Cranfield
UNIVERSITY

Modelling Strains in Helicopters Using Artificial Neural Networks

Mustafa KARAKÖY

School of Industrial and Manufacturing Science

MPhil Thesis

1996

Cranfield UNIVERSITY

School of Industrial and Manufacturing Science

MPhil Thesis

1994-1996

Mustafa KARAKÖY

Modelling Strains in Helicopters Using Artificial Neural Networks

Supervisor: Dr. Alfred VELLA

This dissertation is submitted in partial fulfilment of the requirements for the degree of *Master of Philosophy in Computer Science*

Modelling Strains in Helicopters Using Artificial Neural Networks

Mustafa KARAKÖY

May 21, 1996

Abstract

Strains occurring in moving machinery are difficult and costly to measure whilst the machinery is in normal use. However, occasions arise when the effects of strains need to be determined so that preventative or corrective maintenance may be undertaken to avoid catastrophic failure.

This work describes the use of Artificial Neural Networks in the prediction of the strains occurring in moving parts of helicopters during flight. Previous attempts, by other authors, have shown the task to be achievable for some types of strain but very difficult for others. It is one of the difficult strains that is tackled here.

Data collected by the CAA, MOD and a helicopter manufacturer were made available to the Helicopter Damage Tolerance Group at Cranfield. This group allowed us to use the data for the work that is reported here. The strain data consisted of 67217 points each accosted with 21 more easily measured parameters which are used as inputs to the network. Exploratory work used the first 300 data points from this set and later work used 10% of the data for training, and the whole set for validation.

By employing more hidden neurons than previous attempts, as well as another hidden layer, this study has succeeded in raising the correlation between the actual and predicted signals from below 50% to 88% on the training set and 73% on the validation set. The thesis describes the methods used to achieve these results and the experiments undertaken to find suitable values of the parameters needed for the Artificial Neural Network.

Further development of the work may result in the possibility of accurate condition monitoring of helicopter parts, which may in turn improve both the safety and the economics of the helicopter as a mode of transport.

Acknowledgements

First and foremost, I thank God, Who gave me the strength and opportunity to complete this study successfully.

A special acknowledgement goes to my supervisor, Dr. Alfred Vella. He gave me the necessary inspiration, and his help made a great contribution in preparation of the thesis.

My deep gratitude also goes to the reviewers for their critical and constructive suggestions. They are Humam Maqsud, Carole Peachy-May, Bob Hudson, Carol Vella and Nick Benwell.

I am in debt to Dr. Orhan Çakır for his encouragement and support whenever needed during the time I spent at Cranfield.

Finally, let me thank my live networks, my wife Kamuran, and daughters Nurefşan and Neslihan for their quiet patience through the many lonely hours during the study and preparation of the thesis.

Contents

3 Modelling the System

4 Methodology

5 Discussion of Experimental Results

6 Conclusions

Appendices

A Eperiments

B Flight Parameter Data

Chapter 1

INTRODUCTION

1.1 Problem Definition

In a helicopter, the main and tail rotors, the engine and transmission, a number of other rotating components, and the airframe itself cause vibrations (*Lison 1994*). In general, helicopters are subjected to much more vibratory loading conditions than fixed-wing aircrafts. Consequently, helicopter airframes are subject to fatigue damage occurring primarily at or near threshold crack growth (*Crawford et al 1991, Buller 1994*). Hence, before a helicopter leaves the plant or takes off, its vibrations should satisfy the required specifications (*Taitel et al 1992*); otherwise, excessive vibration may lead to unexpected structural and component damage (*Lison 1994*).

To avoid technical problems and incidents, a number of flights are performed to ensure that the helicopter is tuned well, resulting in increased production and maintenance cost (*Taitel et al 1992*). Preventive maintenance, particularly, is so important not only to avoid undesirable events and states, which cause high emergency maintenance cost (*Matteson 1987*), but also to safeguard the value of investment, operate within law, inspire confidence in flight crew and customers, and provide safe, reliable and airworthy aircraft (*Bateman 1982*).

For a particular component or system, primary maintenance can be defined as one of the following (*Wendlandt 1990, Bateman 1982*):

Hard Time : As a preventive maintenance, it periodically overhauls the component or the system or removes it after certain period of usage based on operating experience or tests as appropriate.

On Condition : As a preventive maintenance, it periodically inspects the component or the system against some appropriate physical limit to determine whether it can continue in service. Again, these limits can be adjusted based on operating experience or tests as appropriate.

<u>Condition Monitoring</u> : This is a maintenance process for items which have neither *Hard Time* nor *On Condition* maintenance. It is accomplished by means of data collection and analysis by which an operator obtains information from the whole population of a system, or item in service.

Beside many benefits of maintenance mentioned above, the disadvantages are almost entirely concerned with the cost of the activity (*Bateman 1982*). Because of the fact that the finance available is always limited, cost acts as a major constraint in balancing the objectives.

1.2 Helicopter Health and Usage Monitoring

Helicopter health and usage monitoring is an attempt to improve reliability and maintainability, and to reduce operating costs. In simple terms, it seeks to identify changes or trends in monitored areas to provide advanced warning of component degradation towards failure, allowing the component to be changed before the failure occurs (*Hill 1994*). Leaving wear to become noticeable is not a viable choice as the helicopter may become dangerous before the damage becomes apparent (*Vella et al 1996*).

In the military field, usage monitoring is defined as below (*Ministry of Defence Procurement Executive 1989*):

"*Usage monitoring is a process which assesses the life consumption of life-limited components, systems and structures by monitoring actual damage exposure (e.g. due to combinations of loads, speeds, temperatures, etc.). Allied to this is recording of information relating to exceedances of placarded limits, whether in emergency conditions or normal operations. The activity relates primarily to components substantiated on a* ***safe-life*** *basis.*"

The civilian equivalent of the definition as provided by the Helicopter Health Monitoring Advisory Group (HHMAG) is succinct but essentially the same (*Civil Aviation Authority 1984*) :

"A process which assesses the life consumption of critical components, systems, and structures by monitoring actual damage exposure."

The potential benefit of usage monitoring is shown in **Figure 1.1**:

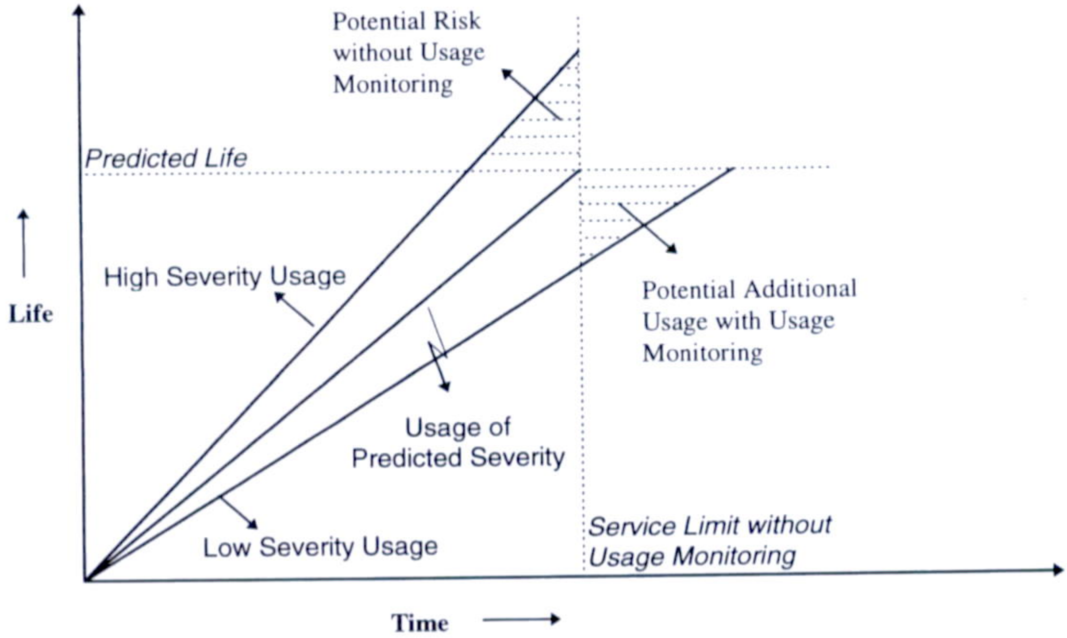


Figure 1.1

Potential Benefit of Usage Monitoring

Usage monitoring not only provides additional usage under conditions of lower severity usage than expected but prevents taking risk of further usage when higher severity usage occurs.

Usage monitoring is distinct from but complementary to health monitoring. (*Western and Boulding 1990*). However, both are usually used together in the form of *health and usage monitoring*. So, we too refer to same thing by use of either term in this thesis.

A number of reports spurred developments in helicopter health and usage monitoring (*Civil Aviation Authority 1984, Royal Aeronautical Society 1984, and Royal Aeronautical Society 1990*). In 1982, upon the implementation of recent advances in design philosophies, manufacturing techniques and new materials for aircrafts, the Chairman of the Civil Aviation Authority (CAA) suggested that the Airworthiness Requirement Board (ARB) review helicopter certification standards. The Helicopter Airworthiness Review Panel (HARP) was set up by ARB in response to this. It announced some recommendations published as a report, which is known as the HARP report (*Civil Aviation Authority 1984*). The report contained 15 wide-ranging recommendations, two of which were directly related to usage monitoring.

One of the recommendations sought closer co-operation between the manufacturer and operator to ensure that fatigue test programmes reflect realistic usage and that revised modes of usage remains within the boundaries of substantiated frequency and severity of loading. The other, in conjunction with this, sought to establish research programmes to obtain in-flight data recording of actual severity and frequency of loads to which a helicopter is exposed.

In the HARP report, some statistical data regarding the helicopters registered in the UK also showed that the rate of accidents per flight was 2.5 times that of fixed-wing ones. Furthermore, 35% of the accidents were caused by technical problems mostly concerning the rotors and transmission area.

The failure of the components in those sections is most often due to fatigue. One of the most important tasks in the design of fatigue loaded component, in other words, calculating or anticipating damage on the component, involves developing load spectra which represent the actual strains experienced by the component during flight (*Turban 1992*). The wear particularly on rotating parts of a helicopter is difficult to estimate without directly measuring the strains that they are subjected to. Therefore, the prediction of strains is still a challenge for researchers in this field who seek not only reliability but also cost-effectiveness. Although some estimation is possible using mathematical or simulation models of structure involved, this does not prove to be of sufficient accuracy and other methods need to be developed (*Vella et al 1996*).

Helicopter usage monitoring can range from recording flight times (e.g. recording flying hours and engine running hours) to complex systems which monitor the structural usage or fatigue consumption of components. Using flying time is similar to the idea of changing a motor car's oil after say 6000 miles of use of the car. But this method suffers from the disadvantage of ignoring actual flight conditions (*Vella et al 1996*).

Flight Condition Recognition (FCR), a better method, involves the continuous monitoring of flight parameters so as to form a flight spectrum, which is then used in damage calculations instead of an assumed spectrum. Again, FCR systems can only recognise predetermined conditions and new manoeuvres will not be recognised (*Hill 1994*).

Another strategy is Direct Load Monitoring (DLM), which directly measures stresses by means of attaching strain gauge instruments during the flight. At first glance, this may look straight forward. But, using this method for rotating components is more problematic. Nor is the data reliable as the telemetry itself is prone to damaging influences (*Vella et al 1996*). One more drawback of this method is high cost of such instrumentation.

There are other techniques which try to relate rotating system loads to stationary system parameters such as Regression Load Analysis, Load Spectra Pattern Recognition, Holometric Load Analysis and Continuous Parametric Techniques (*Hill

1994, Leahy 1994). In this research we use Artificial Neural Networks as an alternative method for the prediction of helicopter strains.

1.3 Prediction of Strains Using ANNs

Many problems, such as ours, do not lend themselves to easy mathematical solution. Amongst these there are classes which have easily generated, but extremely large, search spaces in which good solutions can be found given sufficient search time. A number of efficient search techniques, including Artificial Neural Networks (ANNs), have been developed for this type of problem over recent years.

The aim of this work is to find ANN models of the strain in a rotating part of the helicopter in terms of the measurement made on stationary parts. In other words, the system developed will be used to predict strain values given the values of other measured parameters. It is expected that this can be achieved by first *teaching* or *training* the network using some of the data provided by Ministry of Defence (MOD). Some data will be retained for testing of the trained system. Once the network has been trained we may then use it to predict strain without the use of strain gauges. Hence, we will be able to contribute to health and usage monitoring.

1.4 Structure of the Thesis

In this chapter, we have introduced the problem and the method, namely ANNs used to find candidate solutions to the problem at hand. In chapter two, we introduce the basic concepts of ANNs. The ways in which ANNs can be used to model helicopter strain data are discussed in chapter 3. In chapter 4, we discuss the methodology used in this work. The results of the experiments are given in chapter 5. The thesis ends with a discussion of the results and a note of the potential for further work in chapter 6.

Chapter 2

ARTIFICIAL NEURAL NETWORKS

2.1 Introduction

Artificial Neural Networks (ANNs) are considered as one of the major tools for solving problems in the field of *Artificial Intelligence* (AI). Their origins go way back to the 1940s when neurophysiologist Warren McCulloch and logician Walter Pitts introduced the first mathematical model of a biological neuron.

The learning behaviour of artificial neural networks was first treated extensively by Frank Rosenblatt in early 1960s. However, a growing interest towards ANNs emerged especially after 1980. Workers such as Rumelhart, McClelland, Widrow, Kohonen and Grossberg made great contributions to the development in ANNs by suggesting their own versions. Hence, problems such as that of exclusive-or, which originally contributed to the demise of ANNs in late 1960s, have been overcome using new learning techniques (e.g. backpropagation) (*Picton 1994*). One of the most important publications during this period, which stimulated the resurgence of interest in ANNs was "Parallel Distributed Processing" written by Rumelhart and McClelland (*1986*).

Artificial/Neural Networks or *Parallel Distributed Processing* or *Connectionism* are synonymous terms (*Aleksander and Morton 1990*) and refer to computational models of the human brain (*Pham and Liu 1995*).

The most important feature of ANNs which makes such computational models very appealing for problems where there is little or incomplete understanding is their adaptive nature, where 'learning by examples' replaces traditional 'programming' in solving problems. Another key feature is parallelism which enables fast computations of solutions when implemented in customised hardware (*Hassoun 1995*). Along with some other attractive characteristics, these features make ANNs viable computational models for a wide variety of problems, including pattern classification, function approximation, clustering, forecasting, prediction etc.

The field of ANNs, therefore, attracts the interest of scientists from a number of diverse disciplines. Engineers are interested in applying them to a variety of problems such as pattern recognition and optimisation; biologists are concerned with issues related to our understanding of the human brain; mathematicians and computer scientists are intrigued by the computational capabilities of non-linear dynamic models of ANNs and are also interested in the uniqueness and convergence of heuristic training methods; psychologists are interested in models of learning and cognition and finally philosophers use ANN models to question the ability of machines to perform cognitive functions (*Mammone and Zeevi 1991*).

2.2 Artificial Neuron

The human brain is superior to a digital computer at many tasks such as processing of visual information. Even a one-year-old baby is much better and faster at recognising objects, faces etc. than the most advanced AI systems (*Lisboa 1992*). Only in tasks based primarily on simple arithmetic does the computer outperform the brain!

The key features of the brain in succeeding at these complex tasks can be explained in these terms: It is robust and fault tolerant, flexible (i.e. adaptive), parallel, small and compact and able to deal with noisy and fuzzy information. It is these features which provide the motivation for studying neural computation (*Hertz et al 1991*).

ANNs are networks of interconnected elements called *processing elements* (PEs). These elements are also called as *neurons* or *artificial neurons* since in most cases they have been inspired by studies of biological nervous systems. (The exceptions are Boolean (i.e. weightless) networks such as WISARD (*Picton 1994*)). In fact, ANNs are not faithful models of biological neural or cognitive phenomena (*Hassoun 1995*). Nevertheless, they are certainly an attempt at creating machines working in a similar way to the human brain.

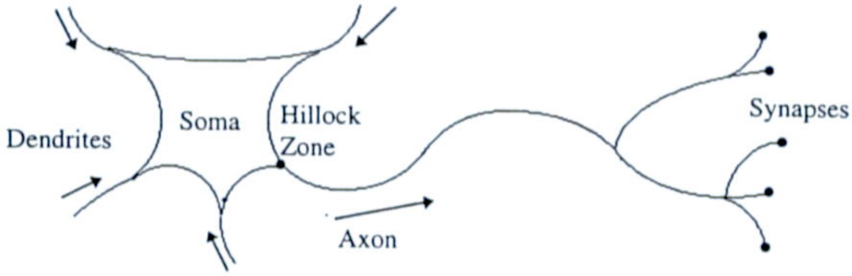


Figure 2.1
Simplified Model of an Organic Neuron

It is said that the human brain contains about 10^{11} neurons connected to each other (*Rumelhart and McClelland 1986*, *Aleksander and Morton 1990*). A simplified model of a biological neuron (see **Figure 2.1**) consists of a cell body or soma, a set of fibers entering the cell body (i.e. dendrites) and one special fiber leaving the soma, called the axon (*Veelenturf 1995*). It is believed that information flow amongst biological neurons takes place like this: A neuron fires or transmits an electrical signal to the other neurons if the electrical potential at the hillock zone is above a certain value. The transfer of electrical pulses from one neuron to others also depends on another factor namely 'synaptic transmission efficiency' at synapses, i.e., connection points of the neuron's axon and the succeeding neurons' dendrites.

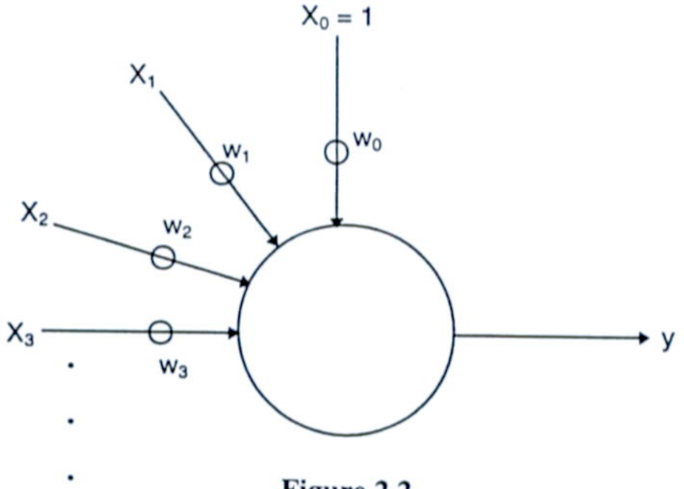


Figure 2.2
Artificial Model of a Neuron

The biological model can easily be simulated by an artificial neuron such as the one shown in **Figure 2.2**. It has n binary inputs which are 0 or 1 and a binary output. Each input signal is multiplied by a weight value, which corresponds to synaptic transmission efficiency in the biological model, so as to give the effect of the input onto the neuron. The constant input called bias, $x_0 = 1$, and the weight w_0 realise a threshold equal to - w_0 most often denoted by T.

The output of this model is specified by:

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i > -w_0 \\ 0 & \text{otherwise} \end{cases}$$

The artificial neuron is the basic processing unit of an artificial neural network. Neural networks composed of layers of such interconnected units (i.e. neurons) have been studied extensively by many researchers. At this point, it is necessary to understand the computational capabilities of these processing units.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 2.1

Truth Table of Function $y = x_1x_2 + x_3$

With a single unit, for example, we can realise a restricted[1] class of binary functions/predicate logic. Let us assume we have a binary function $y = x_1x_2 + x_3$. If we now consider the values of x_1, x_2, and x_3 as the inputs of a single PE and y as the output, we can select the weights w_0, w_1, w_2, and w_3 in such a way that the output behaviour of the PE yields the same results as the function the truth table of which is shown in **Table 2.1**. From the output definition of the artificial neuron we can investigate the following inequalities[2]:

1. $w_0 \leq 0$	for the argument (0, 0, 0)
2. $w_0 + w_3 > 0$	for the argument (0, 0, 1)
3. $w_0 + w_2 \leq 0$	for the argument (0, 1, 0)
4. $w_0 + w_2 + w_3 > 0$	for the argument (0, 1, 1)
5. $w_0 + w_1 \leq 0$	for the argument (1, 0, 0)
6. $w_0 + w_1 + w_3 > 0$	for the argument (1, 0, 1)
7. $w_0 + w_1 + w_2 > 0$	for the argument (1, 1, 0)
8. $w_0 + w_1 + w_2 + w_3 > 0$	for the argument (1, 1, 1)

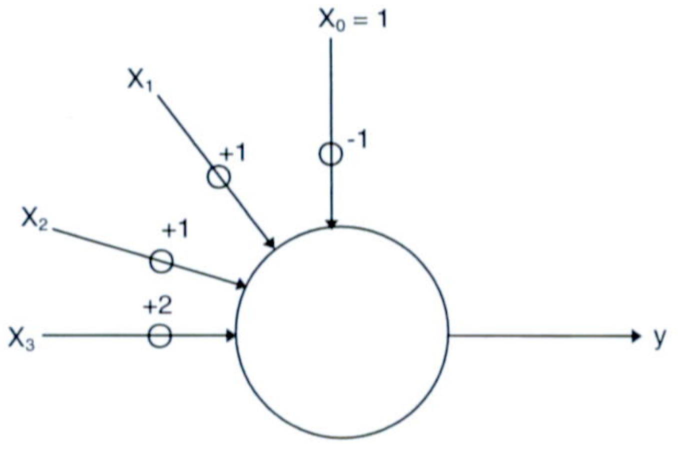


Figure 2.3

Prototype for the Function $y = x_1x_2 + x_3$

[1] The restrictions with one neuron will be examined in detail in the section 'The Perceptron'.
[2] In general case of n variables we have to investigate 2^n inequalities.

One of the weight vectors which satisfy these inequalities is (-1, 1, 1, 2). Thus, the prototype shown in **Figure 2.3** realises this function.

2.3 Activation Function

We can modify the 'artificial neuron' to obtain more functionality and flexibility by applying different function (called an *activation function*) to the weighted sum of the inputs[3] (*Williams 1986*). This function specifies how the neuron will scale its response to incoming signals and produces the neuron's final response (It will be seen later that there is a need to use a *continuous* activation function for some learning rules). The typical activation functions scaled in the range (0,1) are[4] :

Type of Functions	***Formulae***
Threshold-logic	$y = \begin{cases} 1 & net > 0 \\ 0 & net \leq 0 \end{cases}$
Threshold-linear	$y = \begin{cases} 1 & net \geq \dfrac{1}{2\mathrm{k}} \\ 0 & net \leq -\dfrac{1}{2\mathrm{k}} \\ \dfrac{1}{2} + \mathrm{k}net & \text{otherwise} \end{cases}$
Logistic (Sigmoid)	$y = 1/(1 + \exp(-\mathrm{k}net))$
Hyperbolic Tangent	$y = (1 - \exp(-2\mathrm{k}net))/(1 + \exp(2\mathrm{k}net))$
Radial Basis (Gaussian)	$y = \exp(-net^2/2\mathrm{k}^2)$

Table 2.2

Typical Activation Functions

[3] Recall that it was a simple, threshold-logic function so far.

[4] '*net*':total weighted sum; 'k':sharpness factor. See section 4.3.1 for shapes and flexible forms of functions.

2.4 Structure of ANNs

In terms of their structures, ANNs are classified into two groups:

- Feedforward networks
- Recurrent (Feedback) networks

2.4.1 Feedforward Networks

In these networks, the PEs are grouped into layers and the inputs flow from the input layer through to the output layer via unidirectional connections without feedback as illustrated in **Figure 2.4**.

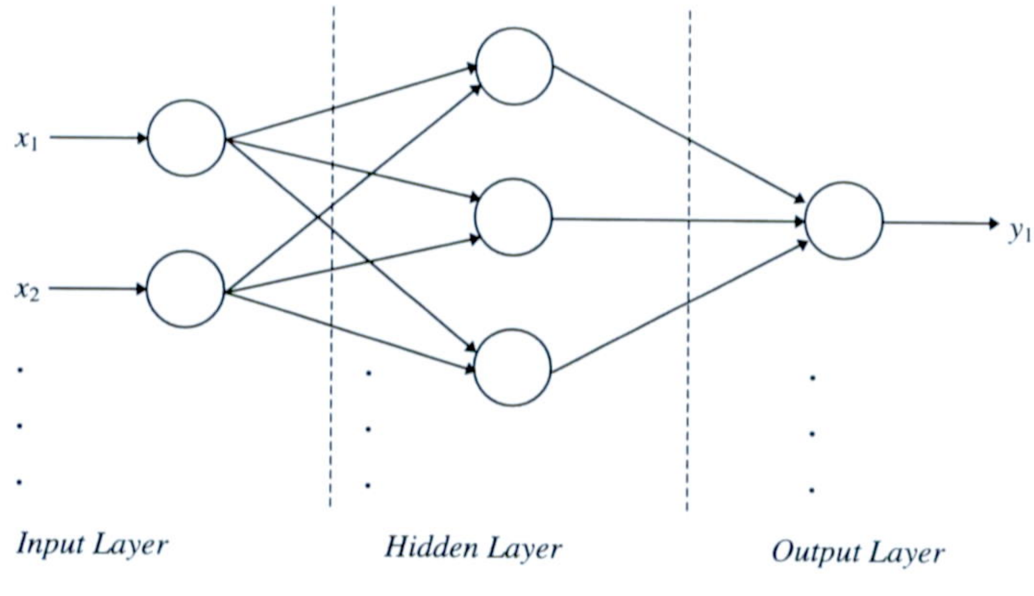


Figure 2.4
Feedforward Network Structure

Input layer nodes, unlike other 'neurons', have no functionality other than to serve as 'fan-outs' (i.e. buffers) passing data to each neuron in the next layer. The number of intermediate (hidden) layers may be more than one, or there may be no intermediate layer in the structure at all. However, in most cases hidden layers are present. A network with only input and output layers is generally called a single-layer network. Multi-layer Perceptron (MLP) and Learning Vector Quantization (LVQ) networks are the best known examples of this type.

2.4.2 Recurrent Networks

In a recurrent network, the outputs of some or all PEs are fedback to themselves and/or PEs in the preceding layers. In other words, signals can flow in both forward and backward directions. A simple example of such a network with fully connected PEs is shown in **Figure 2.5**.

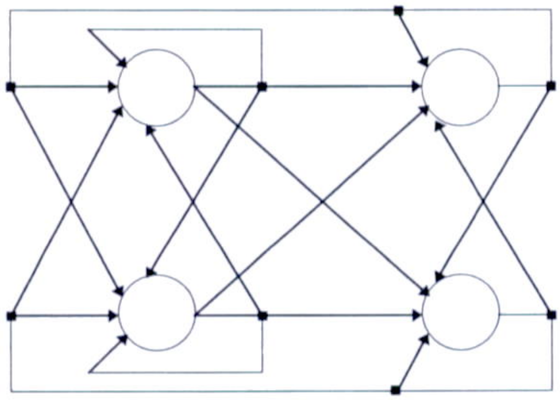


Figure 2.5

An Example of Recurrent Network Structure

In this example, there is no distinction between layers and units. Input to the network consists of clamping an initial state vector. Hopfield and Elman networks are examples of this type (*Pham and Liu 1995*).

2.5 Learning (Weight Updating) Methods

One of the most significant attributes of a neural network is its ability to learn by interacting with its environment or with an information source. Learning in a network is normally accomplished through an adaptive procedure, known as a *learning rule* or *learning algorithm* whereby the weights of the network are incrementally adjusted so as to improve a predefined performance measure over time.

In the context of artificial neural networks, the process of learning is best viewed as an optimisation process. More precisely, the learning process can be viewed as 'search' in a multidimensional parameter (i.e. weight) space for a solution, which gradually optimises a prespecified objective (criterion) function. This view allows us to unify a wide range of existing learning rules which otherwise would have looked more like diverse variety of learning procedures.

Learning rules are categorised as supervised learning, unsupervised learning, or reinforcement learning.

2.5.1 Supervised Learning

In supervised learning (also known as *learning with a teacher* or *associative learning*), each input pattern is associated with a specific desired target pattern. Usually, the weights are synthesised gradually, and at each step of the learning process they are updated so that the error between the network's output and a corresponding desired target is reduced. Examples of this method include the *delta rule*, the *generalised delta rule* or *backpropagation algorithm* and the *learning vector quantization* (LVQ) *algorithm*.

2.5.2 Unsupervised Learning

In unsupervised learning, there is no teacher signal. In other words, the desired outputs do not need to be known. Here, the objective is to 'categorise' or discover features or regularities in the training set. The weights and the outputs of the network are usually expected to converge to representations that capture the 'statistical' regularities of the input data. *Hebbian learning*, *competitive learning and self-organising feature-map learning* are examples of unsupervised learning.

2.5.3 Reinforcement Learning

Reinforcement learning is a process of 'trial and error' designed to maximise the expected value of a criterion function. It involves updating the weights in response to an 'evaluative' teacher signal. This differs from supervised learning where the teacher signal is the 'correct' answer. In its simplest form, reinforcement learning is based on the idea that if an action is followed by an 'improvement' in the state of affairs, then the tendency to produce that action is strengthened, i.e. reinforced. Otherwise, the tendency of the system to produce that action is weakened. An example of a reinforcement learning algorithm is the *genetic algorithm* (GA).

2.6 Example Neural Networks

There are many different types of ANNs with differences in architecture, learning strategies and weight updating methods. Each network has different strengths which make it suitable for a specific problem. Below, we introduce some paradigms with particular emphasis on the work undertaken in this study.

2.6.1 The Perceptron

The perceptron network, developed by F. Rosenblatt was the first neural network to emerge. Rosenblatt aimed to use this network mainly for 'pattern recognition' problems (*Rosenblatt 1958*). Because of this, it was called *Perceptron* - an aid to perception (*Veelenturf 1995*). The perceptron network is strictly feedforward[5] (see **Figure 2.4** for feedforward structure) and undergoes supervised learning.

The perceptron network established the concept of the artificial neuron, which is still used today (*Maren 1990a*). The number of 'neurons' is defined by the number

[5] There is no hidden layer in the perceptron model developed by F.Rosenblatt.

of outputs of the system modelled[6]. Each neuron in a perceptron computes a weighted sum of its inputs, and passes this sum into a non-linear thresholding function. This function is the threshold-logic activation function in Rosenblatt's model[7] (See output definition of threshold-logic function in **Table 2.2** for the perceptron's output).

In order to clarify the behaviour of a perceptron network, we will give an example: Let us assume we are to map one set of input vectors, denoted by *class A*, to 0, and another set, *class B*, to 1. For the sake of simplicity we will consider only 2 inputs; otherwise, it is difficult to illustrate patterns.

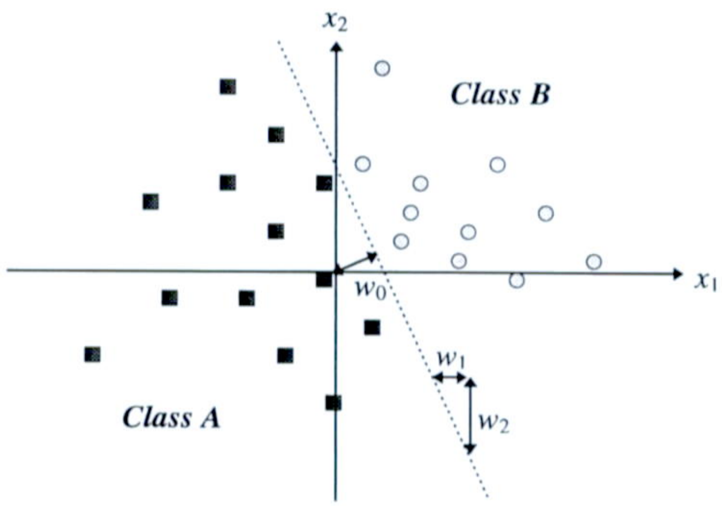


Figure 2.6

An Example of Linearly Separable Functions

The perceptron network (with a single neuron), can establish a line in the input space. The illustration in **Figure 2.6** shows the effects of weight values on the line. The slope of the line is defined by weights w_1 and w_2 whereas w_0 (bias) off-sets the line from the origin.

---

[6] Recall that input layer nodes do not behave like artificial neurons.

[7] It is also possible in the perceptron that activation function produces (1 , -1) instead of (1, 0).

If a straight line separates two classes as in the above example, the output function is called *linearly separable*. Clearly, not all problems have only two inputs. When the number of inputs, n, is more than two, the pattern space is n-dimensional and a n-1-dimensional hyperplane separates the two classes.

The training method for single-layer perceptrons is a variant of the delta rule, developed by Widrow and Hoff (*Stone 1986*). Training starts by assigning random initial values to all connection weights. Usually, these values are within some limits, such as -3 to +3. Then each pattern from the training set is presented to the network. The error, δ, is measured as the difference between the desired output, d, and the actual output from the non-linear output of the perceptron, y. By applying the learning rule to this δ, weights are modified. During modification, no one pattern is allowed to influence a given weight too much because the weights should take values enabling overall response to all of the patterns in the training set (*Maren 1990a*). So, the learning rule involves an equation for the i th weight as follows :

$$\delta = d - y$$

$$\Delta w_i = \alpha \delta x_i$$

$$w_{i(new)} = w_{i(old)} + \Delta w_i$$

The training set is needed to be shown to the network a number of times[8] since the relatively small changes are applied to the weights. The magnitude of the changes are adjusted by means of α, called the *learning rate*. During training, the weights are updated either after each pattern is presented to the network (as in the case of local learning) or after the whole set is presented to the network (as in the case of global learning)[9] (*Picton 1995*).

The perceptron convergence theorem states that whether in local learning or in global learning, the weight vector converges to a solution after some finite iterations

[8] Showing the whole training set once is called an *epoch*.

[9] In global learning the mean value of the errors for the whole training set is used.

if the samples are linearly separable. However, not all problems are linearly separable; for example, of the binary functions with two arguments, the exclusive-or and its inverse are not. If the set of samples is not linearly separable then the 'separating' hyperplane will oscillate. Similar behaviour occurs when the data set contains contradictory samples.

The limitation of only being able to implement linearly separable functions can be overcome using a multi-layered perceptron. Nevertheless, the learning method for the perceptron, namely, the perceptron learning rule, cannot be extended to the multi-layer perceptron because there are no target values for hidden layer PEs to adjust their incoming connection weights.

2.6.2 Multi-Layer Perceptron

Minsky and Papert first showed that *exclusive-or*-like (i.e. non-linearly separable) problems cannot be performed by the perceptron (*Minsky and Papert 1969*). They also pointed out that if there is a layer composed of a large enough set of hidden PEs, providing 'internal representations' of input patterns, then a solution that performs a mapping from inputs to outputs through these PEs can be found. However, whilst there was a very simple guaranteed learning rule for all problems that can be solved without hidden PEs, namely the perceptron learning rule, there was no equally powerful mechanism for learning in multi-layer perceptron (MLP) networks, which have hidden layers.

Later on, one of the responses to this need was from the PDP research group with the *generalised delta rule* based on the gradient descent method of the mean squared error (MSE) (*Rumelhart et al 1986*). In this method, the error at the output layer is 'backpropagated' to previous layers, allowing incoming weights to these layers to be modified. A network using this rule, therefore, is also called *backpropagation network* (BPN), referring to the learning method rather than referring to its structure where all information flow is feedforward.

In a MLP, if a weight is to be adjusted regardless of in which layer it is, its effect on the outputs of the network has to be known. Backpropagation or the generalised delta rule uses the derivative of the error function with respect to that weight, like in the original delta rule. Thus, the error at the output is reflected back to the preceding layer PEs in proportion to their outgoing weight values (i.e. effects on the output layer PEs). The change, Δw_{ji}, in the weight of a connection between the i th PE in the previous layer and the j th receiver PE in the current layer is as follows:

$$\Delta w_{ji} = \alpha \delta_j x_i$$

$$w_{ji(new)} = w_{ji(old)} + \Delta w_{ji}$$

where α is the learning rate, δ_j is the error factor for the PE and finally x_i is the i th connection to the PE. The error factor depends on whether the PE is in the output layer or in a hidden layer. For the output layer PEs:

$$\delta_j = f'(net_j)(d_j - y_j)$$

where f ′ is the derivative of the PE's activation function, net_j is the total weighted sum of inputs to the PE, d_j is the desired target value for the j th output and y_j is the PE's actual output. For the hidden layer PEs, there is no target and hence no 'error' concept. Therefore, the cumulative of error factors of following layer PEs times 'corresponding' outgoing weights is used instead of 'error'. So, starting from the latest hidden layer PEs, the error factors for hidden layer PEs are calculated as following:

$$\delta_j = f'(net_j) \sum_k w_{kj} \delta_k$$

It is noticed that the derivative of the activation function plays an important role in backpropagation learning. The threshold-logic activation function, on which the perceptron is based, is discontinuous and hence will not suffice for the backpropagation algorithm. Similarly, a linear activation function will not suffice either as there is no advantage in using a hidden layer in that case[10]. Thus, the new activation function needs to be *continuous* and *non-linear* (*Rumelhart et al 1986*). The sigmoid logistic function is the most common and is the one used by Rumelhart et al. On the other hand, replacing the sigmoid activation function by gaussian or other types is also possible as long as they are differentiable. Yet, the advantage of choosing one of them or using a different activation function in different layers is not completely understood (*Hassoun 1995*).

The weight updating process in a MLP network can take place after the presentation of each pattern or after the presentation of the whole training set, and a number of epochs are required for the MLP to be properly trained as in the case of perceptron learning. To make an improvement on training time a variety of methods can be employed, for example, updating weights after a number of iterations (i.e. *batch training*), or adding a 'momentum' term which gives the effect of the preceding weight change (i.e. *momentum* or *smoothing training*), or using both principles together (*Hecht-Nielsen 1992*). However, once again there is no firm rule saying which method is generally better than others.

2.6.3 Learning Vector Quantization

Learning Vector Quantization (LVQ) is a supervised form of competitive learning[11] developed by Kohonen for adaptive pattern classification (*Kohonen 1989*). The LVQ network, based on this learning mechanism, develops an approximation to the decision regions of each class.

The LVQ network has a unique feedforward structure and is composed of three layers: An input (buffer) layer, a hidden layer (called Kohonen layer), and an output layer. The numbers of PEs in the input layer and the output layer are defined by the

[10] The multi-layer networks using a linear activation function can be reduced to single-layer networks.
[11] Recall that earlier competitive learning is mentioned to be amongst unsupervised learning methods.

number of inputs and the number of classes in which the input patterns are scattered, respectively. The number of PEs in the Kohonen layer must be at least the same as the number of PEs in the output layer (in this case one Kohonen layer PE for each class). The input layer is fully connected to the Kohonen layer. The Kohonen layer, on the other hand, is partitioned into groups of PEs so that each group is assigned to a particular class. Finally, each output layer PE is linked to a different group of PE(s). In other words, the output layer is partially connected to the previous layer as opposed to the MLP network.

The LVQ network, like any other network employing competitive learning, involves a somewhat different 'neuron' (i.e. PE) concept (*Maren 1990b*). The output layer connection weights are fixed to '1' while an adaptive weight vector representing a 'template' for a particular class is associated with each Kohonen layer PE. When an input pattern is presented to the network, the Kohonen layer PE whose vector is the closest (in a Euclidean sense) to the input vector wins the 'competition'. Then, this PE yields a '1' whereas all other PEs are forced to yield a '0'. Consequently, only the output PE dedicated to the class of the current pattern becomes 'active' provided that the network has been properly trained.

The training process begins with the initialisation of 'template' vectors. i.e., the weight vectors of PEs in the Kohonen layer. After the presentation of an input pattern to the network, each Kohonen layer PE calculates the Euclidean distance between input vector and its weight vector as follows:

$$d_i = \|w_i - x\| = \sqrt{\sum_{j=1}^{n} (w_{ij} - x_j)^2}$$

These distances are used to select the 'winner' PE. This competition is carried out regardless of class assignments, and the winner PE may or may not be in the correct group (which is assigned to the class of the current input pattern) as determined by the 'teacher' signal. Then, only the winner PE's weight vector is updated according to the following learning rule (standard LVQ) :

$$\Delta w_i = \begin{cases} \alpha(x - w_i) & \text{if the winner is in the correct group} \\ -\gamma(x - w_i) & \text{if the winner is in the incorrect group} \end{cases}$$

$$w_{i(new)} = w_{i(old)} + \Delta w_i$$

where α and γ are learning rates, and the index i designates the winner PE. This equation moves the winner PE's weight vector a small distance towards or away from the input vector. Kohonen suggests that the convergence speed can be improved if each vector has its own adaptive learning rate (*Kohonen 1990*). Other algorithms developed for improvement are a 'conscience' mechanism (*DeSieno 1988*) and LVQ2 (*Kohonen 1989*).

Chapter 3

MODELLING THE SYSTEM

3.1 Introduction

The flight data used in this work was generated from a total of ten hours of flight tests on a *Westland Lynx Helicopter XZ170 Mk 9*. The total data, containing a wide range of helicopter manoeuvres, consisted of 48 channels of flight parameter data and three channels of helicopter load spectra measured by means of strain gauges. For this work an eighty second sample was taken which included two major manoeuvres and some non-transitional flights so as to keep experimental time to a minimum and minimise the amount of computer disk space required.

Of the three load spectra channels, *the main rotor spider arm control* was selected for study. Previous studies on this problem concluded that this strain signal was the most difficult to predict, and concentrated on prediction of the other 2 signals (*Hill et al 1995*, *Vella et al 1996*).

3.2 The System Modelled

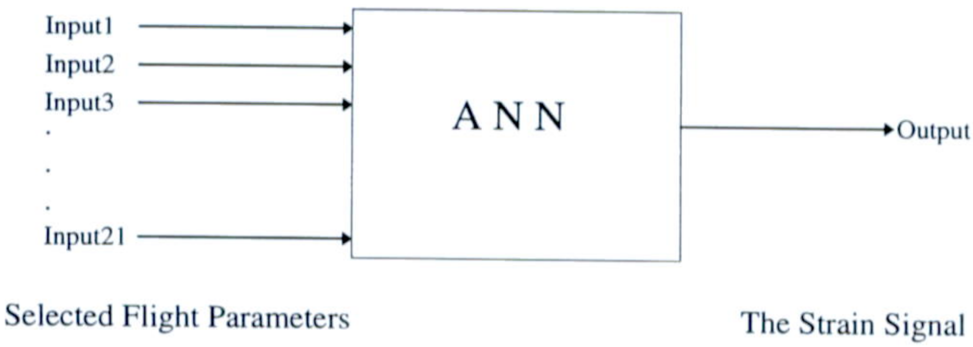


Figure 3.1
A Simple Illustration of the System

An ANN system can be thought of as a black-box like in **Figure 3.1**. Despite there being 48 flight parameters, a number of them were omitted on the basis of domain knowledge. More precisely, experts suggested that some parameters have almost no effect on strain in the airframe or rotorhead (*Lison 1994*). In the end, 21 flight parameters were used on the basis of discussions with helicopter operators and the manufacturers of the Lynx (*Hill et al 1995*).

3.2.1 Flight Parameter Data

The parameters mentioned above and their sample rates are shown in **Table 3.1**.

Number	Description	Rate (Hz)
Input 1	Collective Lever Position	26.25
Input 2	Collective Servo Position	840
Input 3	F/A Cyclic Stick Position	26.25
Input 4	Control Cyclic Stick Position	26.25
Input 5	F/A Cyclic Servo Position	840
Input 6	Control Cyclic Servo Position	840
Input 7	Tail Rotor Pitch Angle	26.25
Input 8	Rudder Pedal Position	26.25
Input 9	Barometric Altitude	26.25
Input 10	Barometric Airspeed	26.25
Input 11	Lateral Acceleration	26.25
Input 12	Main Rotor Speed	26.25
Input 13	Outside Air Temperature	26.25
Input 14	Pitch Attitude	26.25
Input 15	Pitch Rate	26.25
Input 16	Roll Attitude	26.25
Input 17	Roll Rate	26.25
Input 18	Aerodynamic Yaw	26.25
Input 19	Normal Acceleration Centre of Gravity	26.25
Input 20	Yaw Rate	26.25
Input 21	Main Rotor Hub Torque	840
Output	Strain in Main Rotor Spider Arm Control	840

Table 3.1

Flight Parameter Data

Apart from the parameters given above it was also suggested by my supervisor that an 'artificial' parameter, namely, *sawtooth waveform at rotor frequency* (R), would increase performance of the ANN system. Since 'outside air temperature' had a virtually constant value, it was replaced with this 'artificial parameter'. Otherwise, it would not have had any effect other than unnecessarily doubling the bias inputs of the first hidden layer PEs in the network, i.e., behaved like a second bias input for first hidden layer PEs.

As can be seen from **Table 3.1**, the data was gathered using two different sample rates and the 2 sets needed to be of the same size before presentation to the network. Therefore, it was necessary to sample the high rate, 840 Hz, channels or 'balloon' the low rate channels. The latter option was taken due to the strain channel having a high data rate, so as to preserve as much information as possible.

3.2.2 Scaling

Generally it is best for all inputs to the network to be of the same order of magnitude. Since the bias value is 1, they should be of that order of magnitude. Thus, before being presented to the network, channels were scaled between zero and one as follows :

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$$

3.3 Hardware

The flight parameter data were stored on a PC format. The experiments were undertaken on three different PC models. Because of this fact, experimental durations depended on the PC model on which the experiment was carried out. The PC configurations and their relative training times in a given task were as follows :

No.	Microprocessor	Co-processor	Speed (MHz)	Duration
1	486 SX	-	25	30
2	486 DX	Built-in	33	3
3	486 DX4	Built-in	100	1

Table 3.2
PC Models Used in the Experiments

To give an idea in comparison of them let us give this example: If a training session takes 1 day on model 3, roughly it takes 3 days on model 2 whereas it takes 1 month on model 1. Therefore, if we had only used model 1 it would not have been possible to do all the experiments we did, even if we had 10 years.

3.4 Software

A number of companies develop commercial software for neurocomputing. HNC ExploreNet 3000 software package was available to us for study. This software runs under Windows. It gives users some flexibility to try out a variety of different ANN paradigms to develop a unique system. Some brief information is given below about the specifications of the package and the applications developed by means of it.

3.4.1 The Program Package - ExploreNet

In the package, applications are created by means of *modules* and the *connections* between modules. Each module in an application represents a software device to perform some specific task. There are 9 different module types available for this reason. These are listed below :

Module Type	Function
File	handles data flow from or to a file or a group of files.
Form	sets up interactive screen forms.
Pipe	enables to communicate with other *Windows* programs.
Data	performs mathematical and logical computations.
Network	provides predefined ANN paradigms.
Graph	displays two-dimensional graphs.
Chart	provides column bar charts.
Tile	provides tiled graphical display of vector or matrix data.
Image	views data in the form of an image.

Table 3.3
Module Types

To identify data manipulated by a module, variables called *items* are created. They are either input items or output items of a module. They are used to pass information from one module to another over the *mappings* in the connection in between. In other words, mappings in the connection between the modules transfer the values from one or more output items of one module to input items of another module. Thus, connections also determine which of the connected modules will be executed first. Unlike those of other modules, all items in network modules are created automatically once the specifications of the network has been defined.

All modules also have a group of predefined items called *control items*. They are built into modules when modules are created, and influence the behaviour of the modules in various ways. Apart from these, there are 4 more 'global', control items in an application, which affects the whole application rather than any single module[1].

[1] For more information refer to *HNC ExploreNet 3000 Manual Release 2.0*.

3.4.2 Features of the Package

ExploreNet has some useful mechanisms giving control over application details and operation. They are *tasks*, *sessions*, and *watchpoints*.

A task is a series of statements that set the values of items in any module of the application. By setting the value of items, especially control items, we can influence how the modules of the application behave. For example, 'learning' in a network module can be switched on and off. In addition to the statements, a task may specify that the application be run through a certain number of cycles after the statements have been executed. To run a task, however, it must be included in a session, which is a series of tasks. Both mechanisms (i.e., tasks and sessions) ultimately complete a run after a fixed number of cycles. On the other hand, another mechanism, watchpoints, enables us to cease the currently running application cycle, task or session under a condition. A watchpoint is a conditional expression which tests that condition.

Another feature is that the network modules allow the repeated saving of network files (including weight values) through the use of auto-incrementing file names for different learning levels. It is accomplished by means of an auto-incrementing file name counter. If the file name to which the network data is saved is defined by one or several '#' characters at the end of the file name then the contents of that counter are added to the file name before it is saved. Hence, it is possible to save and load different learning statuses of a network to and from different files in tasks, e.g., if we give the name "exp1-##.wts" to 'weight save file' option then we can save a hundred different learning levels of the network on files "exp1-00.wts" through "exp1-99.wts" and given the same name for 'weight load file' option we can restore any of those learning levels through the associated file.

One more important feature is *fast train*. It is an alternative mode of running an application. It is controlled by a task and lasts the duration of the task. During a 'fast train' task, all network modules in the application do not run initially. Instead, they simply save each record to the network module in a memory buffer. After a prescribed number of records have been accumulated, the network modules alone are run a specified number of times and then the task finishes.

3.4.3 Applications

Four applications were developed for the experiments undertaken. One distinguishing factor in the applications was the use of two different ANN paradigms. One network module cannot represent several different paradigms because a network module's items are determined according to the paradigm chosen for the module[2]. If 'paradigm' option of the network module is changed it loses the mappings in the connections with other modules since old items no longer exist.

The other factor was whether training or testing was being performed. To perform training and testing in the same application leads to some difficulties when different learning statuses of the network are concerned[3]. Taking the above facts into account, the applications developed are as follows :

Description	Purpose
LTRAIN	to train LVQ network
LTEST	to test LVQ network
MTRAIN	to train MBPN[4] network
MTEST	to test MBPN network

Table 3.4
Applications Developed

[2] Recall network modules' items are created automatically, not by the user.
[3] Refer to *HNC ExploreNet 3000 Manual Release 2.0*.
[4] MBPN stands for Multi-Layer Backpropagation Network.

