, and treat the coupling factors between the two sub-systems as a nonlinear measurable influence. Hence, the identification model for each sub-system includes both linear and nonlinear parts. The mathematical representation of the above discussion may be written as

$$\hat{y}_1(k+1) = a_{11}y_1(k) + a_{12}y_1(k-1) + b_{11}u_1(k) + b_{12}u_1(k-1) + \\ + NN_{21}[y_2(k), y_2(k-1), u_2(k), u_2(k-1)]$$

$$\hat{y}_2(k+1) = a_{21}y_2(k) + a_{22}y_2(k-1) + b_{21}u_2(k) + b_{22}u_2(k-1) + \\ + NN_{12}[y_1(k), y_1(k-1), u_1(k), u_1(k-1)] \qquad (5.1)$$

5.1 Experimental BP-based architecture for system identification

The series-parallel configuration is chosen for our electrothermal furnace system identification purpose. The series-parallel identification architecture is depicted in Fig. 5.1. Note the past values of the input and the output signals of the plant are achieved though Tapped Delay Line (TDL) operators to form the input vector to the back-propagation neural network, whose output signal $\hat{y}_p(k)$ corresponds to and is to represent an estimate of the plant output y_p at the instant of time k.

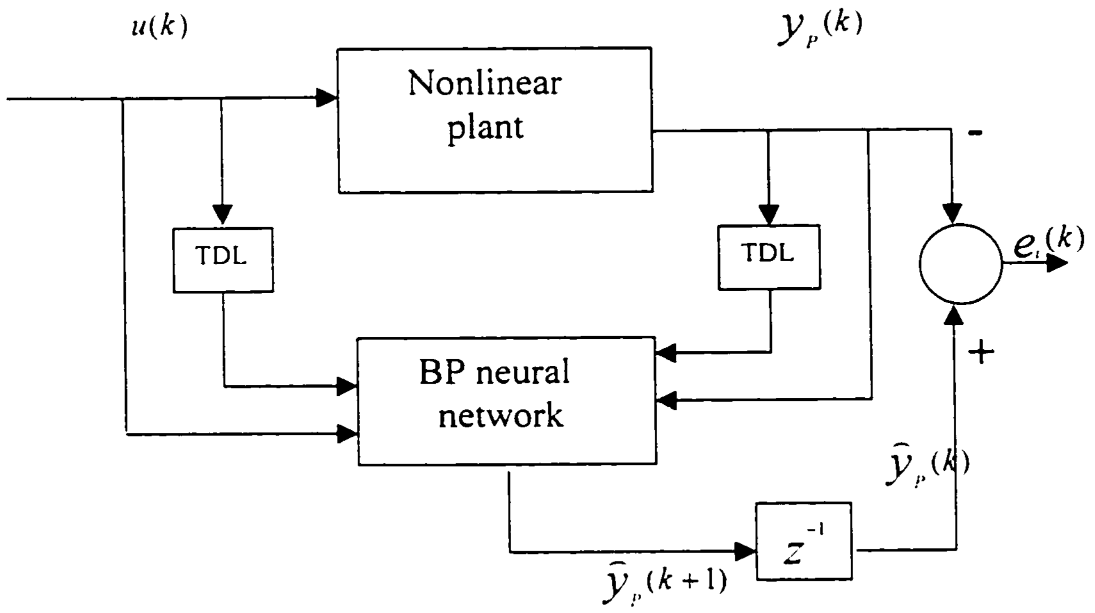


Fig. 5.1 Identification of the nonlinear system using BP network

The backpropagation training procedure is an iterative gradient decent based algorithm designed to minimize the mean square error between the actual output of a multi-layer feedforward network and the desired output. It requires usage of a continuously differentiable nonlinear activation function. In the following section we assume to employ a sigmoidal logistic nonlinearity where the activation function is as shown in Figure 5.2.

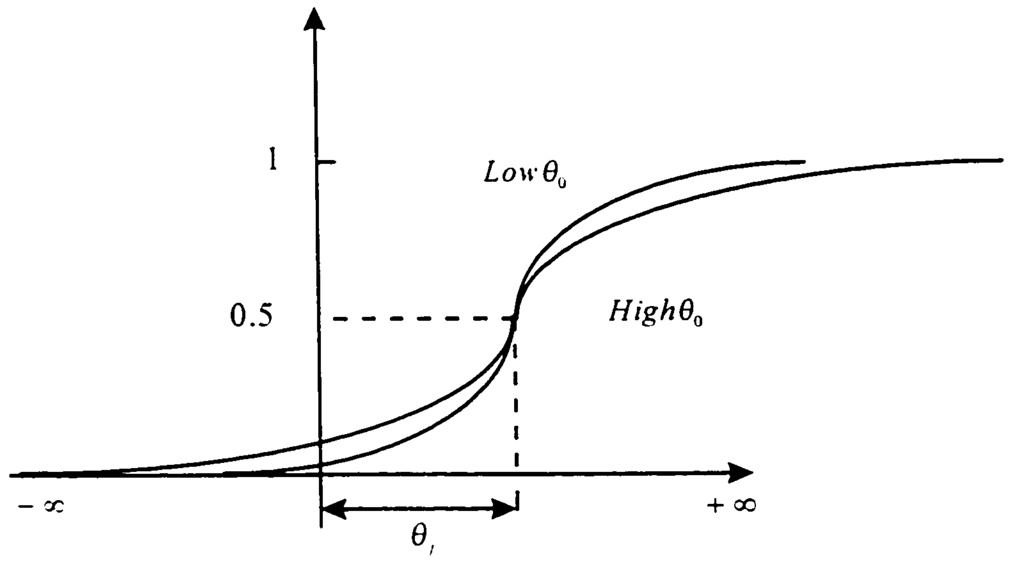


Fig. 5.2 Sigmoidal activation function

$$f(x) = f(Net_j) = \frac{1}{1+e^{-(Net_j - \theta_j)/\theta_0}} \tag{5.2}$$

where θ_j is the threshold of the *j*th neuron.