Chapter 4

METHODOLOGY

4.1 Introduction

Solving a problem on a computer requires a certain number of steps (time complexity), a certain memory size (space complexity), and a certain length of algorithm (Kolmogorov complexity). In an ANN simulation, the number of computations is a measure of time complexity, the number of units is a measure of space complexity and the number of weights (degrees of freedom) where the algorithm is 'stored' is a measure of Kolmogorov complexity (*Hassoun 1995*).

Problems that require complex algorithms make the most use of ANNs. On the other hand, the theory of ANNs has not yet been sufficiently developed in the way that, say, a radio or a bridge is engineered. In general, designing an ANN system is an 'empirical' process. It requires a combination of careful research and planning, educated guesswork, and outright trial-and-error. More precisely, in formulating ANN solutions for practical problems, the resulting time, space and Kolmogorov complexities should be minimised so as to obtain an efficient system. The strategy used for this purpose during the experiments undertaken is explained in this chapter.

4.2 Paradigm Selection

Once there has been a decision to use an ANN approach for the problem at hand, a wide range of ANNs has to be taken into consideration. Although there is no firm rule defining the most suitable type for a given problem, some networks have been found useful in specific applications. (*Jones and Franklin 1990*).

Networks trying to approximate a function or mapping from a subset, i.e., examples of the mapping's action, are called mapping networks. There are basically two types of mapping networks: *feature-based* networks and *prototype-based* networks (*Hecht-Nielsen 1990*). A feature-based network implements a functional input/output relationship that is expressed in terms of a general, modifiable functional form. This functional form is modified via the adaptive setting of weights so as to fit the specific mapping that is to be approximated. Prototype networks, on the other hand, operate by creating a set of specific input/output examples that statistically represent the action of the mappings. The LVQ network is an example of prototype-based networks, whereas the backpropagation network (BPN) is an example of feature-based networks[1].

The BPN is without a doubt the most widely used network. It is a universal function approximator. In principle, there is no continuous input/output mapping it cannot approximate. However, its ability to generalise from a set of input/output pairs has no such guarantee (*Stone and Thorton 1995*). The LVQ network, on the other hand, was originally developed to solve pattern classification problems. It develops an approximation to the decision regions of each class. It is reported that it has been used successfully to classify speech spectra (*Galindo 1995*).

At the very beginning of the research the intention was to use the two networks not only to compare the performances of both types but also to carry out further experiments with the more promising one.

The package used provides a variety of BPNs. The Multilayer Backpropagation Network (MBPN) paradigm is a version of BPN which is derived from the work of Rumelhart and the PDP group. There is also another paradigm called BPN. One of the differences is that a BPN paradigm only allows zero or one hidden layer while a MBPN paradigm supports up to three hidden layers. Likewise, it offers a choice of five parameterised activation functions against BPN's single option of logistic function. One of the previous works attempted to use the BPN paradigm and produced poor results on the prediction of the strain signal that is used here, although the results for the other strain signals were quite good (*Vella et al 1996*). To build a successful ANN model MBPNs were employed due to their broader range of choice of network parameters.

[1] Recall the theory of these networks from chapter 2.

The other paradigm applied to the problem was LVQ. This type had not been used earlier for the prediction of strains at Cranfield, so no results was available on the performance of the LVQ network on the prediction of strains.

4.3 Defining Network Parameters

Choosing a proper network structure is essential in practical applications. Whatever network paradigm is selected it is crucial that the parameters of the network be chosen carefully in order to exploit the full potential of the network. For example, a large network with many PEs and in turn with a large number of weighted connections can learn training data with minimal errors. However, it easily loses its generalisation ability and results in poor robustness (*Mrazova 1995*). In other words, an excessive number of weights can easily cause overfitting and thus a large error on data outside the training set. A small network, however, is unlikely to be overfitted but may not be able to learn the training set well (*Murata 1995*).

In the software used here, the network parameters are categorised as *load-time parameters* and *run-time parameters*. Load-time parameters such as numbers of PEs in the layers and the activation function of PEs define the topology of the network. Therefore, they need to be specified when the network is loaded; they cannot be modified during the training phase. In contrast, run-time parameters, which define processing equations such as learning rate, may be specified at any time. Some run-time parameters are also defined together with load-time parameters so as to restrict their values; for example, the learning method when the MBPN paradigm is used.

4.3.1 Parameters of MBPN

Having selected the MBPN, it was necessary to specify its parameters. Load-time and Run-time parameters of MBPN and their input forms are shown in

Figure 4.1.

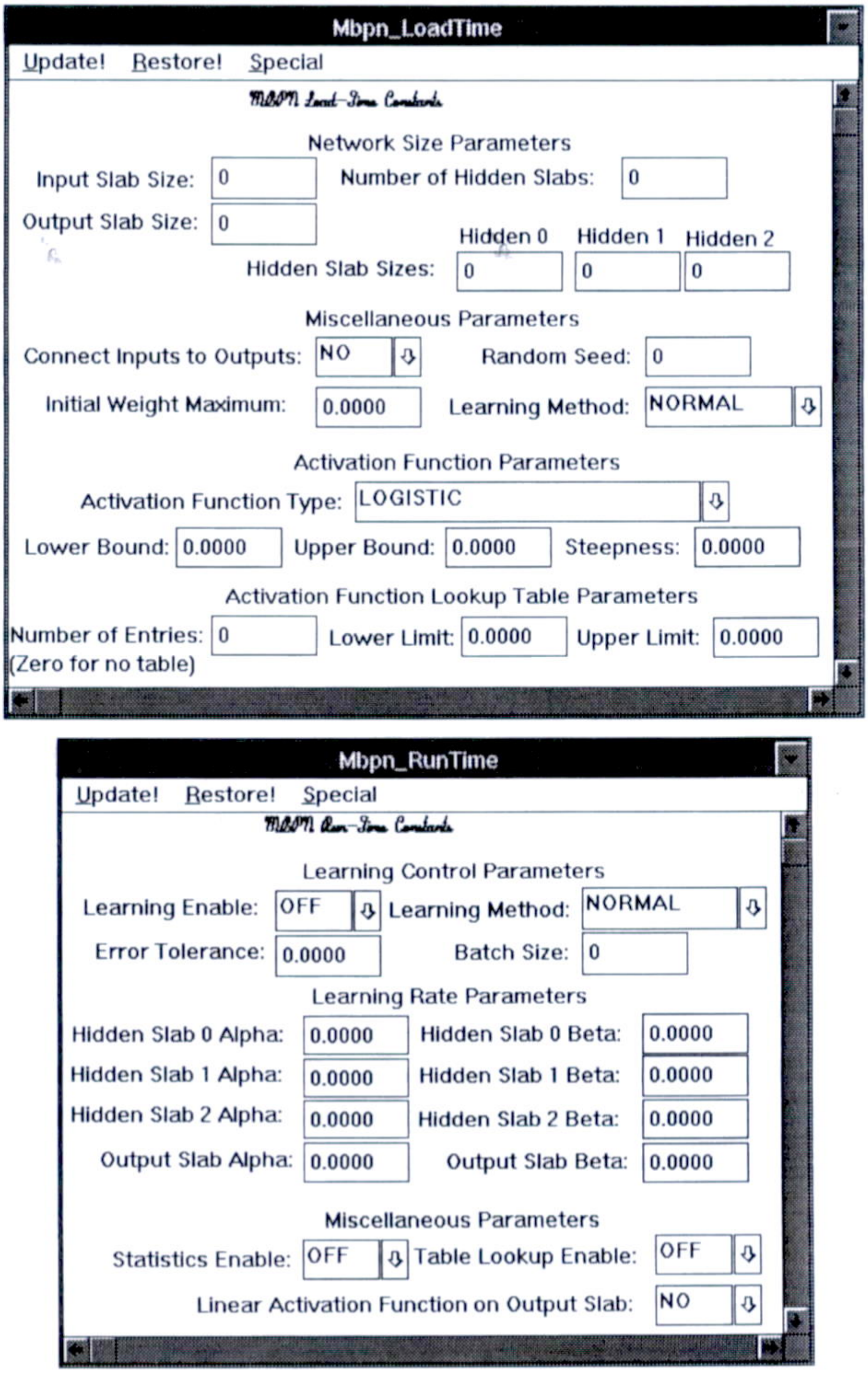


Figure 4.1
Default Values of Parameters of MBPN Paradigm[2]

[2] Glossary: In the tables shown above the word slab is also known as layer. In addition, alpha means learning rate and beta momentum rate.

When using a BPN the first thing to be done is to define the number of layers and the number of PEs in each layer. There is always an input and an output layer, and it is obvious what the number of PEs in these layers must be as they depend on the nature of the problem. By comparison, specifying an optimal number of hidden layers and then the number of PEs in each one is not easy, because the structure that will best fit is not fully understood. More precisely, it is never clear that the final model is accurate until it is tested.

Here are some other points on defining the number of hidden layers and their PEs. The Kolmogorov Theorem states that any continuous function of n variables can be implemented exactly by an ANN with one hidden layer comprising $(2n + 1)$ PEs (*Hecht-Nielsen 1990*). Kolmogorov's Theorem was a first step in this direction. However, several other ideas arising from subsequent experience have been put forward since then. One of these suggests that there is no need for more than two hidden layers in processing real-valued input data and the accuracy of approximation is controlled by the number of PEs, not the number of layers (*Picton 1994*). On the other hand, the backpropagation approximation theorem suggests that, given $\varepsilon > 0$ and any continuous function of n variables, a BPN with one hidden layer is enough and can approximate the function to within ε mean-squared-error accuracy (*Hecht-Nielsen 1992*). However, it is further emphasised that one hidden layer is not necessarily the most efficient number; using more layers could result in a smaller number of PEs in the whole network.

The best advice based on current knowledge is that if the outputs of a ANN need to be continuous functions of the inputs, then using two hidden layers may be more reasonable. Otherwise, a single hidden layer should be sufficient, depending on the nature of the problem (*Maren et al 1990*).

Taking into consideration all the factors above, MBPNs with both one and two hidden layers were used in the experiments. For single-hidden-layer MBPNs, experiments were performed with different numbers of PEs in the hidden layer. A constant number of PEs were used in the experiments with two-hidden-layer MBPNs (i.e., 20 PEs in each hidden layer) to investigate the effects of other network parameters.

Another important network parameter is the learning rate, which relates to the training procedure, not to the network's structure. The learning rate has a significant effect on the learning speed and efficiency. A high learning rate may result in faster learning, but makes the network subject to oscillation. A small learning rate, on the other hand, may stabilise the process, but results in slower learning. Moreover, it is likely to lead to a local minimum (*Maren et al 1990*). Beginning with a high learning rate and later decreasing it as learning progresses sounds plausible. However, what is too big and what is too small are the issues here. Clearly, it depends on the error surface of the problem. Furthermore, the starting point on the surface (i.e. initial weight values) also has an impact on the performance of a particular learning rate as different starting points are likely to result in different minima. Once again considering previous work on this problem and the facts above, two different learning rates (0.1 and 0.5) were chosen for MBPNs as initial values.

Another parameter considered for change was the activation function. The MBPN provides five parameterised functions for this purpose. The *logistic* is the most widely used one in BPN applications. Besides the *hyperbolic tangent*, the *inverse tangent* and the *threshold-linear* are used to some extent. All four of these 'sigmoid' functions are available in the package[3]. The other activation function is *Gaussian*.

Activation functions have parameters that determine the maximum and minimum values. The usual choices are 1 for the maximum and 0 or -1 for the minimum. One reason for the magnitude of these values being about 1 is that the bias input is 1, and inputs should usually be of the same order of magnitude. The forms and formulae of these functions are shown below:

[3] They are called 'sigmoid' functions, meaning S-shaped, because of the general shape of their graphs.

The Logistic Function:

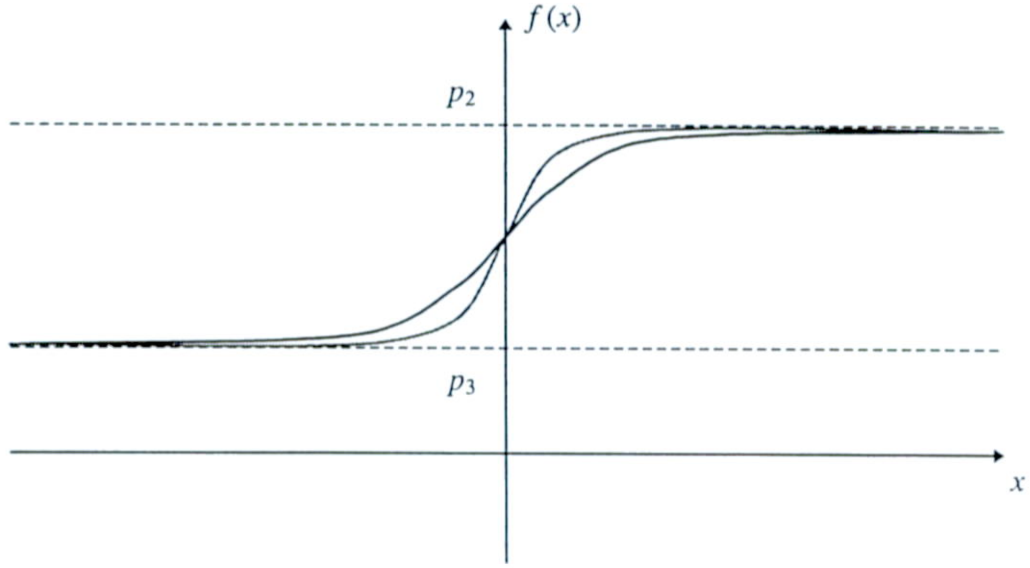


Figure 4.2
Logistic Activation Function

The logistic activation function is defined by

$$f(x) = \frac{p_2 - p_3}{1 + e^{-p_1 x}} + p_3$$

where p_3 is the lower limit, p_2 is the upper limit, and p_1 determines the slope at $x = 0$; the larger the value, the steeper the slope. p_1 must be positive and p_2 must be greater than p_3. **Figure 4.2** shows the form of logistic function for two different values of p_1.

The Inverse Tangent Function:

The inverse tangent function is defined by

$$f(x) = \frac{p_2 + p_3}{2} + \frac{p_2 - p_3}{\pi} \tan^{-1}(p_1 x)$$

where p_3 is the lower limit, p_2 is the upper limit, and p_1 determines the steepness. Its graph is similar to that of the logistic function.

The Hyperbolic Tangent Function:

The hyperbolic tangent function is defined by

$$f(x) = \frac{p_2 + p_3}{2} + \frac{p_2 - p_3}{2}\tanh(p_1 x)$$

where p_3 is the lower limit, p_2 is the upper limit, and p_1 determines the steepness. Its graph is similar to that of the logistic function as well.

The Threshold-Linear Function:

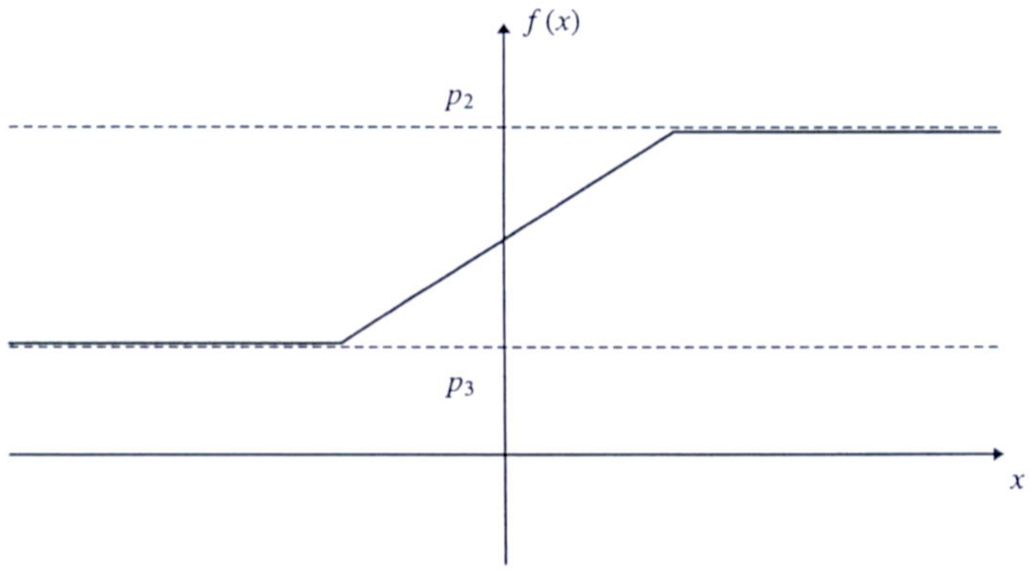


Figure 4.3

Threshold-Linear Activation Function

The threshold-linear function is defined by

$$f(x) = \begin{cases} p_2 & \text{if } x \geq \dfrac{p_2 - p_3}{2p_1} \\[2ex] p_3 & \text{if } x \leq -\dfrac{p_2 - p_3}{2p_1} \\[2ex] \dfrac{p_3 + p_2}{2} + p_1 x & \text{otherwise} \end{cases}$$

Again, p_3 is the lower limit, p_2 is the upper limit, and p_1 determines the slope.

The Gaussian Function:

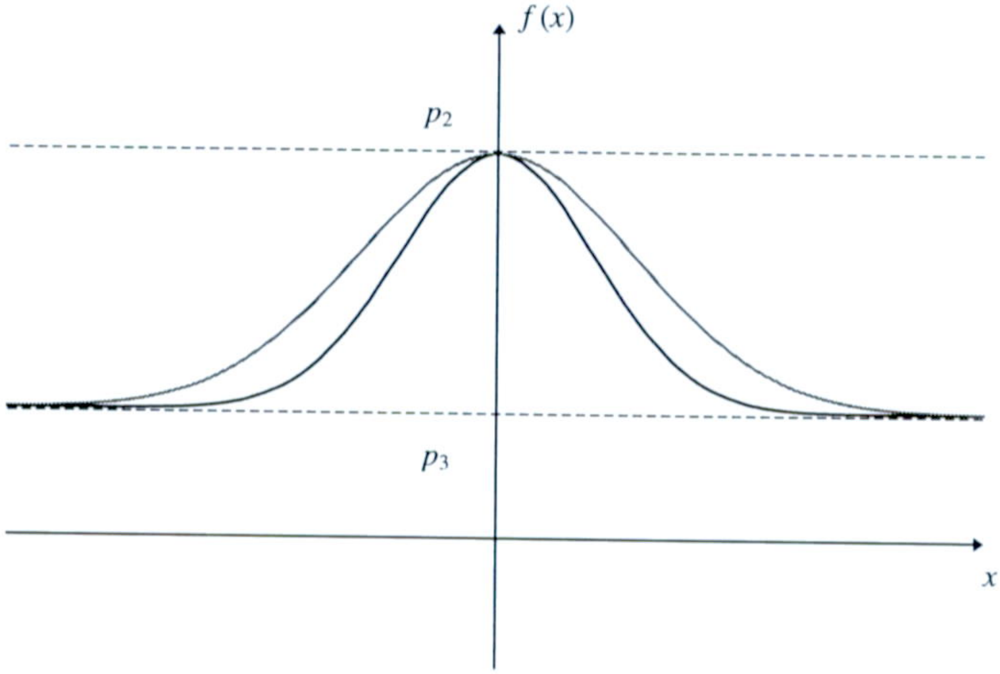


Figure 4.4
Gaussian Activation Function

The Gaussian function is defined by

$$f(x) = p_3 + (p_2 - p_3)e^{-x^2/2p_1^2}$$

where p_3 is the lower limit, p_2 is the upper limit, and p_1 determines the sharpness of the peak; the smaller the p_1 is, the sharper the peak. **Figure 4.4** shows the form of Gaussian function for two different values of p_1.

As can be seen in the figures, the sigmoid functions allow a PE to define one half of the space of its inputs as giving a high output value, and the other half a low output value. The Gaussian function is bump-shaped and as opposed to sigmoid functions qualitatively allows a PE to define the region of the space of its inputs near a hyperplane as calling for a high output and the rest of the space a low output. It is suggested that the Gaussian function may be more useful than sigmoid functions when the input space regions of interest are more localised. Logistic and Gaussian activation functions were used for experiments with MBPNs.

4.3.2 Parameters of LVQ

The LVQ paradigm defines six load-time and nine run-time parameters. These are shown in their input forms in **Figure 4.5**.

Once again, the first issue is to specify the number of PEs in the hidden layer, i.e., Kohonen layer[4] (*Maren 1990b*). The number of classes defines the minimum number of these PEs and defining the number of PEs per class will determine the exact number of PEs in the hidden layer.

Because the problem was not a classification problem originally, it was necessary firstly to divide the range of strain values into a number of sub-ranges which would then represent the classes. Of course, the number of classes in this case also determines the precision of the results, i.e., the more classes that are defined the

[4] Recall the LVQ network structure

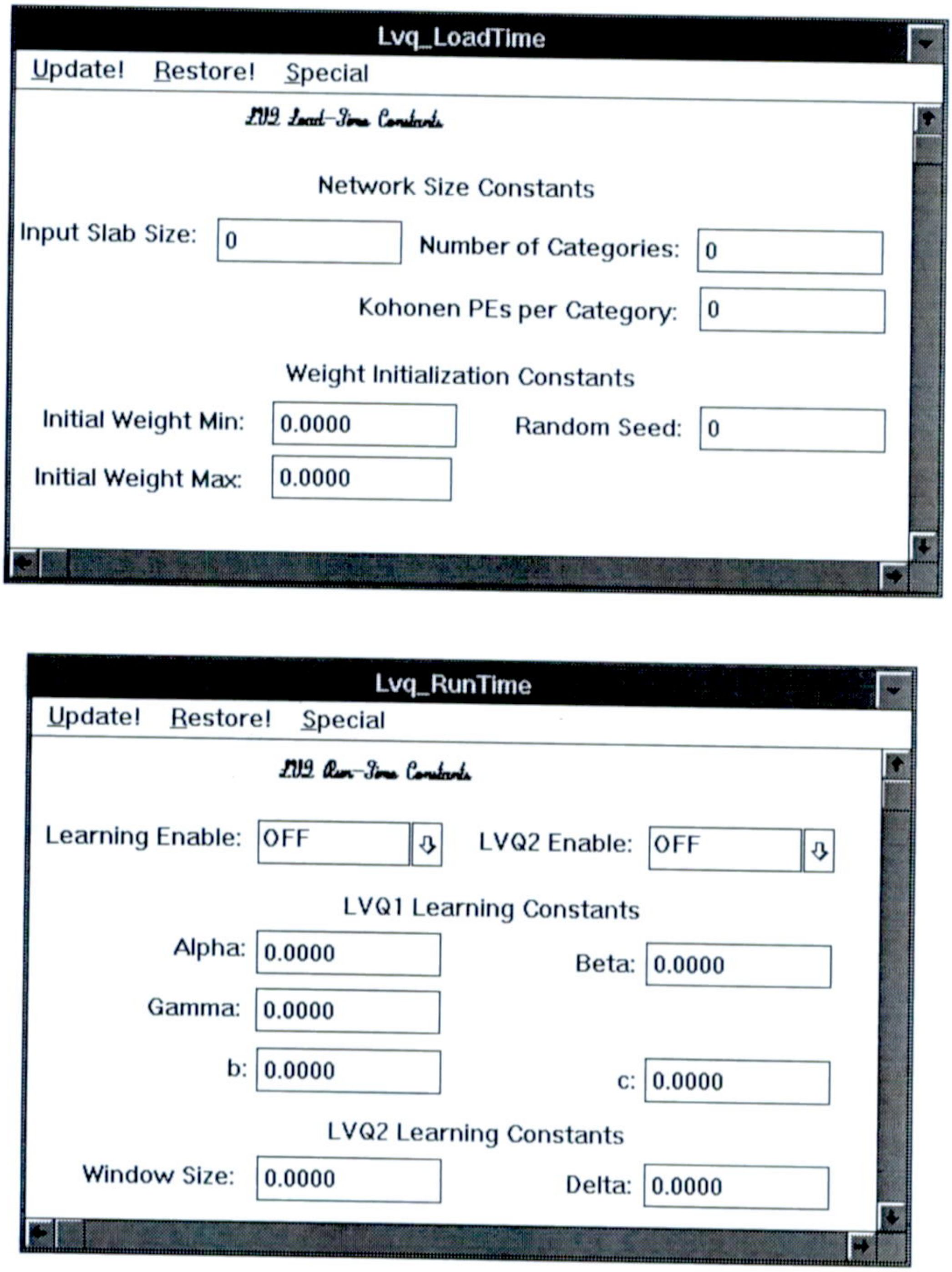


Figure 4.5

Default Values of Parameters of LVQ[5]

[5] Recall weight updating in LVQ from section 2.6.3. LVQ2 learning is not activated during the experiments undertaken. For further information refer to *HNC ExploreNet 3000 Manual Release 2.0.*

higher the precision obtained. However, more classes will also increase the number of PEs in the network, which in turn increases the training time. A hundred classes were specified as follows:

$$Nc = \left(100 \ * \ \frac{x - x_{\min}}{x_{\max} - x_{\min}} \right)$$

where Nc denotes the class number, x is the current value, and $x_{\min}$ and $x_{\max}$ are the minimum and the maximum values of the strain respectively.

Having defined the number of classes, it is necessary to specify the number of PEs per class. This was again one of the points examined. The LVQ networks with a different number of PEs in the hidden layer were employed.

The next important issue was to select the training mode and then specify a good learning rate. Since there was no *a priori* knowledge about the performance of the LVQ on the current problem, typical values given in the manual for these parameters were taken as the initial values.

4.4 Training Procedure

During training an ANN is expected to change its weights until they converge to a set which models the training set. Thus, learning in an ANN normally involves the repeated presentation of the training data to the network[6]. Learning is a hard process in a sense that the loading of an arbitrary mapping onto a 'faithful' network architecture requires exponential time irrespective of the learning algorithm. In other words, the training time scales up exponentially in the size of the problem as the

[6] Recall learning algorithms from Chapter 2.

problem size increases, i.e., as the pattern dimension and/or the number of input patterns increases (*Hassoun 1995*).

One of the problems in the current work was the amount of data. A number of different ANNs were employed, aiming to reproduce a small part of the strain (i.e., the first 300 data points). This was to attain an insight into the performance of different ANNs. Further experiments were then conducted with promising ANNs.

4.4.1 Representative Nature of Training Data

One of the vital aspects of training is the training data itself. More precisely, the 'quality' of the output data (i.e., how close a network's results are to actual data) depends on the representative nature of the original training data in relation to the scope of the problem domain as well as the training method and the network structure (*Helliwel et al 1995*).

The first part of the experiments, working on 300 patterns, sought 'candidate' ANNs as mentioned earlier. Then, the huge amount of data had to be dealt with. In order to sample the flight in a representative way, a mixing procedure was performed. Training 10% of the mixed data was tried for the whole 80 second data samples (67217 points). By doing this, not only an attempt was made to provide a representative training set and thus reduce the training time but it also had the benefit of presenting them to the network in a random order. Otherwise, if the network had been trained on the data presented to it sequentially, it would have 'forgotten' the early manoeuvres (i.e., different characteristics which lie in different parts of the signal) as later ones were presented to the network (*Vella et al 1996*).

4.4.2 Training Mode Selection

The importance of the hardware configuration in training has been shown in the preceding chapter. The package used also offers an alternative mode of training, namely *fast train* mode[7]. This feature was exploited as much as possible in order to minimise the training time, but the problem was that in this mode the size of training set had a limitation irrespective of PC configuration[8]. The limit was about 700 patterns for the network modules in the applications developed. Although this was far enough for the first part of the experiments, using 300 patterns, it could not be applied directly in the major part of the experiments, using 6721 patterns.

The method used in this case was to split the training set into sub-sets so that the 'fast train' mode could be applied. For this purpose the training set (6721 patterns) was divided into 10 sub-sets each of which comprised 672 patterns except the final one which comprised 673 patterns. On one hand, it was necessary to present each sub-set to the network a number of times in order to exploit the 'fast train' mode and reduce the training time; on the other hand, excessively increasing the number of iterations on a single sub-set, without showing others, would have resulted in losing the advantage of presenting the training data in random order. Thus, after some experiments the number of continuous iterations on a single sub-set was kept to a maximum of a 100 as an optimal number for this dilemma[9]. Hence, it was possible to have ten times as many iterations with 'normal' mode during a certain amount of time. Needless to say better results were recorded by means of the 'fast train' mode in a particular time period.

4.5 Assessing the Results

In order to compare the target signal with the outputs of the ANNs, it is necessary to decide on suitable criteria. One method may be a visual assessment, comparison of the shapes of actual and predicted strain signals. However, a mathematical measure should be used to be more scientific, i.e., to measure how accurate the predictions are. Initially, this was done by recording the *correlation*

[7] Recall it from chapter 3 section 'The Features of the Package'.
[8] The limit which was the same for all three PC configurations is probably defined by the package.
[9] Refer to experiments 12 and 12a in appendix A.

4.5.1 Correlation Coefficient

The correlation coefficient gives a measure of the linear relationship between two sets of data points. The higher the correlation, the stronger the linear relationship. A strong linear relationship results in a correlation near 1 while a weak one is close to 0. Given two sets of $x=\{x_1, x_2, x_3,..., x_N\}$ and $y=\{y_1, y_2, y_3,..., y_N\}$, the correlation coefficient (r) is formulated by

$$\left.\begin{aligned} S_x &= \sum_{i=1}^{N} x_i \\ S_y &= \sum_{i=1}^{N} y_i \\ S_{xx} &= \sum_{i=1}^{N} (x_i)^2 \\ S_{yy} &= \sum_{i=1}^{N} (y_i)^2 \\ S_{xy} &= \sum_{i=1}^{N} x_i y_i \end{aligned}\right\} \quad r = \frac{N\,S_{xy} - S_x S_y}{\sqrt{(N\,S_x - (S_{xx})^2)\,(N\,S_y - (S_{yy})^2)}}$$

where N is the number of elements in each set. It was used as the main measure of the network's performance.

4.5.2 Mean Absolute Error

Workers in the field of ANNs traditionally use the mean absolute error (MAE) in assessing the results, but it is difficult to say what the value of the MAE should be. The MAEs were evaluated as a measure of how close the predictions were. For the two sets given above, it is calculated as follows:

$$MAE = \frac{\sum_{i=1}^{N} | x_i - y_i |}{N}$$

4.5.3 Maximum Error

The maximum error was the last measurement recorded during the experiments. It was used as complementary information concerning the performance of ANNs. Although it was not crucial to our problem, it might be critical for some problems as it shows the maximum distance between two sets in terms of prediction.

Chapter 5

DISCUSSION OF EXPERIMENTAL RESULTS

5.1 Introduction

This chapter discusses the results of experiments which were undertaken during a period of just over four months. Each section describes and assesses a separate stage of the work. The details of experiments are given in appendix A.

5.2 Comparing the Results of LVQ and BPN

As initial work, a part (i.e., the first 300 points) of the strain signal was used as indicated in chapter 4. In this part, running in parallel with the backpropagation (BPN) development was Learning Vector Quantization (LVQ) implementation. The purpose of this was not only to make comparison between two types but also to carry on further experiments with the more promising one[1].

Although a number of different structures were used, the LVQ produced very poor results in all cases[2]. In the experiments with LVQ, the number of categories (i.e., classes) was 100, and the number of PEs in the Kohonen layer was determined by 'the number of PEs per category' network parameter. Nevertheless, no promising results were obtained with this type even trying to teach a small part of the strain signal. **Figure 5.1** shows the performance of different LVQ structures (which have 1, 2, 3, 5, and 7 PEs per category) with respect to correlation coefficient and mean absolute error between the desired and predicted signals after 500 epochs.

At first glance, **Figure 5.1** suggests that the optimal number of PEs in the Kohonen layer should be 5. Nevertheless, increasing the number of iterations as far as 8000 with this structure still led us to very poor and oscillating network results[3]. At this point, even trying different learning rates did not improve them.

---

[1] Recall from chapter 4.
[2] Refer to experiments 1-5 in appendix A.
[3] See Figure A-4.1 in appendix A.

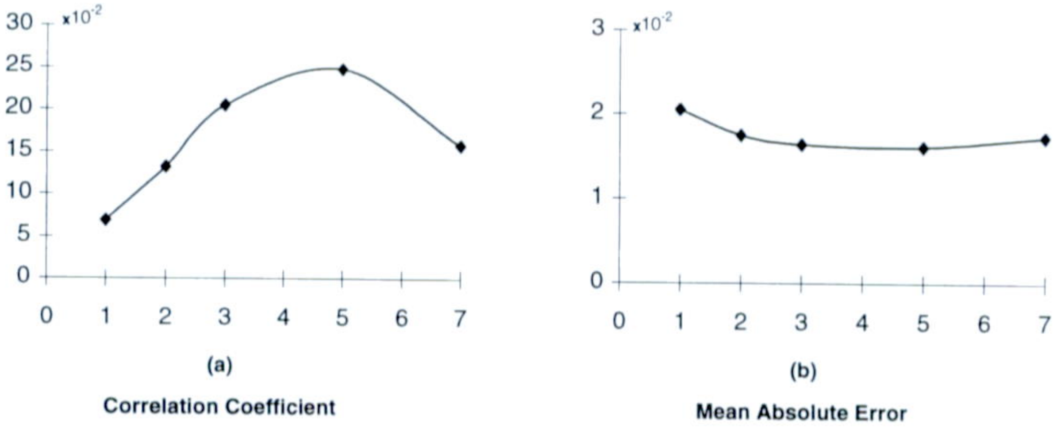


Figure 5.1

Performances of Different LVQ Structures

The intuition from these results was that the input patterns producing the same or very close results are far from each other in the input space in terms of Euclidean distance. If input patterns of a class are scattered over too many regions in the input space, the LVQ is very unlikely to reach a solution unless it has enough PEs to represent all of the regions. Providing such a condition is not always possible. The more scattered the input patterns of a class, the more PEs in the Kohonen layer the LVQ requires. Even so, there is no guarantee that the LVQ will converge to a good solution. Therefore, it was felt that the LVQ type network was likely to fail in learning the strain signal since the number of PEs in the Kohonen layer was subject to and restricted by the hardware (e.g., using 7 PEs per category took a considerable time even in training until 500 epochs).

On the other hand, the initial results with the BPN looked promising. The results obtained from the BPN were consistent with respect to increase in performance (i.e., the correlation between desired and predicted signals was increasing as the number of iterations increased while the mean absolute error in between was decreasing)[4]. Yet, defining some network parameters such as 'initial weight maximum', 'learning method' and 'learning rate' seemed crucial in order to

[4] Refer to experiments 7-9 in appendix A.

stimulate learning. For example, applying 'smoothing' learning method along with the learning rate value of 0.1 and the momentum rate value of 0.8 led the network to get stuck after few iterations[5]. Similarly, giving 'the initial weight maximum' parameter a high value (e.g., 1.0) caused the same situation. Meanwhile, using 'normal' backpropagation learning method along with the learning rate values of 0.1 and/or 0.5 yielded very good results.

Because of the more promising results in the initial experiments, the BPN was chosen for the further experiments.

5.3 Effects of Learning Rate on Performance

The effects of learning rate in theory was mentioned in the preceding chapter[6]. Most of those effects were experienced during the experiments undertaken (e.g., using a value of 1.0 for learning rate did not stimulate the learning process, which meant it was too high).

Figure 5.2

Performances of Two Learning Rates on 300 Points

[5] Refer to experiment 6 in appendix A.

[6] Recall from section 4.3.1.

Using learning rate values of both 0.1 and 0.5 yielded good results in the initial experiments with the two-hidden-layer BPN[7]. In this part, the use of a learning rate value of 0.5 gave better results than those from 0.1. The performances of both rates against the number of iterations on the same structure are shown in **Figure 5.2**.

However, this comparison was less clear when the same network was trained with the 10% of the strain signal (in the major part of the experiments)[8]. In this case, performances of both rates on the training set (not on the validation set) were close to each other (see **Figure 5.3**), although they reached different minima on the error surface. After final improvement attempts on the results (it will be explained later in section 'Effects of Training Mode'), a learning rate value of 0.5 yielded better results on the training set[9]. In contrast, the validation results suggested that the results obtained by means of a learning rate value of 0.1 be better than those of 0.5 based on the experiments with the two-hidden-layer BPN.

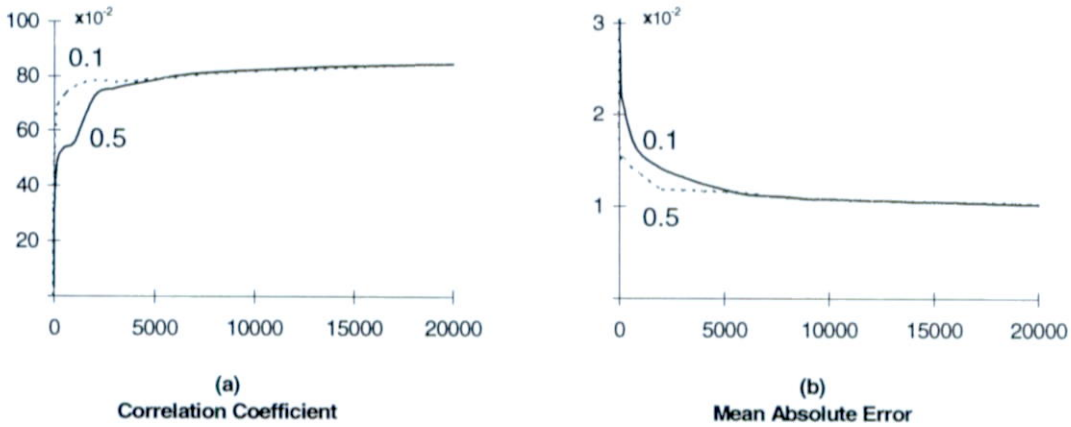


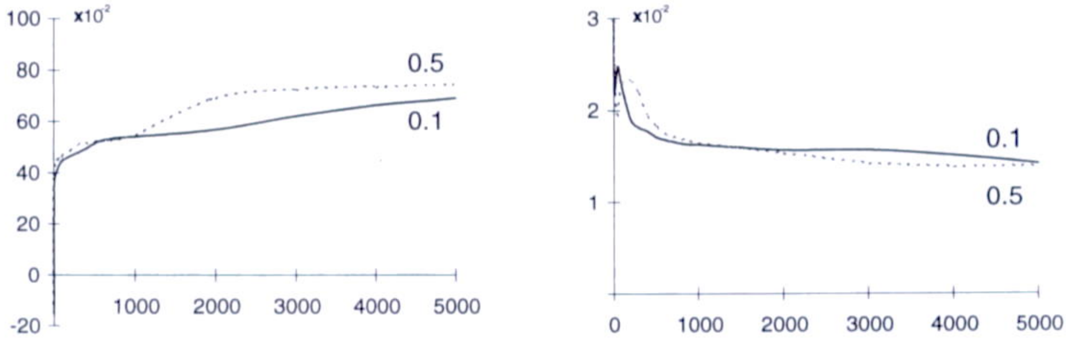
Figure 5.3

Performances of Two Learning Rates
with theTwo-Hidden-Layer BPN

---

[7] Refer to experiments 8 and 9 in appendix A.
[8] Refer to experiments 12a and 13 in appendix A.
[9] Refer to experiments 12b and 13a in appendix A.

In addition to this, the use of a learning rate value of 0.5 on the single-hidden-layer BPN again resulted in faster learning of the training set[10]. **Figure 5.4** shows the learning performances of both rates based on the single-hidden-layer BPN with 40 PEs in the hidden layer.



(a)
Correlation Coefficient

(b)
Mean Absolute Error

Figure 5.4

Performances of Two Learning Rates
with the Single-Hidden-Layer BPN

Another empirical result concerning learning rate was that the performance of a learning rate depended on the number of inputs and their magnitudes. In other words, the learning rate value needed to be chosen considering the number of inputs. The more inputs, the lower the learning rate. For the two-hidden-layer BPN employed, the number of inputs for the first hidden layer PEs was fixed to 21 (i.e., flight parameter data) excluding the bias. It was 20 for the second hidden layer and output layer PEs. So, the numbers of inputs of all PEs were almost the same and using the same learning rate for the hidden and output layers did not cause a problem. More precisely, the learning rate values of both 0.1 and 0.5 produced good results in this case (whereas a learning rate value of 1.0 did not stimulate learning).

[10] Refer to experiments 14 and 15 in appendix A.

On the other hand, the number of inputs of the output layer PE was increased from 40 to 60 when the single-hidden-layer BPN was used so as to examine the effects of different structures (it remained 21 for the hidden layer PEs due to the nature of the problem)[11]. Even, when the number of PEs in the hidden layer was increased to 50, there was still no trouble. Nevertheless, the network got stuck with the learning rate value of 0.5 when it was increased to 60. This problem was overcome by decreasing the learning rate value of the *output layer* (e.g., to 0.1) when the network started learning. In addition, reducing the learning rate value of the hidden layer did not seem to be a good idea, presumably, as there was no change in the numbers of inputs of the hidden layer PEs[12].

5.4 Effects of the Structure

One of the aspects of the network architecture is the activation function (also called *micro-structure*) which can be used to improve the structure.

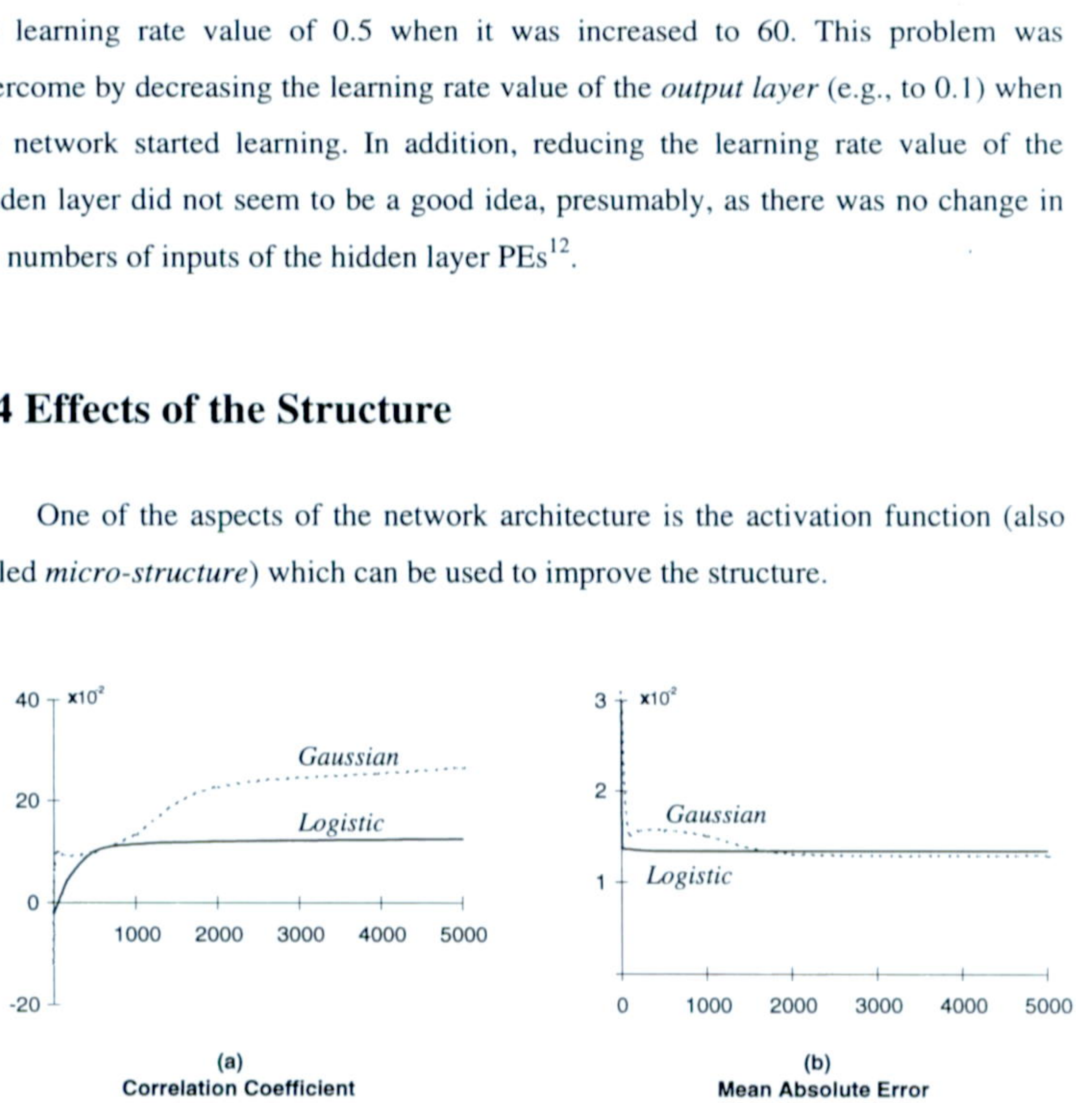


Figure 5.5

Performances of Two Activation Functions

[11] In the hidden layer, 40,41,43,50, and 60 PEs were used respectively. Refer to experiments 14-18.
[12] Refer to experiments 18-20.

Logistic and Gaussian activation functions were used in the initial experiments with the BPN[13]. The performance of these functions on the same structure along with the same learning rate are shown in **Figure 5.5**.

In the above case, a two-hidden-layer BPN was employed. Although one reference suggests that using two hidden layers may be more reasonable when the output needs to be a continuous function of the inputs[14], both the single-hidden-layer BPN and the two-hidden-layer BPN produced a close overall performance in the experiments conducted. The distinctive point was that the performance of the two-hidden-layer BPN both on the training set and the validation set was very close compared to those of single-hidden-layer BPN[15]. Consequently, the single-hidden-layer BPN gave better results on the validation set while the other structure learned the training set more accurately and faster[16].

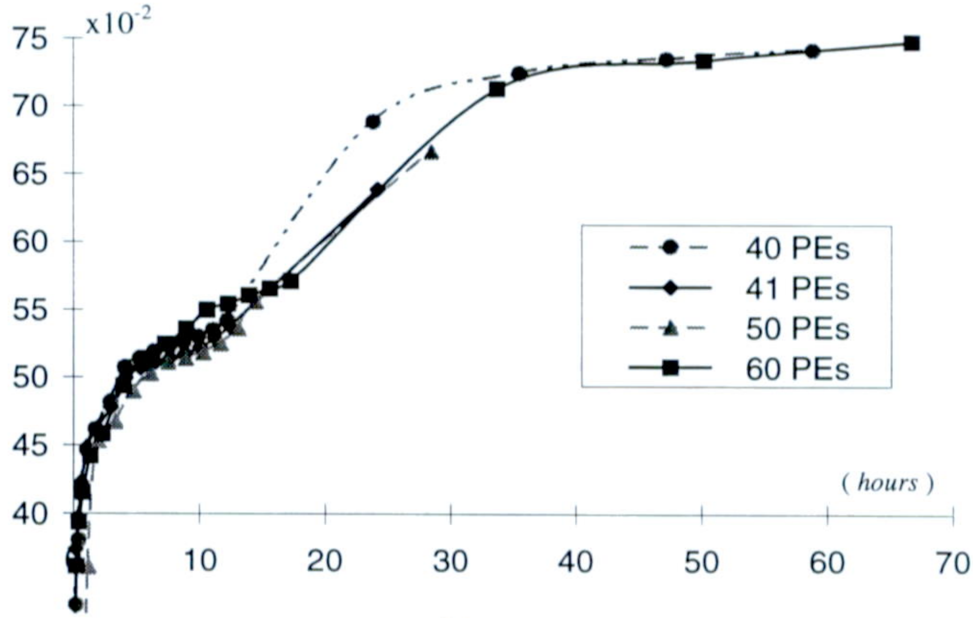
One more point about the structure was the use of different numbers of PEs in the hidden layer of the single-hidden-layer BPN. In this case, the empirical result concerning the learning rate was indicated earlier. Performances of these structures[17] against time (not iterations) depending upon the *PC Model 3* are shown in **Figure 5.6**.

5.5 Effects of Training Mode

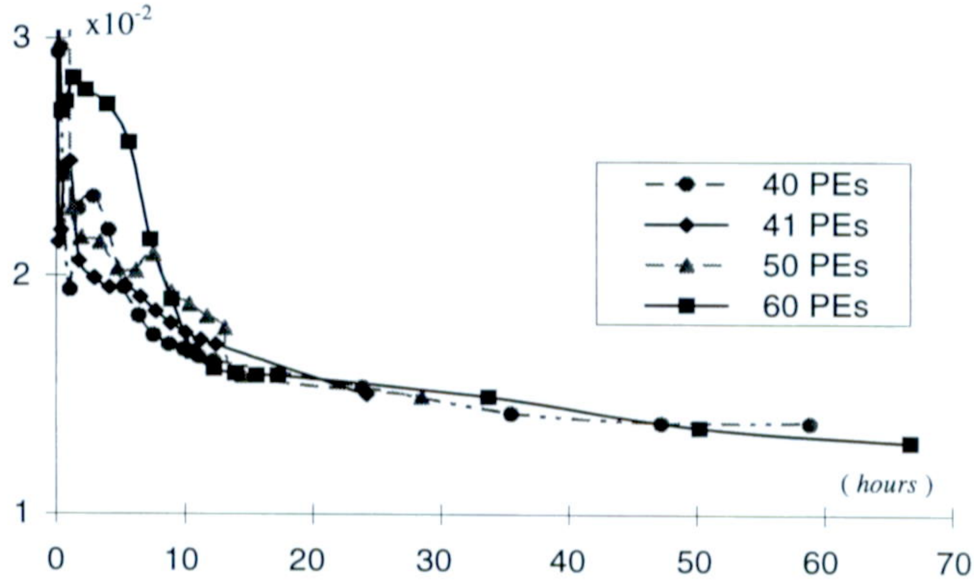
The reason for the use of 'fast train' mode and its optimisation for the major part of the experiments were explained in the preceding chapter[18]. Performances of 'fast training' and 'normal training' in time scale (depending upon the *PC Model 3*) are shown in **Figure 5.7**. Clearly, the performance obtained from 'normal training' is better with respect to the number of iterations[19]. Nonetheless, reaching a high number

[13] Refer to experiments 7 and 8 in appendix A.
[14] Recall from chapter 4, section 'Parameters of MBPN'.
[15] This issue will be discussed in detail in section 5.6.
[16] Refer to experiments 12a,13, and 21 in appendix A.
[17] Refer to experiments 15, 16, 17,18, and 21 in appendix A.
[18] Recall from chapter 4, section 'Training Mode Selection'.
[19] Refer to experiments 10 and 12a in appendix A.

of iteration (e.g., 20,000) and at the same time having faster learning with 'normal train' mode, was impossible as indicated in the preceding chapter.