5.2 The neural network-based identification results

We utilized the series-parallel identification architecture discussed in the previous section for our nonlinear electrothermal furnace system. A network with 13 neurons in the hidden layer is selected for the identification of the nonlinear part of the process. The selected architecture is shown in Fig. 5.3

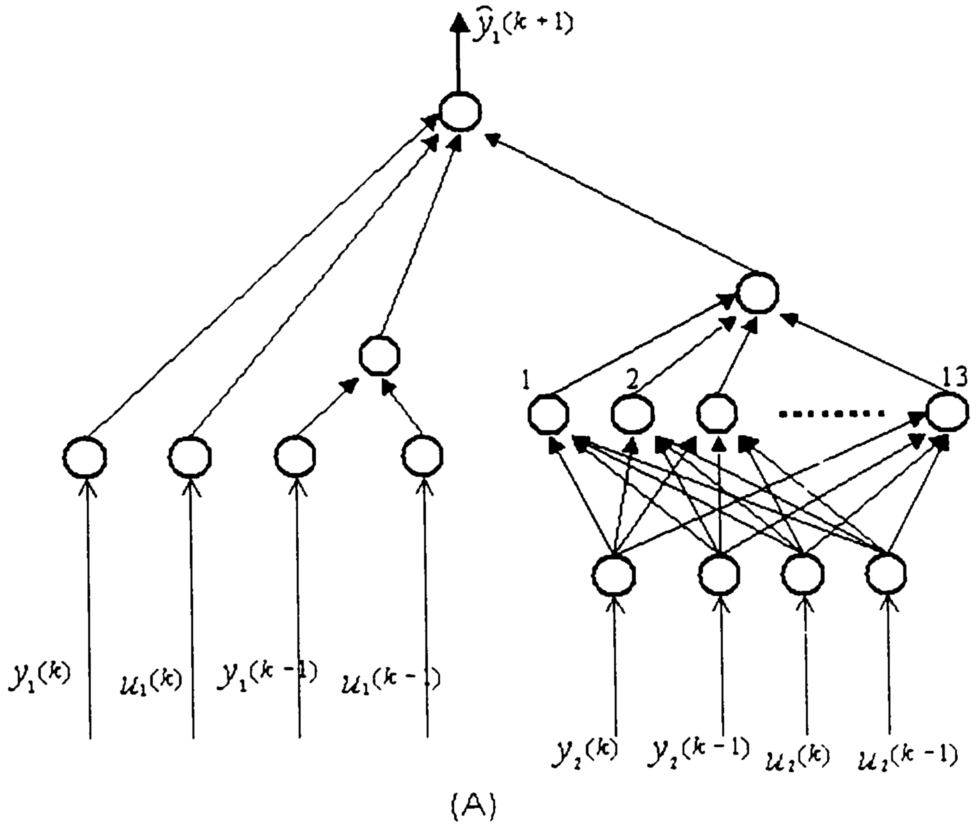
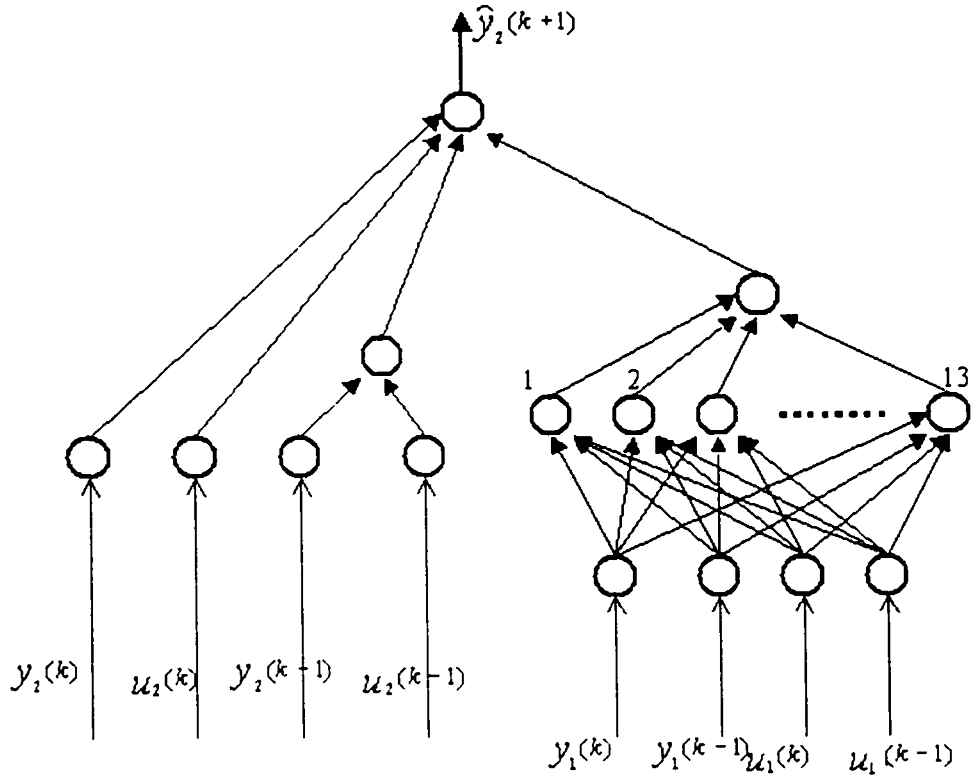


Fig. 5.3 Structure of the BP neural network identifier for the furnace system

(A) identification of the first output y_1



(B)

Fig. 5.3 Structure of the BP neural network identifier for the furnace system

(B) identification of the second output y_2

Note that the following specific information is used in the above neural network-based identifier

Sigmoidal function: $y = \left(\frac{18}{1+e^{-0.3x}} - 12\right)$;

Learning rate: $\eta = 0.025$, $\alpha = 0$

Sampling period: T = 90 sec.

The experimental results after 537 epochs for our system identification are shown in Fig. 5.4 and Fig. 5.5. The desired error requirement is set a priori to 0.2.

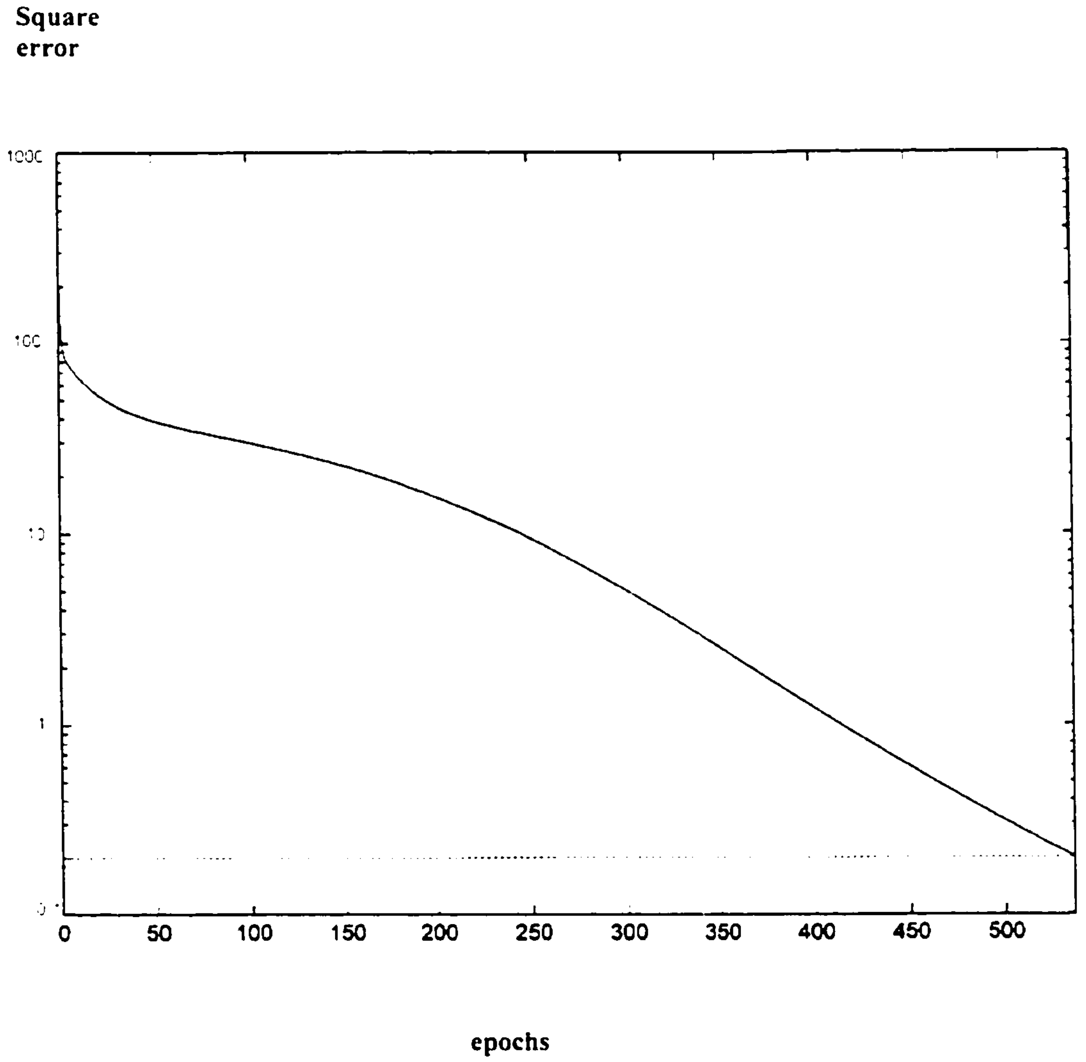


Fig. 5.4 The sum-squared error for the first output Y1

The partial mathematical representation for the estimate of the first output y_1 may now be expressed as follows:

$$\hat{y}_1(k+1) = 0.770142y_1(k) + 0.069923y_1(k-1) + 0.052985u_1(k) + 0.044987u_1(k-1) +$$
$$+ NN_{21}[y_2(k), y_2(k-1), u_2(k), u_2(k-1)] \qquad (5.4)$$

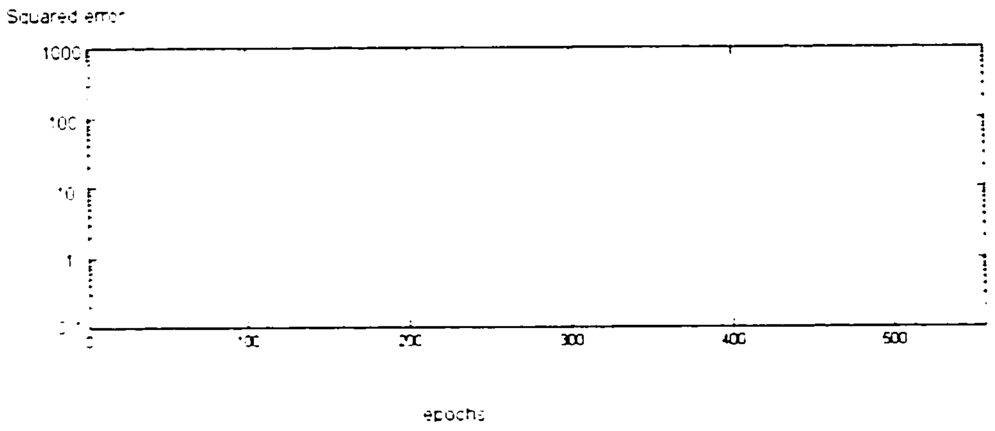


Fig. 5.5 The sum-squared error for the second output Y2

The output y_2, however did not achieve the error specification of 0.2. After 537 epochs it reached the value of 0.285 as shown in Fig. 5.5. Since it is sufficiently close to the desired error specification. we will use this result for characterizing the output y_2, to obtain

$$\hat{y}_2(k+1) = 0.768243y_2(k) + 0.075334y_2(k-1) + 0.059652u_2(k) + 0.042035u_2(k-1) + \\ + NN_{12}[y_1(k), y_1(k-1), u_1(k), u_1(k-1)] \quad (5.5)$$

Note that in equations (5.4) and (5.5). NN_{21} and NN_{12} are the corresponding neural networks that are trained to identify the nonlinear dynamics of the process. The experimental results obtained above are satisfactory as the parameters are reasonably similar to the actual experimental data. Furthermore, note that the linear model estimates for y_1 and y_2 are quite similar to each other based on our estimation technique.

5.3 The experimental results for a neural PID controller

In the previous chapter we have discussed the structure of our proposed PID neural network controller (refer to section 4.1.2). For sake of convenience the expressions are provided again below:

$$u(k) = u(k-1) + K\sum_{i=1}^{3} w'_i(k)x_i(k)$$

$$w'_i(k) = \frac{w_i(k)}{\sum_{i=1}^{3}|w_i(k)|}$$

$$w_1(k+1) = w_1(k) + \eta_I z(k)u(k)x_1(k)$$

$$w_2(k+1) = w_2(k) + \eta_P z(k)u(k)x_2(k)$$

$$w_3(k+1) = w_3(k) + \eta_D z(k)u(k)x_3(k)$$

where

$$x_1(k) = e(k);$$

$$x_2(k) = \Delta e(k);$$

$$x_3(k) = \Delta^2 e(k) = e(k) - 2e(k-1) + e(k-2)$$

and η_I, η_P, η_D are learning rates associated with the integral, proportion and derivative terms.

For the experimental results the training rates are selected as η_I = 180, η_P = 5000, η_D = 22, with K = 0.05. The closed-loop results for both a conventional PID controller as well as our proposed neural network PID controller are shown in Figures 5.6 and 5.7.