(a)
Correlation Coefficient



(b)
Mean Absolute Error

Figure 5.6

Performances of Different BPN Structures

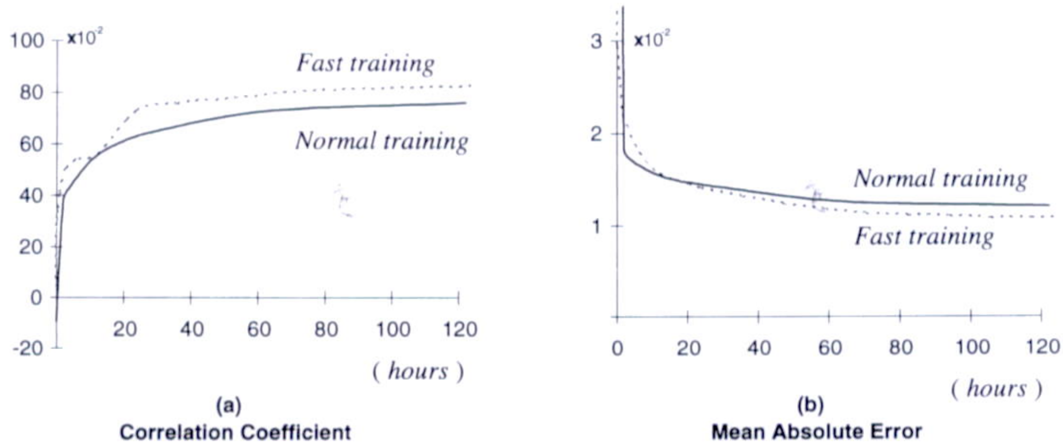


Figure 5.7

Performances of Training Methods

As can be seen from **Figure 5.7**, applying the 'fast train' mode made a substantial contribution in terms of reducing the training time (i.e., obtaining better results in a particular time period)[20]. It was also noticed that applying the 'fast train' mode ultimately caused the network to reach an even worse minimum on the error surface compared to the 'normal train' mode. In this case, applying 'normal training' in a certain number of iterations (e.g., about 2,000) improved the results in terms of performance and training time when the learning of the network considerably slowed down in 'fast training'[21].

5.6 The Ability of Generalisation

Figure 5.8 shows the development of learning process and generalisation ability of a network theoretically with respect to error between desired and actual values, using a representative training set (*Hassoun 1995*).

[20] Approximately, time for 100 iterations in 'normal training' was equal to time for 1,000 iterations in 'fast training'.

[21] Refer to experiments 12b, 13a, and 12a and compare their results with those of 12a, 13 and 21 in appendix A respectively.

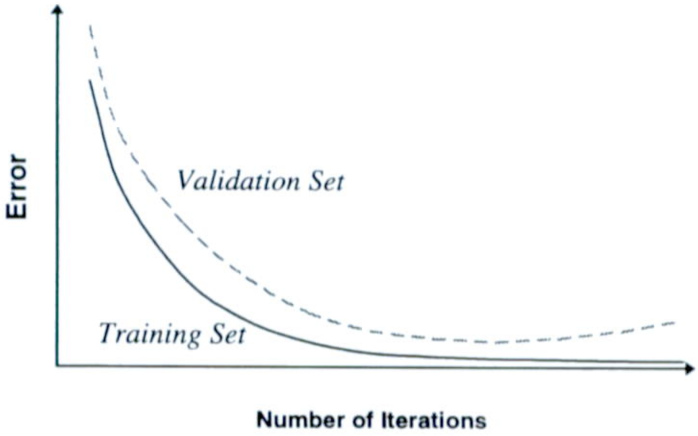


Figure 5.8

Development of Learning Process *in Theory*

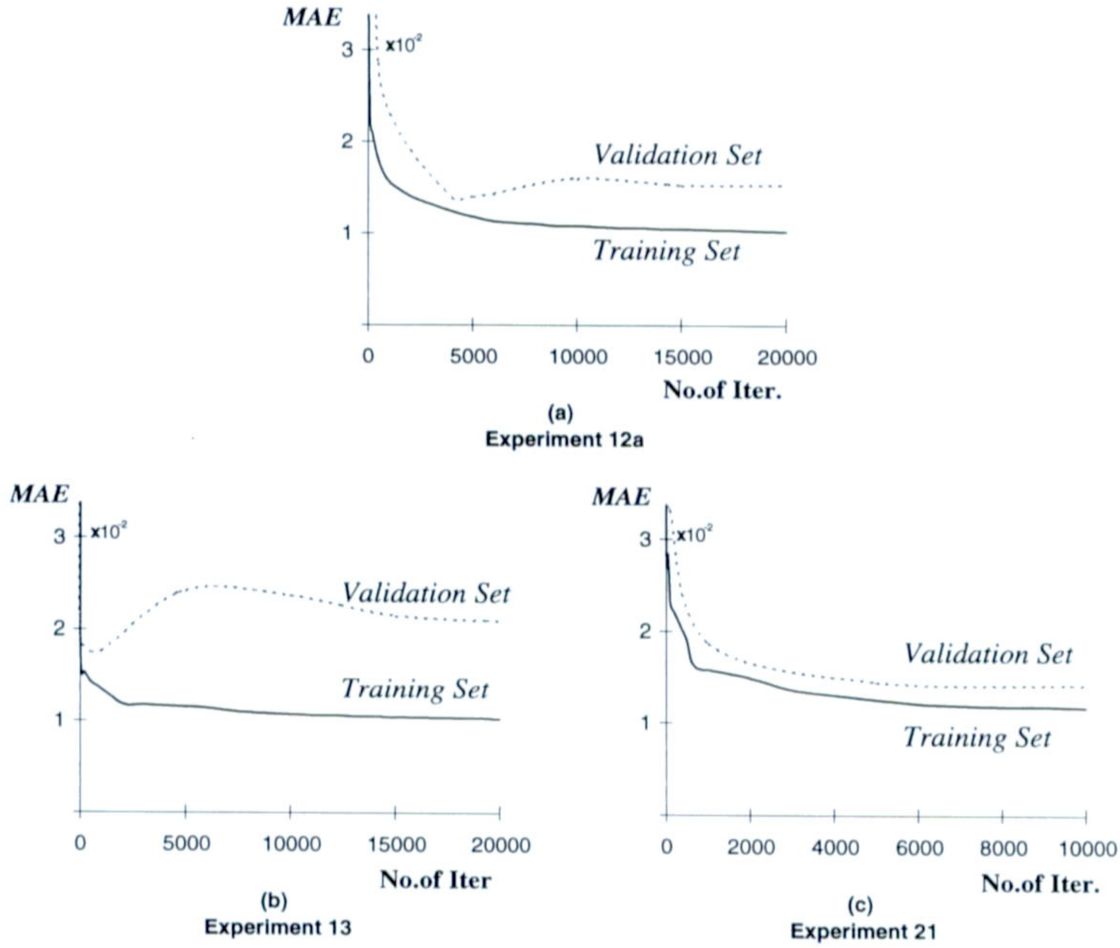


Figure 5.9

Development of Learning *in the Experiments*

For some of the experiments, this situation (based on the mean absolute error between the desired outputs and the network outputs) was realised as shown in **Figure 5.9**. It is clear from **Figure 5.9** that the graph labelled **(c)** bears the closest resemblance to the one shown in **Figure 5.8**. Graph **(c)** is derived from the experiment in which the single-hidden-layer BPN with 60 PEs in the hidden layer was employed. Hence, it is not surprising that the best performance on the validation set was obtained from this structure.

5.7 Representative Data Selection

The importance of the training data was emphasised in the preceding chapter. This factor was also experienced during the experiments conducted. The network which did well on the first 300 points of the strain signal yielded poor results when it was tested on 10% of the strain signal let alone on the whole signal[22]. In other words, it behaved as if it were almost not trained on 10% of the signal although this 10% of the strain signal comprised some of the first 300 points. The first 300 points were unable to represent either 10% of the strain signal or the whole signal. The purpose of using the first 300 points was to seek promising networks which might learn such a signal.

On the other hand, it was realised that 10% of the strain signal could represent the whole set to a certain extent[23]. In addition to this, the results suggested that the training set (i.e., 10% of the strain signal) used in the major part of the experiments may not be enough to represent the whole signal more accurately. Because, a better result on the validation set could somehow be obtained with the current network[24]. Yet, representative nature of the training set seemed to have different strengths on the different structures[25].

---

[22] Refer to experiment 10 in appendix A.

[23] Refer to experiments 10, 12a, 12b, 13, 13a, 21, and 21a in appendix A.

[24] Refer to experiment 12c in appendix A.

[25] Recall from section 'Effects of the Structure'.

5.8 The Best Result So Far

Different approaches suggest that different networks give the best result so far. In terms of learning the training set, the best result was a correlation value of about 88% and a MAE value of 0.0092. Nevertheless, performance of this network on the whole set was not the best[26]. So far, the best performance on the whole set was obtained from a single-hidden-layer BPN with 60 PEs in the hidden layer[27]. It was a correlation value of about 73% and a MAE value of 0.0135.

5.9 Summary

Although there were some other experiments undertaken as a starting point[28], those which are given in appendix A started to be conducted using two different network types, namely LVQ and BPN. First, their learning performances on a small part of the strain signal were observed. Experiments 1-9a were undertaken for this purpose. Then, the major part of the experiments continued with the more promising (i.e., BPN) type. In this part of the study, a number of experiments were conducted not only to find a better structure and improve the results as much as possible but also to explore the effects of network parameters. Experiments 10-21a were undertaken for this purpose.

Experiments 1-5 involve LVQ networks with different structures whereas experiments 6-13b involve a two-hidden-layer BPN and experiments 14-21a involve single-layer BPNs with different structure.

[26] Refer to experiment 13a in appendix A.
[27] Refer to experiment 21a in appendix A.
[28] In order to get acquainted with the package and ANN paradigms as well as the helicopter parameter data itself.

Chapter 6

CONCLUSIONS

6.1 Introduction

In this chapter, the results of the experimental procedures and the overall performance of the system of prediction of helicopter strain are discussed so as to draw a conclusion from the dissertation. First, the modelling of the system is revisited and evaluated. Secondly, potential further developments of the work are presented. Finally, the results from the study are assessed to conclude how well the aims are met.

6.2 Discussion of Modelling

In order to evaluate the effects of the model developed on the results obtained, two issues need to be addressed: The flight parameters used in the model and the network types employed during the experiments.

6.2.1 Flight Parameters

During experiments, ANNs were constructed with 21 inputs to accept time histories for 21 *Lynx* flight parameters. These parameters were selected from the 48 available channels on the basis of the domain knowledge. Of these parameters, 'outside air temperature' was also ignored due to having a virtually constant value in the range used.

Firstly, it was felt that either the training set or the input parameters used in the work caused a limitation on the accuracy that could be achieved with ANNs. Therefore, it may be necessary either to modify this model or to use different training sets which will comprise potential values of influential parameters (including 'outside air temperature'). On the other hand, the critical points in a strain signal in terms of fatigue is the distance between successive turning points. So, the peak-to-peak ranges should be predicted as accurately as possible in order to be able to calculate the fatigue from a predicted strain channel to any degree of accuracy and this point must always be considered whilst modelling the system and evaluating the results.

Another point is that the interpolation of some input parameters (17 out of 21) might also have influenced the accuracy of prediction. Ideally, all of them should have been sampled at the same rate in the original flights. Hence, they could have been presented to the networks without pre-processing.

This study together with the previous work suggests that if the strain to be predicted correlates well with the inputs, the ANN model (at least BPN type network) is very likely to give faster and better results. This is the reason, or at least one of the reasons, why previous attempts were very successful in predicting the other two strain signals, which have a correlation value of over 50% when compared with at least one of the inputs. These previous attempts, however, were poor on the prediction of the strain used in this study, which has correlation values of below 10% when compared with all the inputs. So, they gave a correlation value of over 90% for the other two signals while it was below 50% for the strain predicted here. In this study, the problem was overcome by employing more hidden PEs along with another hidden layer and/or a different activation function. In other words, the lower correlations between the inputs and the output, the more the network requires a complex structure together with more training time.

6.2.2 Network Type

ANNs are a very useful tool for building a model of the relationship between a set of input and output patterns. However, there is always a risk that the generalisation built by means of the training data set may not be much use in practice if it is not the only possible or desired generalisation (any generalisation is just one of many possible generalisations). Yet, it often turns out that the performance of the model whilst it is generalising is sufficient for it to be useful.

Nevertheless, the main issue in the view of ANNs is to find the best architecture that the ANN model will fit because it is not fully understood in theory and this research is no exception. Eventually, this research once again proved the robustness of BPN even though it may suffer from reaching local minima and slow convergence. The initial work, using LVQ and BPN types, suggested that the BPN be more robust when a complex mapping of inputs and the outputs (as here) needs to be found. Of course, it is worth employing different ANN paradigms and comparing all the results obtained from them. But, one intuition from this work is that the networks having *feature-based* approach are likely to give more accurate results for this problem than those of *prototype-based*.

6.3 Further Work

Despite achieving much better results on the prediction of the strain used in this study when compared to previous attempts, the system still needs to be tuned to give even more accurate results. Assuming it has been taught successfully to predict strain from easily measured parameters, it must still be able to cope with the situations very different from its training.

In order to progress this work towards a system completely capable of being implemented to improve both the safety and the economics of the helicopter as a mode of transport the following is recommended:

1. Input parameters and their presentation to the network can be reconsidered to ensure that the ANN system is able to learn the training set easily and thoroughly and also make an accurate generalisation.

2. Employing other network types and/or different learning methods may be investigated to compare and find a better structure and teaching mechanism for this type of problem.

3. Combining the ANN approach with other AI methods and forming a hybrid system may lead to a better system. Genetic Algorithms (GAs) may serve for this purpose but the possibility of a function approximation approach to this problem by GAs was still undergoing study when this work was finished.

4. In order to verify the accuracy of the method it will be necessary to apply it to a wider range of manoeuvres, and hence a larger data set. Needless to say the hardware and software must be able to tackle huge amounts of data.

5. The possibility of transferring the technique to other types of helicopter is also worth investigating.

6.4 Conclusion

This work has shown that by means of an ANN model it is possible to predict strains even in the rotating parts of helicopters given more easily measured parameters at least to a reasonable degree of accuracy. It has also proved that the failure on the prediction of some strain channels in the previous attempts can be overcome by using different ANN structures and/or developing different strategies to deal with some of the problems associated with using ANNs for this type of application. By employing more hidden PEs than previous attempts along with another hidden layer and/or a different activation function, it has been possible to increase the correlation between actual and predicted strain signals from below 50% to 88% on the training set and to about 73% on the validation set.

Yet there is still much work to be done, not only to obtain the results in sufficient accuracy, but also to make the technique robust enough for application in commercial settings. It is the author's belief that further development of the work will contribute to the safety and economics of the helicopter as it provides the accurate condition monitoring of the helicopter parts.

REFERENCES

Aleksander, I., Morton, H., *An Introduction to Neural Computing*, T.J. Press Inc., 1990

Bateman, R.D., *An Investigation into the Problems of Helicopter Maintenance and Maintenance Programmes*, MSc Thesis, College of Aeronautics, Cranfield Institute of Technology, 1982

Buller, R.G., *Fatigue Crack Growth Rates under Helicopter Load Spectra*, MSc Thesis, College of Aeronautics, Cranfield University, 1994

Civil Aviation Authority, *Review of Helicopter Airworthiness*, - Report of the Helicopter Airworthiness Review Panel (HARP) CAP 491, June 1984

Crawford, C.C., Carlson, R.L., Bates, P.R., *Damage Tolerance Analysis for Rotorcraft - What the Issues are*, European Rotorcraft 17th Forum, Berlin, Germany, September 1991

DeSieno, D., *Adding a Conscience to Competitive Learning*, Proceedings of the 2nd Annual IEEE Int. Conf. on NNs, Vol. 1, 1988

Galindo, P.L., *The Competitive Forward-Backward Algorithm (CFB)*, IEE Fourth Int. Conf. on ANNs, pp. 82-85, 26-28 June 1995

Hassoun, M.H., *Fundamentals of Artificial Neural Networks*, the MIT Press, 1995

Hecht-Nielsen, R., *Neurocomputing*, Addison-Wesley Publishing Company, 1990

Hecht-Nielsen, R., *Theory of the Backpropagation Neural Network*, in 'Neural Networks for Perception' (Eds. Maren,A.J.; Harston, C.T.; and Pop,R.M.), Vol. 2, Academic Press Inc., 1992

Helliwell, I.S., Turega, M.A., Cottis, R.A., *Accountability of Neural Networks Trained on "Real World Data"*, IEE Fourth Int. Conf. on ANNs, pp. 218-222, 26-28 June 1995

Hertz, J., Krogh, A., Palmer R.G., *Introduction to the Theory of Neural Computing*, Addison-Wesley Publishing Company, 1991

Hill, K.W.M., *The Application of Usage Monitoring to Military Helicopters*, MSc Thesis, College of Aeronautics, Cranfield University, 1994

Hill, K.W.M., Hudson, B., Irving, P., Vella, A.D., *Loading Spectra, Usage Monitoring and Prediction of Fatigue Damage in Helicopters*, Proceedings of the 18th Symposium of the International Committee on Aeronautical Fatigue, Vol.1, 3-5 May 1995

Jones, D., Franklin, S.P., *Choosing a Network: Matching the Back-propagation Network*, in 'Handbook of Neural Computing Applications' (Eds. Maren,A.J.; Harston,C.T.; and Pop,R.M.), Academic Press Inc., 1990

Kohonen, T., *Self-Organization and Associative Memory*, Springer Verlag, Berlin, 1989

Kohonen, T., *Improved Versions of Learning Vector Quantization*, IEEE Proceedings of the Int. Joint Conf. on NNs, Vol. 1, pp. 545-550, 1990

Leahy, A.M., *Helicopter Fatigue Monitoring - The Way Ahead*, Aerotech 94 HUMS Seminar, Birmingham, January 1994

Lisboa, P.G.J., *Neural Networks - Current Applications*, Chapman & Hall, 1992

Lison, A.C., *The Prediction of Helicopter Airframe Load Spectra from Flight Parameter Data Using ANNs*, MSc Thesis, College of Aeronautics, Cranfield University, 1994

Mammone, R.J., Zeevi, Y., *Neural Networks Theory and Applications*, Academic Press Inc., 1991

Maren, A.J., *Multilayer Feedforward Neural Networks I: Delta Rule Learning*, in 'Handbook of Neural Computing Applications' (Eds. Maren,A.J.; Harston,C.T.; and Pop,R.M.), Academic Press Inc., 1990a

Maren, A.J., *Vector-Matching Networks*, in 'Handbook of Neural Computing Applications' (Eds. Maren,A.J.; Harston, C.T.; and Pop,R.M.), Academic Press Inc., 1990b

Maren, A.J., Jones, D., Franklin, S.P., *Configuring and Optimizing the Back-propagation Network*, in 'Handbook of Neural Computing Applications' (Eds. Maren,A.J.; Harston,C.T.; and Pop,R.M.), Academic Press Inc., 1990

Matteson, T.D., *Do We Really Understand Maintenance?*, Aircraft Design Systems and Operations Meeting, AIAA, New York, 14-16 September 1987

Meade, A.J., *An Applications of Artificial Neural Networks to Experimental Data Approximation*, 31st Aerospace Sciences Meeting & Exhibit, AIAA, Nevada, 11-14 January 1993

Minsky, M.,
Papert, S.,
Perceptrons: An Introduction to Computational Geometry, MIT Press, 1969

MOD Procurement Executive (UK),
Defence Standard 00-970 Volume 2 Chapter 727, Health and Usage Monitoring Systems, 1989

Mrazova, I.,
The Robustness of BP-Networks, IEE Fourth Int. Conf. on ANNs, pp. 234-239, 26-28 June 1995

Murata, J.,
A Structure Designing Method for Feedforward and Recurrent Neural Networks, IEE Fourth Int. Conf. on ANNs, pp. 358-363, 26-28 June 1995

Pham, D.T.,
Liu, X.,
Neural Networks for Identification, Prediction and Control, Springer-Verlag London Limited, 1995

Picton, P.D.,
Introduction to Neural Networks, the Macmillan Press, 1994

Rosenblatt, F.,
The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Psychological Review, 65, pp. 386-408, 1958
and reprinted in 'Neurocomputing' (Eds. Anderson, J. and Rosenfeld,E) , 1988

Royal Aeronautical Society
Helicopter Airworthiness, December, 1984

Royal Aeronautical Society
Helicopter Airworthiness in the 1990s, Health and Usage Monitoring Systems - Experience and Applications, One Day Conference, 29 Nov. 1990

Rumelhart, D.E.,
Hinton, G.E.,
Williams, R.J.,
Learning Internal Representations by Error Propagation, in 'Parallel Distributed Processing' (Eds. Rumelhart,D. and McClelland,J.), the MIT Press, 1986

Rumelhart, D.E., McClelland, J.L., *PDP Models and General Issues in Cognitive Science*, in 'Parallel Distributed Processing' (Eds. Rumelhart,D. and McClelland,J.), the MIT Press, 1986

Stone, G.O., *An Analysis of the Delta Rule and The Learning of Statistical Associations*, in 'Parallel Distributed Processing' Vol. 1. (Eds. Rumelhart,D. and McClelland,J.), the MIT Press, 1986

Stone, J.V., Thornton, C.J., *Can Artificial Neural Networks Discover Useful Regularities*, IEE Fourth Int. Conf. on ANNs, pp. 201-205, 26-28 June 1995

Taitel, H.J., Danai, K., Gauthier, D., *Helicopter Track and Balance with Artificial Neural Nets*, Intelligent Control Systems, ASME, DSC-Vol. 45, pp. 47-54, 1992

Turban, O., *Effect of Load Spectrum Modification on the Damage Tolerance of Aluminium Alloys Used in Aircraft and Helicopter Components*, MSc Thesis, SIMS, Cranfield Institute of Technology, 1992

Veelenturf, L.P.J., *Analysis and Applications of Artificial Neural Networks*, Prentice Hall, 1995

Vella, A.D., Hudson, B., Irving, P., *Estimating Helicopter Strain Using a Neural Network Approach*, in 'Neural Networks and their Applications' (Eds. Taylor, J.G. and Wiley J.), 1996

Wendlandt, N., *Engine Condition Monitoring Applied to Life Usage of Military Aero Engines*, MSc Thesis, SME, Cranfield Institute of Technology, 1990

Western, W.J.H., Boulding, N.J., *UK Military Experience and Viewpoint*, in 'Helicopter Airworthiness in the 1990s', One Day Conference, Royal Aeronautical Society, 29 Nov. 1990

Williams, R.J., *The Logic of Activation Functions*, in 'Parallel Distributed Processing' Vol. 1. (Eds. Rumelhart,D. and McClelland,J.), the MIT Press, 1986

Appendix **A**

EXPERIMENTS

Summary

Experiments 1-5 : Learning Vector Quantization (LVQ) networks which have 1, 2, 3, 5, and 7 PEs in the hidden layer were employed so as to reproduce a small part of the strain signal.

Experiments 6-9a : A two-hidden layer backpropagation network (BPN) having 20 PEs in each hidden layer was employed along with different activation functions, learning methods and learning rates so as to reproduce a small part of the strain signal.

Experiments 10-13b : A two-hidden layer BPN having 20 PEs in each hidden layer was employed along with different learning rates and training methods in order to find an ANN model of the strain.

Experiments 14-21a : Single-hidden layer BPNs having different numbers of PEs in the hidden layer were employed along with different learning rates and training methods in order to find an ANN model of the strain.

Note :

1. In the graphical figures concerning the correlation coefficients, or mean absolute errors, or maximum errors, the X-axis represents the number of iterations on the training set while the Y-axis represents the one indicated in the caption. In these graphical figures, the correlation coefficients, mean absolute errors and maximum errors are scaled up by 100 to avoid clumsy numbers on the Y-axis (e.g., 0.0150 is pointed as 1.5).

2. For figures showing the first 300 points of predicted and actual strain together, it was not possible to use more accurate legends due to the software by which the figures were drawn. In all cases, the predicted strain curve is heavier (darker) than the actual strain curve.

Experiment Definition

No.	: 1
Directory	: LVQ\TEST1
Tool	: HNC Explore Net
Applications	: *LTrain* for training
	LTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

Paradigm	: Learning Vector Quantization (*LVQ*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Categories	: 100
Kohonen PE p. Ctg.	: 1

Training Parameters

Learning Rate (α)	: 0.05
Learning Rate (β)	: 0.05
Learning Rate (γ)	: 0.05
Bias Coefficient (c)	: 0.5303
Win Freq. Coeff. (b)	: 0.1
LVQ2 Enable	: OFF
Random Seed	: 5
Initial Weight Min.	: 0.0
Initial Weight Max.	: 0.0

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)
0	0.00	4.19
5	- 16.34	1.39
10	- 1.04	1.46
20	15.00	1.56
50	- 4.08	1.63
100	13.55	1.55
200	2.26	1.88
500	6.89	2.05

Table A-1.1
Experiment Results on *Training Set*

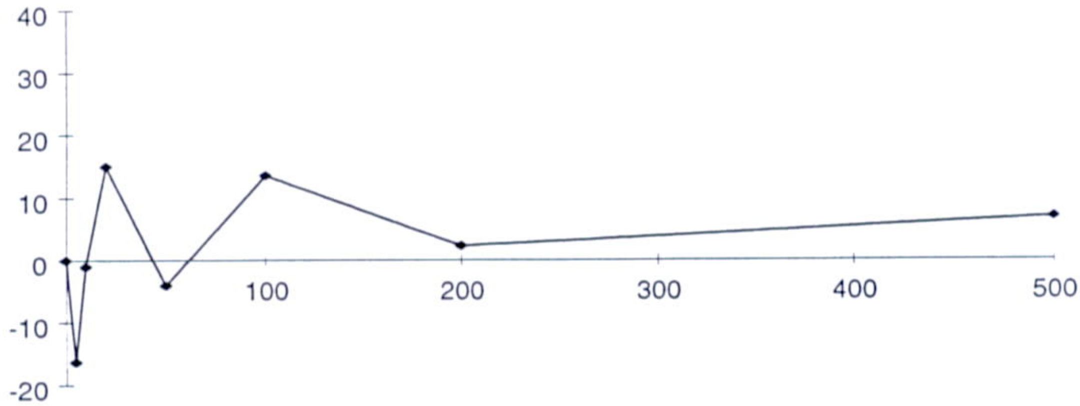


Figure A-1.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

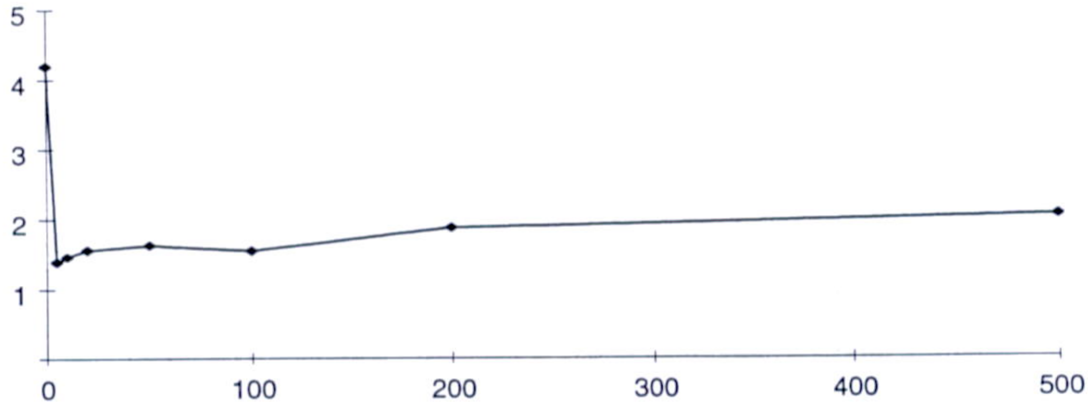


Figure A-1.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

Experiment Definition

No.	: 2
Directory	: LVQ\TEST2
Tool	: HNC Explore Net
Applications	: *LTrain* for training
	LTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

Paradigm	: Learning Vector Quantization (*LVQ*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Categories	: 100
Kohonen PE p. Ctg.	: 2

Training Parameters

Learning Rate (α)	: 0.05
Learning Rate (β)	: 0.05
Learning Rate (γ)	: 0.05
Bias Coefficient (c)	: 0.5303
Win Freq. Coeff. (b)	: 0.1
LVQ2 Enable	: OFF
Random Seed	: 5
Initial Weight Min.	: 0.0
Initial Weight Max.	: 0.0

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)
0	0.00	4.19
5	- 11.12	1.53
10	- 8.09	1.39
20	- 6.26	1.41
50	12.61	1.51
100	- 8.07	2.06
200	15.83	1.85
500	13.21	1.75

Table A-2.1
Experiment Results on *Training Set*

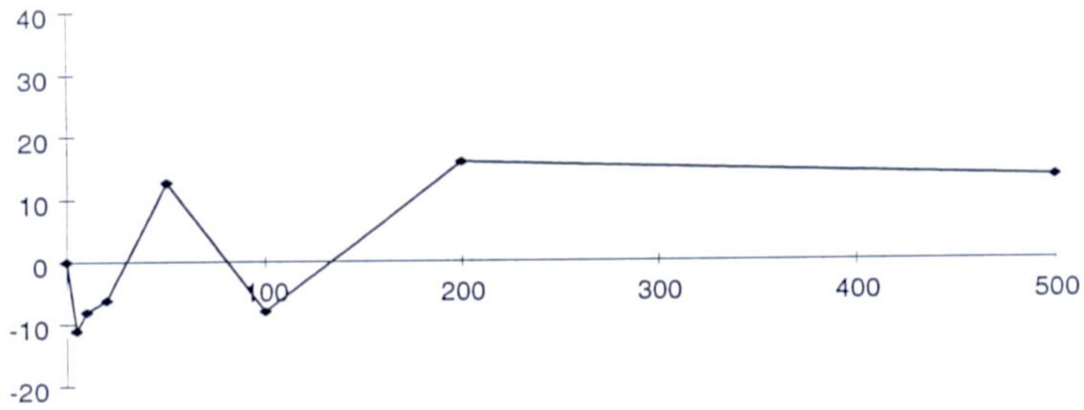


Figure A-2.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

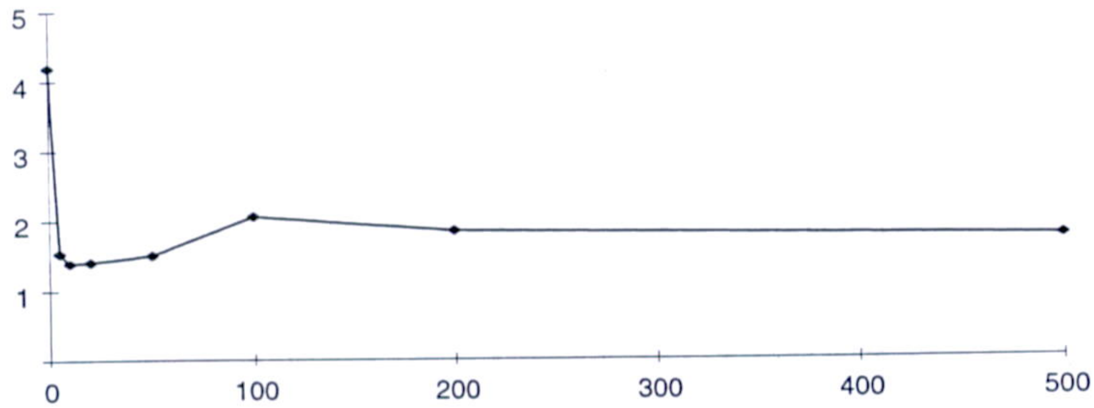


Figure A-2.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

Experiment Definition

No.	: 3
Directory	: LVQ\TEST3
Tool	: HNC Explore Net
Applications	: *LTrain* for training
	LTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

Paradigm	: Learning Vector Quantization (*LVQ*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Categories	: 100
Kohonen PE p. Ctg.	: 3

Training Parameters

Learning Rate (α)	: 0.05
Learning Rate (β)	: 0.05
Learning Rate (γ)	: 0.05
Bias Coefficient (c)	: 0.5303
Win Freq. Coeff. (b)	: 0.1
LVQ2 Enable	: OFF
Random Seed	: 5
Initial Weight Min.	: 0.0
Initial Weight Max.	: 0.0

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)
0	0.00	4.19
5	12.61	1.70
10	- 9.24	1.40
20	8.10	1.47
50	0.75	2.01
100	- 8.30	2.00
200	12.65	1.68
500	20.49	1.64

Table A-3.1
Experiment Results on *Training Set*

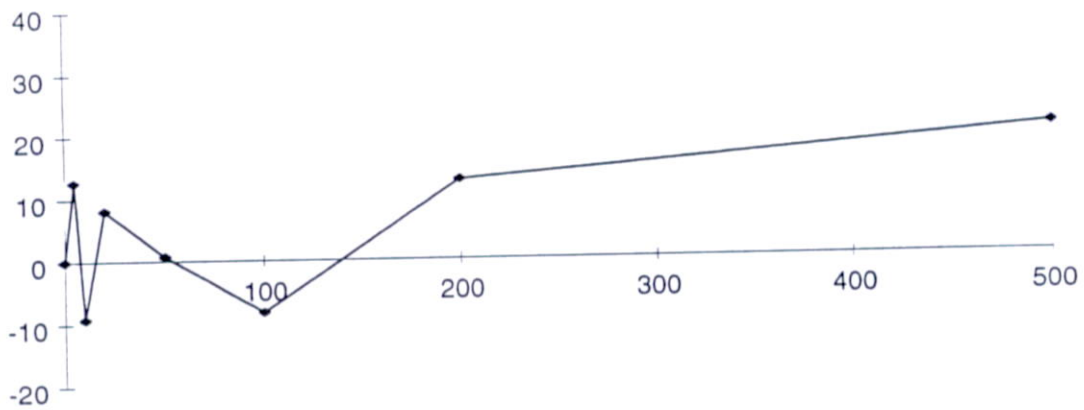


Figure A-3.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

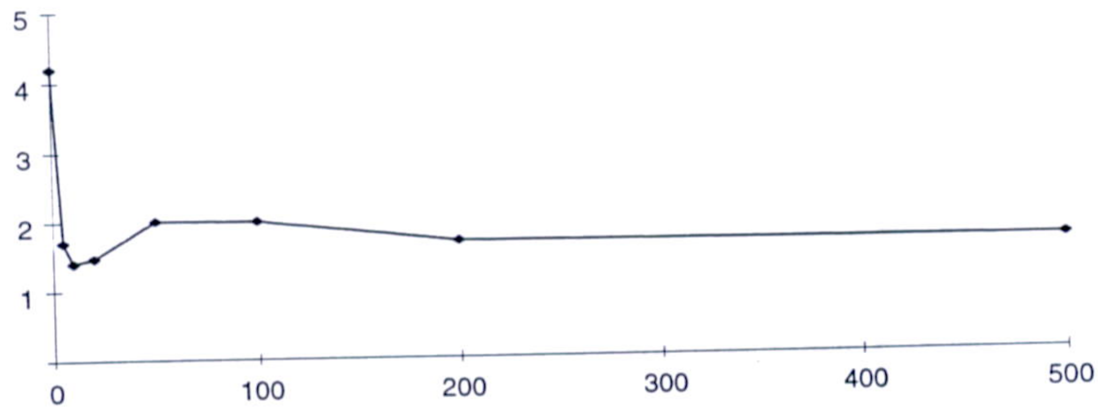


Figure A-3.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

Experiment Definition

No.	: 4
Directory	: LVQ\TEST4
Tool	: HNC Explore Net
Applications	: *LTrain* for training
	LTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

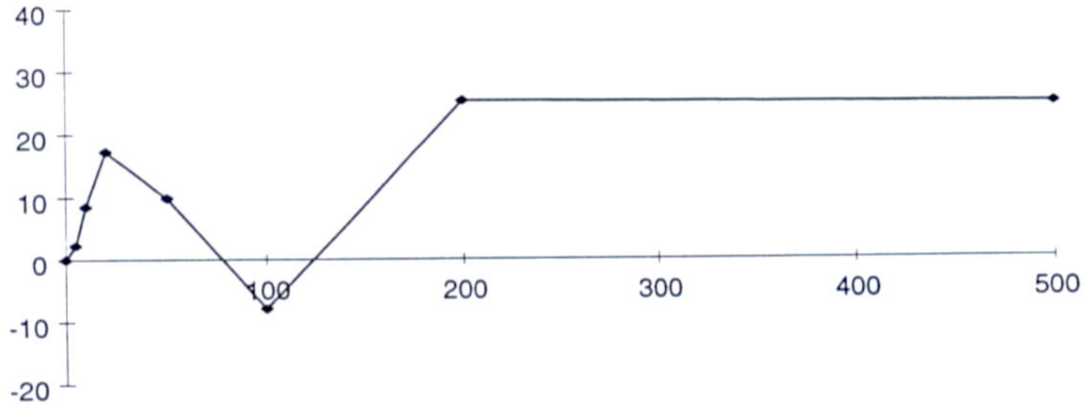
Paradigm	: Learning Vector Quantization (*LVQ*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Categories	: 100
Kohonen PE p. Ctg.	: 5

Training Parameters

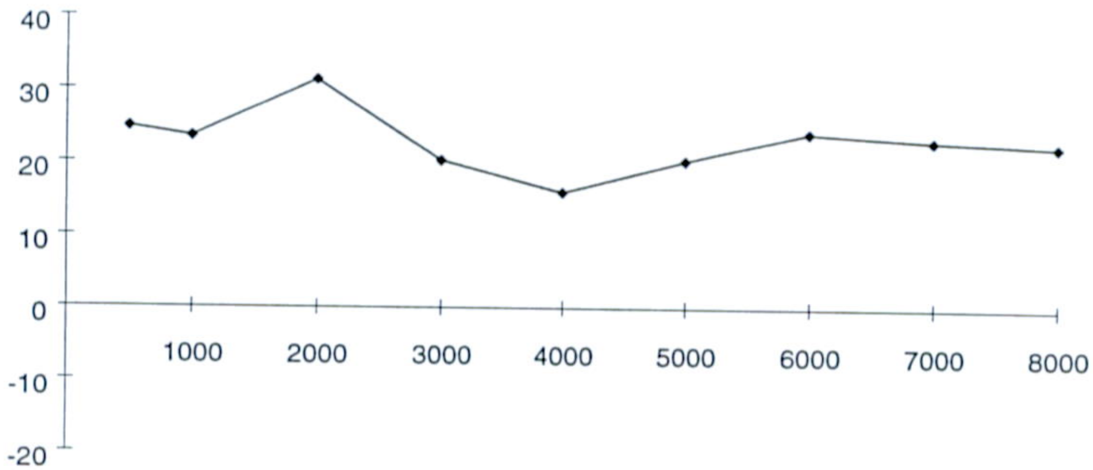
Learning Rate (α)	: 0.05
Learning Rate (β)	: 0.05
Learning Rate (γ)	: 0.05
Bias Coefficient (c)	: 0.5303
Win Freq. Coeff. (b)	: 0.1
LVQ2 Enable	: OFF
Random Seed	: 5
Initial Weight Min.	: 0.0
Initial Weight Max.	: 0.0

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)
0	0.00	4.19
5	2.27	1.34
10	8.53	1.35
20	17.25	1.37
50	9.85	2.07
100	- 7.92	2.37
200	25.30	1.59
500	24.79	1.61
1000	23.51	1.77
2000	31.30	1.56
3000	20.25	1.81
4000	15.95	1.74
5000	20.22	1.63
6000	24.12	1.55
7000	23.08	1.65
8000	22.48	1.66

Table A-4.1
Experiment Results on *Training Set*



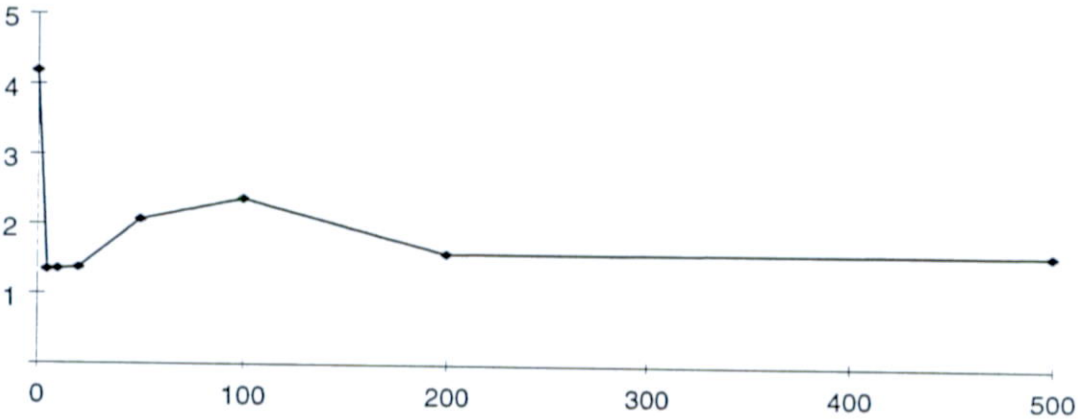
(a)



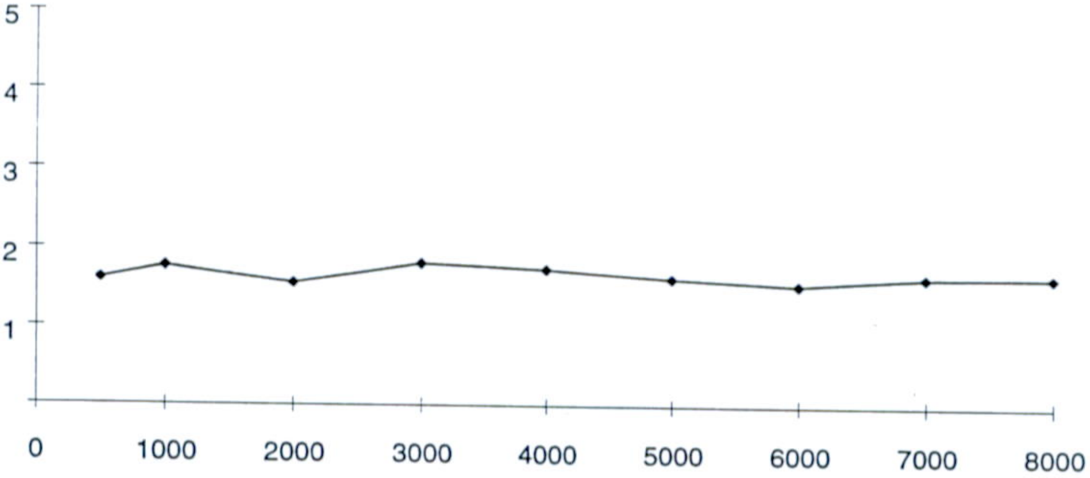
(b)

Figure A-4.1

Correlation Coefficient Vs Number of Iterations for *Training Set*



(a)



(b)

Figure A-4.2

Mean Absolute Error Vs Number of Iterations for *Training Set*

Experiment Definition

No.	: 5
Directory	: LVQ\TEST5
Tool	: HNC Explore Net
Applications	: *LTrain* for training
	LTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

Paradigm	: Learning Vector Quantization (*LVQ*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Categories	: 100
Kohonen PE p. Ctg.	: 7

Training Parameters

Learning Rate (α)	: 0.05
Learning Rate (β)	: 0.05
Learning Rate (γ)	: 0.05
Bias Coefficient (c)	: 0.5303
Win Freq. Coeff. (b)	: 0.1
LVQ2 Enable	: OFF
Random Seed	: 5
Initial Weight Min.	: 0.0
Initial Weight Max.	: 0.0

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)
0	0.00	4.19
5	2.27	1.34
10	- 8.09	1.39
20	6.29	1.75
50	7.39	1.64
100	- 4.34	2.47
200	18.98	1.78
500	15.78	1.73

Table A-5.1
Experiment Results on *Training Set*

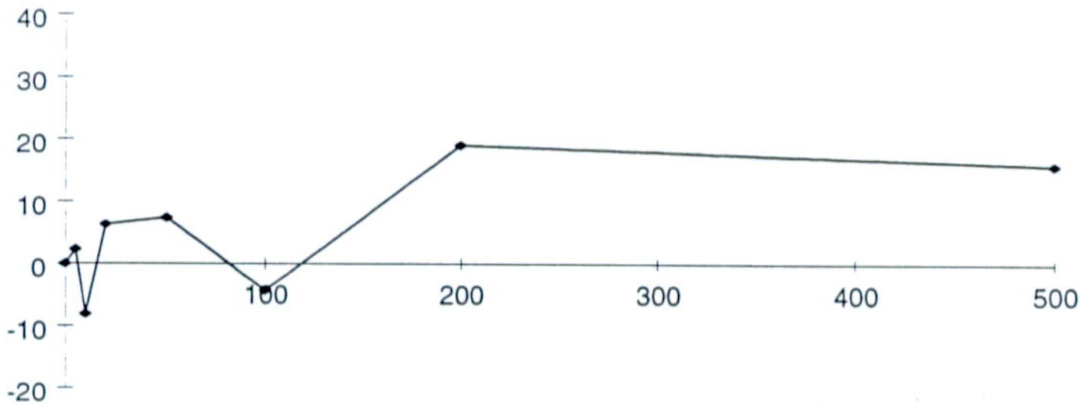


Figure A-5.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

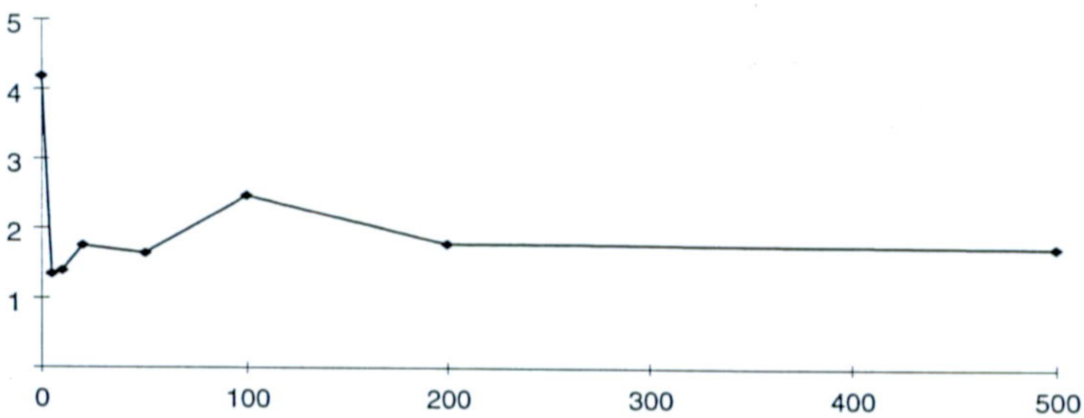


Figure A-5.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

Experiment Definition

No.	: 6
Directory	: MBPN\TEST6
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Smoothing
Learning Rate (α)	: 0.1
Momentum (β)	: 0.8
Batch Size	: 10
Activation Function	: Logistic
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 0
Initial Weight Max.	: 0.1
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
0	0.00	28.10	32.32
5	0.00	1.36	4.49
10	- 6.73	1.36	4.49
20	0.00	1.36	4.49
50	- 2.47	1.36	4.49
100	- 4.66	1.36	4.49
200	0.00	1.36	4.49