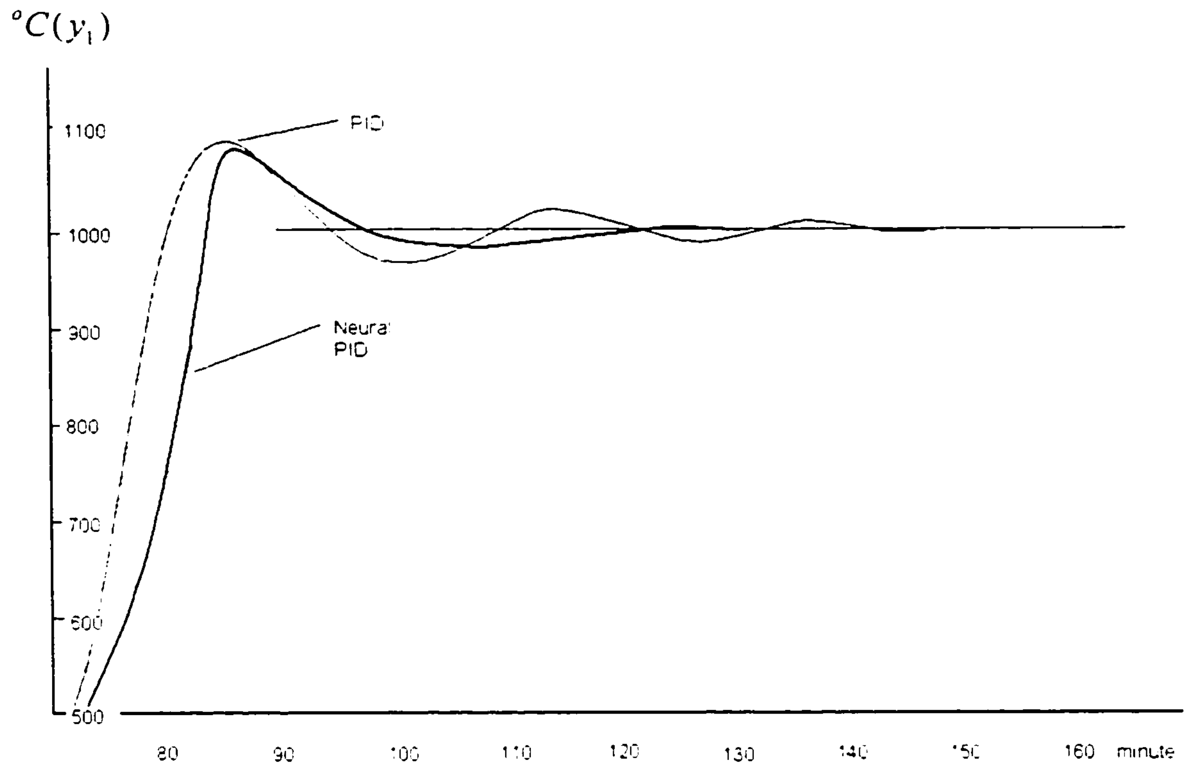


Fig. 5.6 Comparison between PID and neural network-based PID control strategies (y_1)

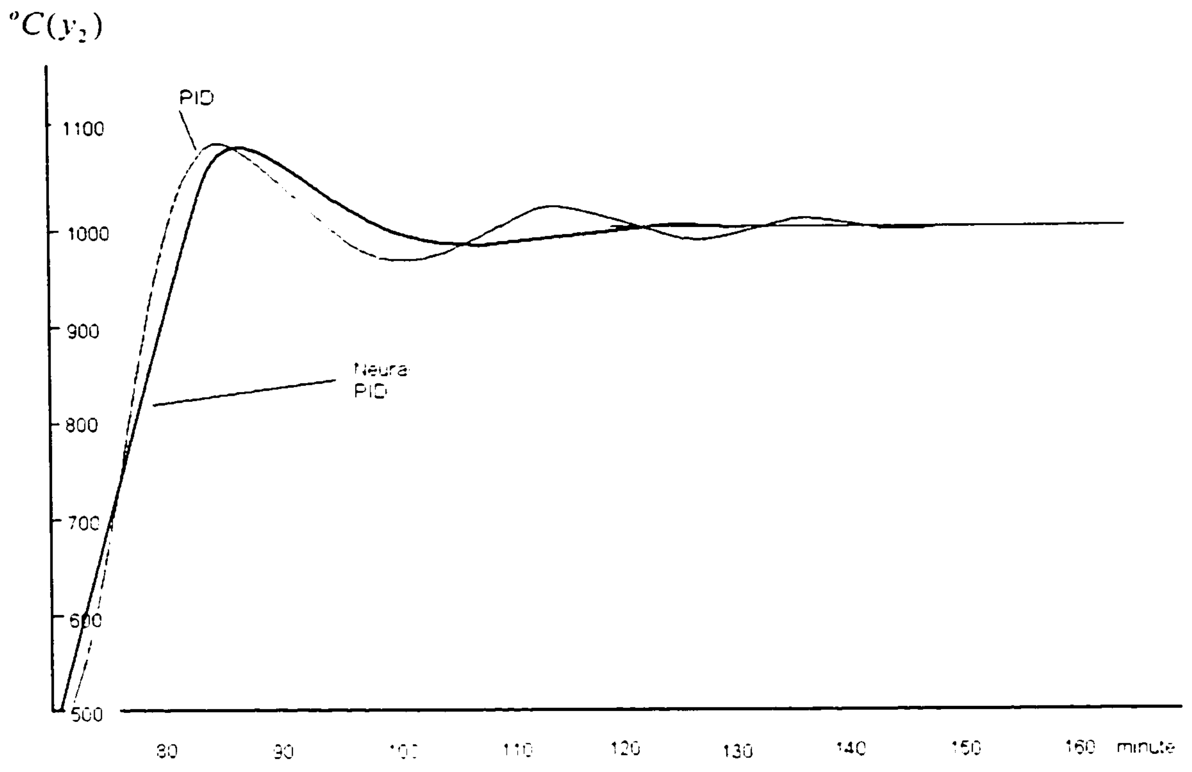


Fig. 5.7 Comparison between PID and neural network-based PID control strategies (y_2)

From the above figures we can conclude that the neural network-based PID controller has better transient as well as steady state performance behaviour compared to the traditional PID controller when applied to the electrothermal furnace system.

Below we now summarize our findings on the performance of the neural network-based PID controller:

Remark 1: The K parameter has to be selected carefully. If K is too large, it may cause instability in the system and if K is too small it will slow down the transient response of the system.

Remark 2: If the output behaves with a large overshoot and appears oscillating excessively, the value of K should be decreased while keeping the values of η_I, η_P, η_D unchanged.

Remark 3: By adjusting the value of η_I it will be also possible to modify the transient response of the system. When the transient response of the system is too fast and the overshoot is too big, the value of η_I should be decreased.

Remark 4: Initially η_D is selected to be small, while the values of K, η_P, η_I have to be adjusted in order to achieve suitable response. After that we can keep the values of K, η_P, η_I and slowly increase η_D for the best result.

Remark 5: By selecting different values for η_D, η_P, η_I, the performance of electrothermal furnace system can be varied as shown in Fig. 5.8.

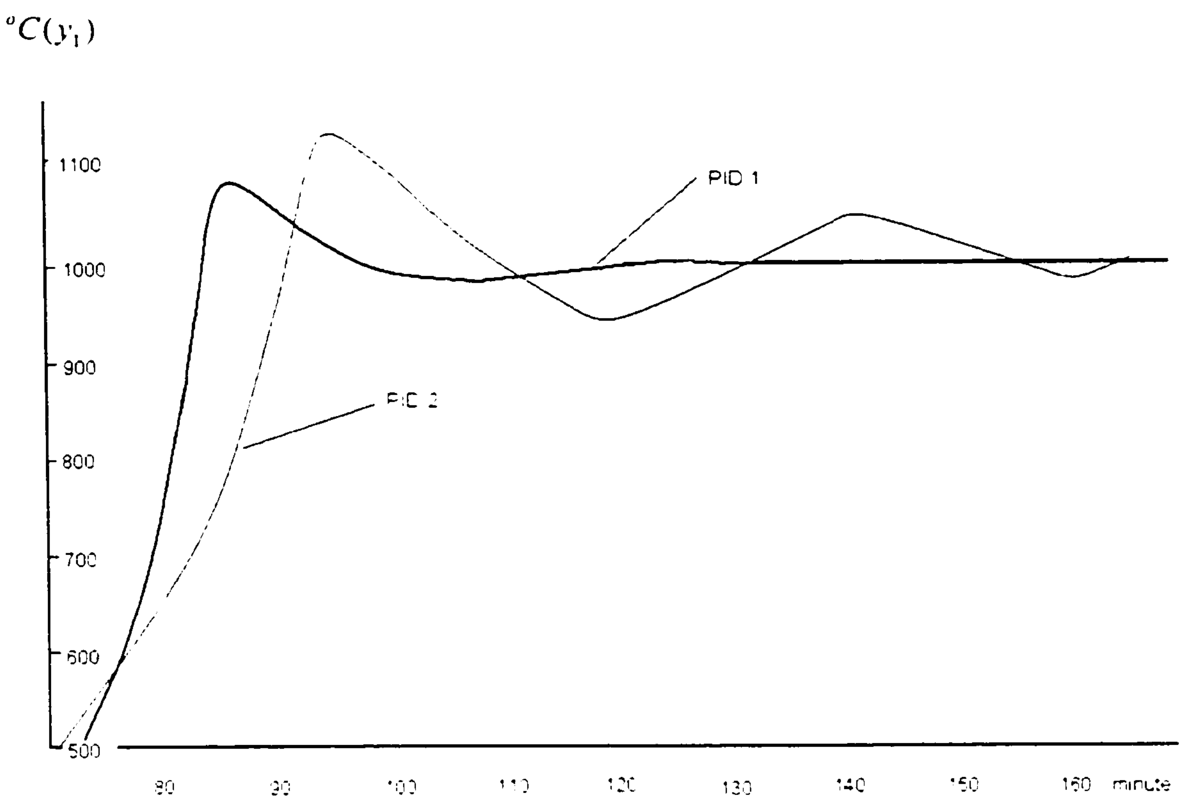


Fig. 5.8 Comparison between different values of η_P, η_I, η_D and identical η_P, η_I, η_D values for the neural network-based PID control strategies (Legend: PID 1 has different η_P, η_I, η_D values whereas PID 2 has the same η_P, η_I, η_D values)

5.4 Nonlinear internal model control

The IMC structure was discussed briefly in previous chapters. As stated earlier the IMC scheme has been shown to encompass a number of different control design techniques developed in the literature. One of the main motivations for employing IMC control strategy is due to its number of desirable properties that were outlined in previous chapters. The proposed IMC architecture and design procedure are presented in the next section.

5.4.1 Neural network internal model control

Based on the mathematical expressions (5.1) and (3.1), the neural network internal model control structure is now constructed below. The proposed configuration is shown in Fig. 5.9.