Table A-6.1
Experiment Results on *Training Set*

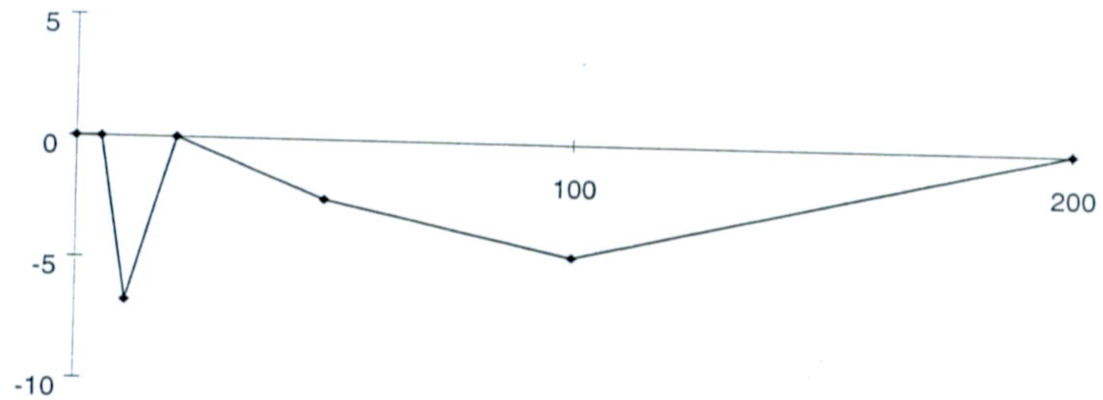


Figure A-6.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

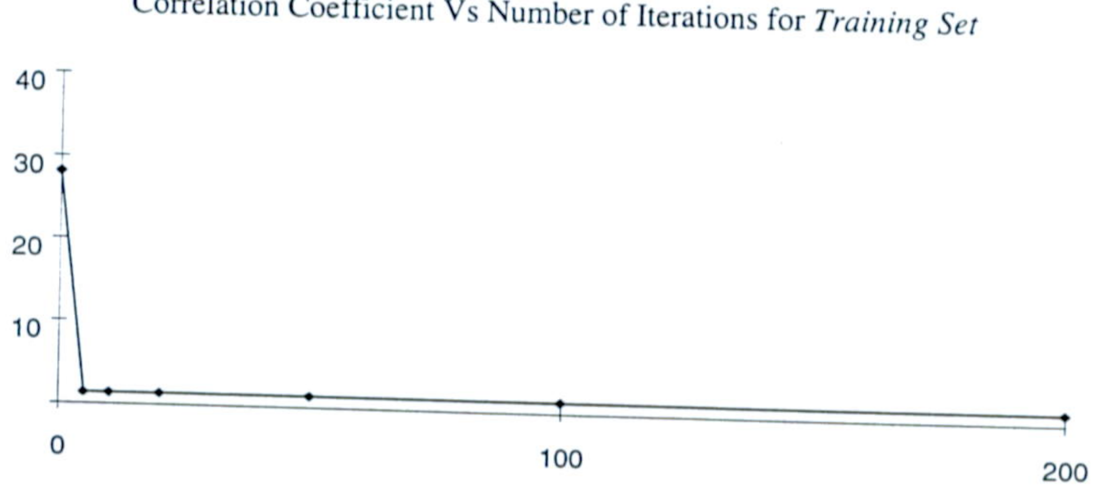


Figure A-6.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

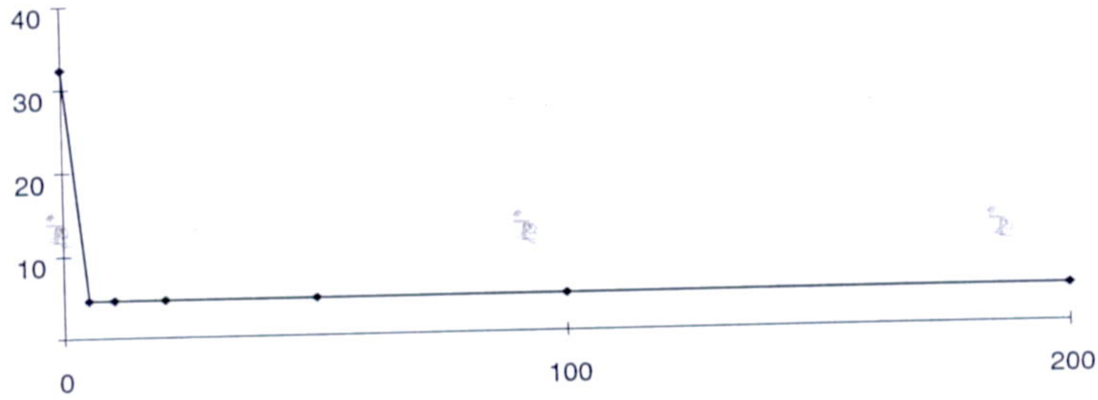


Figure 6.3
Maximum Error Vs Number of Iterations for *Training Set*

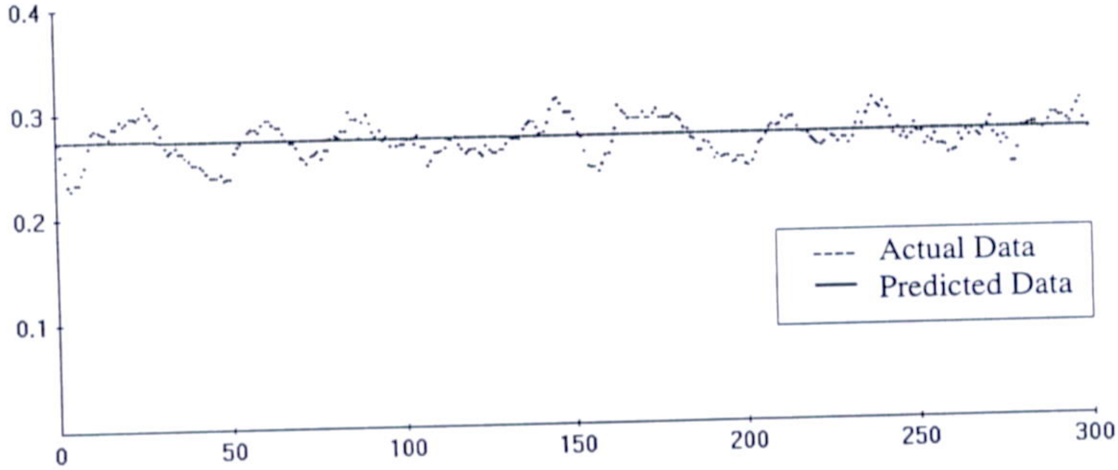


Figure A-6.4
Training Result after *5 Iterations*

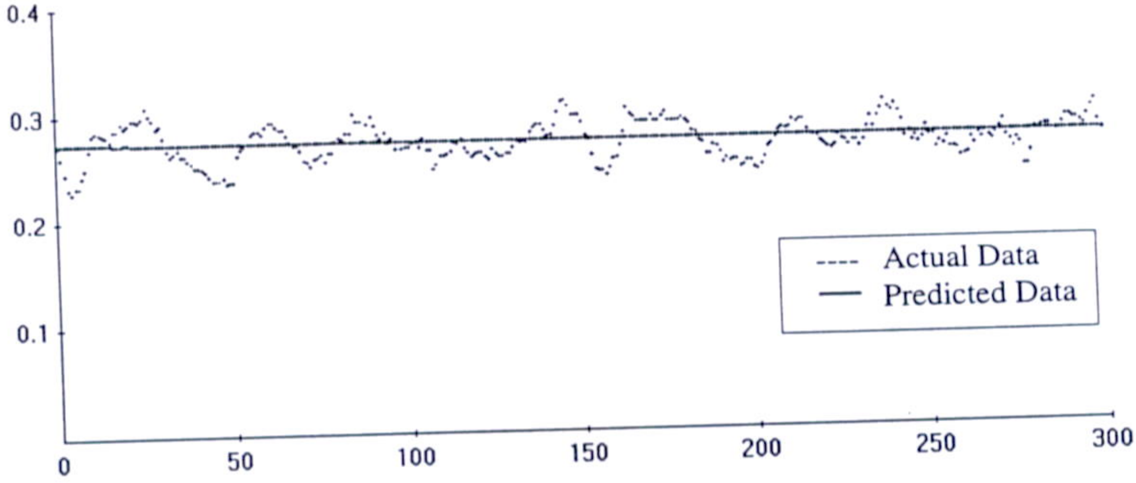


Figure A-6.5
Training Result after *200 Iterations*

Experiment Definition

No.	: 7
Directory	: MBPN\TEST7
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.1
Activation Function	: Logistic
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
0	- 1.67	10.17	14.66
5	- 2.06	1.37	
10	- 1.87	1.37	
20	- 1.39	1.37	
50	- 0.22	1.36	
100	1.76	1.36	
200	5.13	1.35	
500	10.04	1.34	3.91
1000	11.43	1.34	4.18
2000	11.97	1.34	
5000	12.48	1.34	4.25

Table A-7.1
Experiment Results on *Training Set*

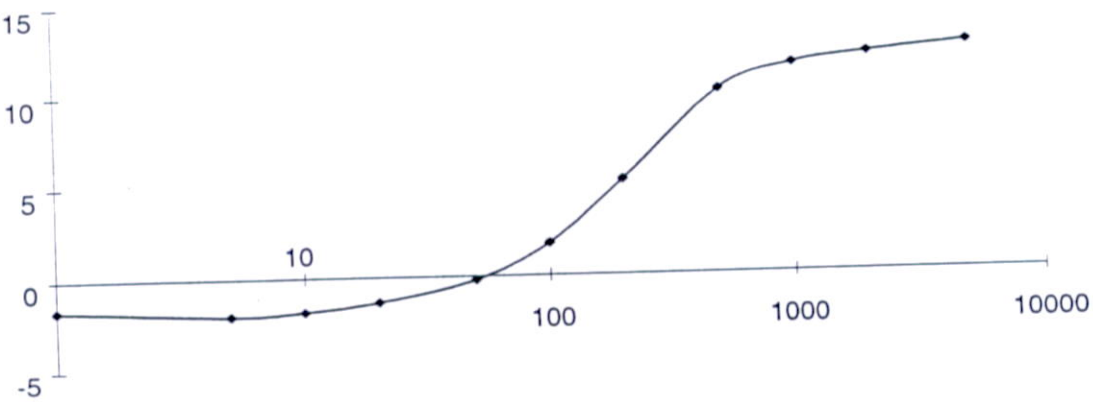


Figure A-7.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

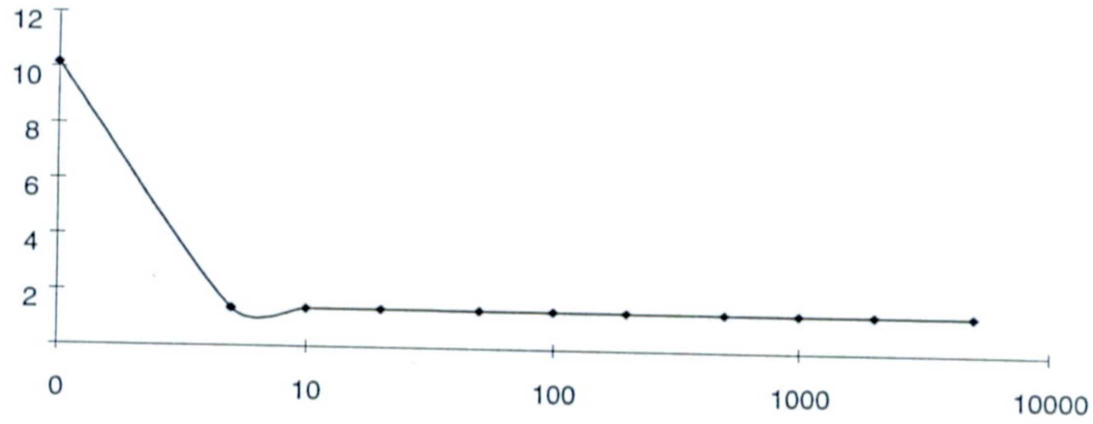


Figure A-7.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

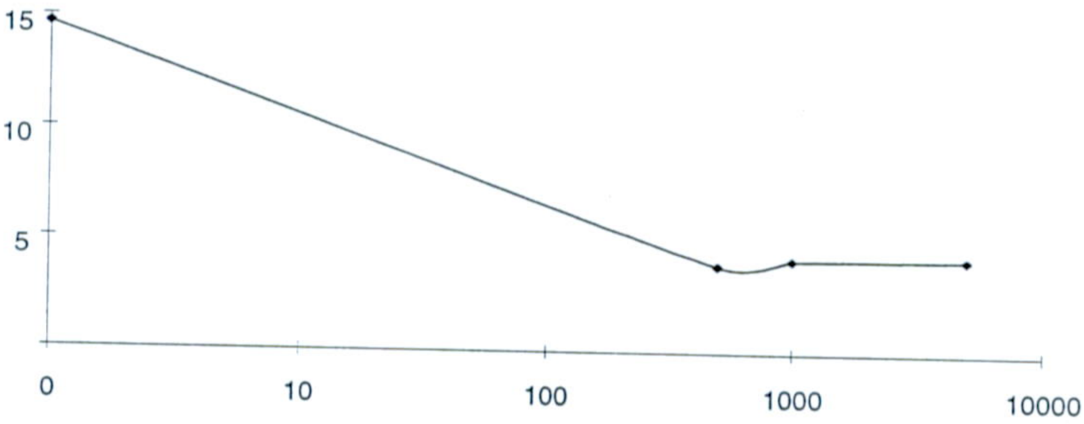


Figure A-7.3
Maximum Error Vs Number of Iterations for *Training Set*

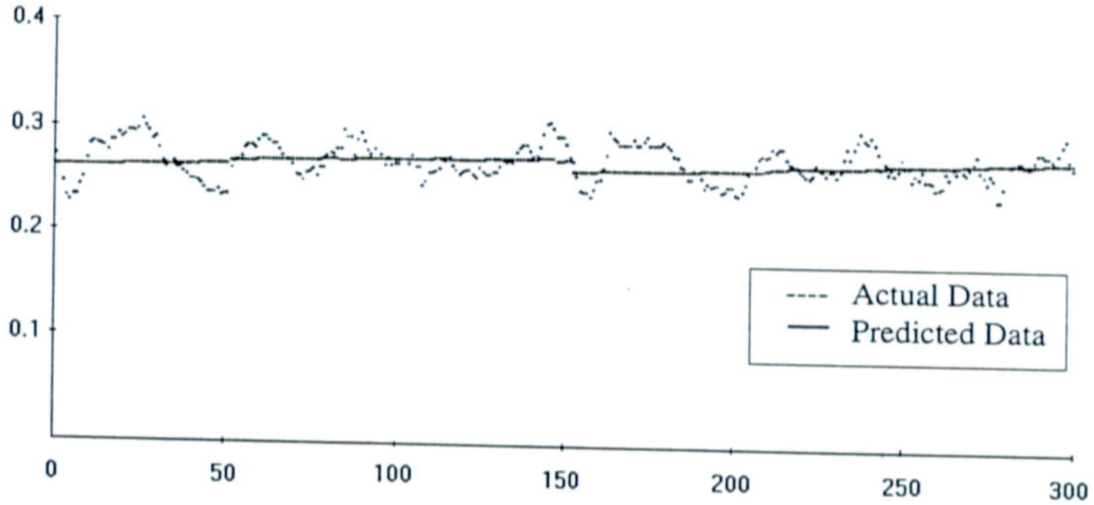


Figure A-7.4
Training Result after *5,000 Iterations*

Experiment Definition

No.	: 8
Directory	: MBPN\TEST8
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.1
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error (x10^{-2})	Maximum Error (x10^{-2})
0	- 12.14	72.03	76.95
5	- 11.67		
10	- 6.41		
20	7.20		
50	9.70		
100	9.33	1.51	5.22
200	8.94		
500	9.71	1.56	5.36
1000	13.12	1.50	5.09
2000	22.52		
5000	26.49	1.28	5.23
10000	35.10	1.33	5.97
15000	39.50	1.33	
30000	58.56	1.23	3.85
40000	71.77	0.92	3.66
50000	75.13	0.83	4.46
60000	77.72	0.77	5.69
70000	81.48	0.71	5.15
80000	85.56	0.64	3.34
90000	87.17	0.61	3.02
100000	87.91	0.59	3.07
110000	88.35	0.58	3.04
120000	88.72	0.57	2.97
130000	89.06	0.57	2.86
140000	89.41	0.56	2.75
150000	89.79	0.55	2.79
160000	90.13	0.54	2.83
170000	90.41	0.53	2.78
180000	90.64	0.52	2.68
190000	90.83	0.52	2.58
200000	91.02	0.52	2.50

Table A-8.1
Experiment Results on *Training Set*

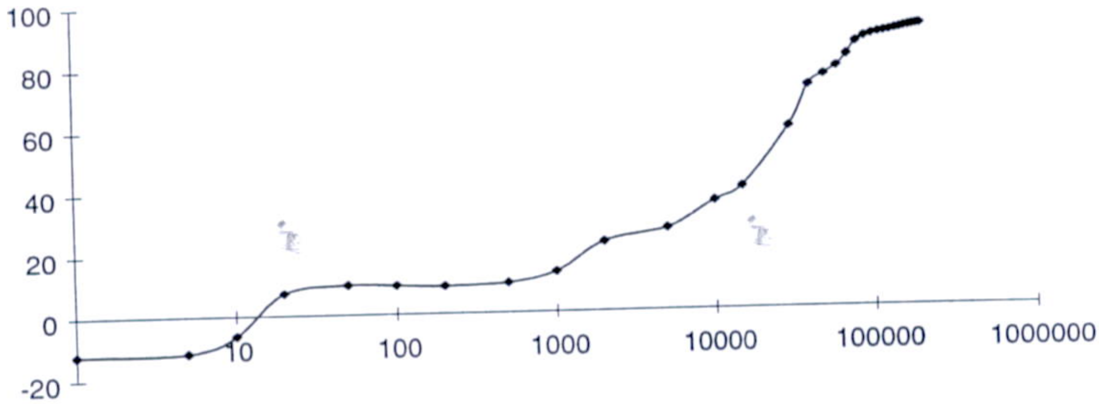


Figure A-8.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

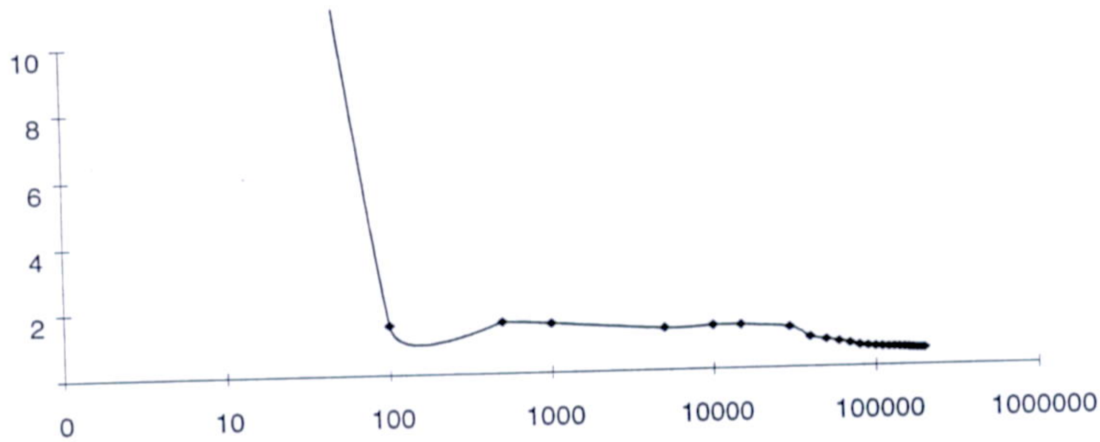


Figure A-8.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

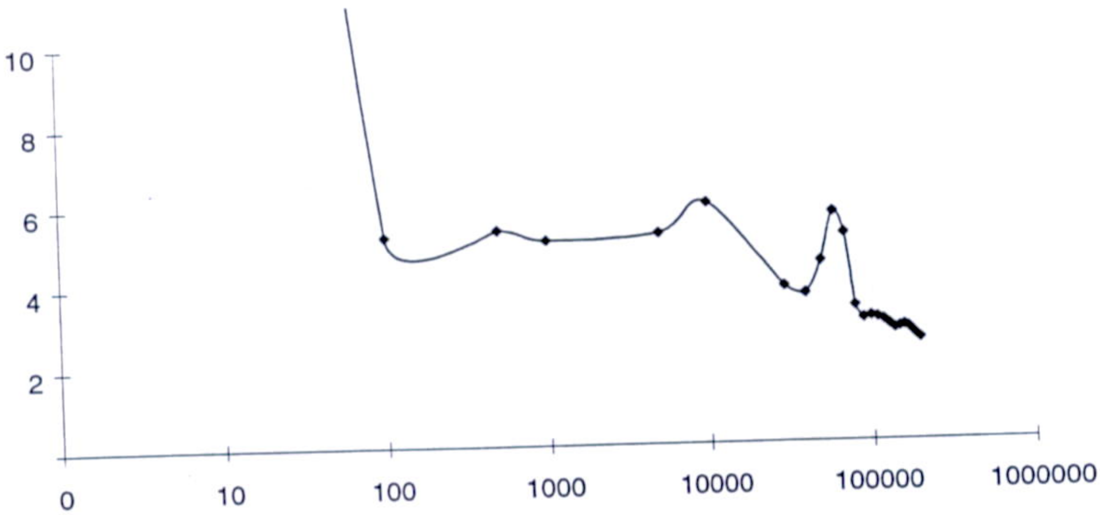


Figure A-8.3
Maximum Error Vs Number of Iterations for *Training Set*

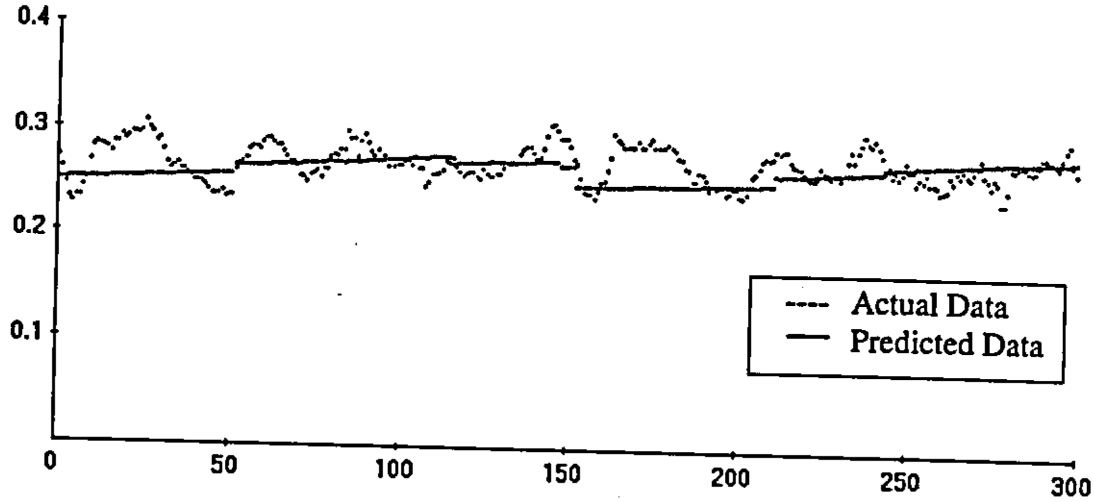


Figure A-8.4
Training Result after *100 Iterations*

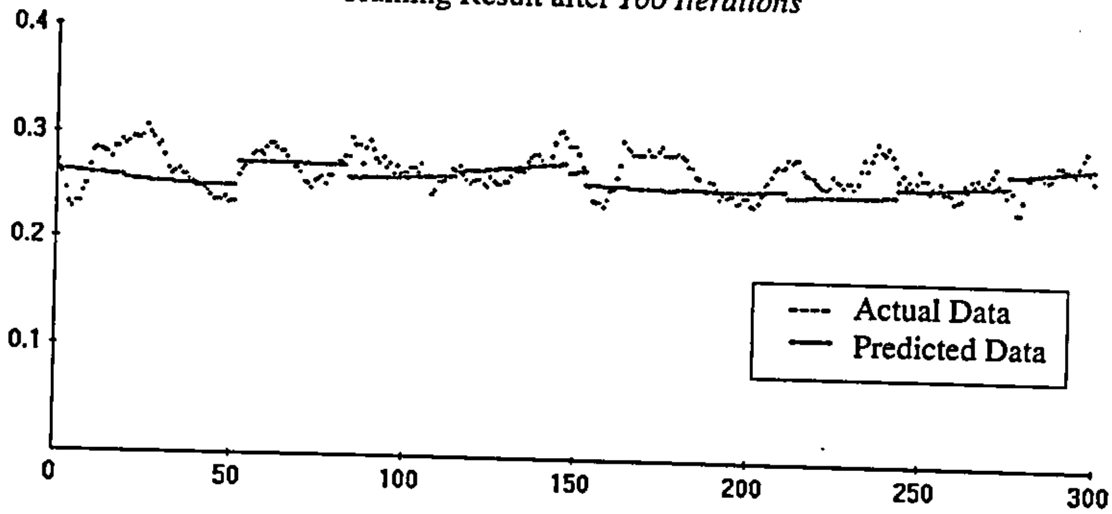


Figure A-8.5
Training Result after *1,000 Iterations*

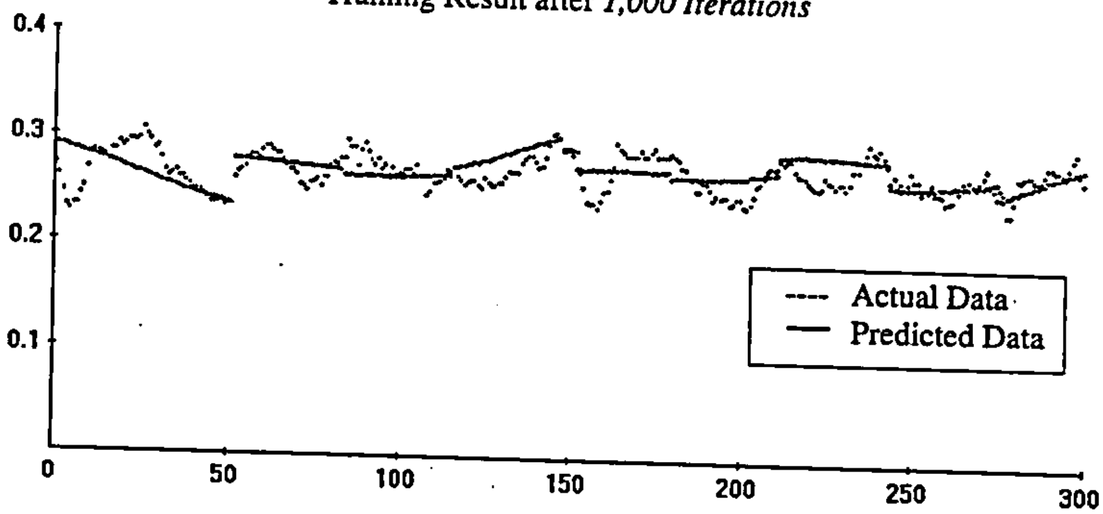


Figure A-8.6
Training Result after *10,000 Iterations*

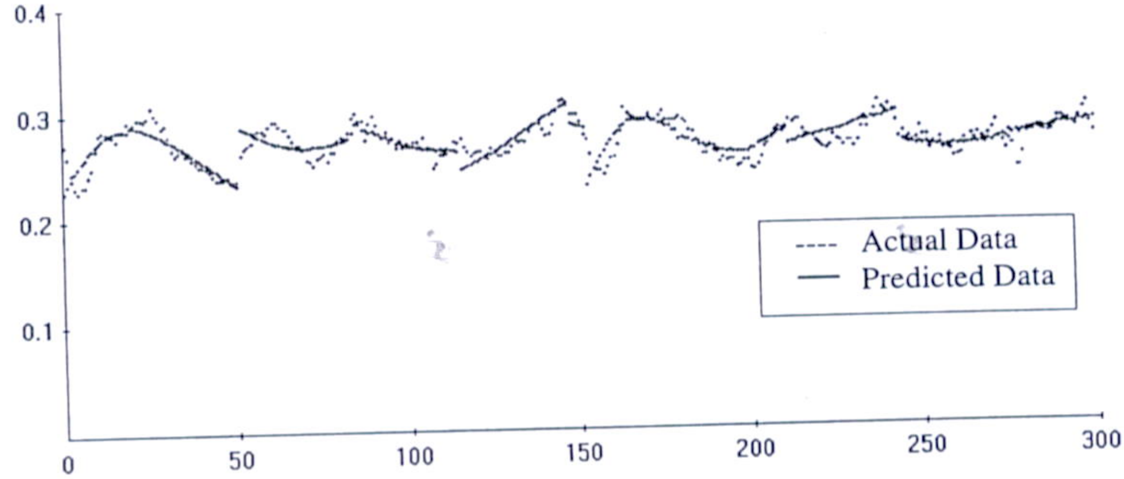


Figure A-8.7
Training Result after *50,000 Iterations*

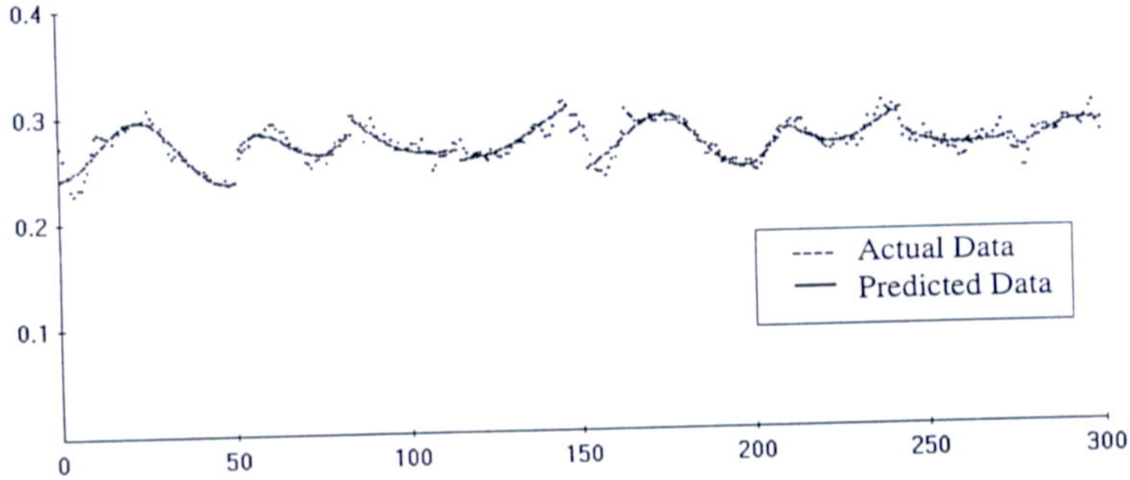


Figure A-8.8
Training Result after *100,000 Iterations*

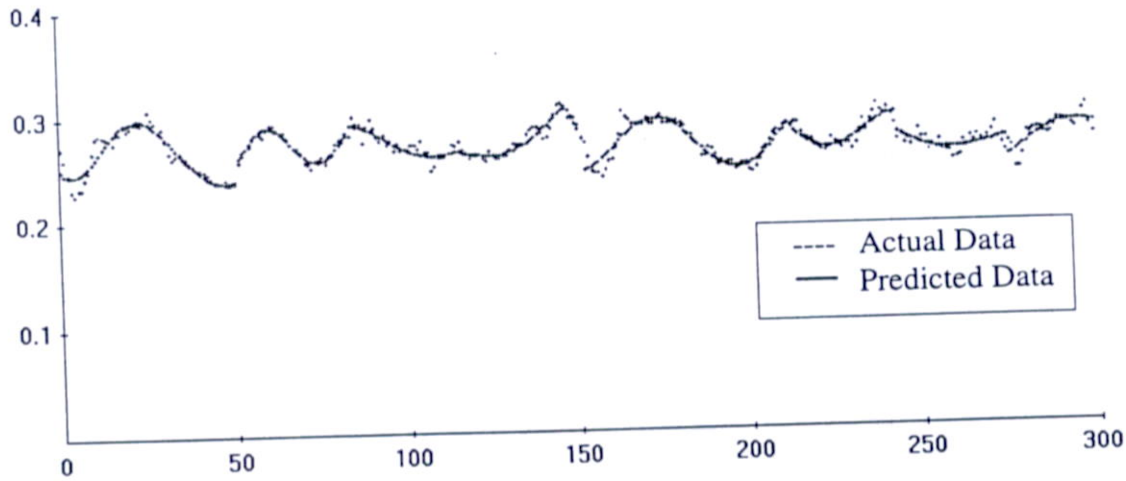


Figure A-8.9
Training Result after *200,000 Iterations*

Experiment Definition

No. [1]	: 8a
Directory	: MBPN\TEST8
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.1 and 0.5 (for comparison)
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

[1] The status of 10,000 iterations of Experiment 8 was used as initial state in this experiment.

Number of Iterations	α = 0.1 Correlation Coefficient (%)	α = 0.1 Mean Absolute Error (x10^{-2})	α = 0.5 Correlation Coefficient (%)	α = 0.5 Mean Absolute Error (x10^{-2})
Initial	35.10	1.33	35.10	1.33
1000	35.96	1.33	25.14	1.45
2000	36.87	1.32	26.45	1.42
3000	37.79	1.31	27.46	1.39
4000	38.69	1.30	28.90	1.35
5000	39.50	1.29	30.05	1.33

Table A-8a.1
Experiment Results on *Training Set*

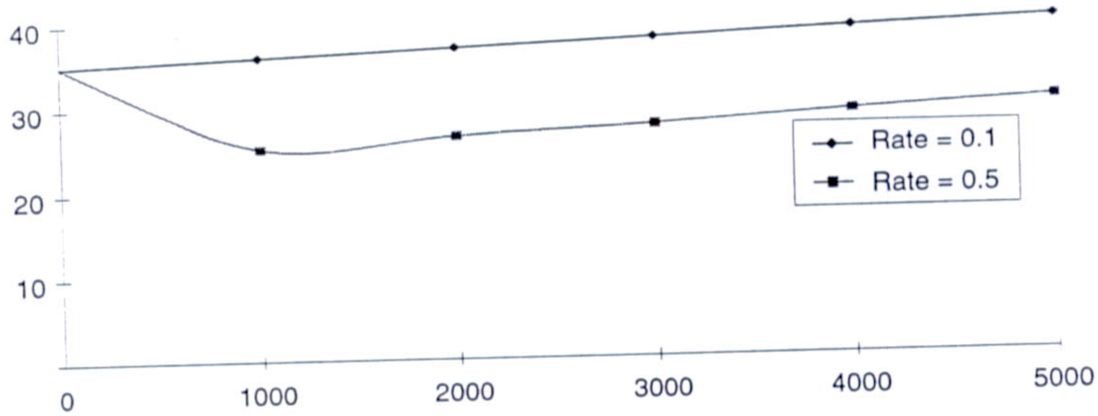


Figure A-8a.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

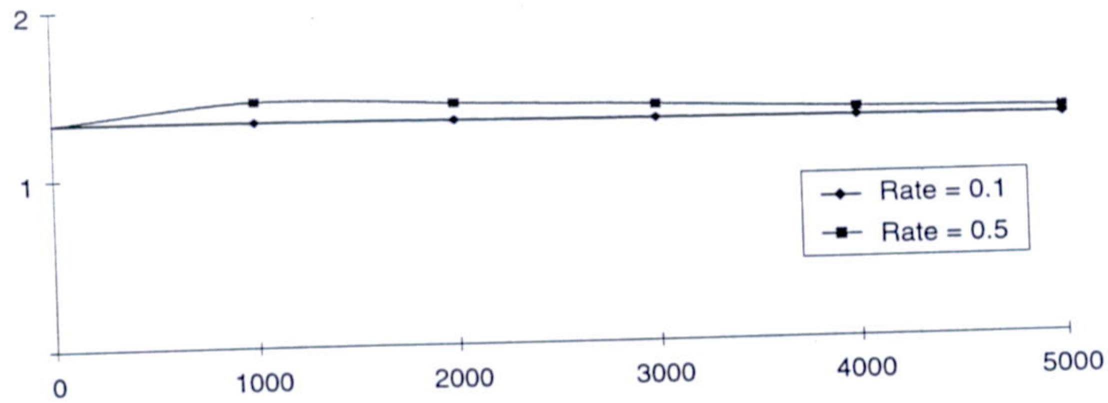


Figure A-8a.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

Experiment Definition

No. [1] : 8b

Directory : MBPN\TEST8

Tool : HNC Explore Net

Applications : *MTrain* for training

MTest for testing

Application Parameters

Size of Input Set : 300

Training Data : V02.DAC (First 300 data)

Sawtooth File : ON

Training Method : Fast training

Network Structure

Paradigm : Multi-Layer Backpropagation (*MBPN*)

Input Layer Size : 21

Output Layer Size : 1

No. of Hidden Layer : 2

Hidden Layer Sizes : 20-20

Input-Output Conn. : OFF

Training Parameters

Learning Method : Normal

Learning Rate (α) : 0.1 and 0.01 (for comparison)

Activation Function : Gaussian

(Lower Limit = 0, Upper Limit =1, Steepness = 1)

Random Seed : 5

Initial Weight Max. : 0.5

Error Tolerance : 0

Lin.Act.Func.on Ou. : OFF

---

[1] The status of 100,000 iterations of Experiment 8 was used as initial state in this experiment..

Number of Iterations	α = 0.1 Correlation Coefficient (%)	Mean Absolute Error ($\times10^{-2}$)	Maximum Error ($\times10^{-2}$)
Initial	87.91	0.59	3.07
5000	88.17	0.59	
10000	88.35	0.58	3.04
15000	88.54	0.58	
20000	88.72	0.57	2.97
25000	88.88	0.57	
30000	89.06	0.57	2.86
35000	89.24	0.56	
40000	89.41	0.56	2.75
45000	89.62	0.55	
50000	89.79	0.55	2.79

Number of Iterations	α = 0.01 Correlation Coefficient (%)	Mean Absolute Error ($\times10^{-2}$)	Maximum Error ($\times10^{-2}$)
Initial	87.91	0.59	3.07
5000	88.45	0.58	
10000	88.54	0.58	2.96
15000	88.61	0.57	
20000	88.68	0.57	2.92
25000	88.76	0.57	
30000	88.81	0.57	2.88
35000	88.87	0.57	
40000	88.92	0.56	2.84
45000	88.98	0.56	
50000	89.01	0.56	2.80

Table A-8b.1
Experiment Results on *Training Set*

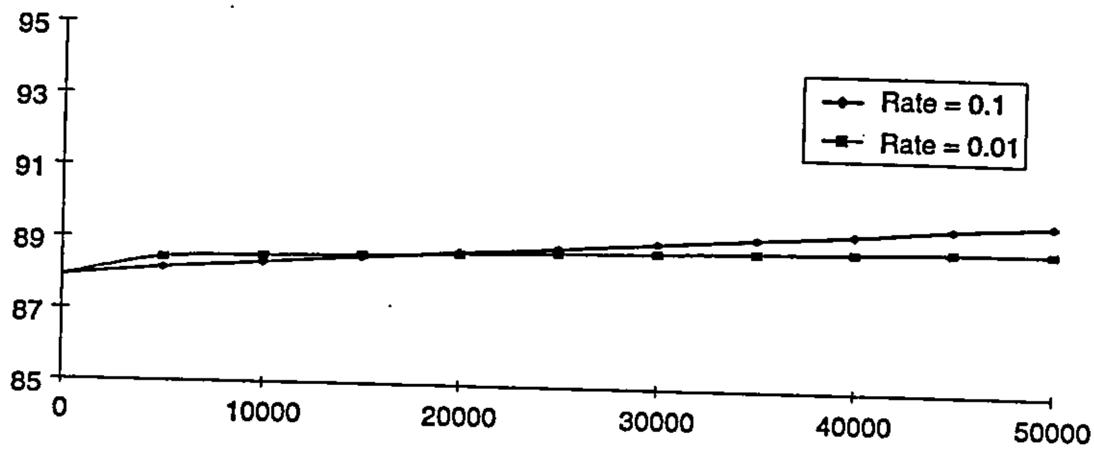


Figure A-8b.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

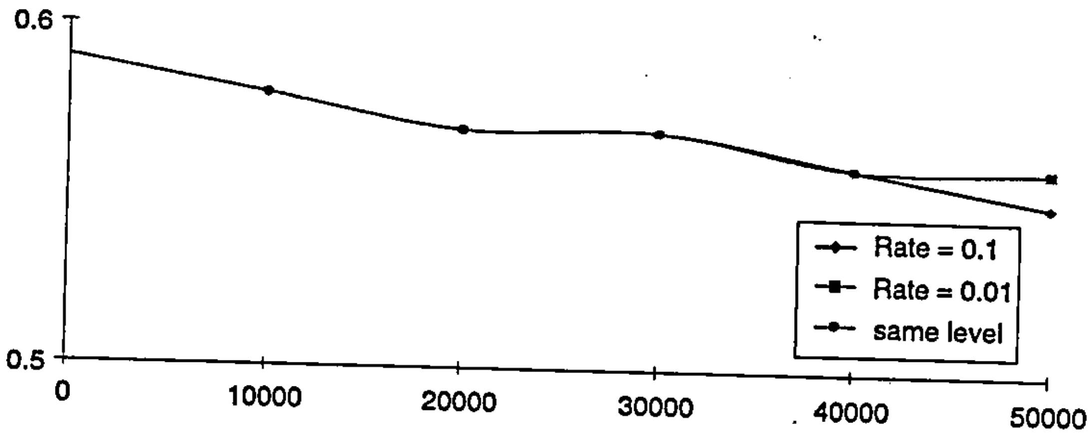


Figure A-8b.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

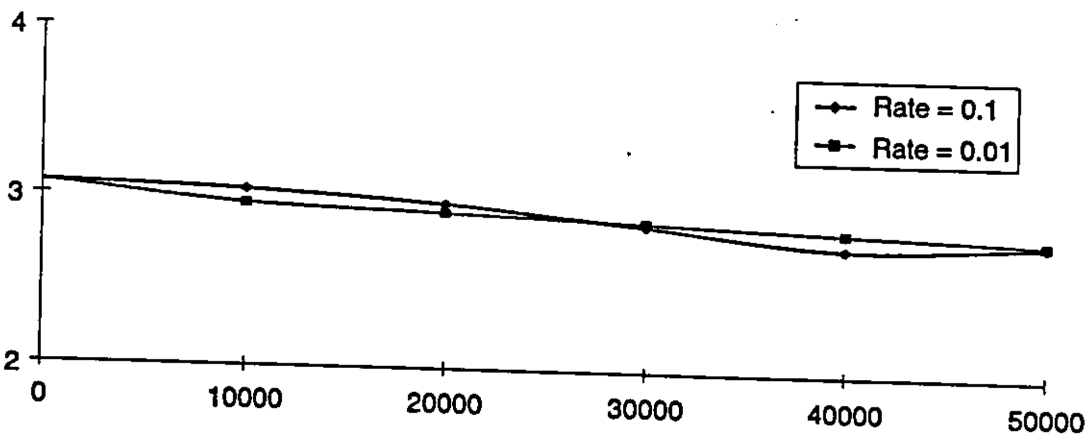


Figure A-8b.3
Maximum Error Vs Number of Iterations for *Training Set*

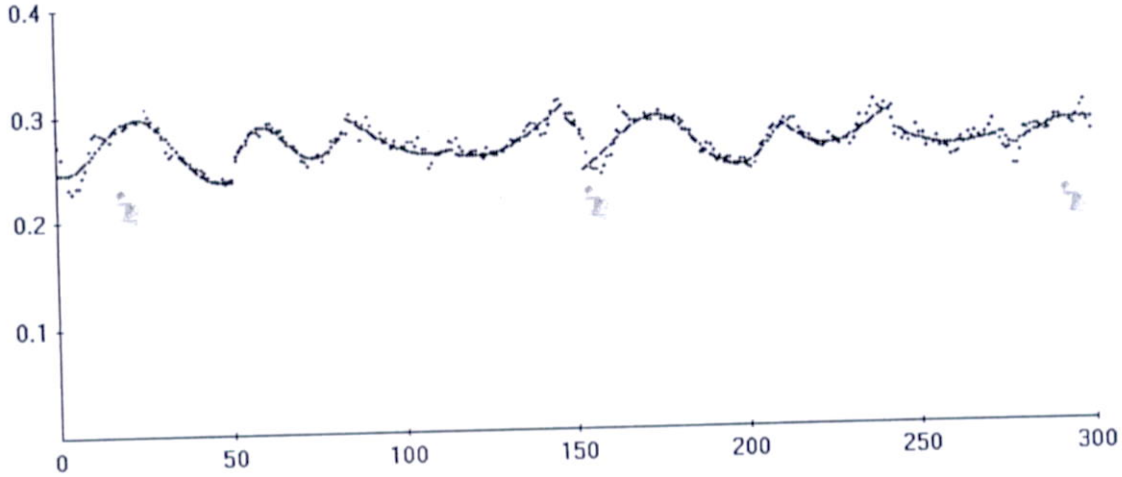


Figure A-8b.4
Training Result after 50,000 Iterations *for* $\alpha = 0.1$ (total epochs = 150,000)

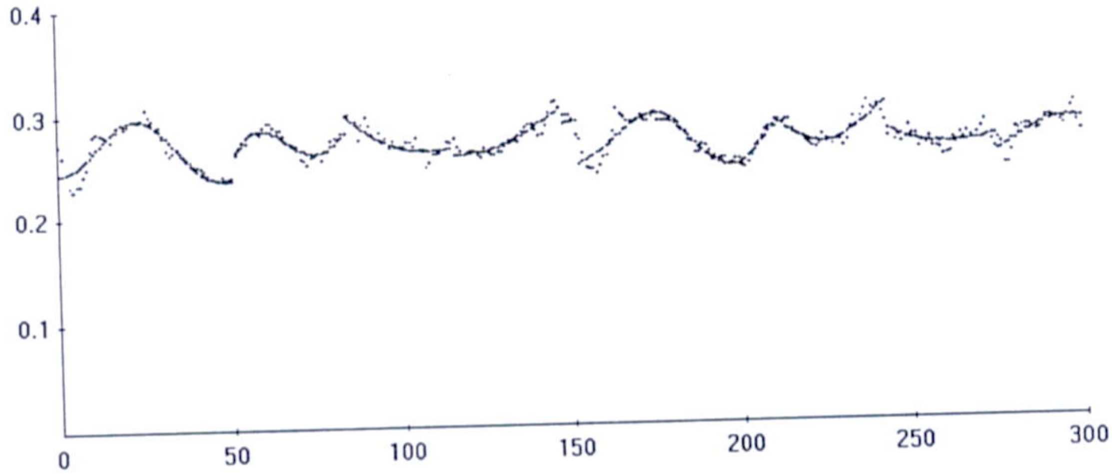


Figure A-8b.5
Training Result after 50,000 Iterations *for* $\alpha = 0.01$ (total epochs = 150,000)

Experiment Definition

No.	: 9
Directory	: MBPN\TEST9
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.5
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
0	- 12.14	72.03	76.95
5	- 12.42	1.72	6.54
10	- 12.86	1.52	5.85
20	- 13.22	1.41	5.06
50	- 3.97	1.38	4.58
100	- 0.69	1.50	4.86
200	0.95	1.71	5.29
500	3.80	1.80	5.46
1000	6.99	1.83	5.51
2000	9.65	1.58	5.30
5000	16.72	1.58	5.08
10000	28.41	1.40	5.35
20000	53.43	1.11	4.05
30000	67.04	0.95	4.55
40000	77.54	0.79	4.07
50000	81.91	0.72	3.52
60000	85.73	0.68	3.06
70000	88.30	0.59	3.06
80000	89.68	0.53	2.75
90000	90.32	0.51	2.64
100000	90.91	0.49	2.55
110000	91.40	0.49	2.42
120000	91.93	0.48	2.41
130000	92.42	0.47	2.51
140000	92.94	0.46	2.57
150000	93.31	0.45	2.55
160000	94.33	0.42	2.04
170000	94.89	0.40	2.02
180000	95.08	0.39	1.99
190000	95.16	0.38	1.97
200000	95.24	0.38	1.96
210000	95.29	0.38	1.95

Table A-9.1
Experiment Results on *Training Set*

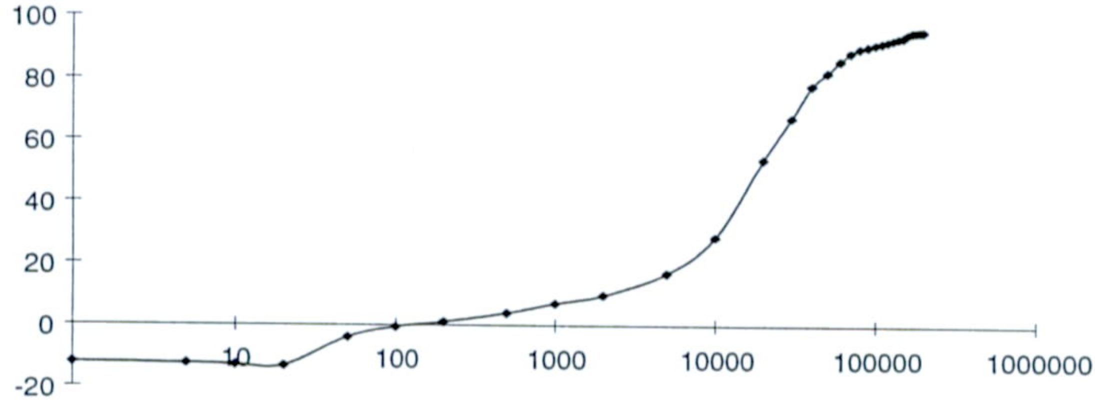


Figure A-9.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

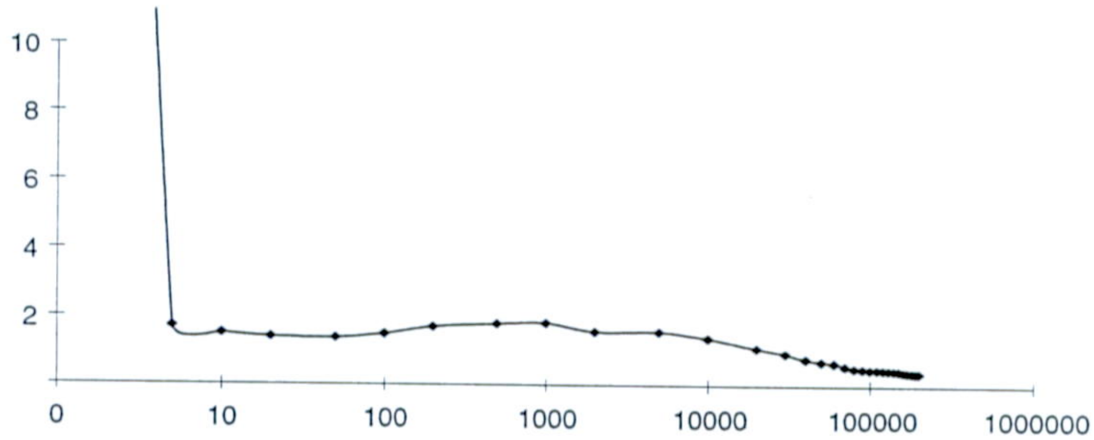


Figure A-9.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

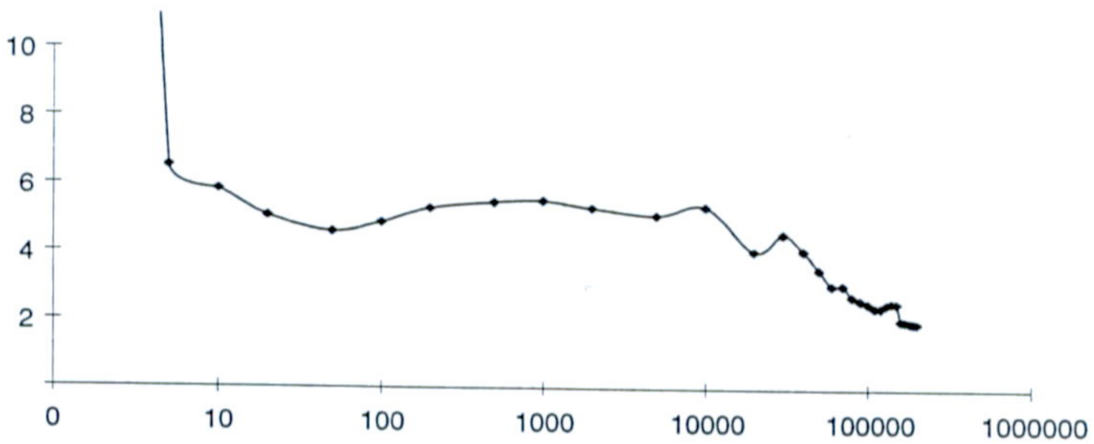


Figure A-9.3
Maximum Error Vs Number of Iterations for *Training Set*

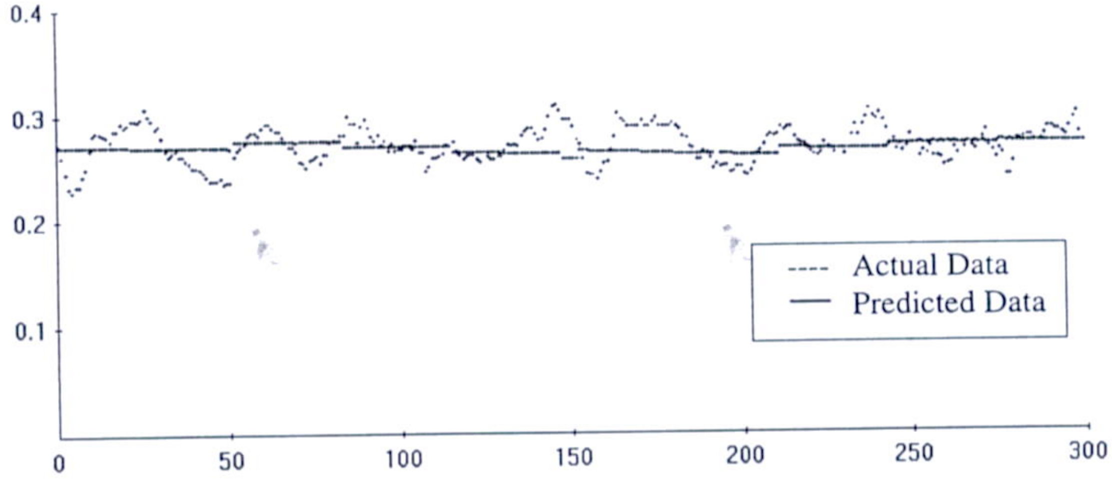


Figure A-9.4
Training Result after *50 Iterations*

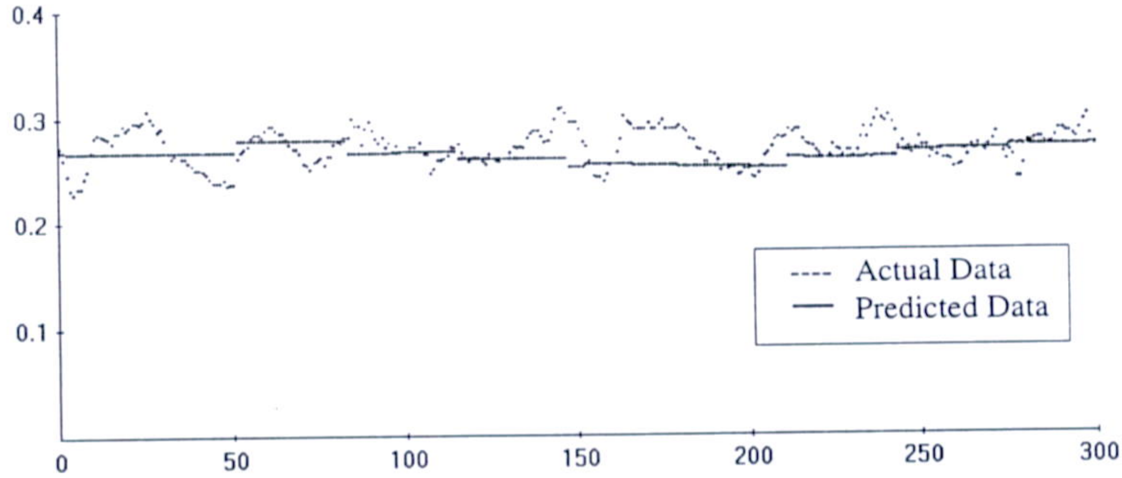


Figure A-9.5
Training Result after *100 Iterations*

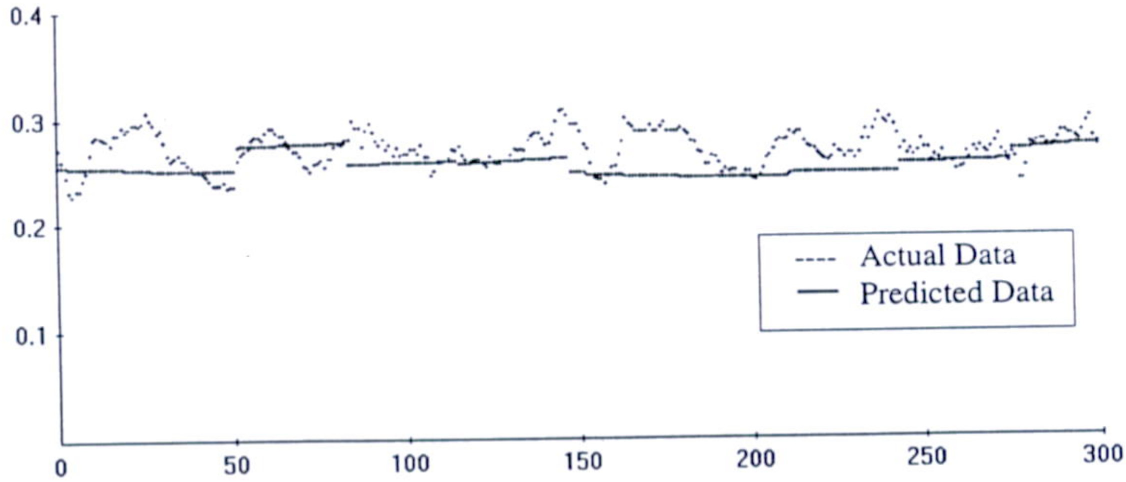


Figure A-9.6
Training Result after *1,000 Iterations*

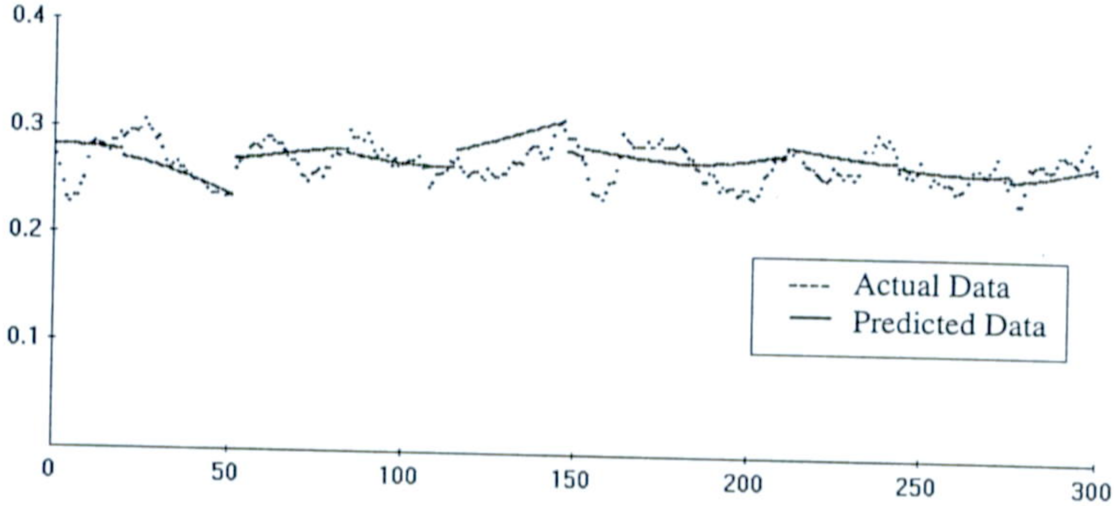


Figure A-9.7
Training Result after *10,000 Iterations*

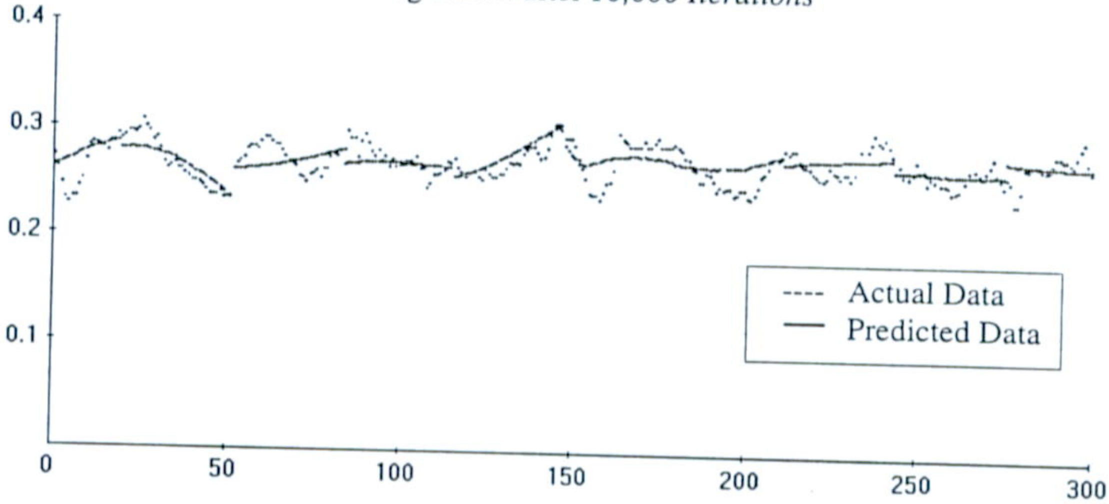


Figure A-9.8
Training Result after *20,000 Iterations*

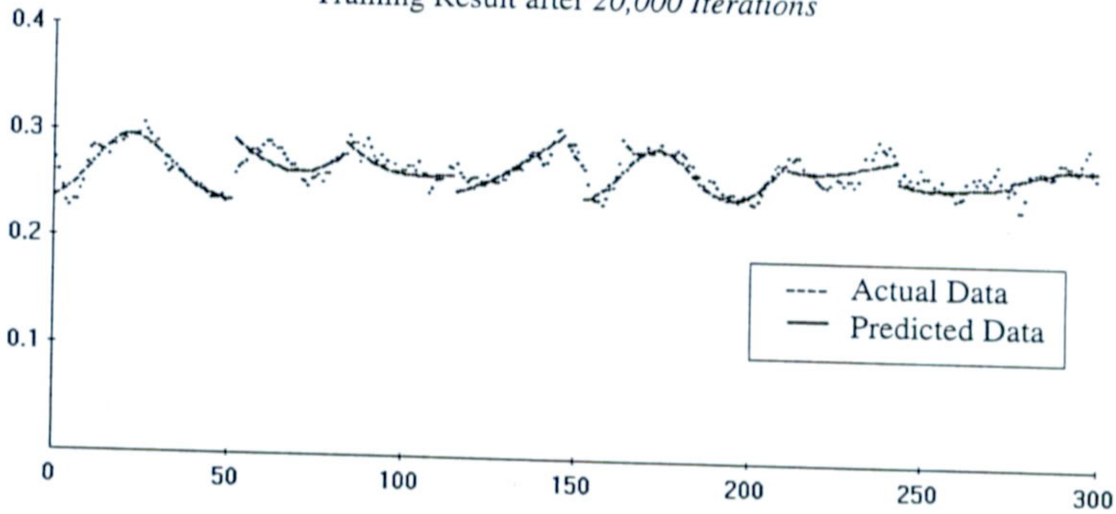


Figure A-9.9
Training Result after *50,000 Iterations*

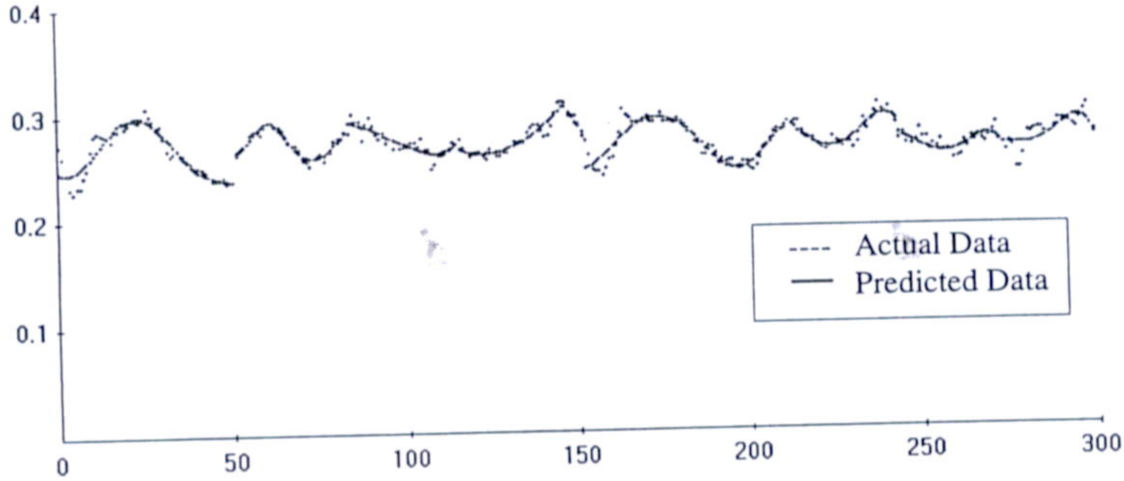


Figure A-9.10
Training Result after *100,000 Iterations*

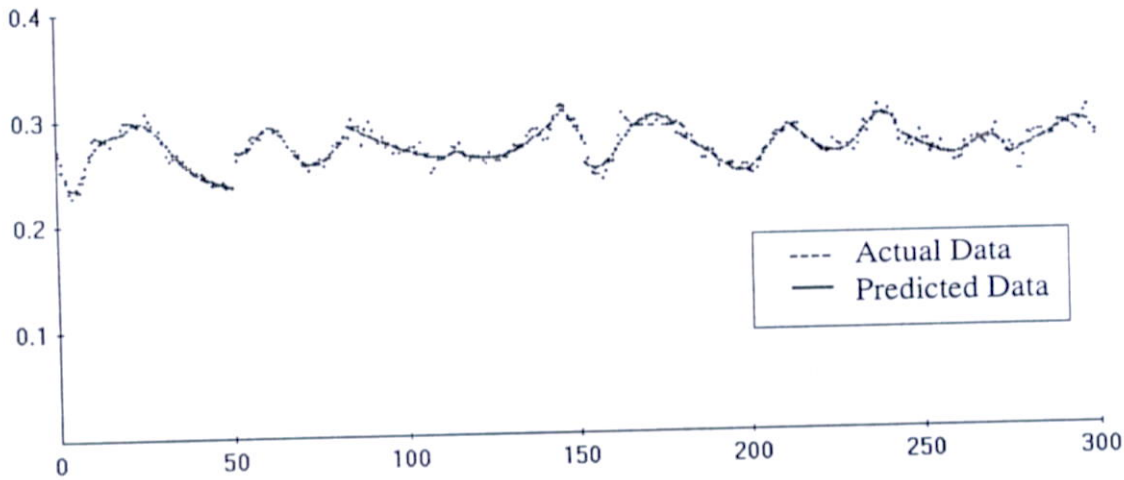


Figure A-9.11
Training Result after *150,000 Iterations*

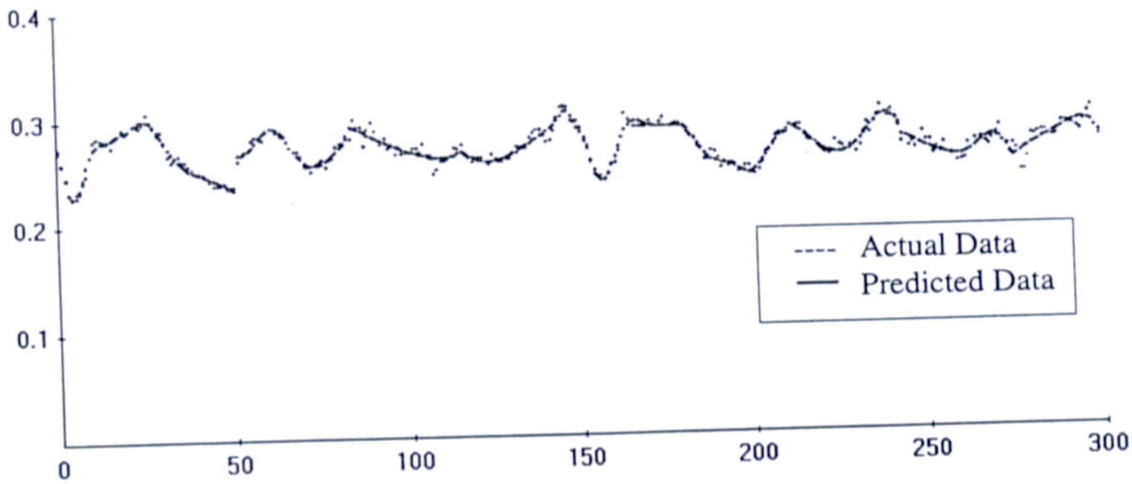


Figure A-9.12
Training Result after *210,000 Iterations*

Experiment Definition

No. [1]	: 9a
Directory	: MBPN\TEST9
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 300
Training Data	: V02.DAC (First 300 data)
Sawtooth File	: ON
Training Method	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.1 and 0.01 (for improvement)
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max. [2]	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

[1] The status of 210,000 iterations of Experiment 9 was used as initial state in this experiment.
[2] Initial weight maximum had no function here as previous weights were used.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
	$\alpha = 0.1$		
Initial	95.29	0.38	1.95
5000	95.36	0.38	1.93
10000	95.34	0.38	1.92
15000	95.37	0.38	1.92
20000	95.38	0.37	1.92
25000	95.39	0.37	1.92
30000	95.39	0.37	1.92
35000	95.41	0.37	1.91
40000	95.40	0.37	1.91
	$\alpha = 0.01$		
45000	95.40	0.37	1.91
50000	95.40	0.37	1.91
55000	95.42	0.37	1.91
60000	95.41	0.37	1.91
65000	95.42	0.37	1.92
70000	95.41	0.37	1.92
75000	95.43	0.37	1.92
80000	95.43	0.37	1.92
85000	95.43	0.37	1.92
90000	95.41	0.37	1.92

Table A-9a.1
Experiment Results on *Training Set*

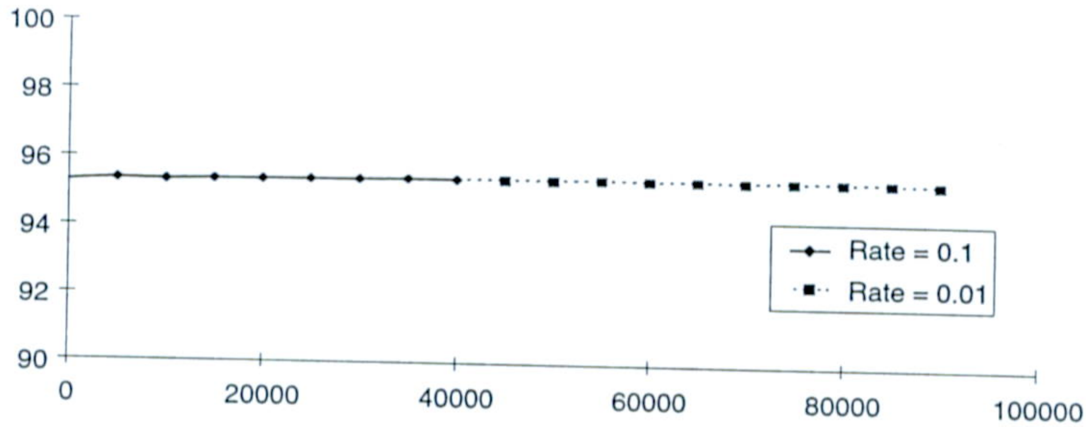


Figure A-9a.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

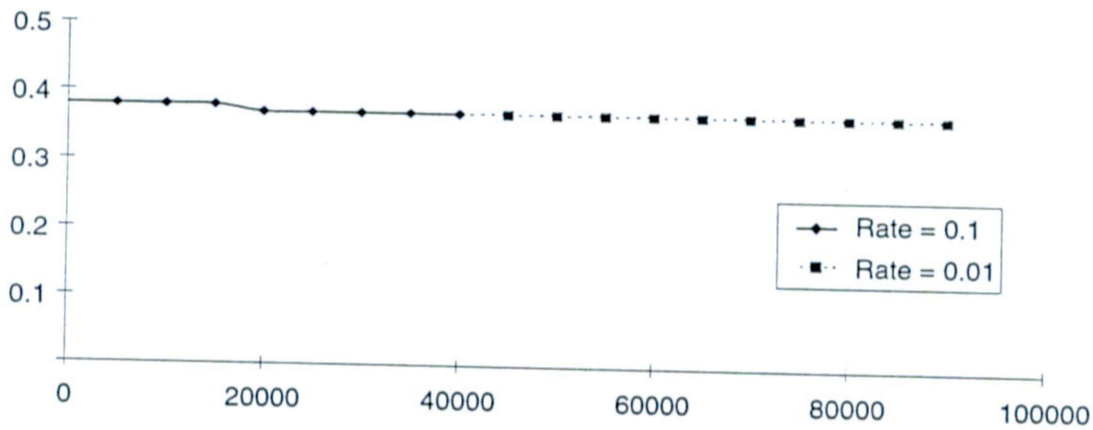


Figure A-9a.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

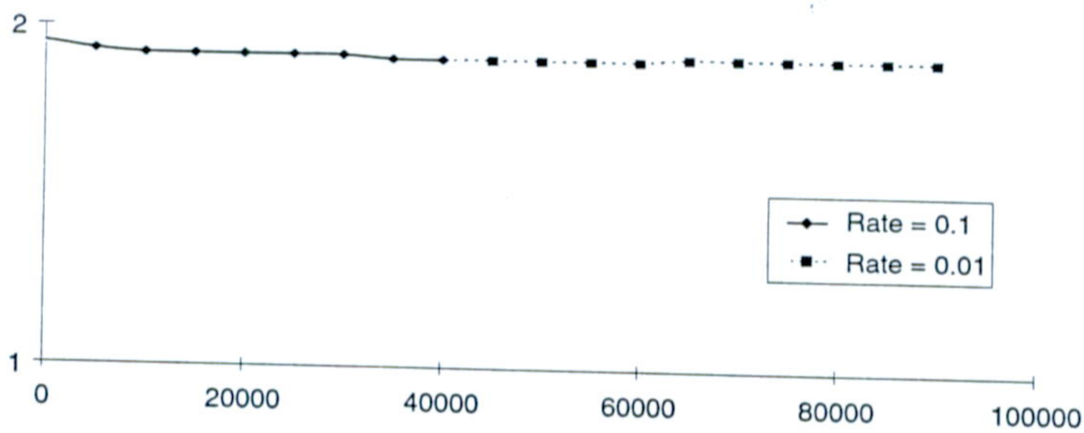


Figure A-9a.3
Maximum Error Vs Number of Iterations for *Training Set*

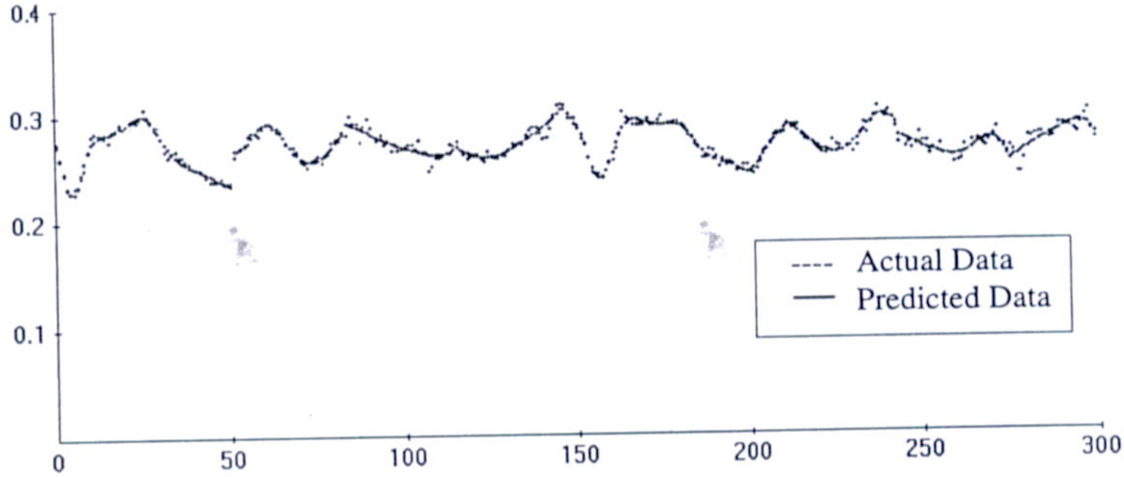


Figure A-9a.4
Training Result after 5,000 Iterations *for* $\alpha = 0.1$ (total epochs = 215,000)

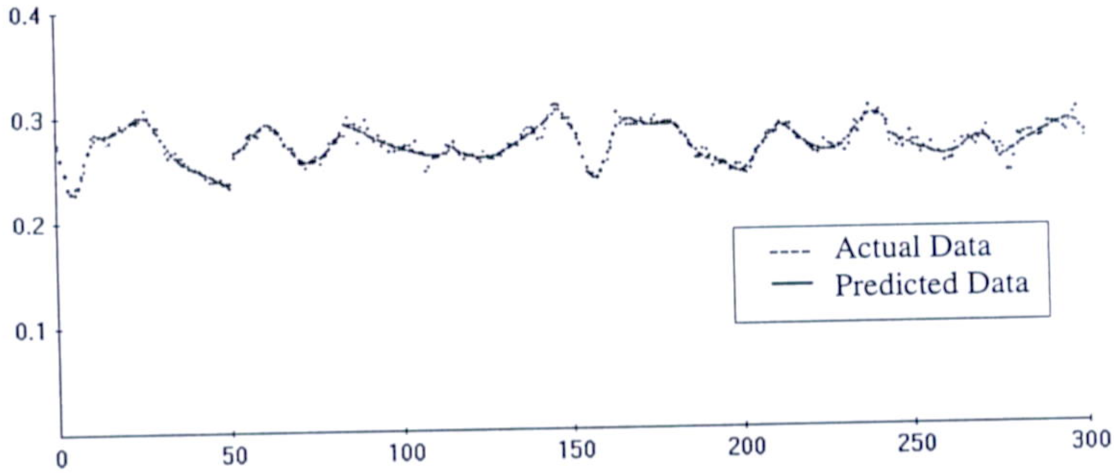


Figure A-9a.5
Training Result after 35,000 Iterations *for* $\alpha = 0.1$ (total epochs = 245,000)

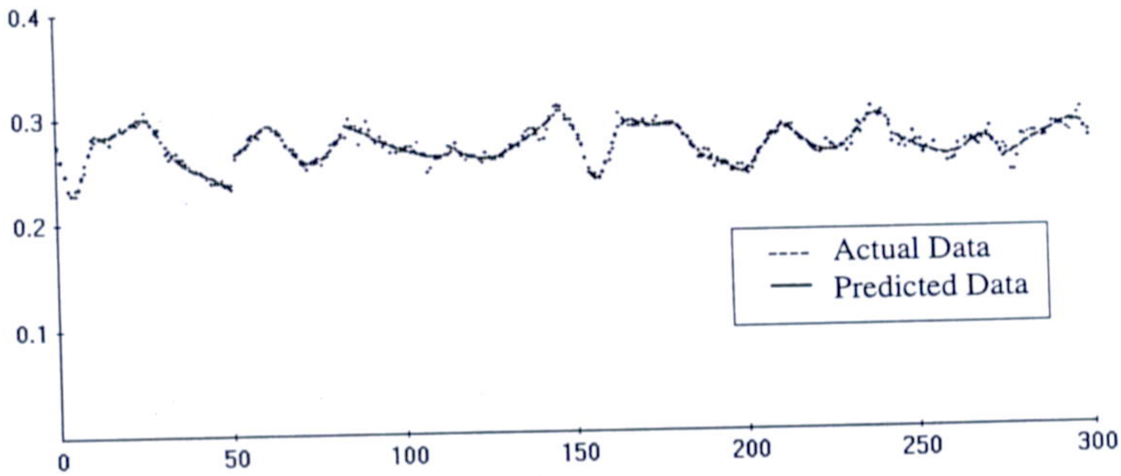


Figure A-9a.6
Training Result after 40,000 Iterations *for* $\alpha = 0.1$ (total epochs = 250,000)

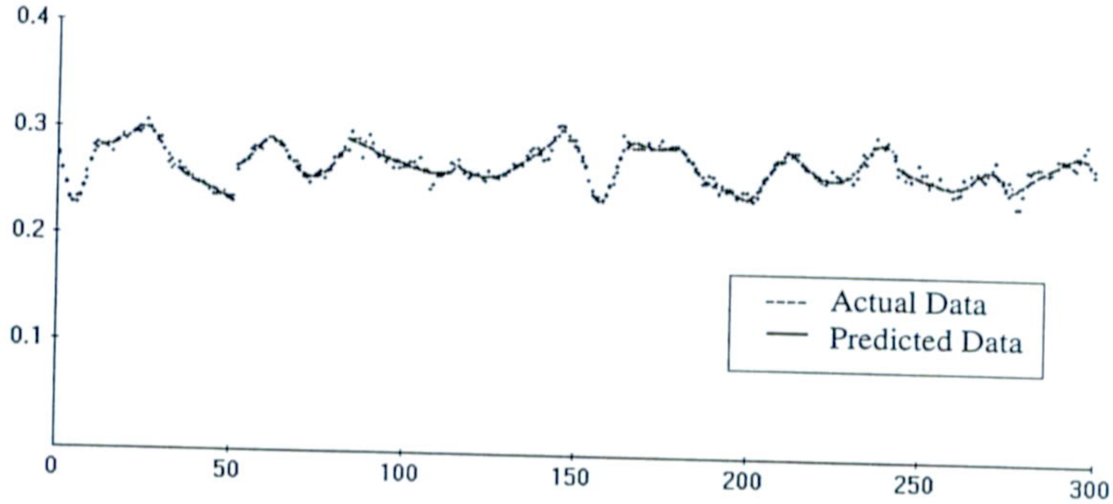


Figure A-9a.7
Training Result after 55,000 Iterations *for* $\alpha = 0.01$ (total epochs = 265,000)

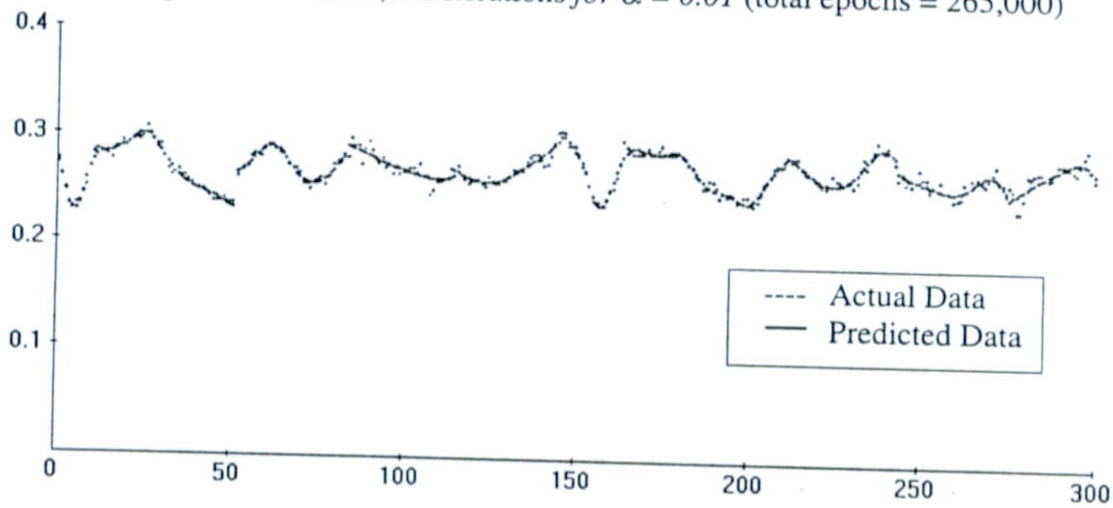


Figure A-9a.8
Training Result after 85,000 Iterations *for* $\alpha = 0.01$ (total epochs = 295,000)

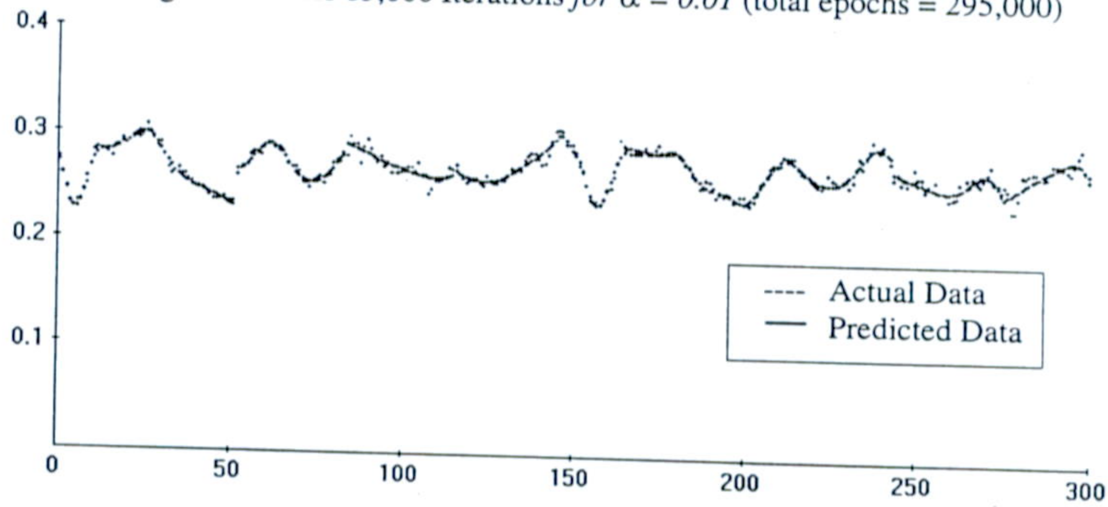


Figure A-9a.9
Training Result after 90,000 Iterations *for* $\alpha = 0.01$ (total epochs = 300,000)

Experiment Definition

No.[1] : 10

Directory : MBPN\TEST10

Tool : HNC Explore Net

Applications : *MTrain* for training
MTest for testing

Application Parameters

Size of Input Set : 6721

Training Data : T02.DAC

Sawtooth File : ON

Training Method : Normal training

Network Structure

Paradigm : Multi-Layer Backpropagation (*MBPN*)

Input Layer Size : 21

Output Layer Size : 1

No. of Hidden Layer : 2

Hidden Layer Sizes : 20-20

Input-Output Conn. : OFF

Training Parameters

Learning Method : Normal

Learning Rate (α) : 0.1

Activation Function : Gaussian
(Lower Limit = 0, Upper Limit =1, Steepness = 1)

Random Seed : 5

Initial Weight Max.[2] : 0.5

Error Tolerance : 0

Lin.Act.Func.on Ou. : OFF

---

[1] The status of 200,000 iterations of Experiment 8 was used as initial state in this experiment.
[2] Initial weight maximum had no function here as previous weights were used.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($x10^{-2}$)	Maximum Error ($x10^{-2}$)
Initial	- 9.31	16.54	69.47
5	37.94	1.83	13.54
10	43.35	1.72	13.30
20	47.84	1.65	12.99
50	54.49	1.56	12.47
100	59.79	1.49	12.03
200	64.96	1.42	11.03
500	73.00	1.26	10.98
1000	76.00	1.21	7.59

Table A-10.1
Experiment Results on *Training Set*

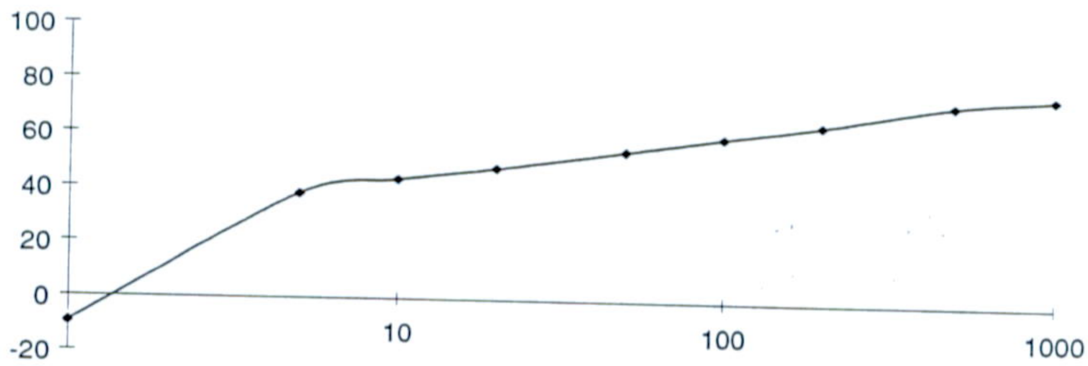


Figure A-10.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

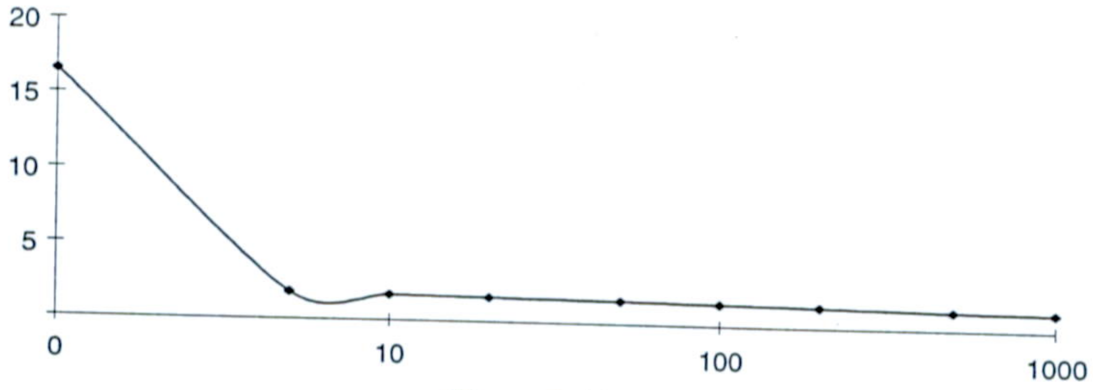


Figure A-10.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

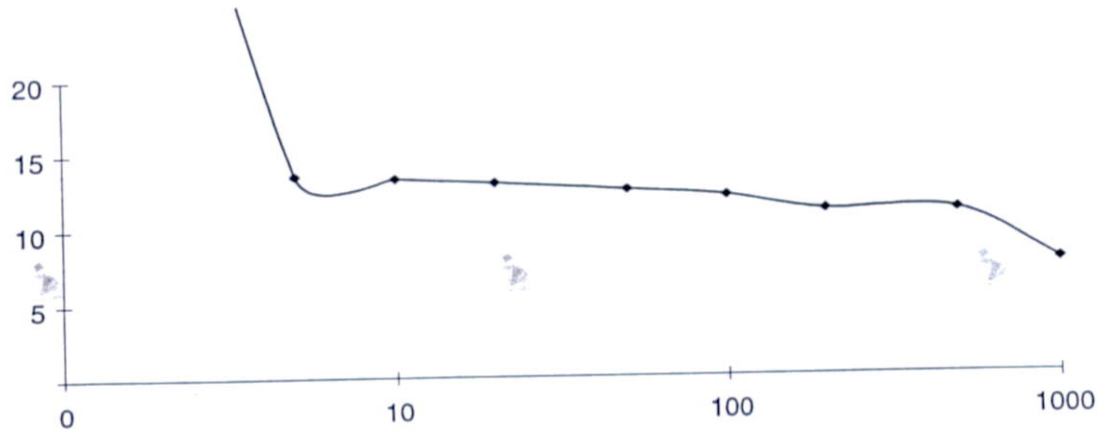


Figure A-10.3
Maximum Error Vs Number of Iterations for *Training Set*

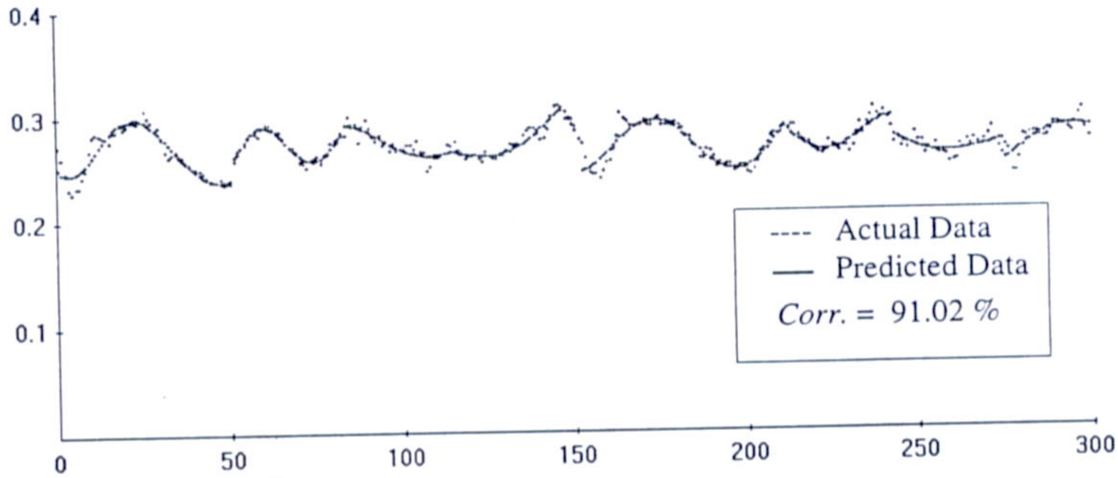


Figure A-10.4
Initial State for *First 300 Data*

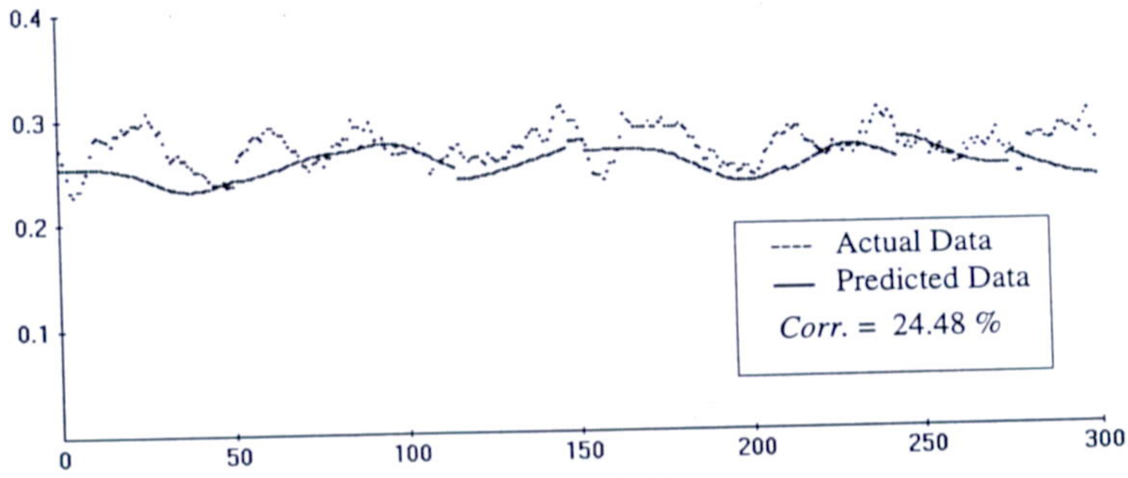


Figure A-10.5
Test Result after 10 Iterations for *First 300 Data*

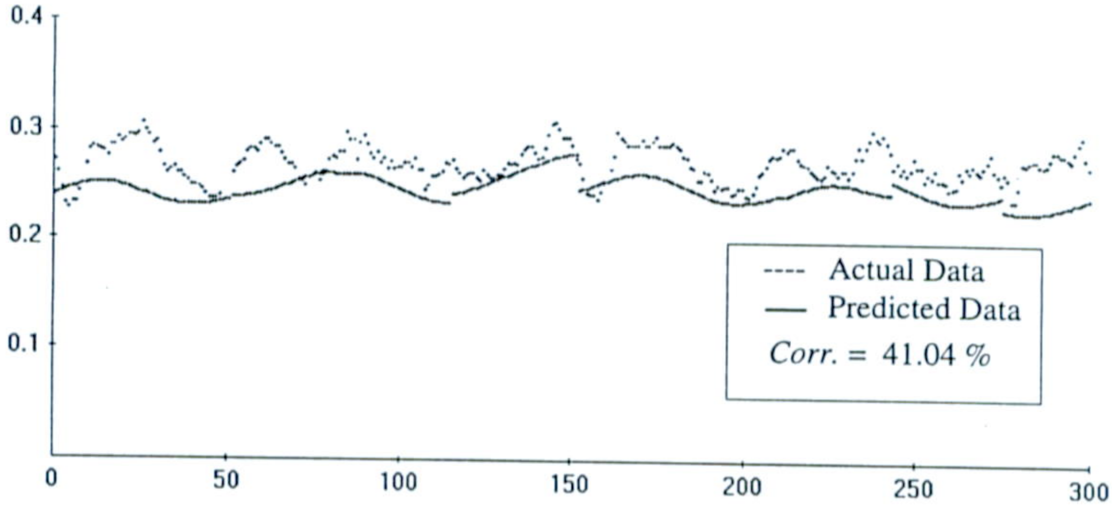


Figure A-10.6
Test Result after 100 Iterations for *First 300 Data*

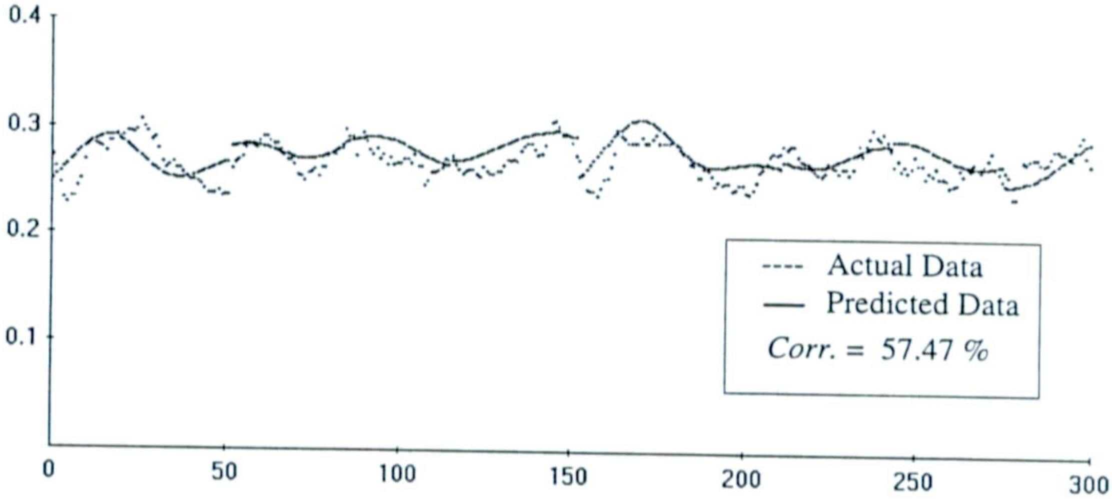


Figure A-10.7
Test Result after 1,000 Iterations for *First 300 Data*

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
100	51.76	2.48	13.76
1000	69.72	1.42	11.23