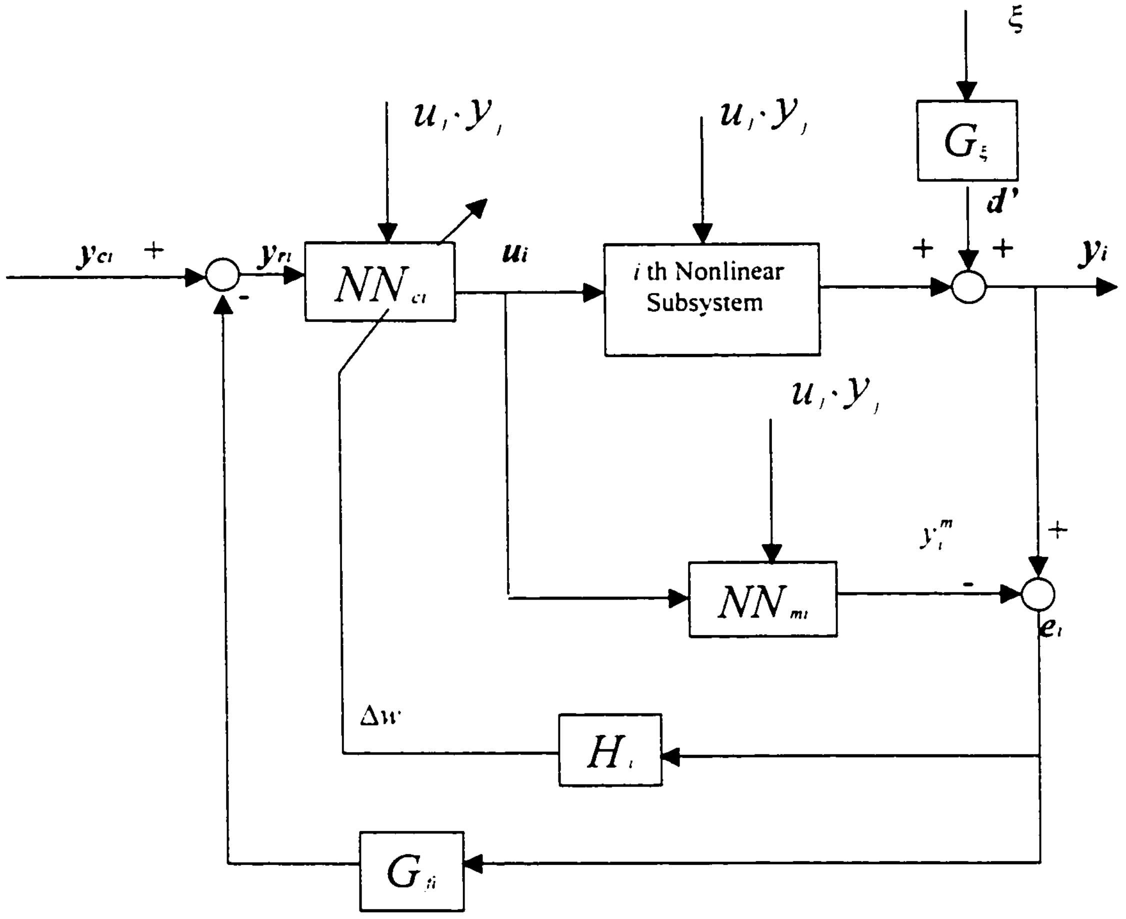


Fig. 5.9 Neural Network IMC Structure

The nonlinear plant to be considered is assumed to be described by

$$y_i(k+1) = f_1(y_i) + f_2(u_i) + f_3(y_j) + f_4(u_j)$$

where for our MIMO electrothermal furnace, we have

$$y_{t1}(k+1) = \alpha_{11}y(k) + \alpha_{12}y(k-1) + \beta_{11}u(k) + \beta_{12}u(k-1) + \gamma_{11}\sin\{\zeta_{11}[y_2(k) +$$

$$+ y_2(k-1)]\} + \gamma_{12}\cos\{\zeta_{12}[u_2(k) + u_2(k-1)]\}$$

and

$$y_{t2}(k+1) = \alpha_{21}y(k) + \alpha_{22}y(k-1) + \beta_{21}u(k) + \beta_{22}u(k-1) + \gamma_{21}\sin\{\zeta_{21}[y_1(k) +$$

$$+ y_1(k-1)]\} + \gamma_{22}\cos\{\zeta_{22}[u_1(k) + u_1(k-1)]\}$$

Based on the identification results for the α and β parameters of the identified model of the plant as in Section 5.2, we obtained for NN_{mi}

$$y_{t1}^{m}(k+1) = 0.770142y_1(k) + 0.069923y_1(k-1) + 0.052985u_1(k) + 0.044987u_1(k-1) +$$

$$+ NN_{21}[y_2(k), y_2(k-1), u_2(k), u_2(k-1)]$$

and

$$y_{t2}^{m}(k+1) = 0.768243y_2(k) + 0.075334y_2(k-1) + 0.059652u_2(k) + 0.042035u_2(k-1) +$$

$$+ NN_{12}[y_1(k), y_1(k-1), u_1(k), u_1(k-1)]$$

The control input to the plant is now given by the neural networks architecture shown in Fig. 5.3 such that

$$u_i(k) = NN_{ci}(y_{ri}, u_j, y_j)$$

where $$y_{ri} = y_{ci} - G_{fi}(e_i)$$

with y_{ci} denoting the desired setting value. The $G_{ri}(e_i)$ is chosen as $G_{ri}(e_i) = 0.9e_i$.

The H_i is chosen as H_i=1.

5.4.2 Performance results for the IMC control applied to the electrohermal furnace system

The proposed neural network internal model control architecture was presented in the previous section. The following specific information is now used for the neural network-based IMC controller. The NN_{ci} neural controller's architecture is identical to that shown in Fig. 5.3. The Gaussion activation function is used for the NN_{ci} neural controllers.

The simulation results are shown in Fig. 5.10. The accuracy is +/- 2 ^{o}C for an operating point of 750 ^{o}C. In steady state the temperature differential between the samples is less than 2 ^{o}C which is better than standard system design requirements. The transient overshoot is about 3.4%. Also, note that both outputs y_1 and y_2 behave very close to each other.

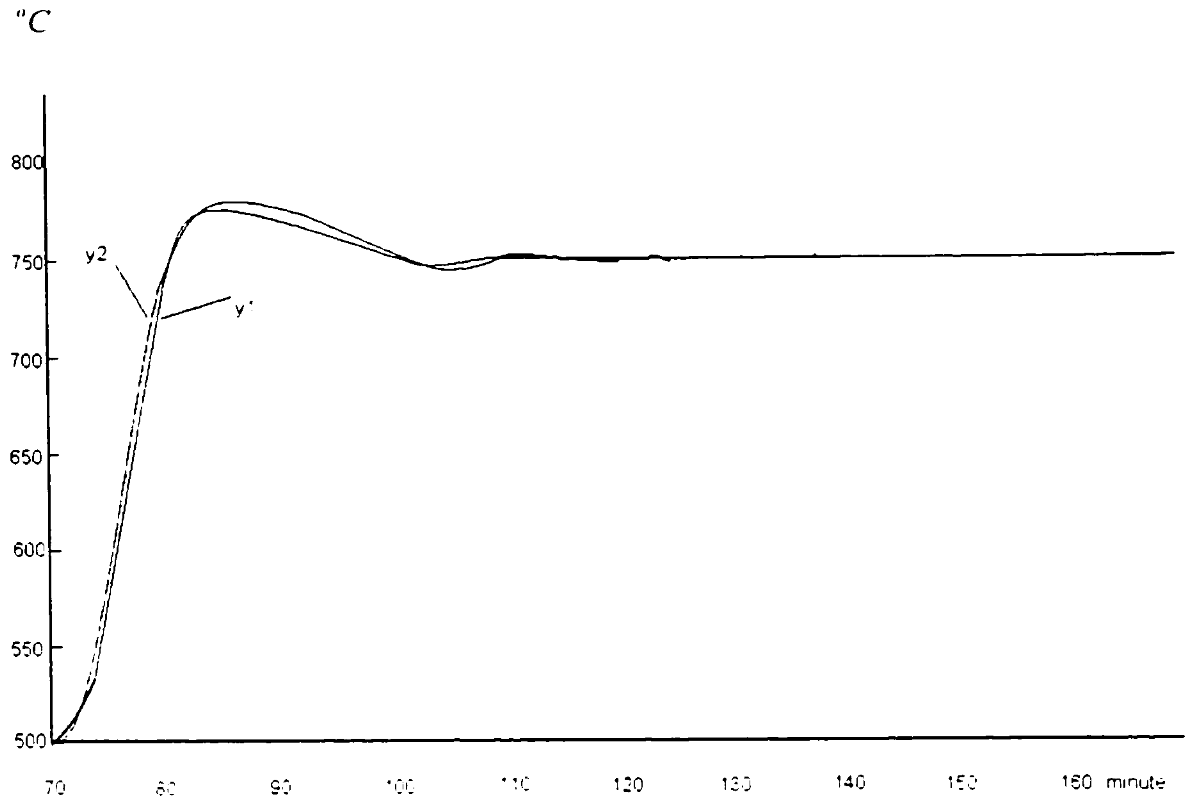


Fig. 5.10 Simulation results of the neural network-based IMC control on electrothermal furnace system