Table A-10.2
Validation Results on *Whole Set*

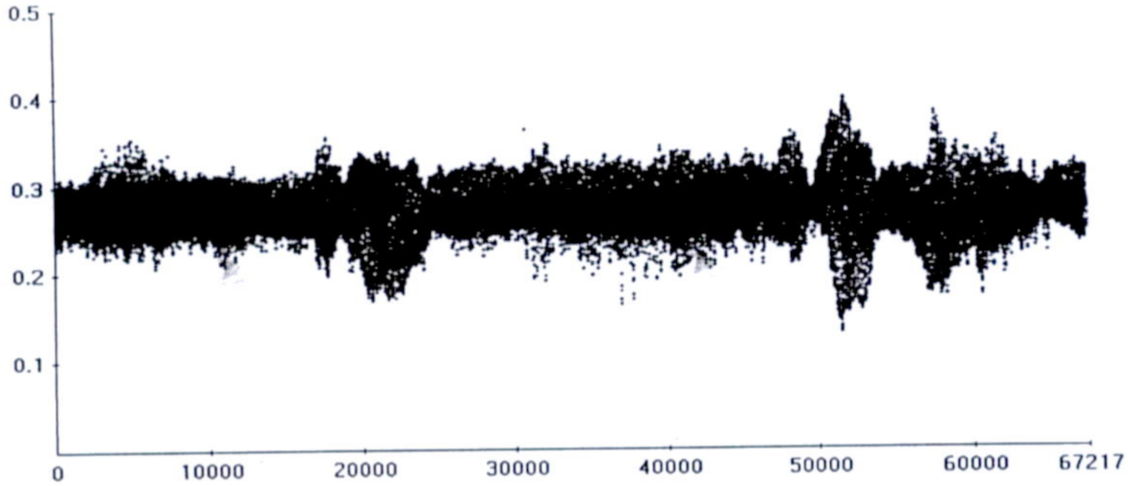


Figure A-10.8
Actual Strain Signal

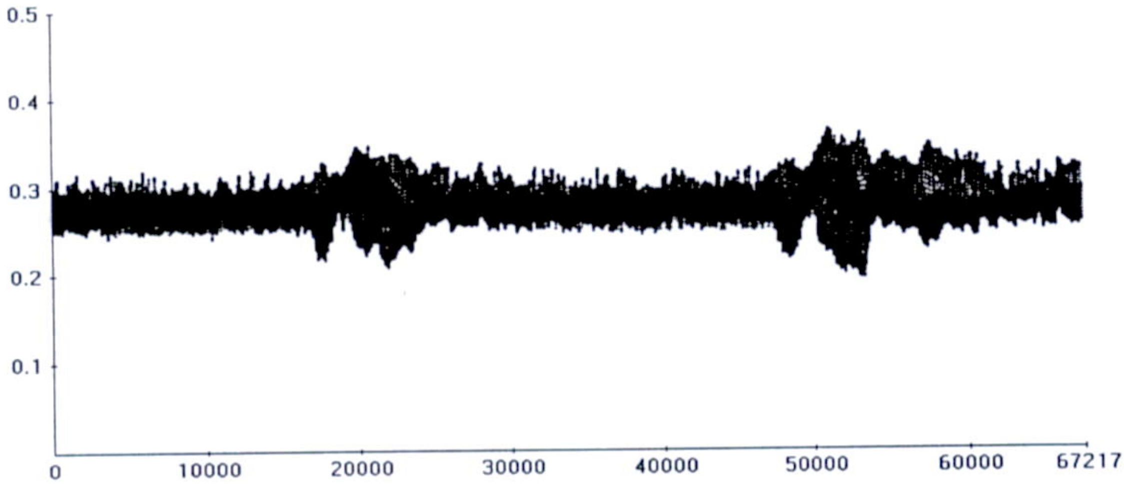


Figure A-10.9
Predicted Signal after *1,000 Iterations*

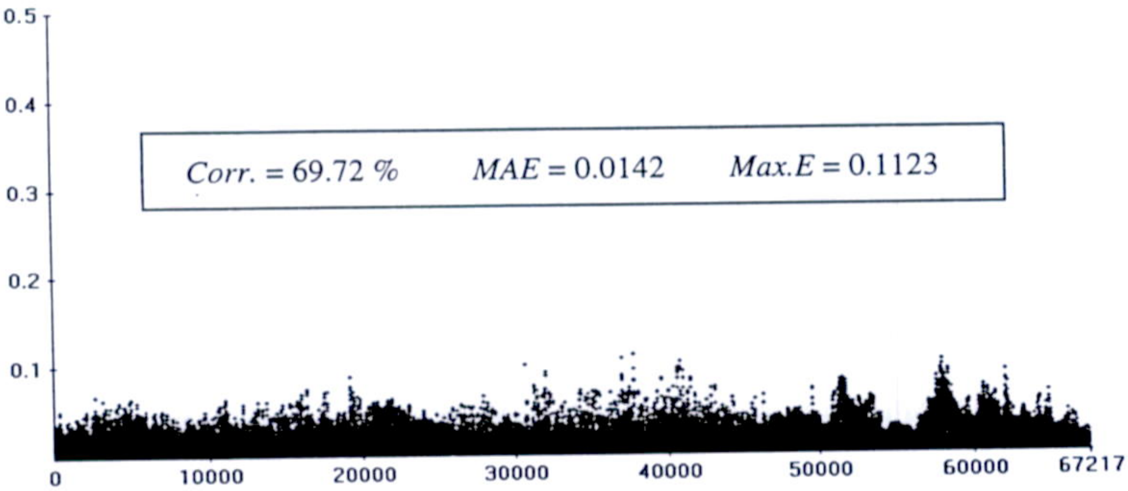


Figure A-10.10
Absolute Difference Between Actual and Predicted Signals

Experiment Definition

No.	: 11
Directory	: MBPN\TEST11
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 6721
Training Data	: T02.DAC
Sawtooth File	: ON
Training Method	: Normal training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.1
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
0	3.21	70.80	82.87
5	30.75	1.77	13.72
10	33.22	1.75	13.61
20	35.86	1.74	13.63
50	41.83	1.68	13.16
100	46.67	1.64	12.57

Table A-11.1
Experiment Results on *Training Set*

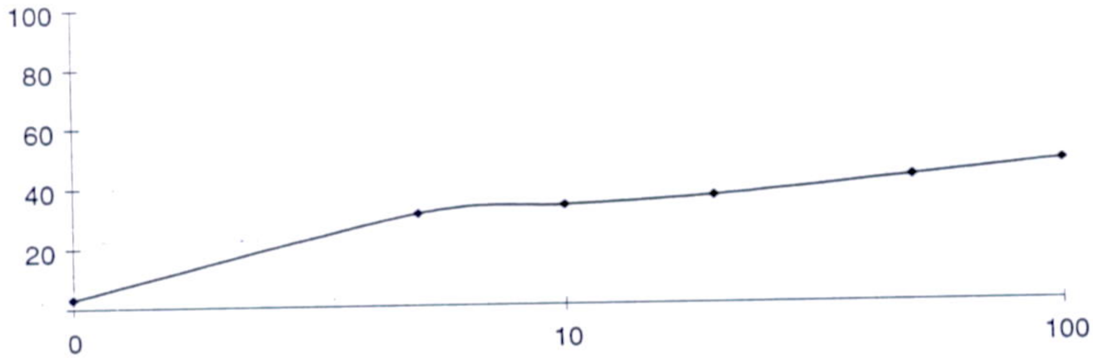


Figure A-11.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

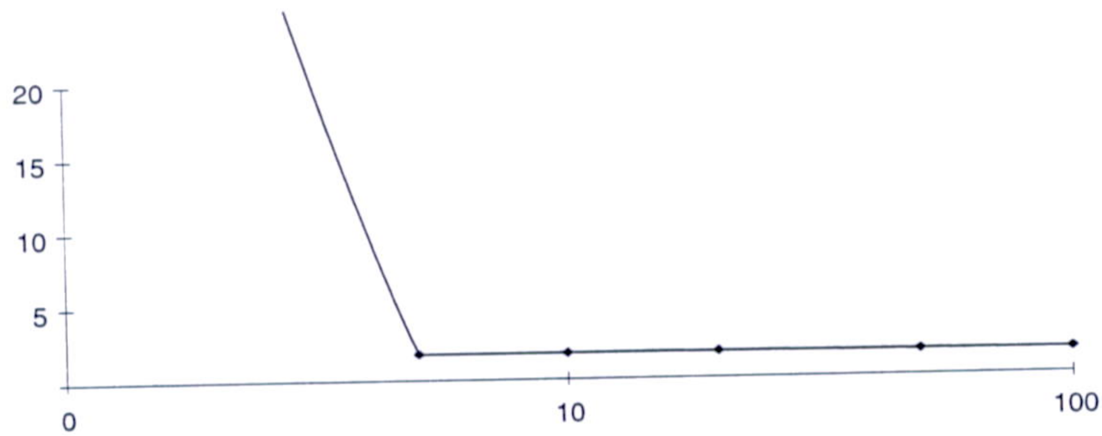


Figure A-11.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

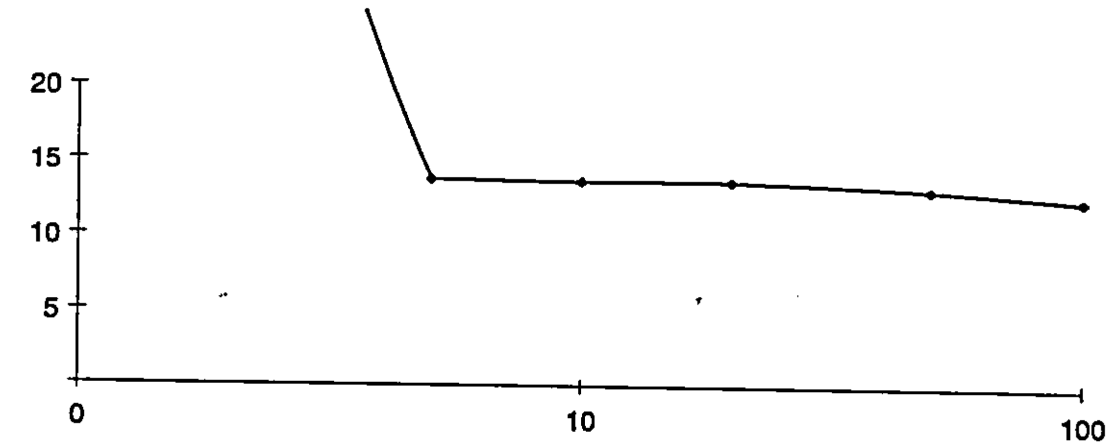


Figure A-11.3
Maximum Error Vs Number of Iterations for *Training Set*

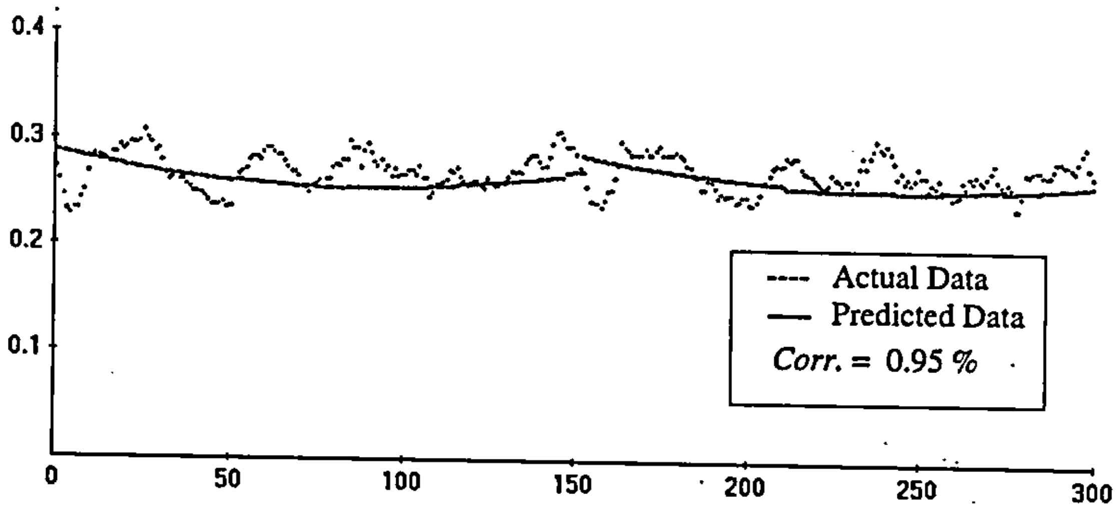


Figure A-11.4
Test Result after 100 Iterations for *First 300 Data*

Experiment Definition

No. : 12

Directory : MBPN\TEST12

Tool : HNC Explore Net

Applications : *MTrain* for training
MTest for testing

Application Parameters

Size of Input Set : 6721

Training Data : T02.DAC

Sawtooth File : ON

Training Method[1] : Fast training

Network Structure

Paradigm : Multi-Layer Backpropagation (*MBPN*)

Input Layer Size : 21

Output Layer Size : 1

No. of Hidden Layer : 2

Hidden Layer Sizes : 20-20

Input-Output Conn. : OFF

Training Parameters

Learning Method : Normal

Learning Rate (α) : 0.1

Activation Function : Gaussian
(Lower Limit = 0, Upper Limit =1, Steepness = 1)

Random Seed : 5

Initial Weight Max. : 0.5

Error Tolerance : 0

Lin.Act.Func.on Ou. : OFF

[1] Each one of ten sub-sets was presented to the network a proper number of times before the next one was presented, resulting in the necessary number of epochs at learning levels shown in Table A-12. First, each one was presented 5 times then 5 times again then 10 times then 30 times so on.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times10^{-2}$)	Maximum Error ($\times10^{-2}$)
0	3.21	70.80	82.87
5	30.26	2.91	14.29
10	32.79	2.94	14.36
20	35.13	2.84	14.31
50	40.73	2.53	14.06
100	45.47	2.21	13.58
200	50.50	2.10	12.83
500	53.03	1.79	14.52
1000	54.43	1.57	15.43

Table A-12.1
Experiment Results on *Training Set*

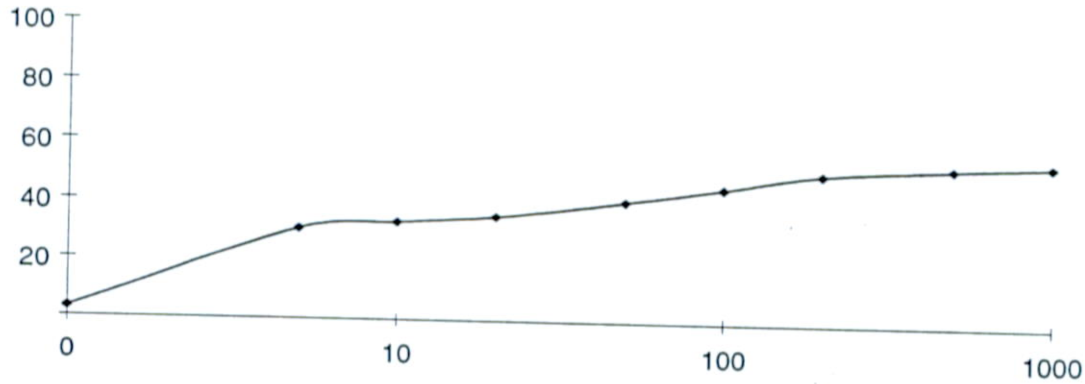


Figure A-12.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

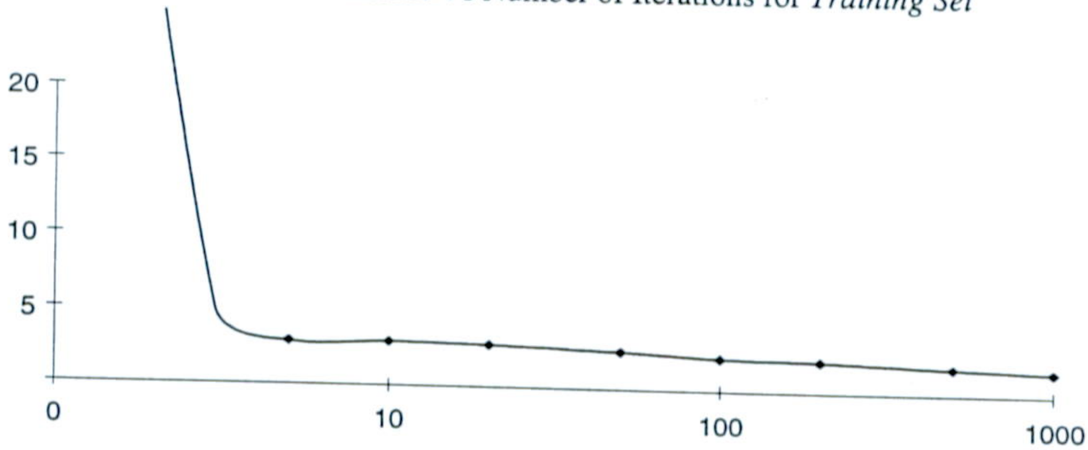


Figure A-12.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

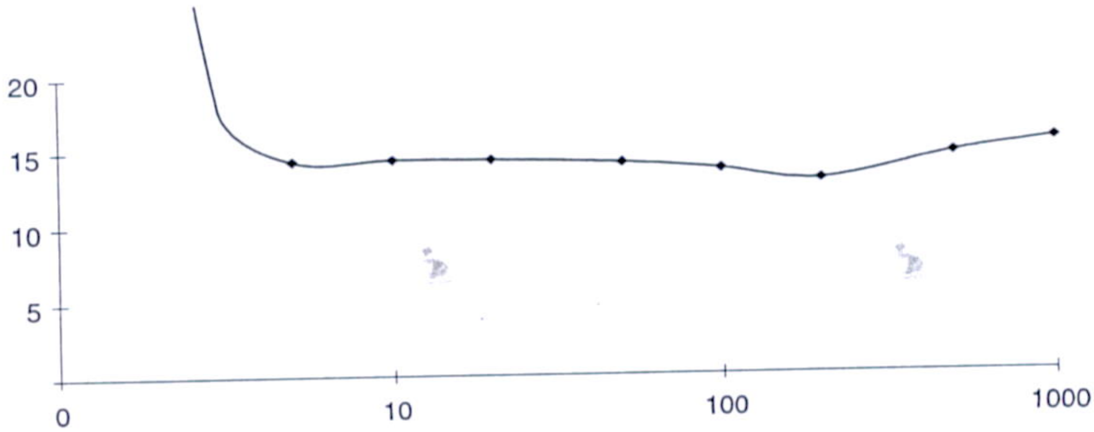


Figure A-12.3
Maximum Error Vs Number of Iterations for *Training Set*

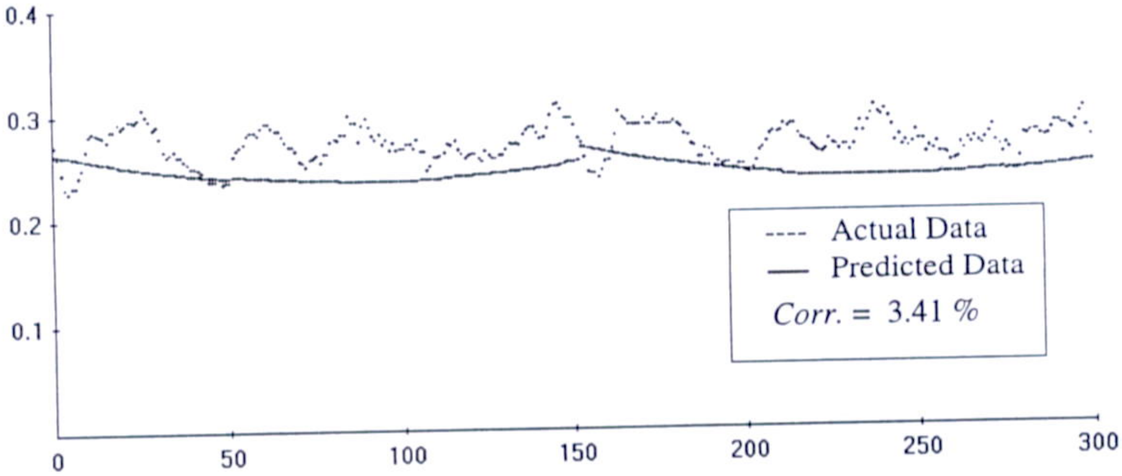


Figure A-12.4
Test Result after 100 Iterations for *First 300 Data*

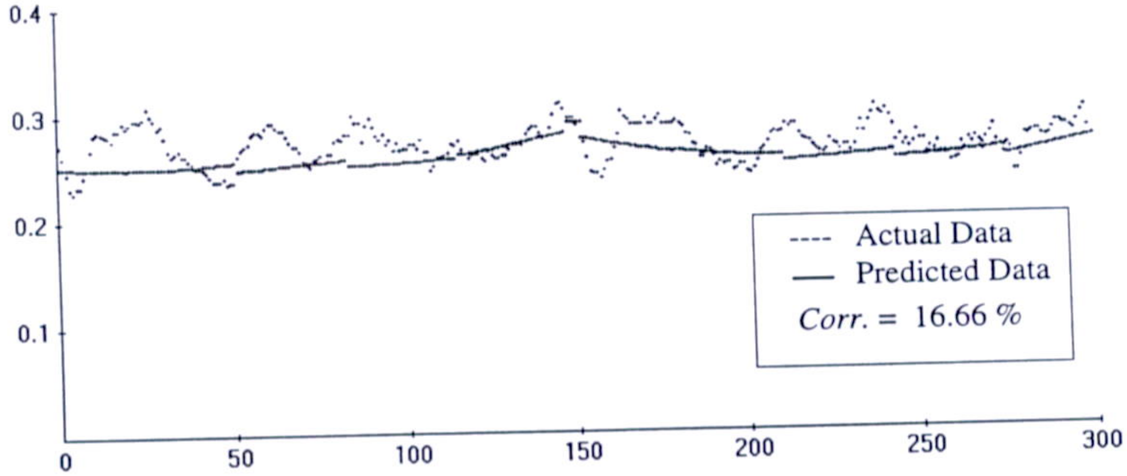


Figure A-12.5
Test Result after 1,000 Iterations for *First 300 Data*

Experiment Definition

No. [1]	: 12a
Directory	: MBPN\TEST12
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 6721
Training Data	: T02.DAC
Sawtooth File	: ON
Training Method [2]	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.1
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

---

[1] The status of 100 iterations of Experiment 12 was used as initial state in this experiment.

[2] Each one of ten sub-sets was presented to the network 100 times (not more) before others were shown at each attempt as opposed to Experiment 12.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($x10^{-2}$)	Maximum Error ($x10^{-2}$)
(Initial) 100	45.47	2.21	13.58
200	50.50	2.10	12.83
500	53.89	1.80	14.02
1000	55.66	1.57	14.63
2000	72.62	1.41	7.54
3000	75.64	1.32	8.30
4000	77.18	1.24	8.76
5000	78.61	1.18	8.90
6000	80.11	1.13	9.07
7000	80.98	1.11	8.74
8000	81.45	1.10	8.18
9000	81.90	1.08	7.70
10000	82.29	1.08	7.30
11000	82.65	1.07	6.99
12000	83.01	1.06	6.78
13000	83.36	1.06	6.75
14000	83.63	1.05	6.73
15000	83.87	1.05	6.72
20000	84.60	1.02	6.62

Table A-12a.1
Experiment Results on *Training Set*

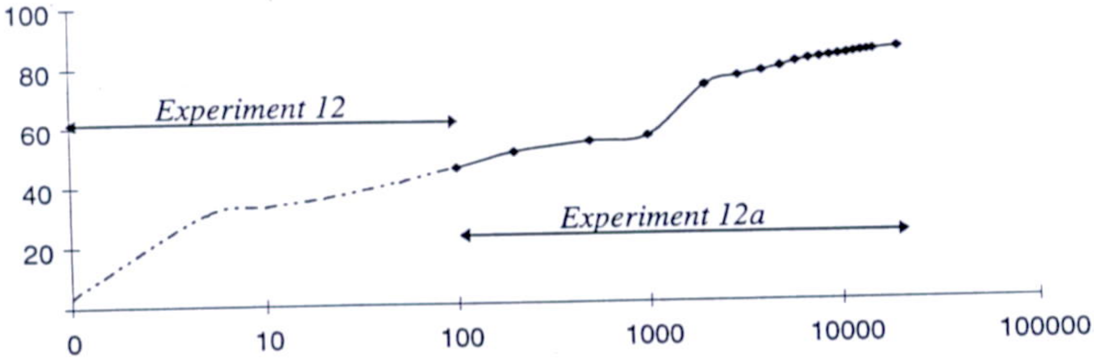


Figure A-12a.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

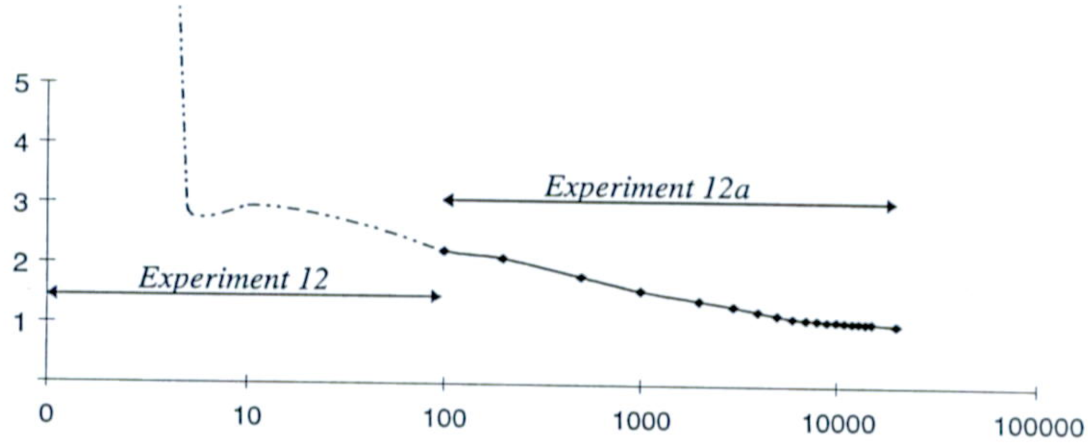


Figure A-12a.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

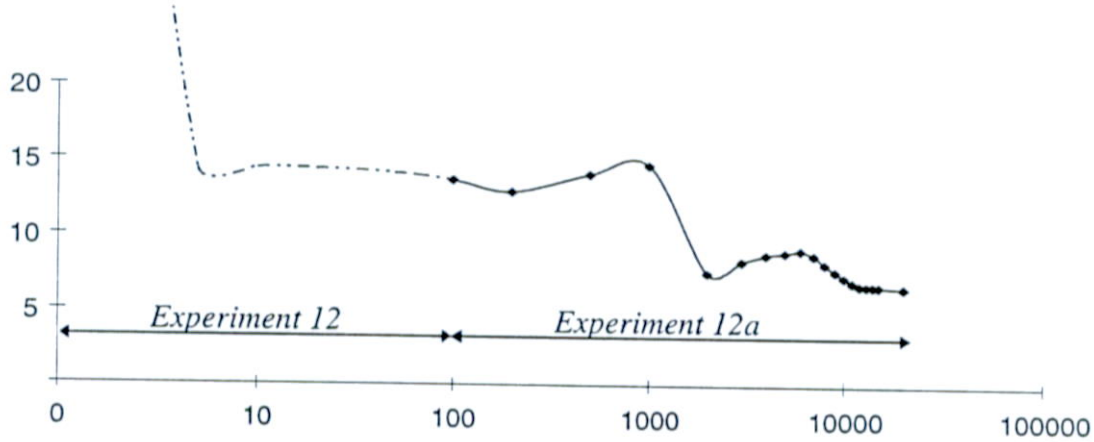


Figure A-12a.3
Maximum Error Vs Number of Iterations for *Training Set*

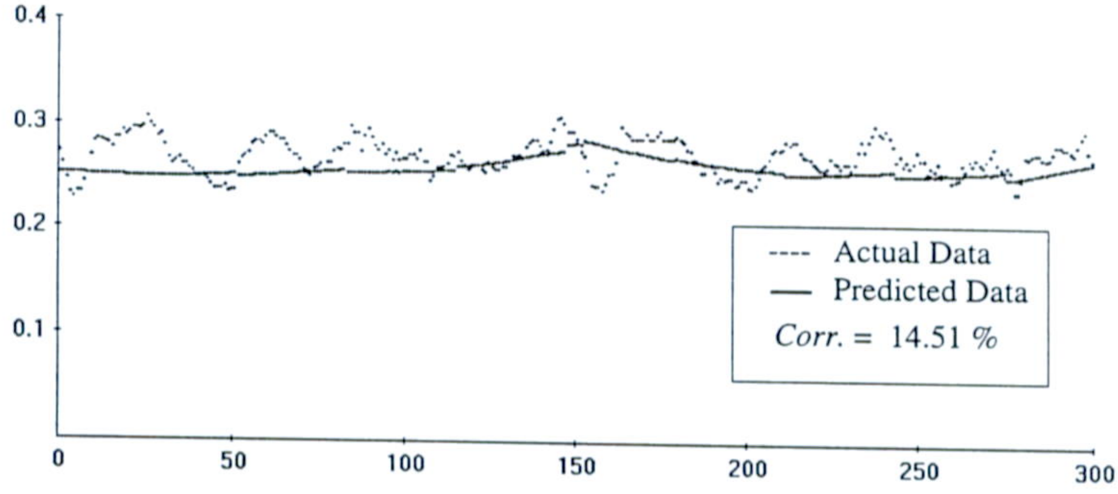


Figure A-12a.4
Test Result after 1,000 Iterations for *First 300 Data*

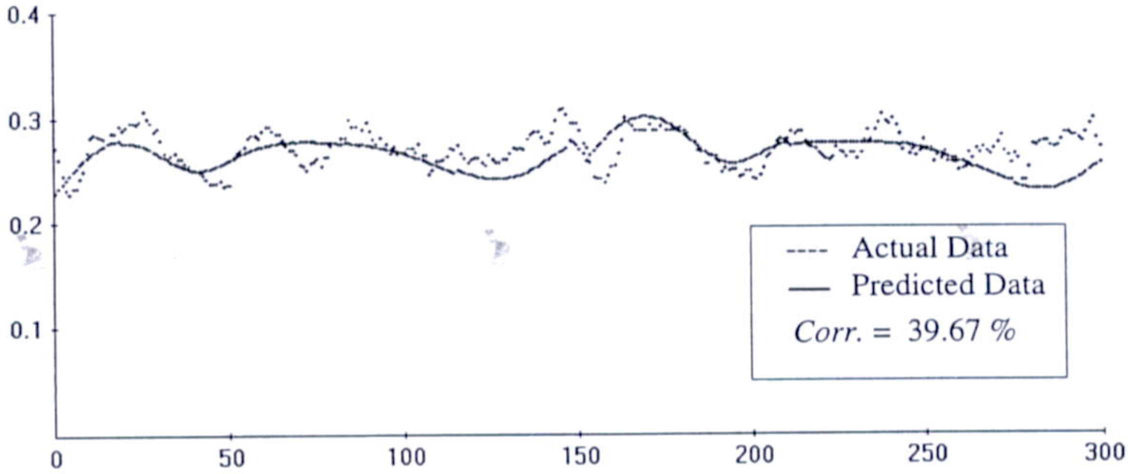


Figure A-12a.5
Test Result after 5,000 Iterations for *First 300 Data*

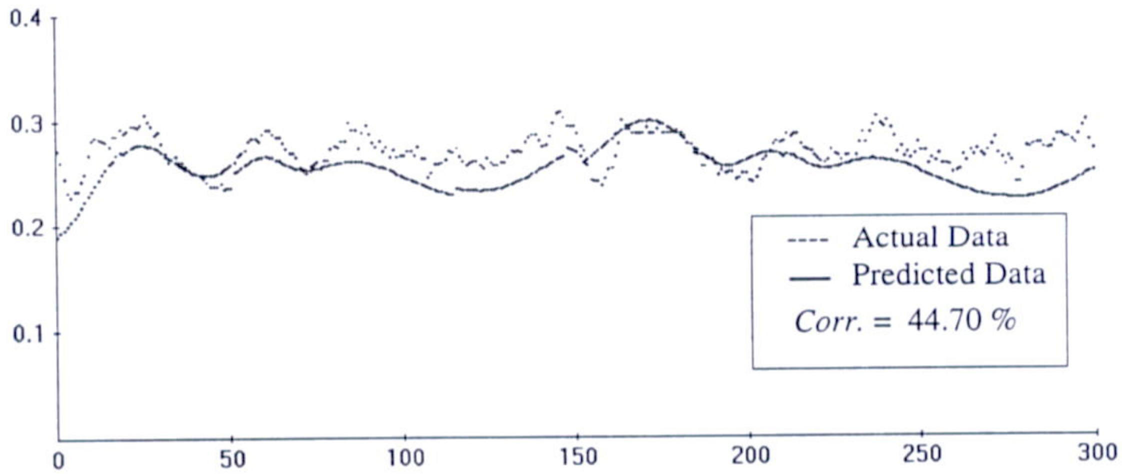


Figure A-12a.6
Test Result after 10,000 Iterations for *First 300 Data*

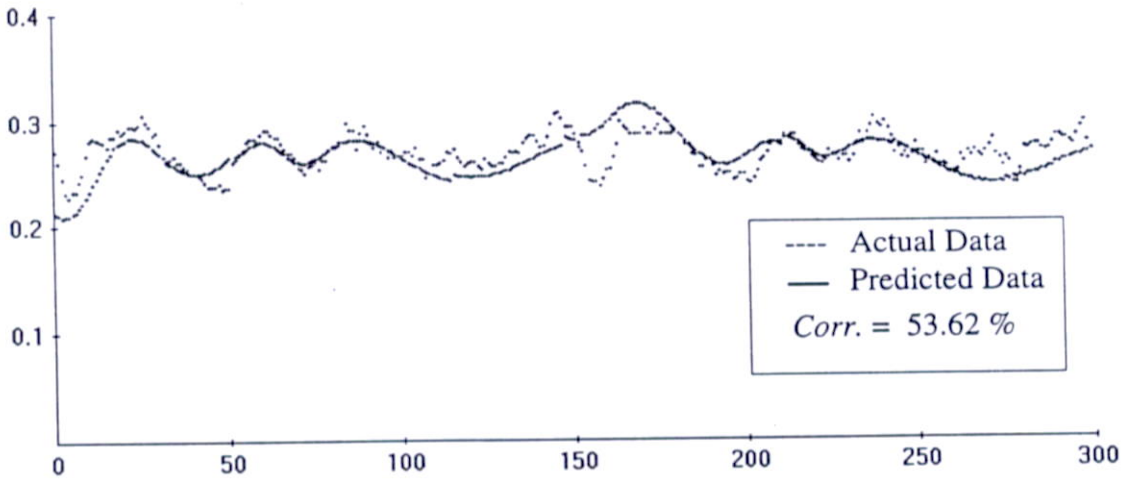


Figure A-12a.7
Test Result after 20,000 Iterations for *First 300 Data*

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
(Initial) 100	42.80	2.86	14.67
4000	68.72	1.40	10.88
5000	69.58	1.39	10.62
6000	69.59	1.42	11.03
10000	66.45	1.59	15.15
15000	67.81	1.52	12.32
20000	67.99	1.52	15.18

Table A-12a.2
Validation Results on *Whole Set*

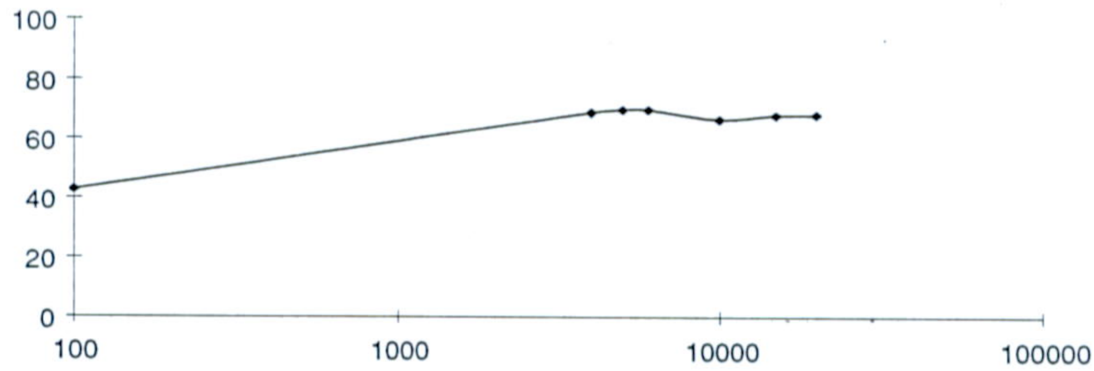


Figure A-12a.8
Correlation Coefficient Vs Number of Iterations for *Whole Set*

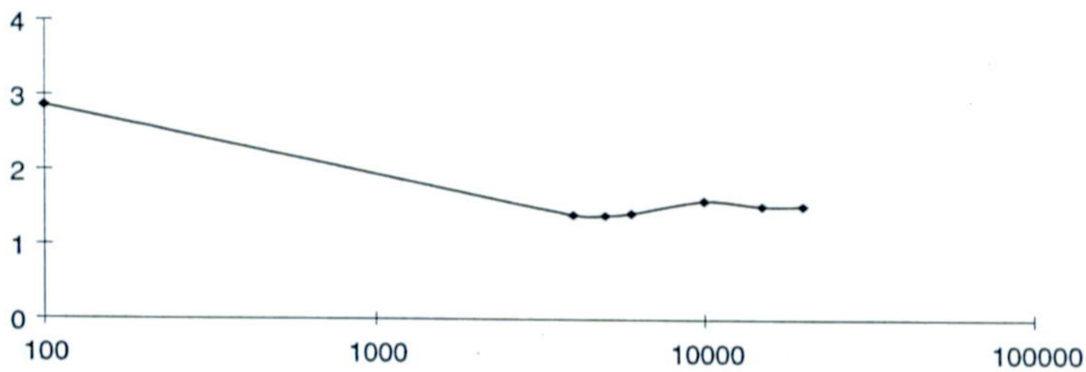


Figure A-12a.9
Mean Absolute Error Vs Number of Iterations for *Whole Set*

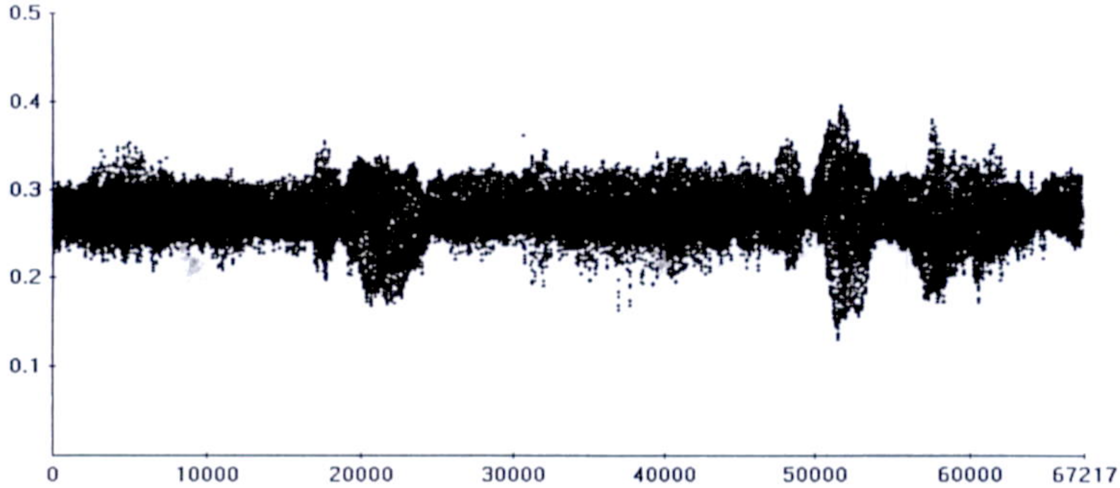


Figure A-12a.10
Actual Strain Signal

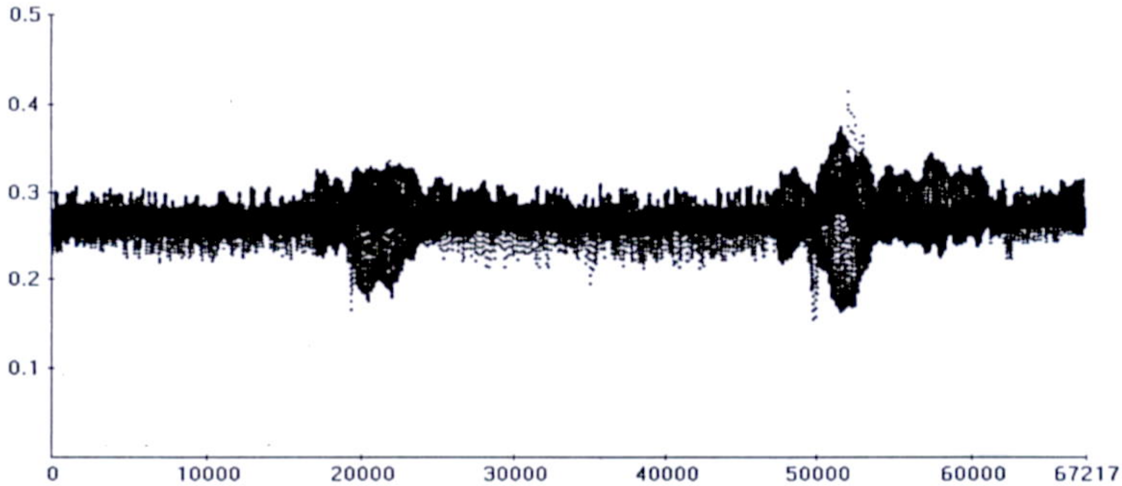


Figure A-12a.11
Predicted Signal after *5,000 Iterations*

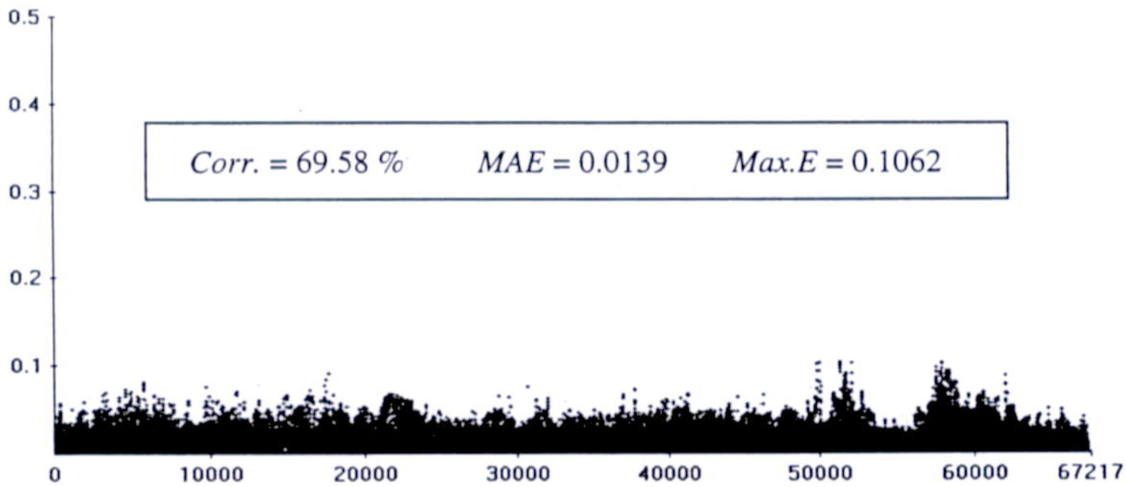


Figure A-12a.12
Absolute Difference Between Actual and Predicted Signals

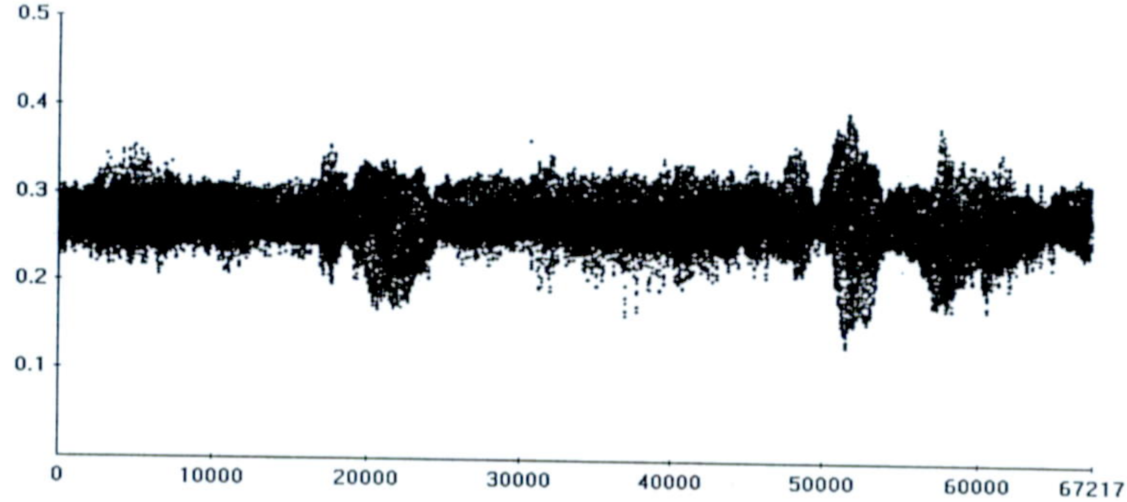


Figure A-12a.13
Actual Strain Signal

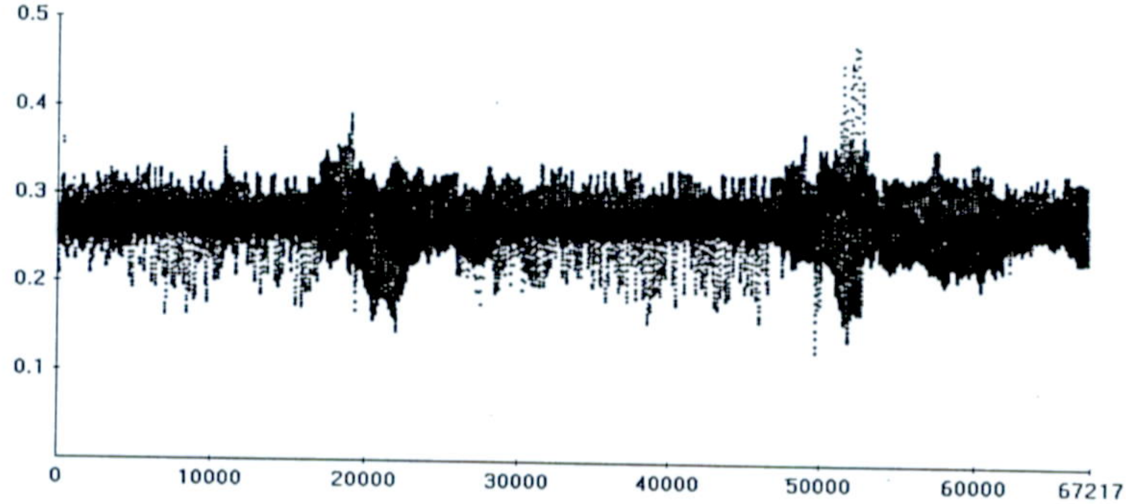


Figure A-12a.14
Predicted Signal after *20,000 Iterations*

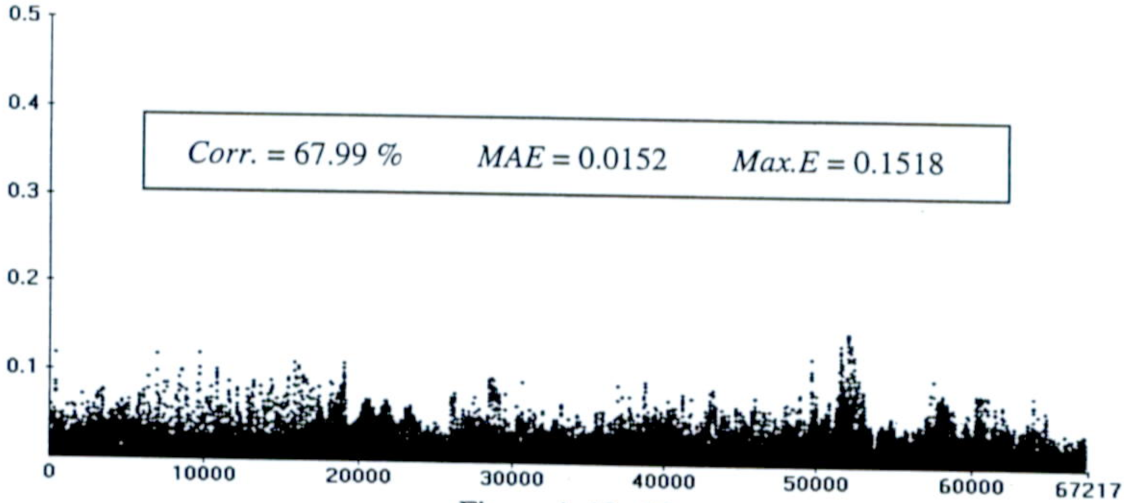


Figure A-12a.15
Absolute Difference Between Actual and Predicted Signals

Experiment Definition

No. [1] : 12b

Directory : MBPN\TEST12

Tool : HNC Explore Net

Applications : *MTrain* for training
MTest for testing

Application Parameters

Size of Input Set : 6721

Training Data : T02.DAC

Sawtooth File : ON

Training Method : Normal training

Network Structure

Paradigm : Multi-Layer Backpropagation (*MBPN*)

Input Layer Size : 21

Output Layer Size : 1

No. of Hidden Layer : 2

Hidden Layer Sizes : 20-20

Input-Output Conn. : OFF

Training Parameters

Learning Method : Normal

Learning Rate (α) : 0.1

Activation Function : Gaussian
(Lower Limit = 0, Upper Limit =1, Steepness = 1)

Random Seed : 5

Initial Weight Max.[2] : 0.5

Error Tolerance : 0

Lin.Act.Func.on Ou. : OFF

---

[1] The status of 20,000 iterations of Experiment 12a was used as initial state in this experiment.
[2] Initial weight maximum had no function here as initial weights were used.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
Initial	84.60	1.02	6.62
100	85.80	0.96	6.75
200	85.88	0.96	6.75
300	85.87	0.96	6.75
400	85.89	0.96	6.74
500	85.88	0.96	6.74
600	85.95	0.96	6.74
700	85.98	0.96	6.74
800	85.96	0.96	6.74
900	85.98	0.96	6.74
1000	86.02	0.96	6.74
1100	86.02	0.96	6.74
1200	86.05	0.95	6.74
1300	86.06	0.95	6.74
1400	86.06	0.95	6.74
1500	86.10	0.95	6.73

Table A-12b.1
Experiment Results on *Training Set*

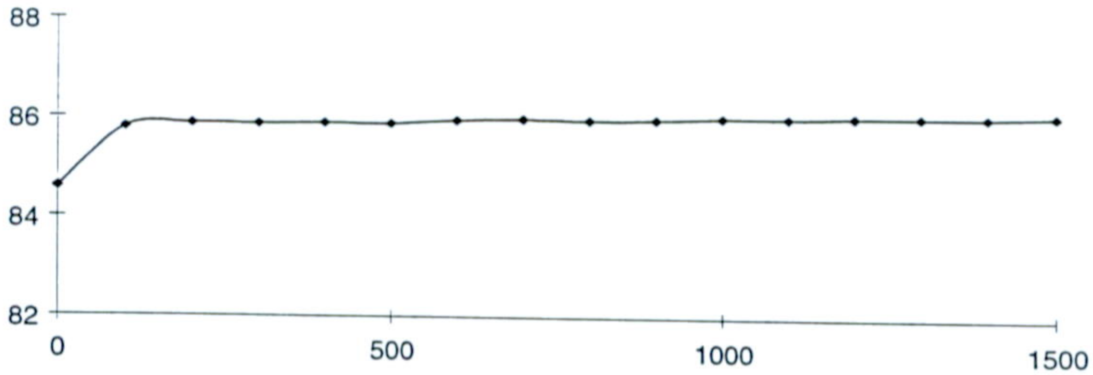


Figure A-12b.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

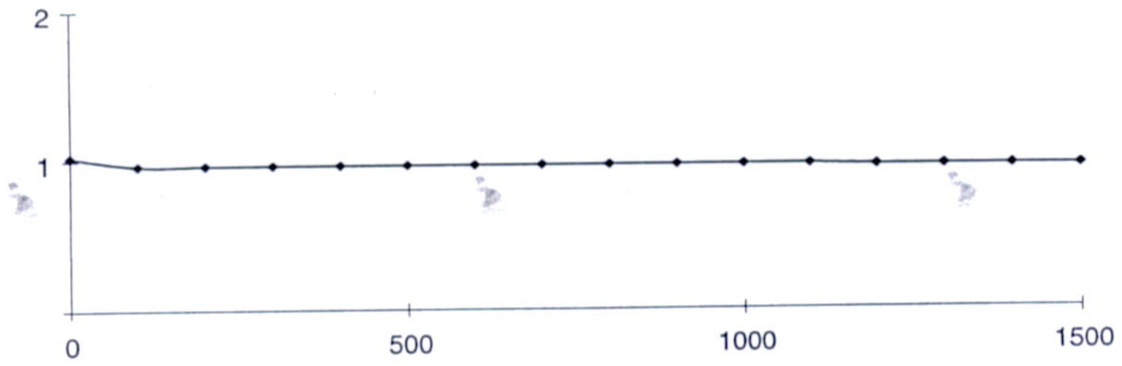


Figure A-12b.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

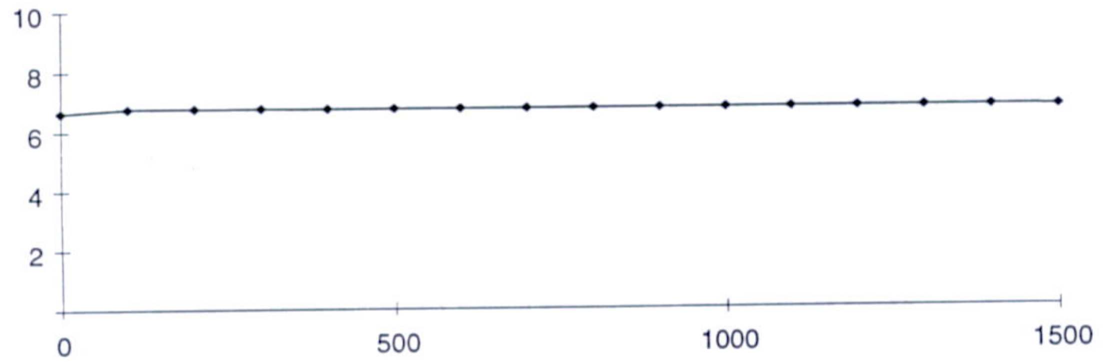


Figure A-12b.3
Maximum Error Vs Number of Iterations for *Training Set*

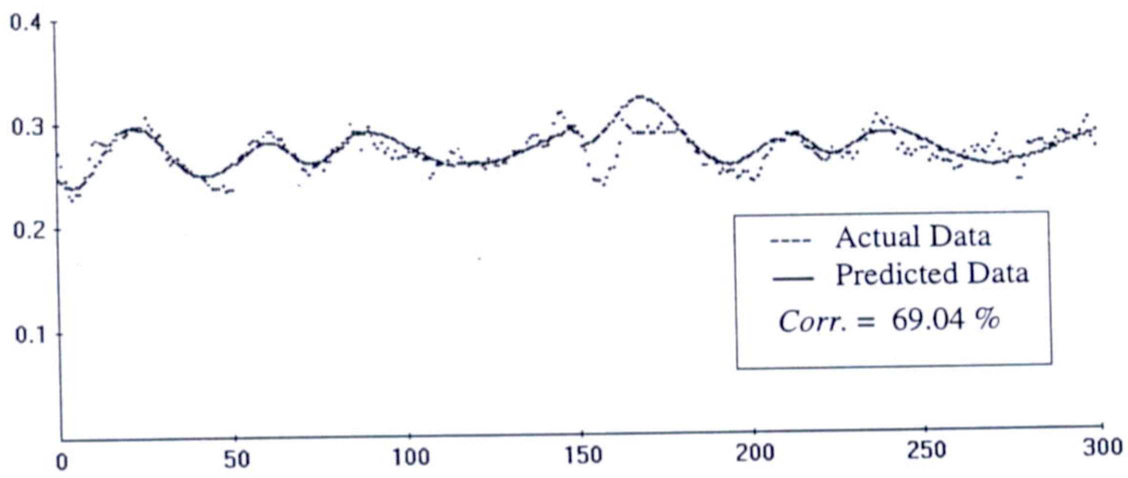


Figure A-12b.4
Test Result after 1,500 Iterations for *First 300 Data*

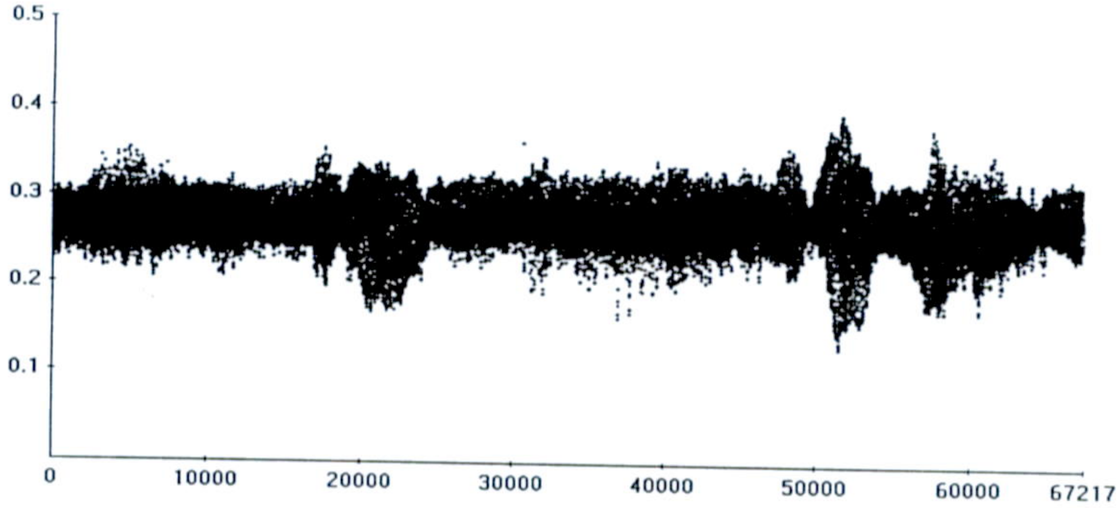


Figure A-12b.5
Actual Strain Signal

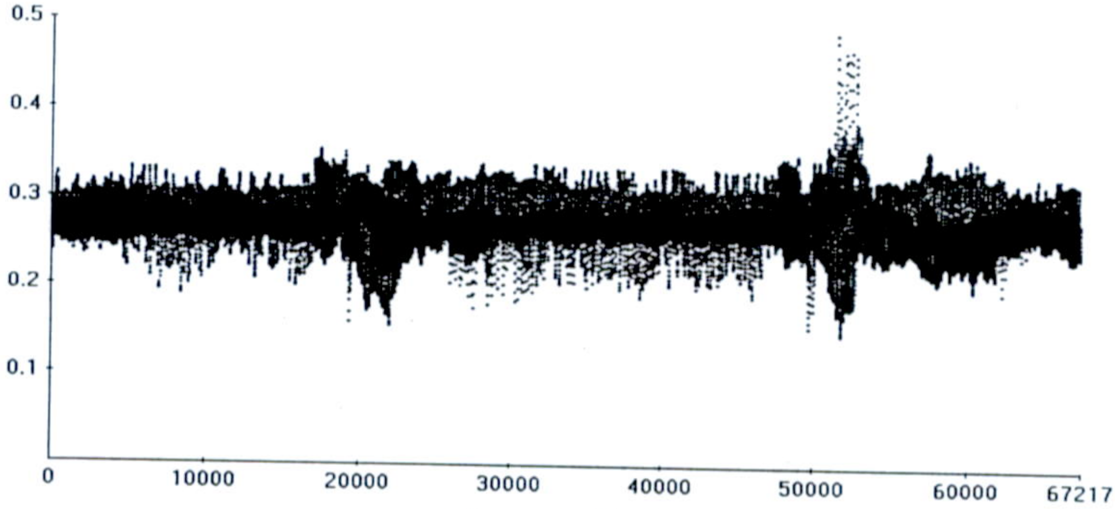


Figure A-12b.6
Predicted Signal after *1,500 Iterations (Total epochs = 21,500)*

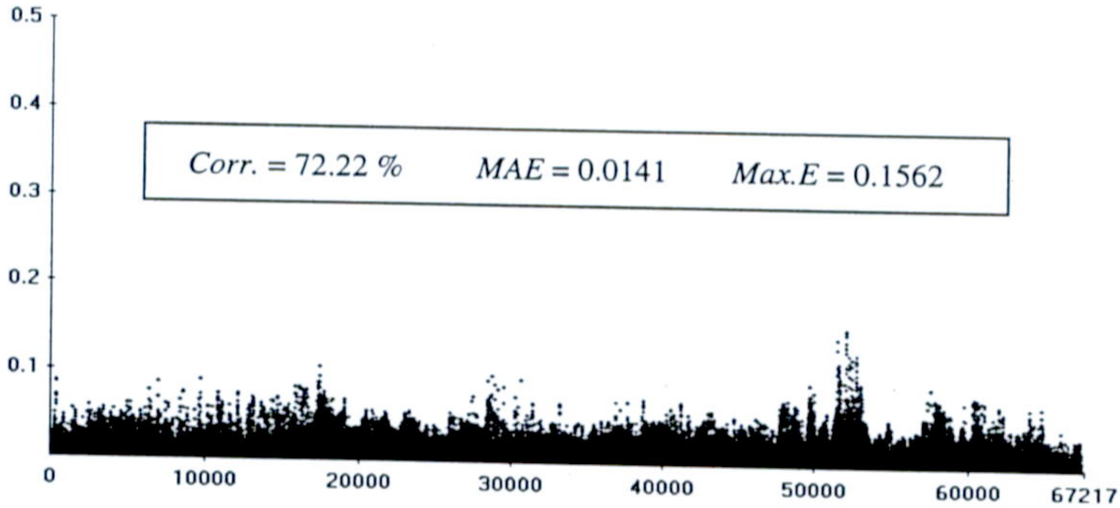


Figure A-12b.7
Absolute Difference Between Actual and Predicted Signals

Experiment Definition

No. [1]	: 12c
Directory	: MBPN\TEST12
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set [2]	: 67217
Training Data	: V02.DAC (Scrambled)
Sawtooth File	: ON
Training Method	: Normal training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.1
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max. [3]	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

---

[1] The status of 5,000 iterations of Experiment 12a was used as initial state in this experiment.

[2] Whole set was used in training.

[3] Initial weight maximum had no function here as initial weights were used.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($x10^{-2}$)	Maximum Error ($x10^{-2}$)
Initial	69.58	1.39	10.62
100	79.05	1.18	11.77
200	80.13	1.13	11.28
500	82.26	1.08	9.02
600	82.37	1.08	8.96
1000	82.92	1.08	9.44

Table A-12c.1
Experiment Results on *Whole Set*

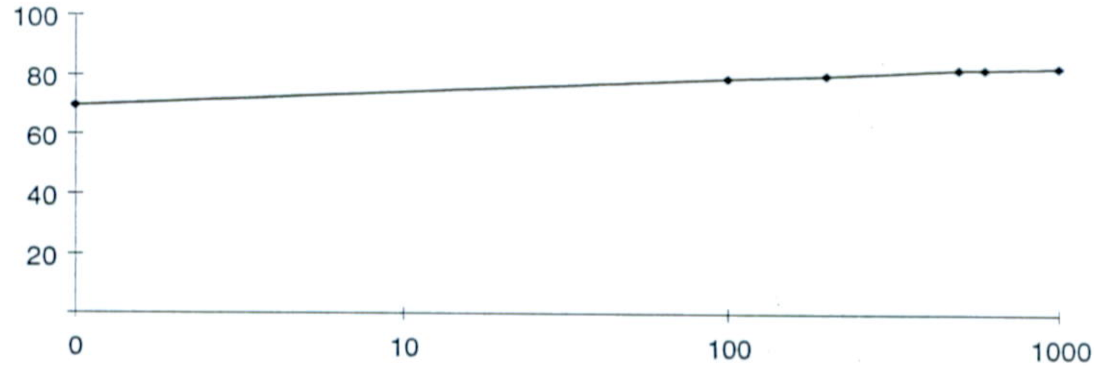


Figure A-12c.1
Correlation Coefficient Vs Number of Iteration for *Whole Set*

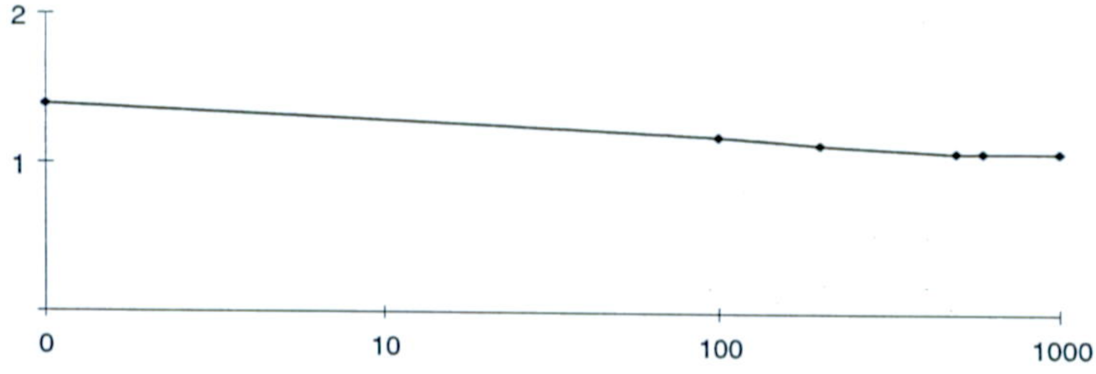


Figure A-12c.2
Mean Absolute Error Vs Number of Iteration for *Whole Set*

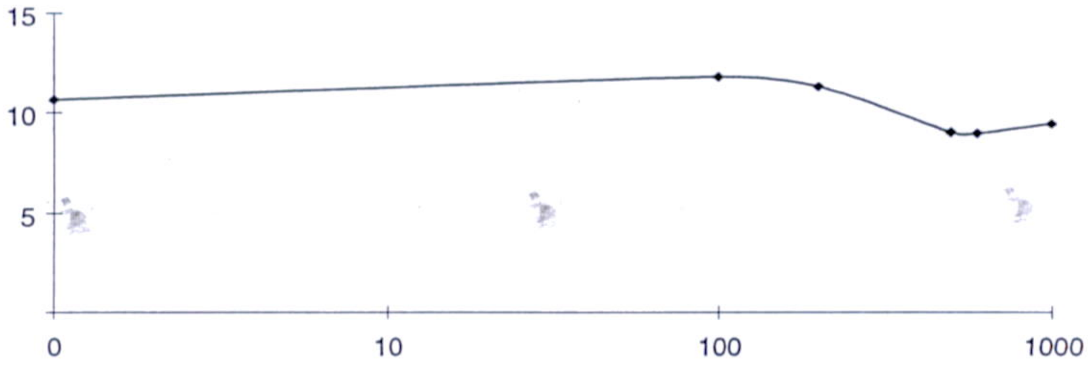


Figure A-12c.3
Maximum Error Vs Number of Iteration for *Whole Set*

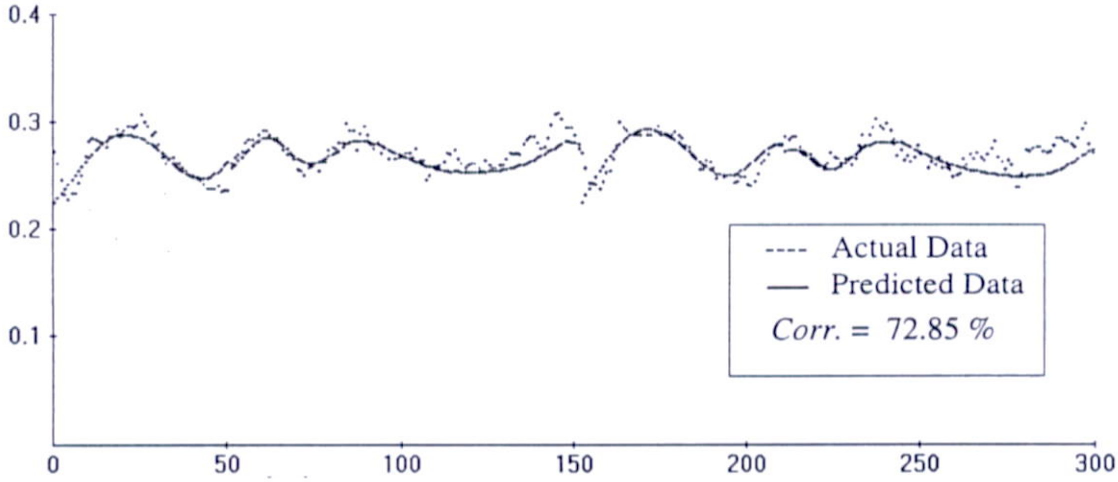


Figure A-12c.4
Test Result after 100 Iterations for *First 300 Data*

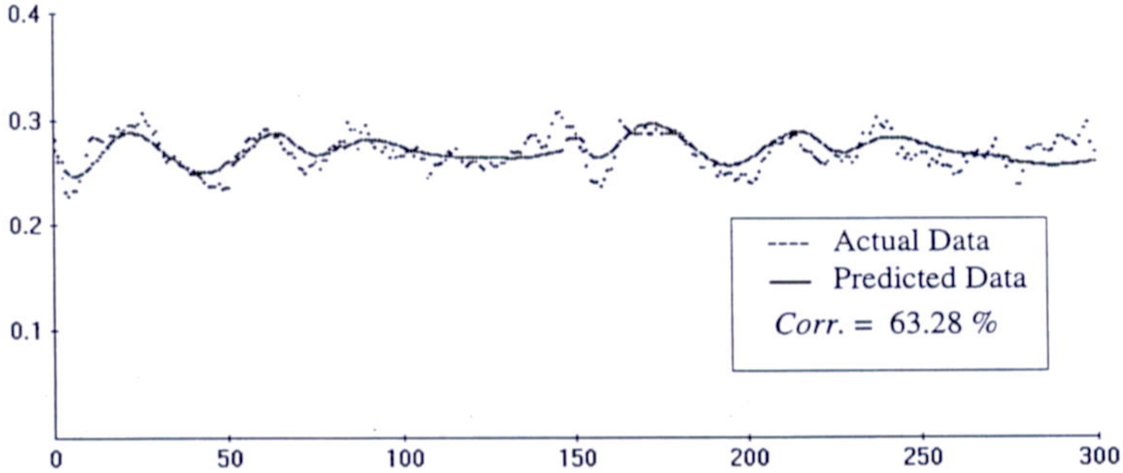


Figure A-12c.5
Test Result after 1,000 Iterations for *First 300 Data*

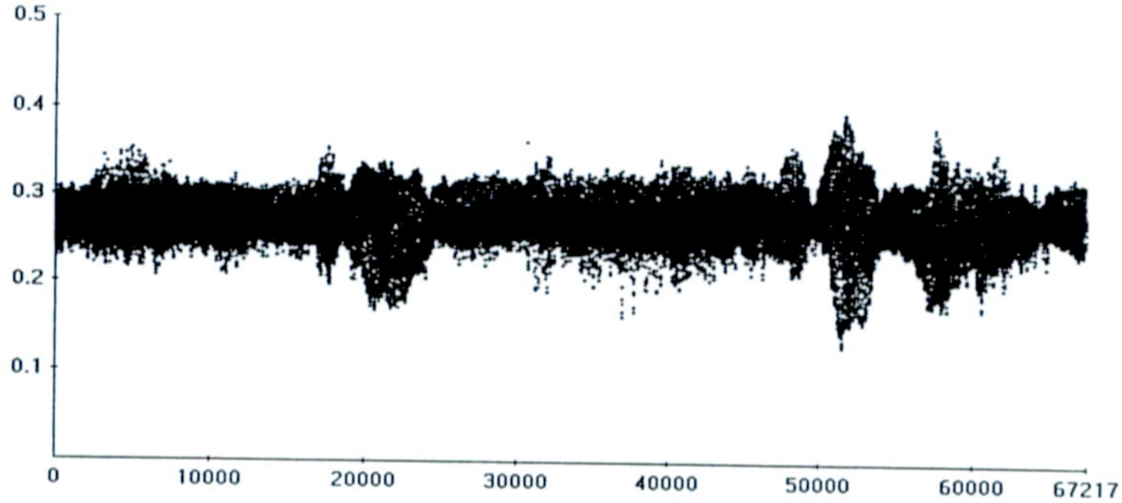


Figure A-12c.6
Actual Strain Signal

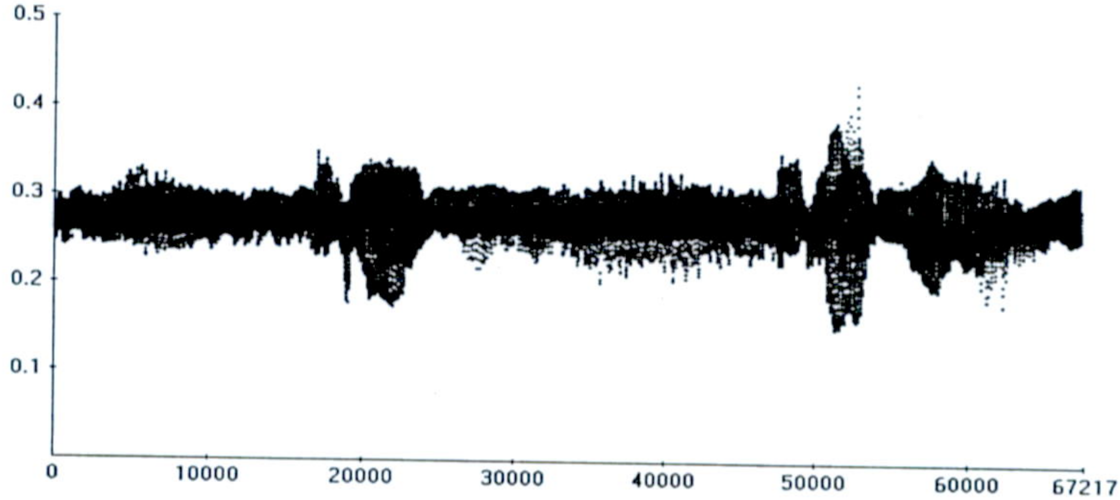


Figure A-12c.7
Predicted Signal after *1,000 Iterations*

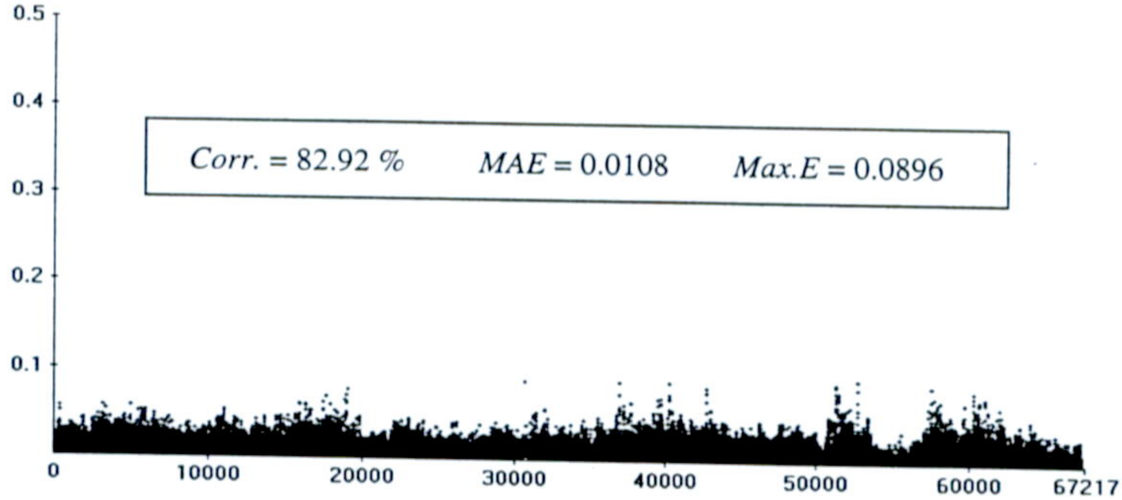


Figure A-12c.8
Absolute Difference Between Actual and Predicted Signals

Experiment Definition

No. [1]	: 13
Directory	: MBPN\TEST13
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 6721
Training Data	: T02.DAC
Sawtooth File	: ON
Training Method [2]	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.5
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max. [3]	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

[1] The status of 200,000 iterations of Experiment 9 was used as initial state in this experiment.
[2] Each one of ten sub-sets was presented to the network as in Experiment 12 until 100 iterations. Then, they were presented as in Experiment 12a.
[3] Initial weight maximum had no function here as initial weights were used.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
0	- 3.73	4.65	22.85
5	32.93	1.86	13.23
10	35.76	1.97	12.89
20	43.60	1.86	13.45
50	53.68	1.57	12.59
100	61.48	1.50	11.80
200	69.41	1.53	8.34
500	72.60	1.43	12.59
1000	75.93	1.35	8.72
2000	78.12	1.18	9.25
3000	77.81	1.17	11.48
4000	78.13	1.16	15.27
5000	78.91	1.15	14.67
6000	79.42	1.14	15.61
7000	80.36	1.11	16.47
8000	80.87	1.09	16.42
9000	81.34	1.08	14.50
10000	81.85	1.07	14.11
11000	82.16	1.06	14.43
12000	82.17	1.06	15.28
13000	82.43	1.05	15.97
14000	82.72	1.05	15.54
15000	83.07	1.04	14.52
19000	84.14	1.03	9.70
20000	84.27	1.02	9.82

Table A-13.1
Experiment Results on *Training Set*

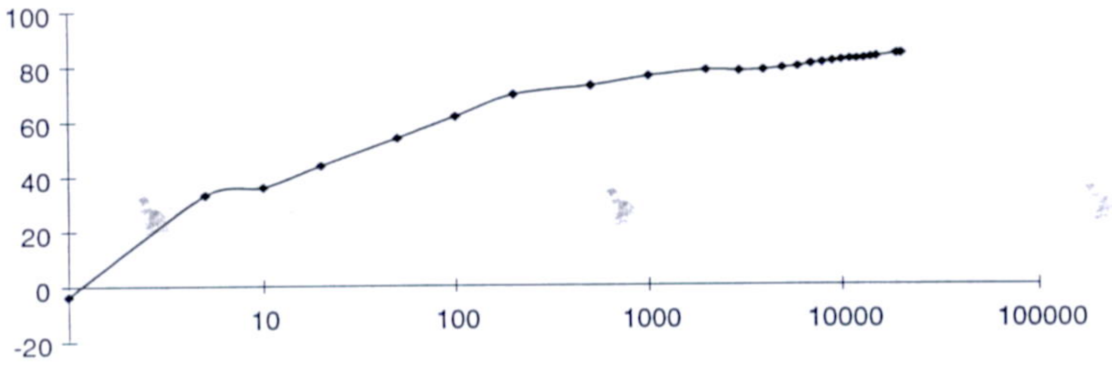


Figure A-13.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

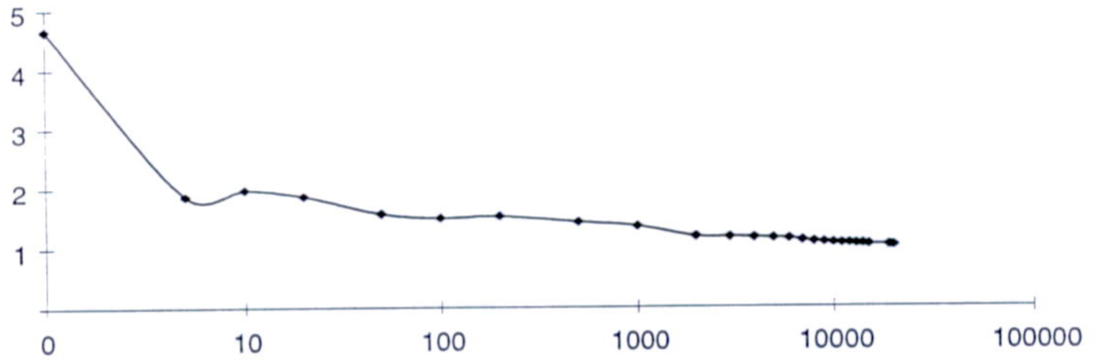


Figure A-13.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

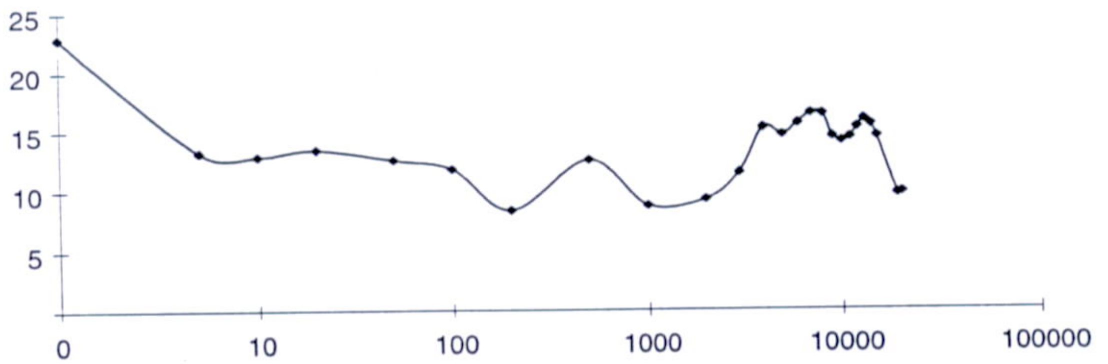


Figure A-13.3
Maximum Error Vs Number of Iterations for *Training Set*

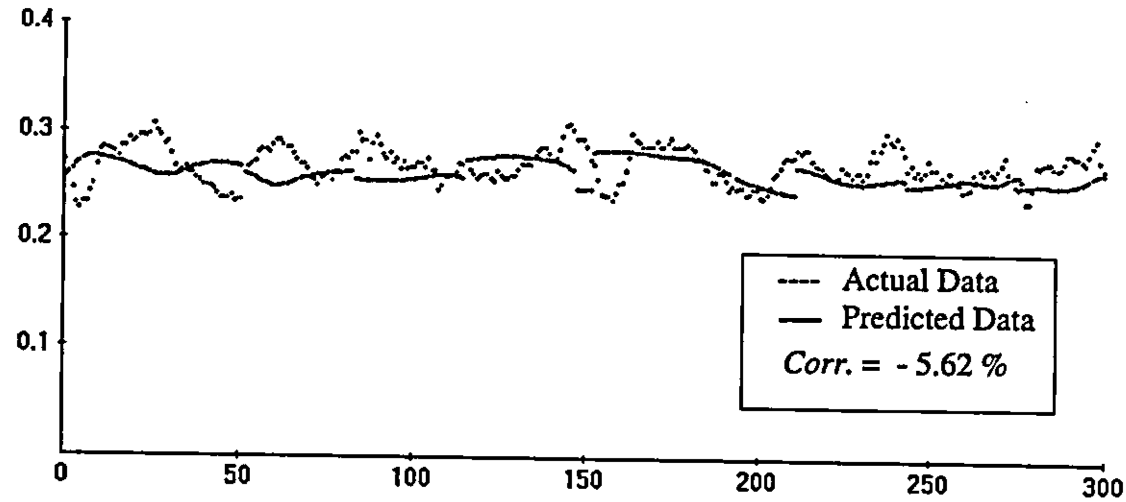


Figure A-13.4
Test Result after 10 Iterations for First 300 Data

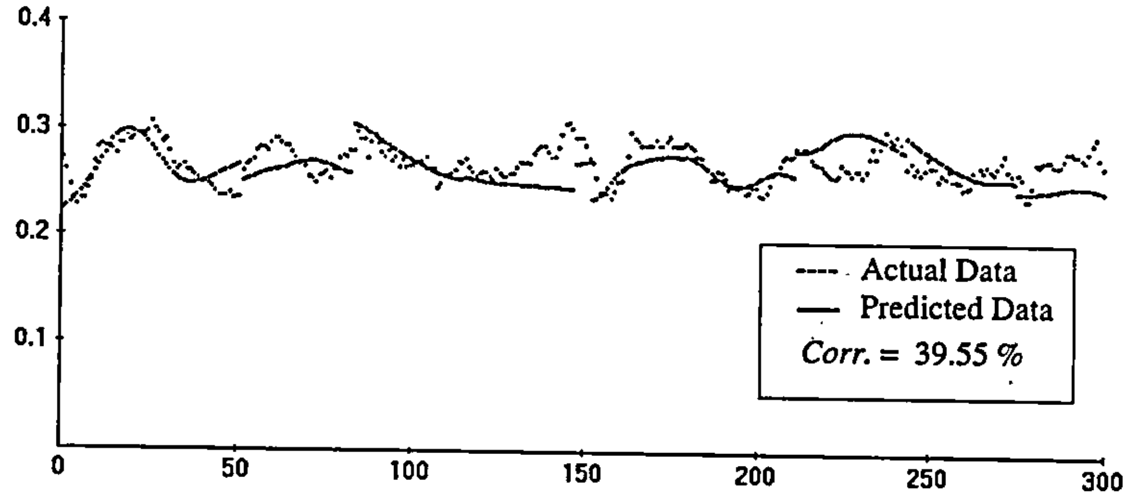


Figure A-13.5
Test Result after 100 Iterations for First 300 Data

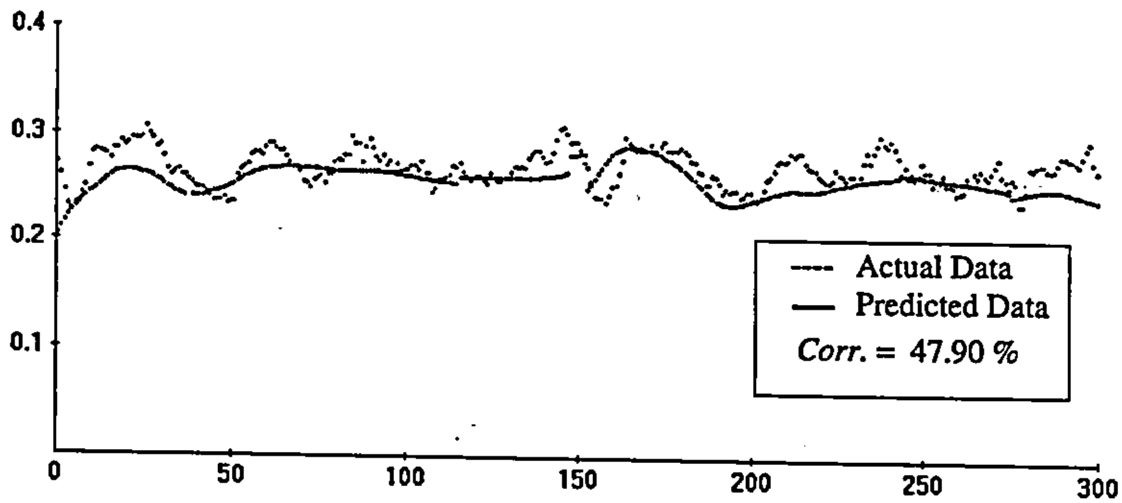


Figure A-13.6
Test Result after 1,000 Iterations for First 300 Data

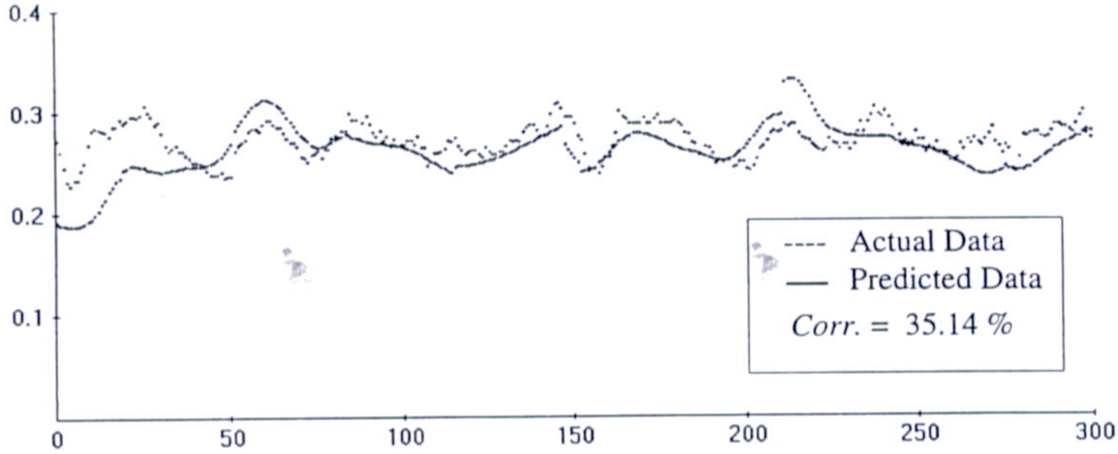


Figure A-13.7
Test Result after 10,000 Iterations for First 300 Data

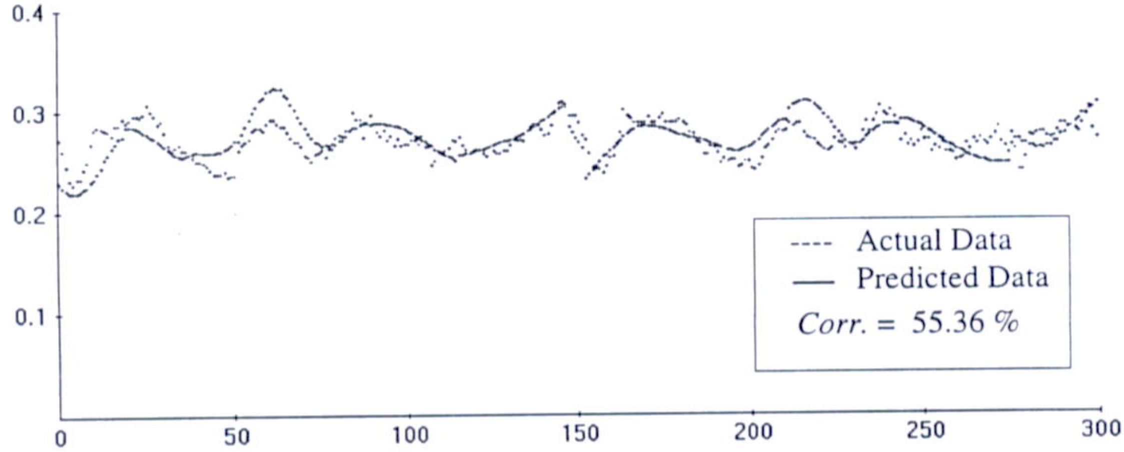


Figure A-13.8
Test Result after 20,000 Iterations for First 300 Data

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($x10^{-2}$)	Maximum Error ($x10^{-2}$)
0	0.86	4.85	24.69
10	36.64	2.18	14.21
100	51.46	1.83	13.83
1000	63.46	1.75	16.18
5000	52.96	2.41	44.10
10000	46.85	2.36	55.93
15000	51.64	2.14	30.72
20000	50.71	2.08	30.35

Table A-13.2
Validation Results on *Whole Set*

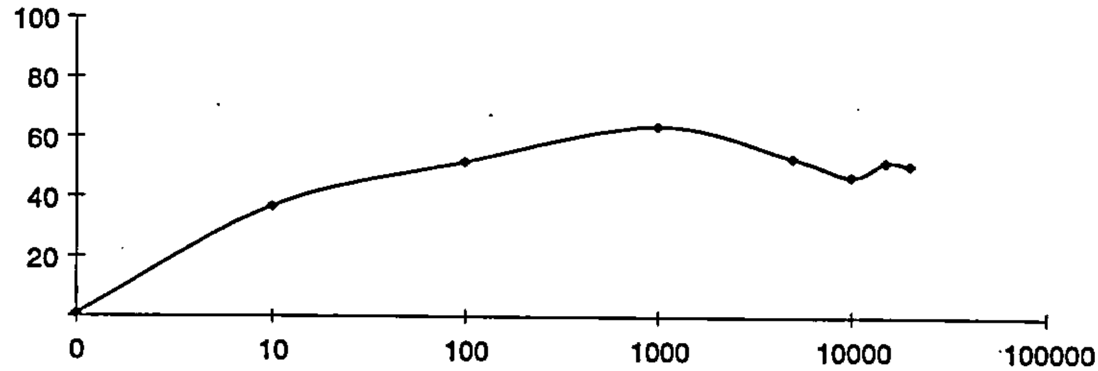


Figure A-13.9
Correlation Coefficient Vs Number of Iterations for *Whole Set*

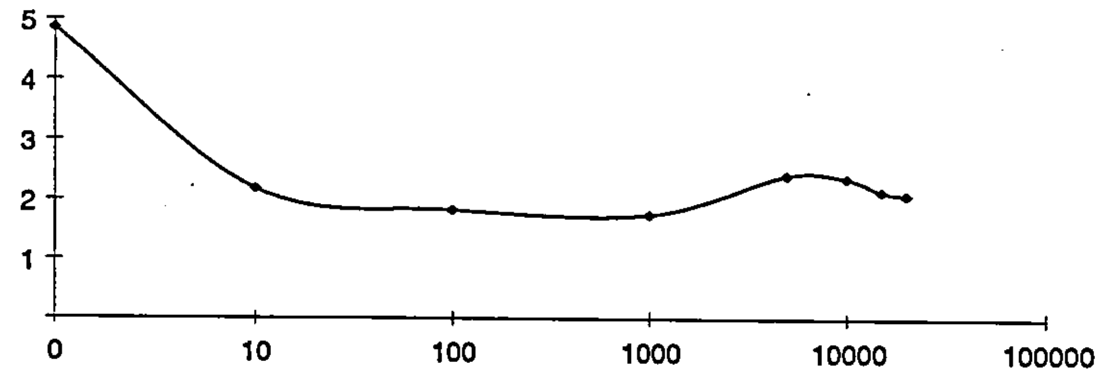


Figure A-13.10
Mean Absolute Error Vs Number of Iterations for *Whole Set*

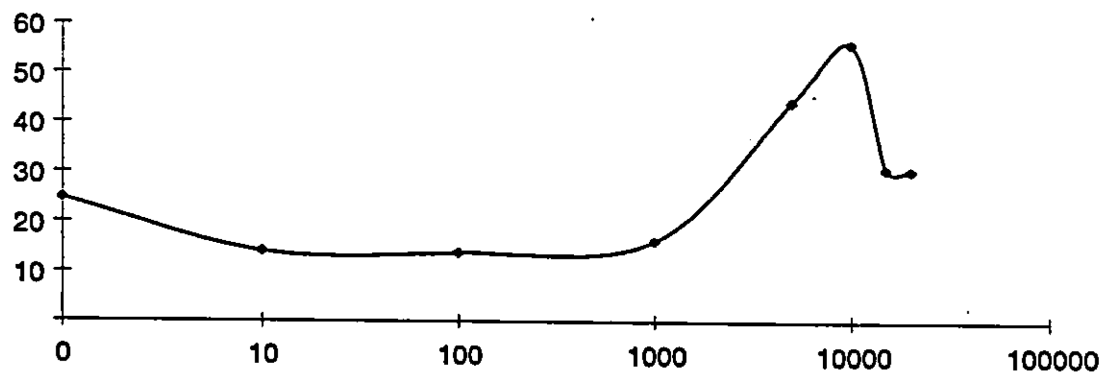


Figure A-13.11
Maximum Error Vs Number of Iterations for *Whole Set*

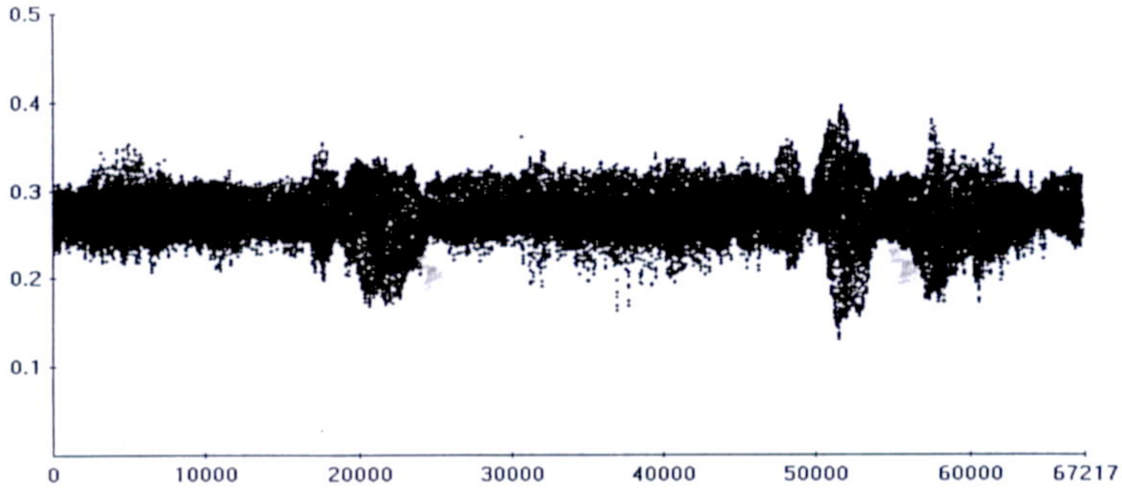


Figure A-13.12
Actual Strain Signal

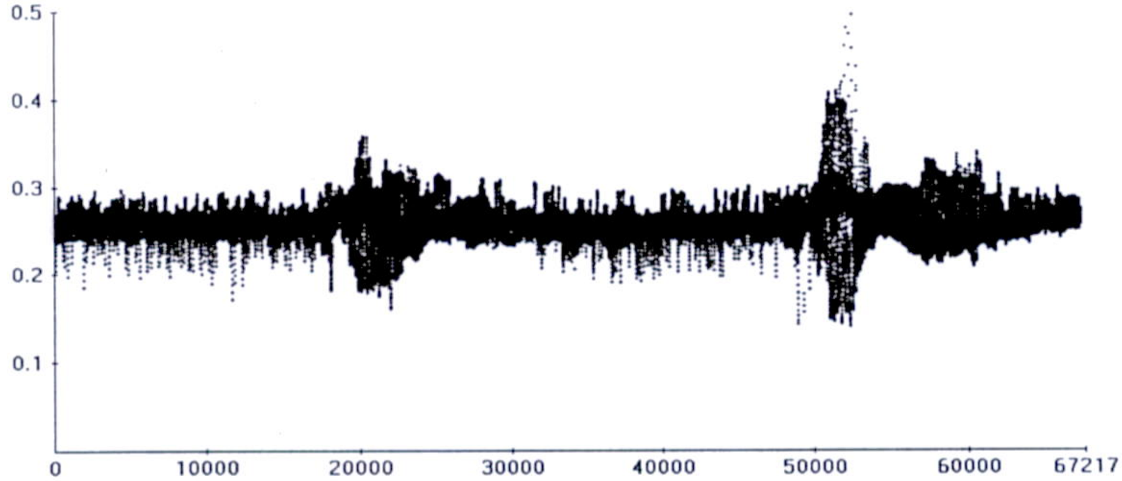


Figure A-13.13
Predicted Signal after *1,000 Iterations*

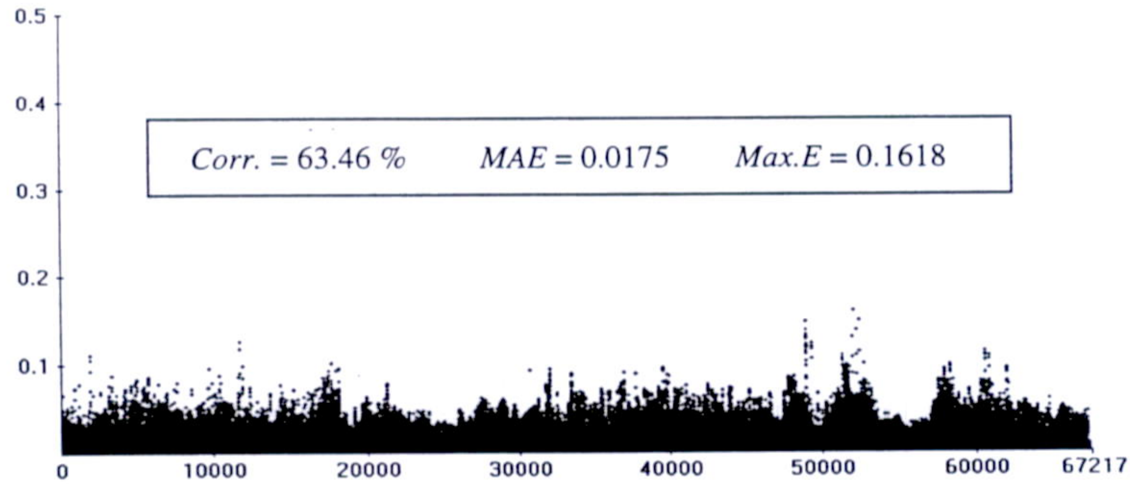


Figure A-13.14
Absolute Difference Between Actual and Predicted Signals

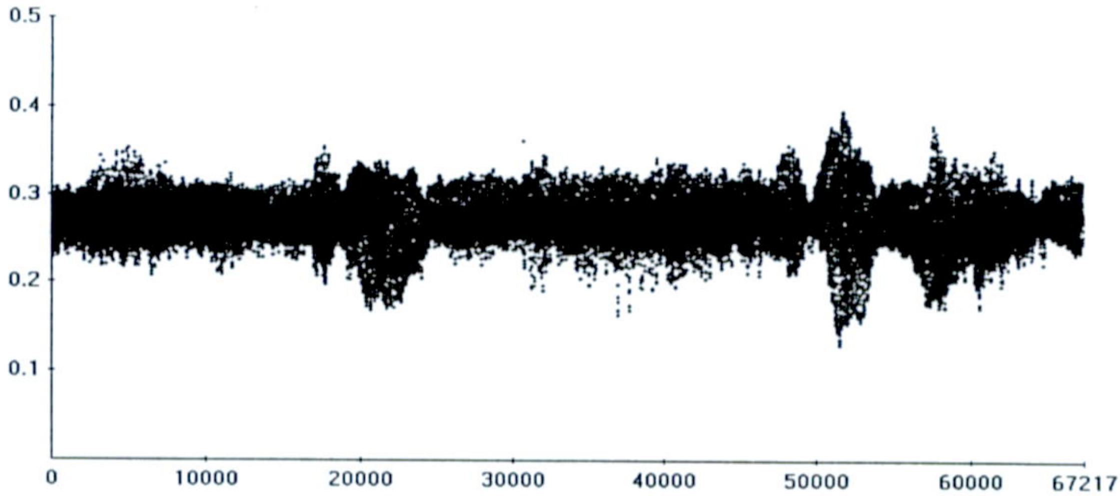


Figure A-13.15
Actual Strain Signal

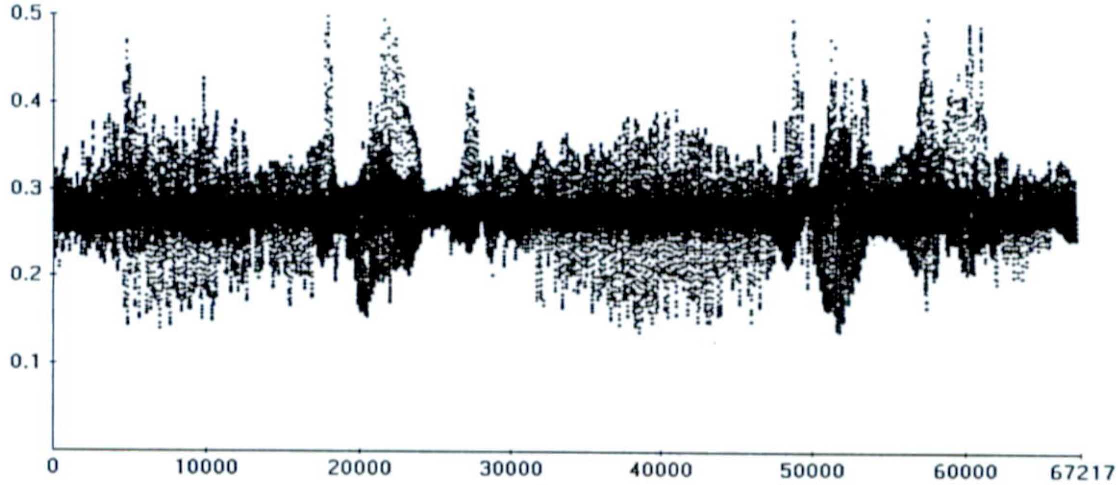


Figure A-13.16
Predicted Signal after *20,000 Iterations*

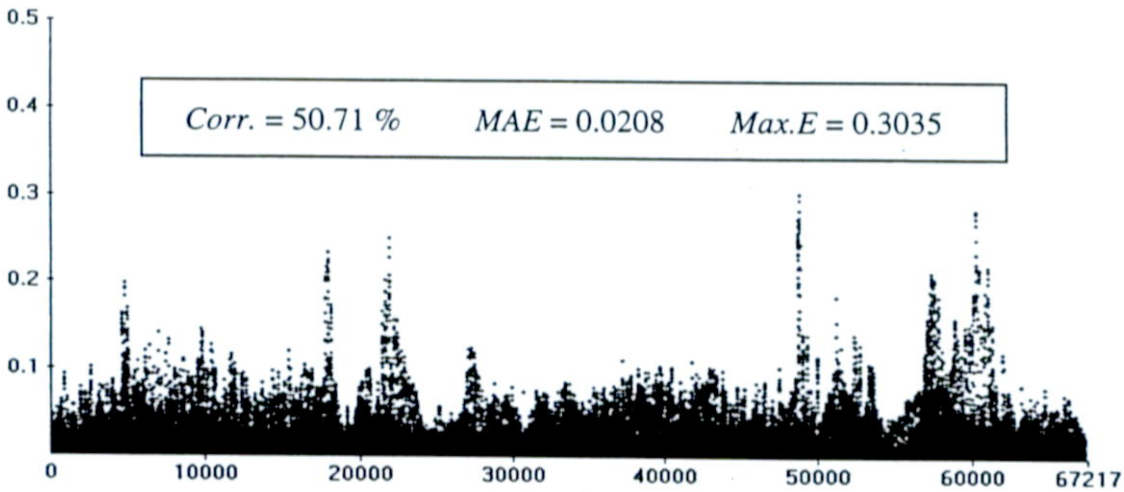


Figure A-13.17
Absolute Difference Between Actual and Predicted Signals

Experiment Definition

No. [1] : 13a

Directory : MBPN\TEST13

Tool : HNC Explore Net

Applications : *MTrain* for training

MTest for testing

Application Parameters

Size of Input Set : 6721

Training Data : T02.DAC

Sawtooth File : ON

Training Method : Normal training

Network Structure

Paradigm : Multi-Layer Backpropagation (*MBPN*)

Input Layer Size : 21

Output Layer Size : 1

No. of Hidden Layer : 2

Hidden Layer Sizes : 20-20

Input-Output Conn. : OFF

Training Parameters

Learning Method : Normal

Learning Rate (α) : 0.5

Activation Function : Gaussian

(Lower Limit = 0, Upper Limit =1, Steepness = 1)

Random Seed : 5

Initial Weight Max. [2] : 0.5

Error Tolerance : 0

Lin.Act.Func.on Ou. : OFF

---

[1] The status of 20,000 iterations of Experiment 13 was used as initial state in this experiment.