5.5 Conclusions

Both experimental and simulation results are presented in this Chapter to demonstrate that feedforward backpropagation networks may be designed and implemented for a nonlinear electrothermal furnace system. The system model is first identified by two three-layer backpropagation neural networks. The experimental and simulation results also show that the neural network-based PID controller as well as neural network-based IMC controller offer stable response and satisfactory output regulation of the electrothermal furnace system. Finally, the neural network-based controllers offer fast dynamic response and can potentially improve the control performance and accuracy of the electrothermal furnace system over a conventional PID controller.

Chapter 6

CONCLUSION AND FURTHER WORK

The objectives of this thesis are to demonstrate an application of a neural network-based strategy for identification and control of an electrothermal furnace temperature control system – which is a highly nonlinear system and is widely used in material testing area – and to illustrate the feasibility and superiority of the proposed scheme as compared to a conventional PID and internal model control techniques.

6.1 Contributions

The thesis demonstrates that neural networks can be used effectively in identifying and controlling an electrothermal furnace system. The methodology in the thesis is based on the electrothermal furnace model identified by using a backpropagation based network combined with both neural network-based PID and internal model based control strategies. Both static and dynamic back-propagation methods are discussed.

The main contributions of the thesis are as follows:

- Three-layer neural networks are developed for the identification of the electrothermal furnace system.
- An experimental electrothermal furnace computer control system is considered.
- A neural network-based PID control strategy is proposed and compared with a convention PID controller experimentally for the electrothermal furnace system.
- A neural network-based IMC control strategy is developed and used for the electrothermal furnace system simulation.

6.2 Conclusions

An application of a neural network-based strategy to temperature control system is presented in this thesis. The electrothermal furnace is a popular instrument widely used in material testing area.

Backpropagation neural networks are trained to identify and control the electrothermal furnace system. The experimental results are quite satisfactory. There is every reason to believe that the same methods can also be used successfully for the identification and control of other multivariable nonlinear system.

Two types of neural network-based controllers (PID and IMC) are proposed. Both of these architectures are used in experimentation and simulation for the electrothermal furnace system.

It is quite feasible to implement the neural network adjustment laws to adapt the PID parameters online. We have also studied invertibility conditions of a class of nonlinear dynamical systems and derived invertibility specification for these systems. We also proposed architectures and algorithms for training networks to represent both feedforward nonlinear dynamic maps and the inverses of these maps.

The outline of the thesis is as follows:

In Chapter 1, a general introduction about biological neuron, neural networks, ANN history and motivation, and outline of the thesis are presented. In Chapter 2, a standard feedforward neural network topology and its principles are reviewed. Attributes such as activation functions, supervised learning, backpropagation network and backpropagation training are discussed. In Chapter 3, the physical electrothermal furnace system, its computer control system and a neural network model for the electrothermal furnace are introduced. In Chapter 4, a neural network-based PID control system and a neural network-based nonlinear internal model control system (IMC) are introduced. The details regarding network learning algorithms, internal model principle and nonlinear internal model control are presented. A novel technique for directly using a neural network in the internal model control of a nonlinear system is proposed. The ability of neural networks to model nonlinear functions and their inverses is exploited. In Chapter 5, a backpropagation identification neural network, PID neural network control and neural network nonlinear internal model control schemes for the electrothermal furnace system are developed and results are presented. In Chapter 6, future work and conclusions are discussed.

6.3 Suggestions for Future Work

Future work could focus on the extension of the proposed approach to other multivariable nonlinear systems. analysis of the robustness of the controller and stability analysis for the closed-loop control system.

References

[1] Simon S. Haykin. "Neural Networks: a comprehensive foundation". Macmillam College Publishing Company. New York 1994.

[2] Laurene Fausett. "Fundamentals of Neural Networks", Prentice-Hall. Inc. Englewood Cliffs. N.J., 1994.

[3] M. R. Tadayon. R.W. Lewis and K. Morgan. "An Implicit-explicit Finite Element Microcomputer Program for Transient Heat Conduction Analysis". Microcomputers in Engineering Applications. Edited By B. A. Schrefler. John Wiley & Sons Ltd. Great Britain. 1987.

[4] Madan M. Gupta. Dandina H. Rao. "Neuro-Control Systems". The Institute of Electrical and Electronics Engineers. Inc., New York. 1994.

[5] V. Rao Vemuri. "Artifical Neural Networks: Concepts and Control Applications". IEEE Computer Society Press. CA. 1992.

[6] K. S. Narendra and K. Parthasarathy. "Identification and Control of Dynamical Systems Using Neural Networks". IEEE Trans. Neural Networks. Vol. 1. No. 1. pp. 4-27, Mar. 1990.

[7] M. Saerens and A. Soquet. "Neural Controller Based on Back-propagation Algorithm". IEEE Proc.-F Vol. 138. No. 1, pp. 55-65. Feb. 1991.

[8] S. R. Chu. R. Shoureshi and M. Tenorio. "Neural Networks for System Identification". IEEE Control Systems, Vol. 10. No. 3. pp. 31-35. Apr. 1990.

[9] D.E. Rumelhart. G. E. Hinton and R. J. Williams. "Learning Representations by Back-Propagating Errors". Nature. Vol. 323. pp. 533-536. Oct. 9, 1986.

[10] E. M. Johansson. F. U. Dowla and D. M. Goodman. "Backpropagation Learning for Multilayer Feed-Forward Neural Networks Using the Conjugate Gradient Method". Int'l J. Neural Systems. Vol. 2. No.4, pp. 291-301, 1992.

[11] K. Hornick. M. stinchcombe and H. White, "Multilayer Feedforward Networks are Universal Approximators". Neural Networks. Vol. 2, No. 5. pp.395-366. 1998.

[12] S. Chen. S. A. Billings and P.M. Grant. "Non-linear System Identification using Neural Networks". Int. J. Control, pp. 1191-1214. 1990. 51.

[13] P. J. Werbos. "Backpropagation through Time: What It Does and How to Do It". Proceedings of the IEEE. October 1990.

[14] Richard P. Lippmann. "An Introduction to Computing with Neural Nets". IEEE ASSP Magazine. Vol. 4. No. 2. pp. 4-22. April 1987.

[15] C.E. Garcia and M. Morari. "Internal Model Control – An unifying review and some new results". Ind. Eng. Chem. Process Des. Dev.. pp. 308-323. 1982. 21.

[16] M. Morari and E. Zafiriou. "Robust Process Control". Prentice-Hall. 1989.

[17] C. G. Economou. M. Morari, and B. O. Palsson. "Internal Model Control. 5. extension to nonlinear systems". Ind. Eng. Chem. Process Des. Dev.. pp. 403-411. 1986. 25.

[18] Reza Nekovei. Ying Sun. "Back-Propagation Network and its Configuration for Blood Vessel Detection in Angiograms". IEEE Trans. on Neural Networks. Vol. 6. No. 1. January 1995.

[19] K. S. Narendra and K. Parthasarathy "Neural Network and dynamical systems. Part II: Identification". Center Syst. Sci., Dept. Electrical Eng.. Yale University. New Haven, CT. tech. Rep. 8902, Feb. 1989.

[20] Hiroshi Ninomiya. Naoki Kinoshita, "A new learning algorithm without explicit error back-propagation", International Joint Conference on Neural Networks Proceedings. Vol. 3, pp. 1389-1392, 1999.

[21] Jun Qing Chen. Jing Ping Jiang, "New methods to train a back-propagation network and their application". International Joint Conference on Neural Networks Proceedings. Vol. 3. pp. 1702-1705. 1999.

[22] N. Bhat and T. J. Mcavoy. "Use of Neural Nets for Dynamical Modelling and Control of Chemical Process Systems". Comput. Chem. Eng.. 14. (4-5). pp. 573-583. 1990.

[23] Bogdan M. Wilamowski. Yixin Chen. Aleksander Malinowski. "Efficient algorithm for training neural networks with one hidden layer" International Joint Conference on Neural Networks Proceedings. Vol. 3. pp. 1725-1728. 1999.

[24] D. Psaltis. A. Sideris and A. A. Yamamura. " A Multilayered Neural Network Controller". IEEE Control System Mag., pp. 17-21. 1988. 8.

[25] Joey Rogers. "Object-Oriented Neural Networks in C++". Academic Press. Inc. San Diego. CA 92101-4495. USA, 1997.

[26] V. K. Sood. N. Kandil. R. V. Patel. K. Khorasani. "Comparative Evaluation of Neural Network Based and PI Current Controllers for HVDC Transmission". PESC Conf. REC.. pp. 553 ~ 560. 1992.

[27] K. J. Hunt and D. Sbarbaro, " Adaptive Filtering and Neural Networks for Realisation of Internal Model Control" Ind. Eng. Chem. Process Des. Dev., pp. 403-411, 1996, 21.

[28] K. S. Narendra and K. Parthasarathy, "Back Propagation in Dynamical Systems Containing Neural Networks", Center Syst. Sci., Dept. Electrical Eng., Yale University, New Haven, CT, tech. Rep. 8905, Mar. 1989.

[29] James A. Freeman, "Simulating Neural Networks", Addison-Wesley Publishing Company, Inc. 1994, New York.

[30] M. Norgaard, O. Ravn, N. K. Poulsen and L. K. Hansen, "Neural Networks for Modelling and Control of Dynamic Systems", Springer - Verlag London Limited, Great Britain, 2000.

[31] Mark Girolami, "Self- Organising Neural Networks – Independent Component Analysis and Blind Source Separation", Springer - Verlag London Limited, Great Britain, 1999.

[32] Tomas Hrycej, " Neurocontrol: towards an industrial control methodology", John Wiley &Sons, Inc. New York, 1997.

[33] Lakhmi C. Jain, Beatrice Lazzerini and Ugur Halici, "Innovations in ART Neural Networks", Physica -Verlag, A Springer - Verlag Company, 2000.

[34] Hongxing Li, C. L. Philp Chen, Han-Pang Huang, "Fuzzy Neural Intelligent Systems – Mathematical Foundation and the Applications in Engineering", CRC Press LLC, Florida, 2001.

[35] R. A. Al-Ashoor and K. Khorasani, " A Decentralized Indirect Adaptive Control for a Class of Two-Time Scale Nonlinear Systems with Application to Flexible-Joint

Manipulators". IEEE Transactions on Industrial Electronics, Vol. 46, No. 5, pp. 1019- 1029, October 1999.

[36] K. J. Hunt and D. Sbarbaro, "Neural Networks for Nonlinear Internal Model Control", IEE Proceedings-D, vol. 138, 231-238, Sept. 1992.

[37] Ian Cloete and Jacek M. Zurada, "Knowledge - Based Neurocomputing". The MIT Press, Cambridge, Massachusetts, 2000.

[38] K. S. Narendra, Y. H. Lin and L. S. Valavani, "Stable Adaptive Controller Design" IEEE Trans. Automat. Contr., vol. 25, pp. 440 – 448, June 1980.

[39] Terrence L. Fine, "Feedforward Neural Network Methodology", Springer - Verlag New York, Inc. USA, 1999.

[40] K. Khorasani, "Adaptive Control of Flexible-Joint Robots", IEEE Transactions on Robotics and Automation, Vol. 8, No. 3, pp. 250-267, April 1992.

[41] D. G. Sbarbaro and P. J. Gawthrop, "Self-organization and Adaptation in gaussian networks", 9th IFAC/IFORS symposium on identification and system parameter estimation, Budapest, Hungary, 1991.

[42] Terrence L. Fine, "Feedforward Neural Network Methodology", Springer - Verlag New York, Inc. USA, 1999.

[43] A.A. Georgiev, "Fitting of Multivariate Functions", Proc. IEEE, pp.970-971, 1987, 75.

[44] F. Girosi and T. Poggio, "Networks and the Best Approximation Property", Biol. Cybernetics, pp. 169-176, 1990, 63.

[45] D. C. Sbarbaro and P. J. Gawthrop. "Self-organization and Adaptation in Gaussian Networks". 9th IFAC/IFORS symposium on identification and system parameter estimation. Budapest. Hungary. 1991.

[46] C. A. Desoer and M. Vidyasagar. "Feedback Systems: Input-Output Properties". Academic Press. London. 1975.

[47] B. Widrow. "Adaptive Inverse Control". Preprints 2nd IFAC workshop on Adaptive Systems in Control and Signal Processing. Lund. Sweden. pp. 1-5.1986.