[2] Initial weight maximum had no function here as initial weights were used.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error (x10^{-2})	Maximum Error (x10^{-2})
Initial	84.27	1.02	9.82
100	87.33	0.96	7.44
200	87.45	0.95	7.31
300	87.51	0.94	7.21
400	87.56	0.94	7.14
500	87.60	0.94	7.09
600	87.64	0.93	7.05
700	87.65	0.93	7.02
800	87.71	0.93	6.99
900	87.71	0.93	6.97
1000	87.75	0.93	6.95
2000	87.93	0.92	6.88

Table A-13a.1
Experiment Results on *Training Set*

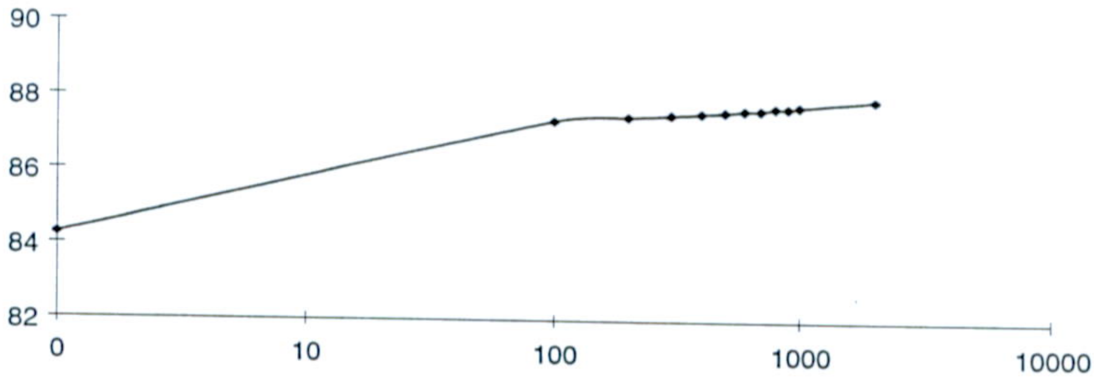


Figure A-13a.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

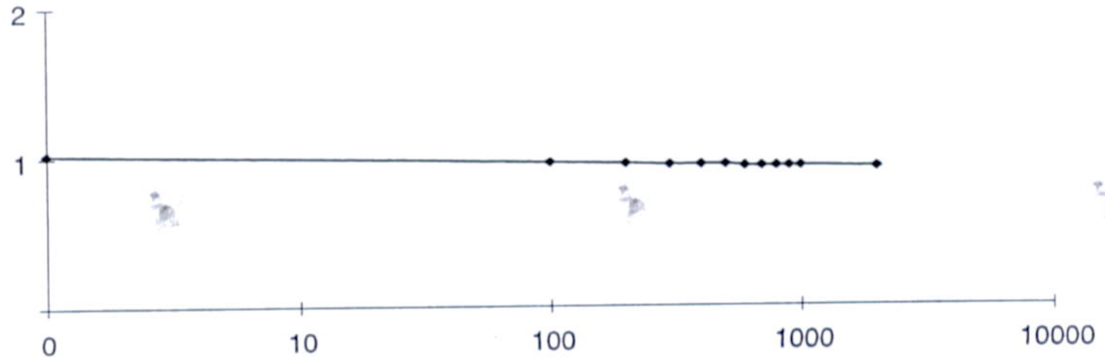


Figure A-13a.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

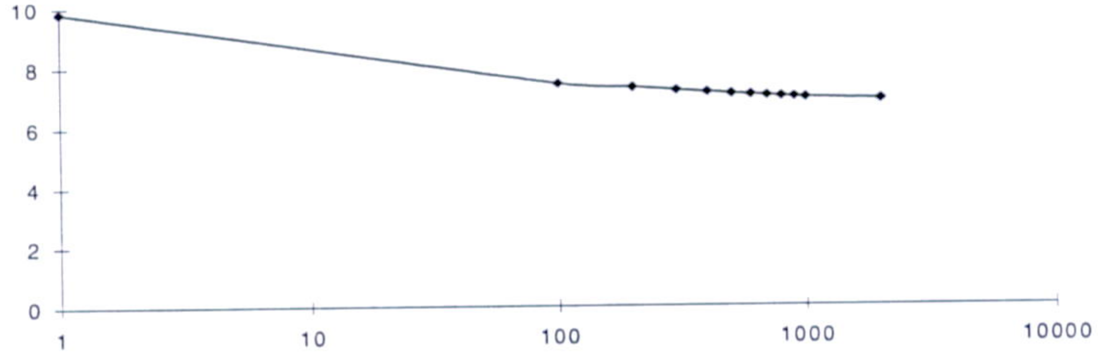


Figure A-13a.3
Maximum Error Vs Number of Iterations for *Training Set*

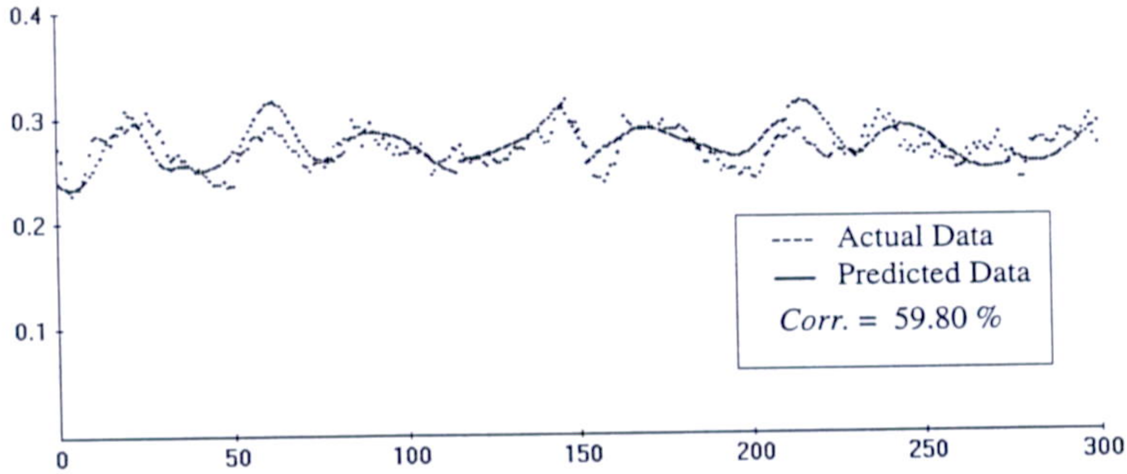


Figure A-13a.4
Test Result after 2,000 Iterations for *First 300 Data*

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
Initial	50.71	2.08	30.35
100	55.76	1.97	38.48
500	61.58	1.73	31.31
1000	63.20	1.67	29.58
1500	64.23	1.64	29.08
2000	64.91	1.63	28.79

Table A-13a.2
Validation Results on *Whole Set*

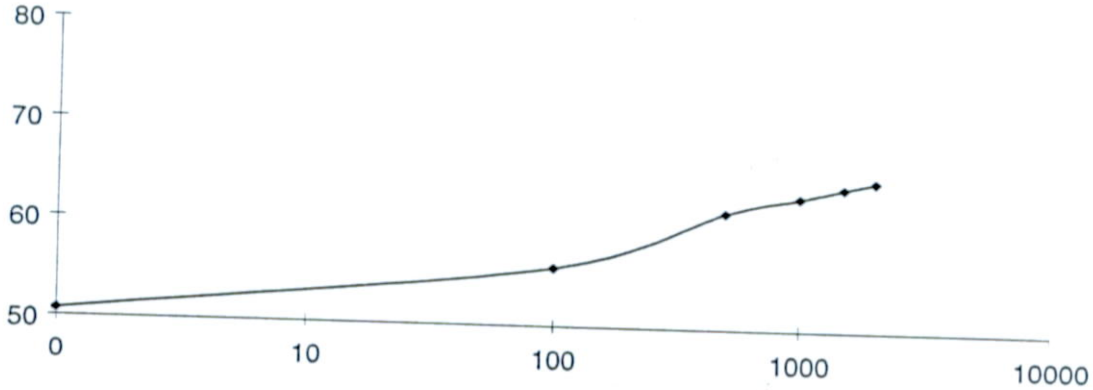


Figure A-13a.5
Correlation Coefficient Vs Number of Iterations for *Whole Set*

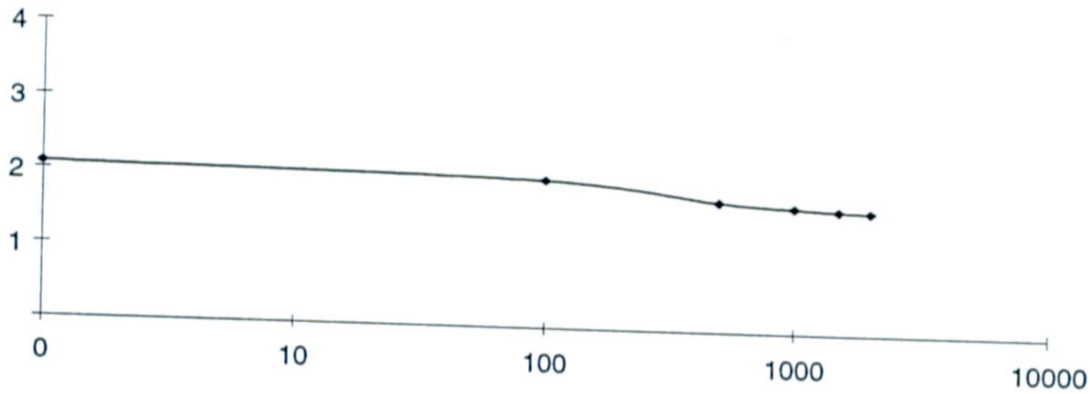


Figure A-13a.6
Mean Absolute Error Vs Number of Iterations for *Whole Set*

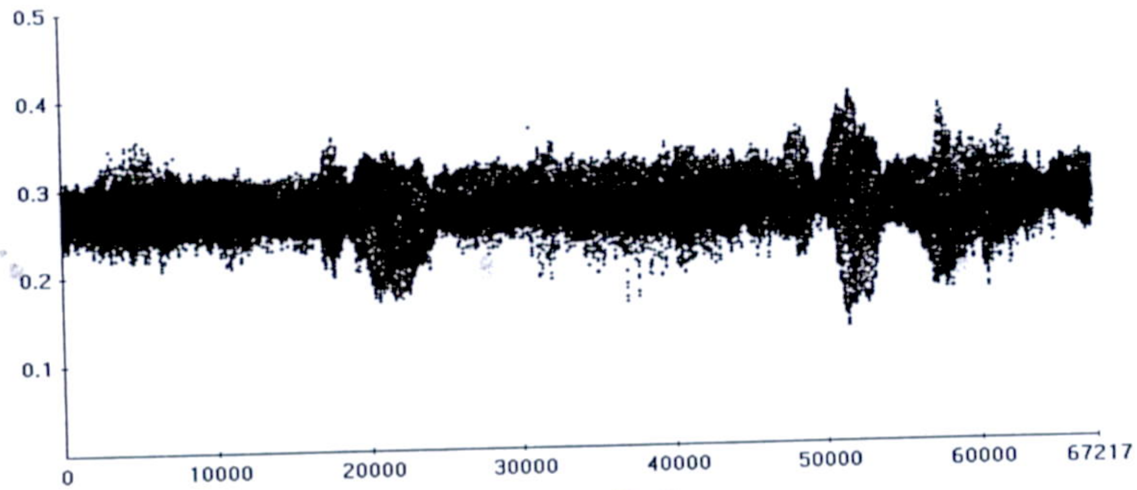


Figure A-13a.7
Actual Strain Signal

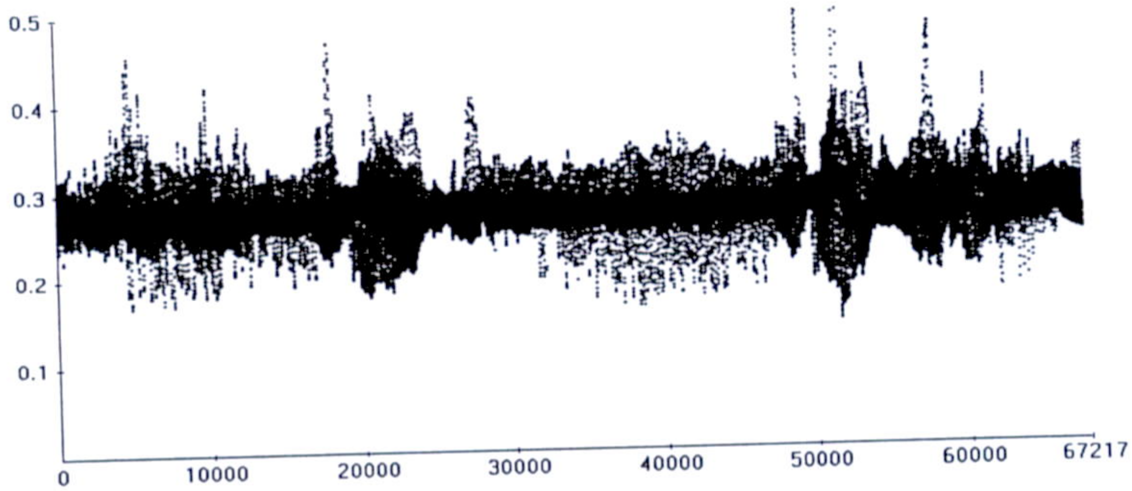


Figure A-13a.8
Predicted Signal after *2,000 Iterations (Total epochs = 22,000)*

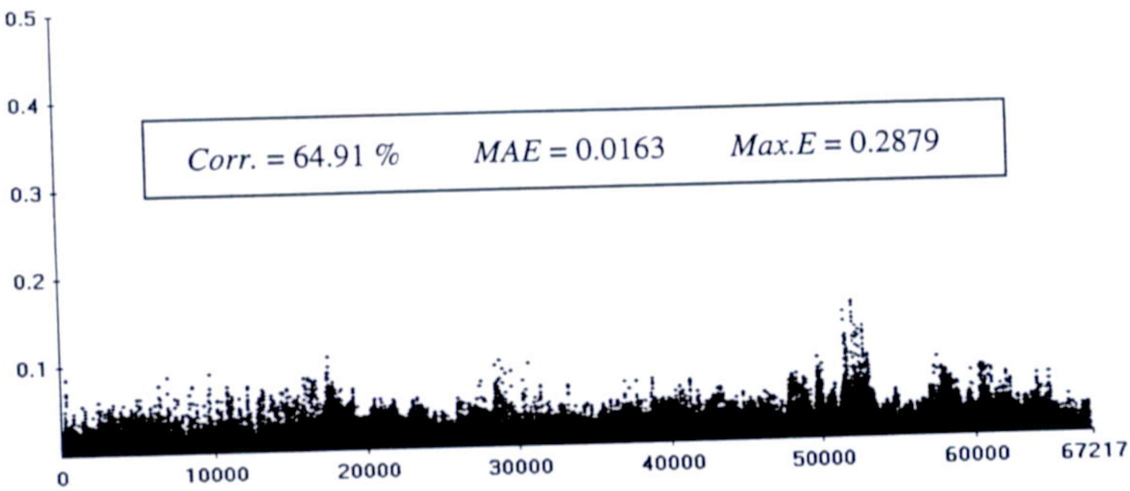


Figure A-13a.9
Absolute Difference Between Actual and Predicted Signals

Experiment Definition

No. [1]	: 13b
Directory	: MBPN\TEST13
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 6721
Training Data	: T02.DAC
Sawtooth File	: ON
Training Method [2]	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 2
Hidden Layer Sizes	: 20-20
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.5
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max. [3]	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

[1] The status of 500 iterations of Experiment 13a was used as initial state in this experiment.
[2] Each one of ten sub-sets was presented to the network as in Experiment 13.
[3] Initial weight maximum had no function here as initial weights were used.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
Initial	87.60	0.94	7.09
100	85.15	1.00	9.37
200	85.04	1.00	9.50
300	84.97	1.01	9.59
400	84.93	1.01	9.66
500	84.83	1.01	9.71

Table A-13b.1
Experiment Results on *Training Set*

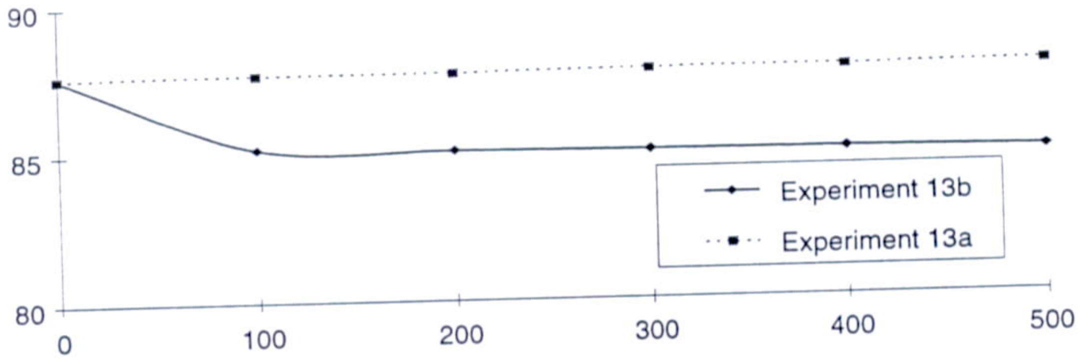


Figure A-13b.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

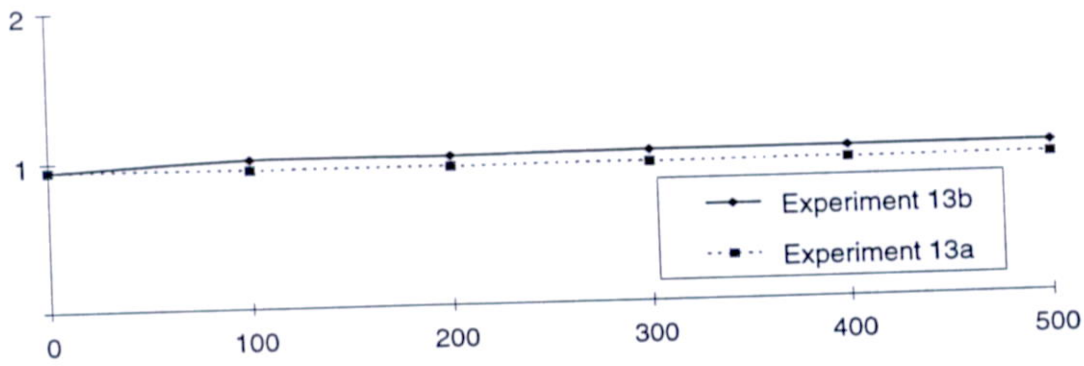


Figure A-13b.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

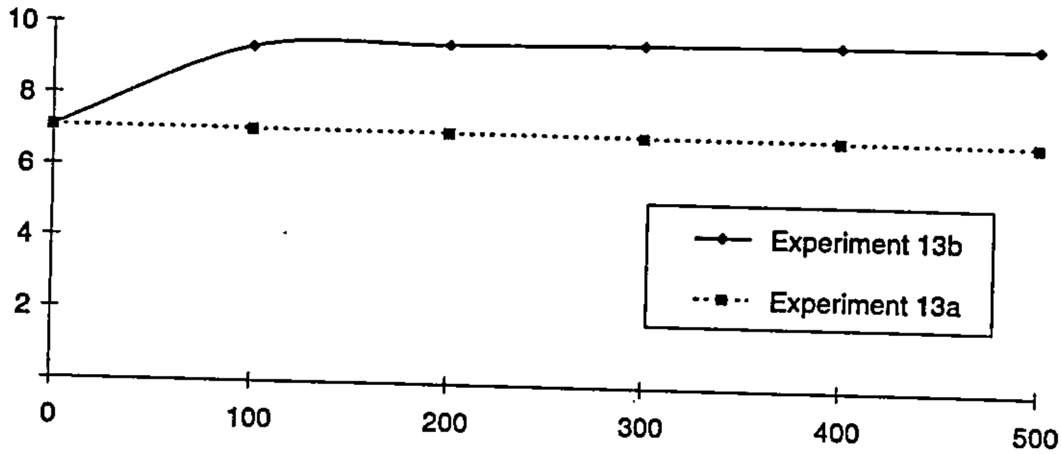


Figure A-13b.3
Maximum Error Vs Number of Iterations for *Training Set*

Experiment Definition

No. : 14

Directory : MBPN\TEST14

Tool : HNC Explore Net

Applications : *MTrain* for training
MTest for testing

Application Parameters

Size of Input Set : 6721

Training Data : T02.DAC

Sawtooth File : ON

Training Method[1] : Fast training

Network Structure

Paradigm : Multi-Layer Backpropagation (*MBPN*)

Input Layer Size : 21

Output Layer Size : 1

No. of Hidden Layer : 1

Hidden Layer Size : 40

Input-Output Conn. : OFF

Training Parameters

Learning Method : Normal

Learning Rate (α) : 0.1

Activation Function : Gaussian
(Lower Limit = 0, Upper Limit =1, Steepness = 1)

Random Seed : 5

Initial Weight Max. : 0.5

Error Tolerance : 0

Lin.Act.Func.on Ou. : OFF

---

[1] Each one of ten sub-sets was presented to the network as in Experiment 13.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error (x10^{-2})	Maximum Error (x10^{-2})
0	- 15.80	29.65	57.15
5	32.85	2.22	12.98
10	35.80	2.21	13.20
20	38.58	2.27	13.34
50	41.53	2.48	13.48
100	44.57	2.24	13.33
200	46.44	1.90	12.94
300	47.87	1.81	12.45
400	49.39	1.77	11.92
500	51.34	1.71	11.13
600	52.73	1.68	10.70
700	53.29	1.66	11.04
800	53.62	1.64	11.24
900	53.88	1.63	11.42
1000	54.07	1.63	11.60
2000	56.76	1.57	13.71
3000	62.15	1.57	11.89
4000	66.33	1.51	11.23
5000	69.07	1.42	10.82

Table A-14.1
Experiment Results on *Training Set*

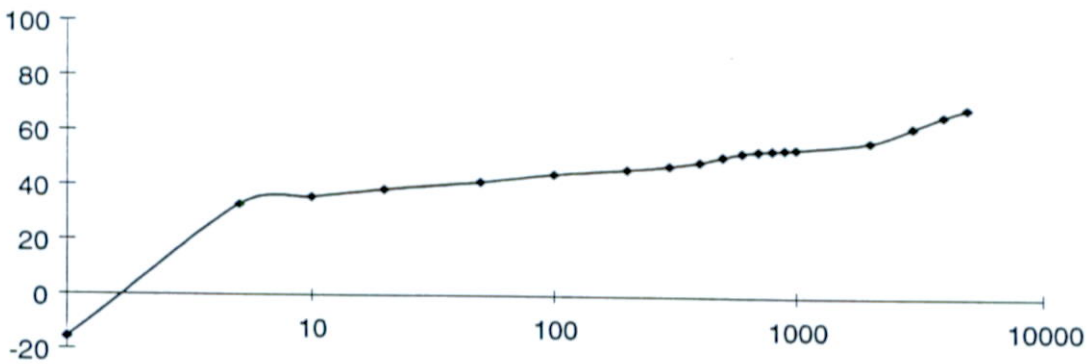


Figure A-14.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

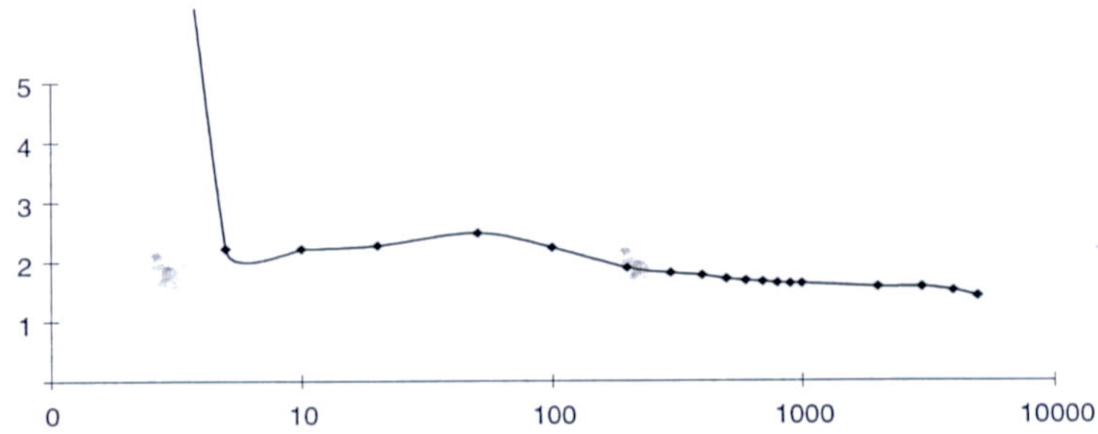


Figure A-14.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

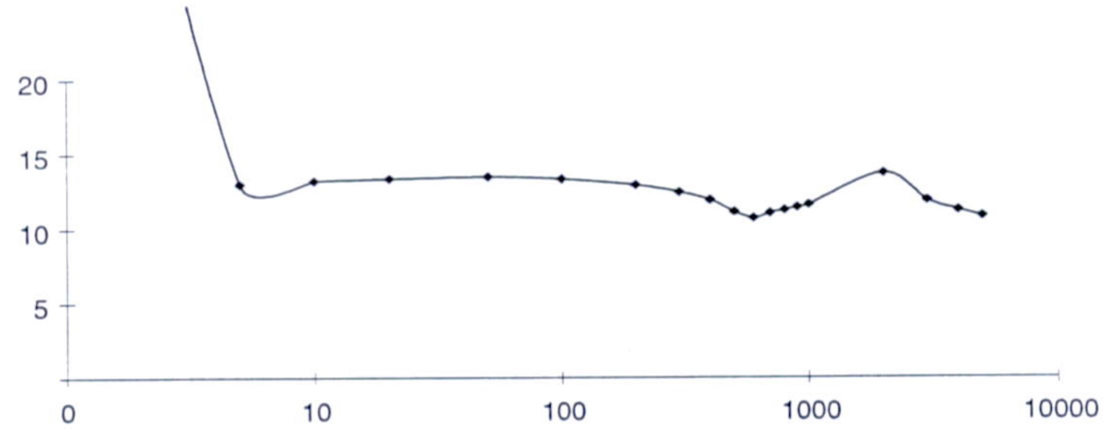


Figure A-14.3
Maximum Error Vs Number of Iterations for *Training Set*

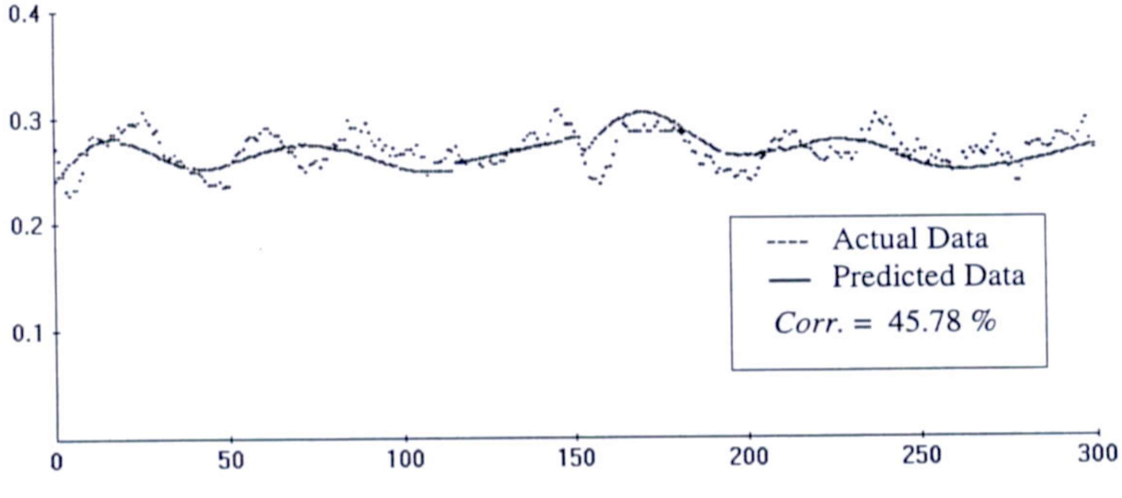


Figure A-14.4
Test Result after 5,000 Iterations for *First 300 Data*

Experiment Definition

No.	: 15
Directory	: MBPN\TEST15
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 6721
Training Data	: T02.DAC
Sawtooth File	: ON
Training Method[1]	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 1
Hidden Layer Size	: 40
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.5
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

[1] Each one of ten sub-sets was presented to the network as in Experiment 13.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
0	- 15.80	29.65	57.15
5	33.21	2.94	13.76
10	38.04	2.96	13.91
20	41.79	2.42	13.46
50	44.62	1.94	13.51
100	46.14	2.28	14.17
200	48.15	2.33	14.46
300	50.68	2.19	13.43
400	51.32	1.95	13.69
500	51.90	1.83	13.95
600	52.24	1.75	14.52
700	52.56	1.71	15.19
800	52.99	1.69	15.66
900	53.49	1.66	15.73
1000	54.18	1.64	15.06
2000	68.84	1.53	11.20
3000	72.41	1.42	12.77
4000	73.53	1.38	11.87
5000	74.17	1.38	10.59

Table A-15.1
Experiment Results on *Training Set*

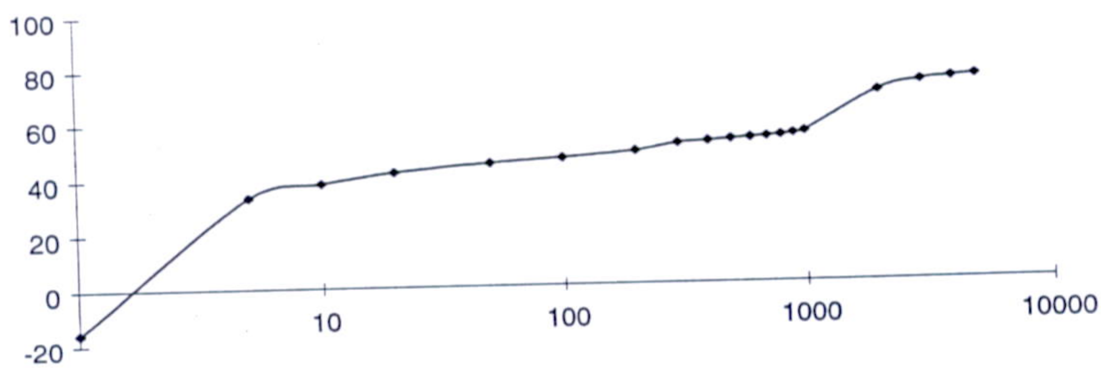


Figure A-15.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

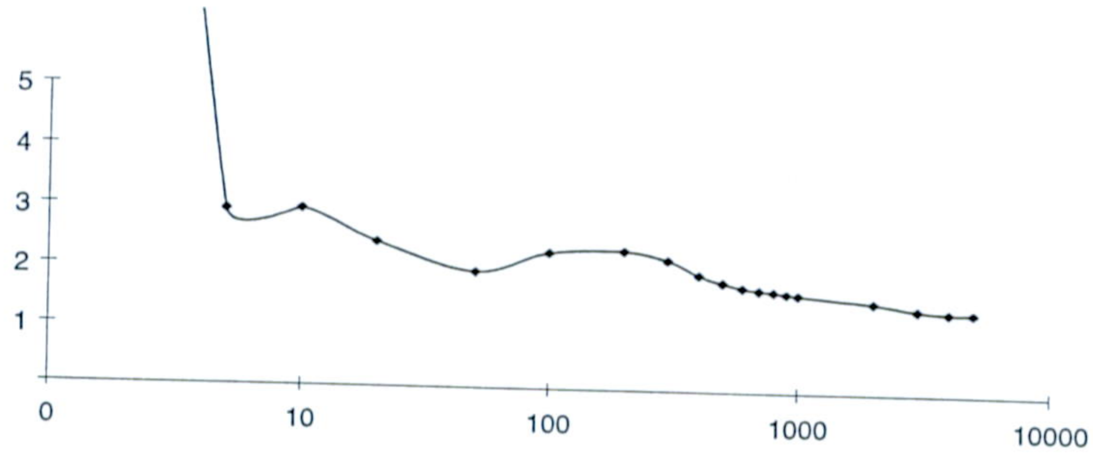


Figure A-15.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

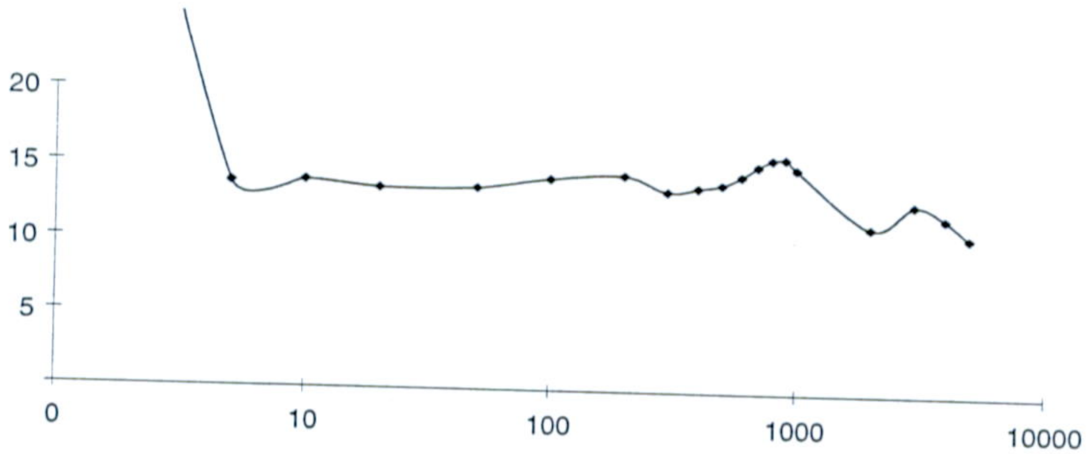


Figure A-15.3
Maximum Error Vs Number of Iterations for *Training Set*

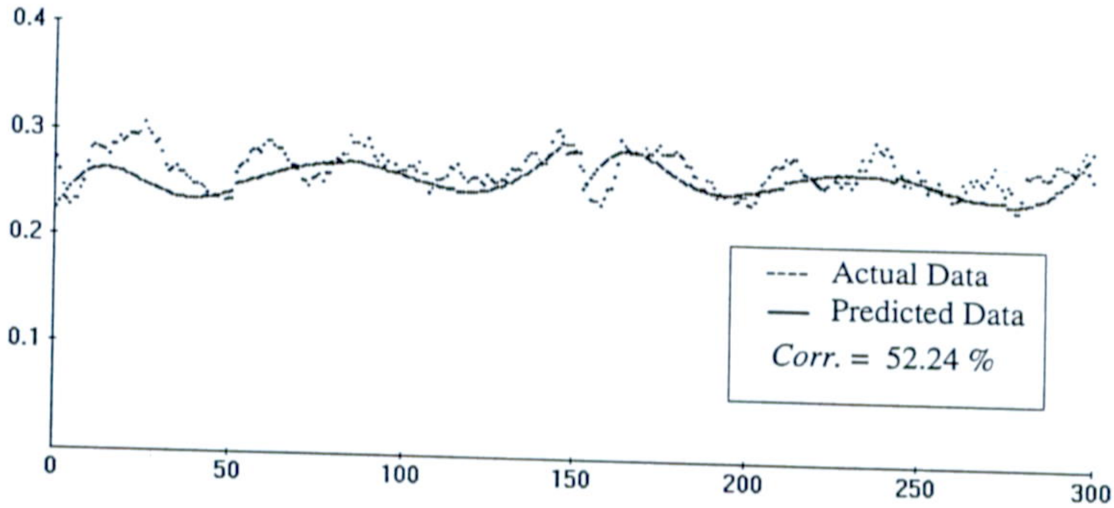


Figure A-15.4
Test Result after 5,000 Iterations for *First 300 Data*

Experiment Definition

No.	: 16
Directory	: MBPN\TEST16
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 6721
Training Data	: T02.DAC
Sawtooth File	: ON
Training Method[1]	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 1
Hidden Layer Size	: 41
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.5
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

[1] Each one of ten sub-sets was presented to the network as in Experiment 13.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
0	- 6.43	5.50	23.67
5	37.37	2.14	12.90
10	39.73	2.19	13.19
20	42.28	2.46	13.95
50	44.84	2.48	14.21
100	46.10	2.06	13.63
200	47.77	1.99	13.53
300	49.94	1.95	12.95
400	50.64	1.96	12.76
500	51.10	1.91	12.90
600	51.40	1.85	13.21
700	51.74	1.80	13.50
800	52.20	1.76	13.91
900	52.86	1.73	14.52
1000	53.76	1.71	15.28
2000	63.87	1.50	15.59

Table A-16.1
Experiment Results on *Training Set*

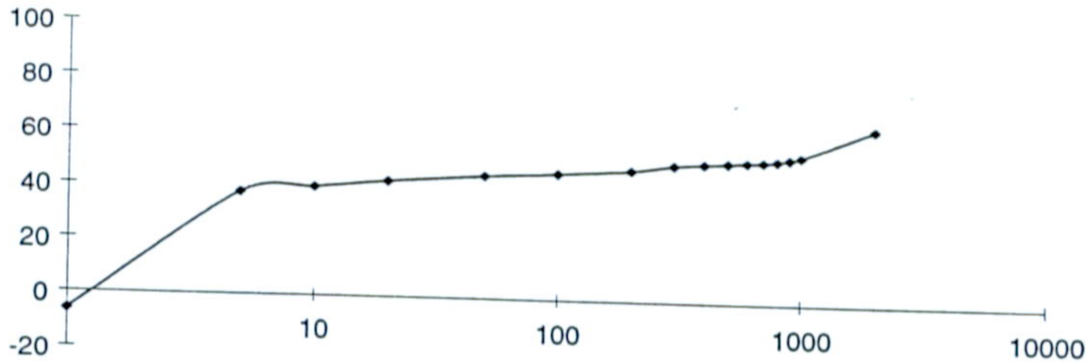


Figure A-16.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

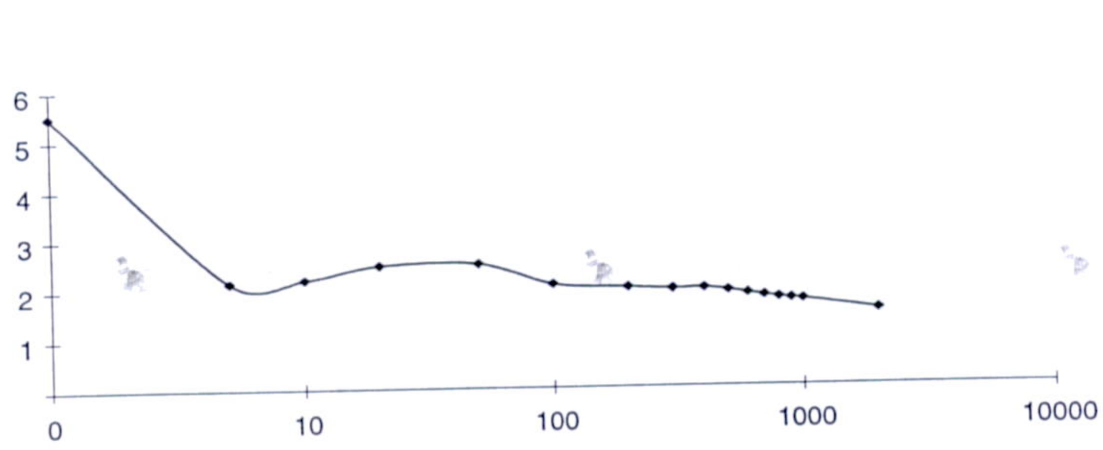


Figure A-16.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

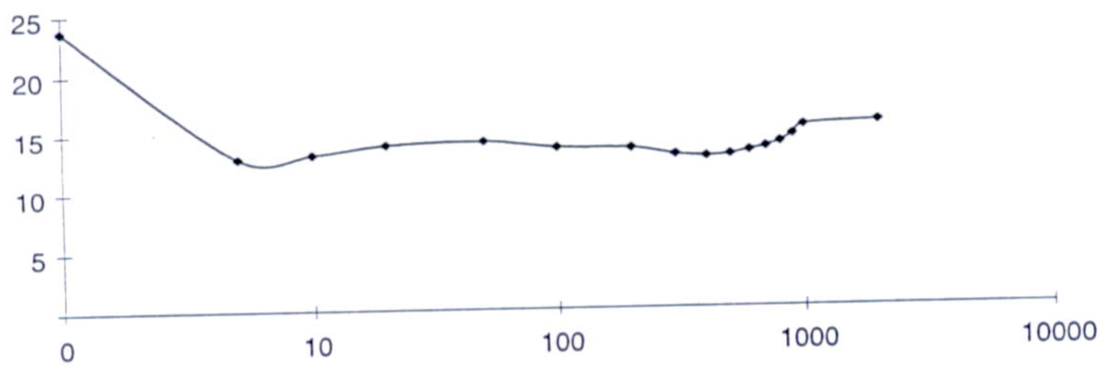


Figure A-16.3
Maximum Error Vs Number of Iterations for *Training Set*

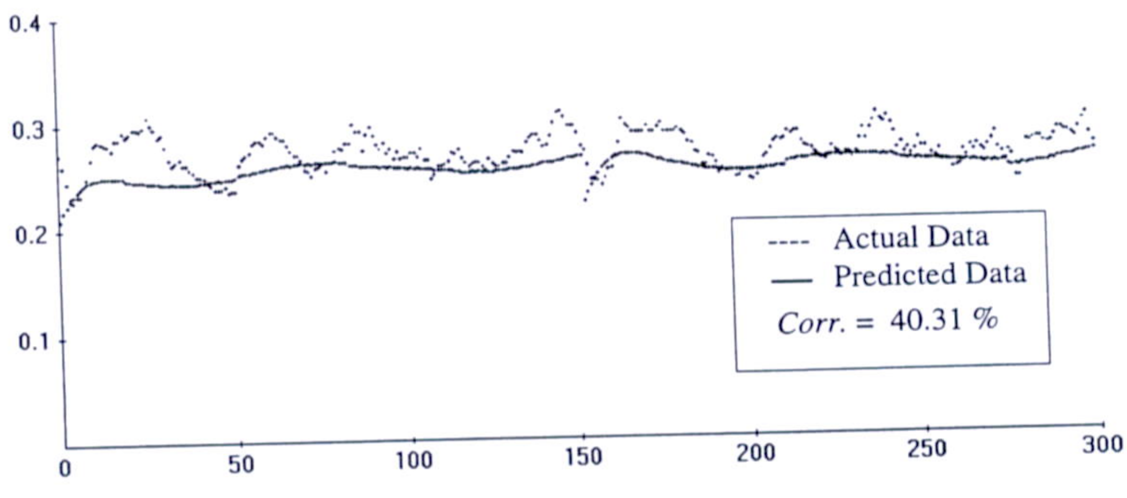


Figure A-16.4
Test Result after 2,000 Iterations for *First 300 Data*

Experiment Definition

No.	: 17
Directory	: MBPN\TEST17
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 6721
Training Data	: T02.DAC
Sawtooth File	: ON
Training Method[1]	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 1
Hidden Layer Size	: 43
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.5
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

[1] Each one of ten sub-sets was presented to the network as in Experiment 13.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error (x10^{-2})	Maximum Error (x10^{-2})
0	6.77	51.06	67.96
5	35.02	2.75	13.88
10	39.25	2.56	14.16
20	42.37	2.39	14.16
50	44.92	2.21	14.07
100	46.19	2.17	14.09
200	47.66	2.12	14.04
300	50.04	1.96	13.29
400	50.75	1.91	12.76
500	51.12	1.85	13.31
600	51.39	1.80	13.68
700	51.81	1.77	13.85
800	52.44	1.75	13.85
900	53.16	1.71	13.56
1000	53.78	1.69	12.91

Table A-17.1
Experiment Results on *Training Set*

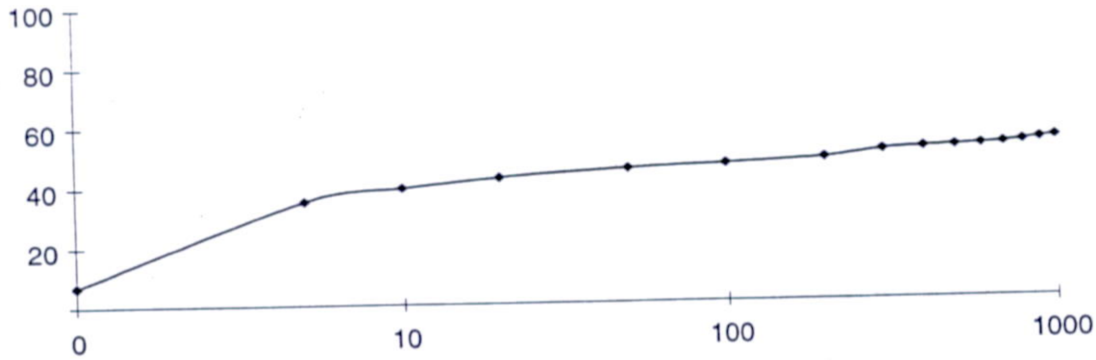


Figure A-17.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

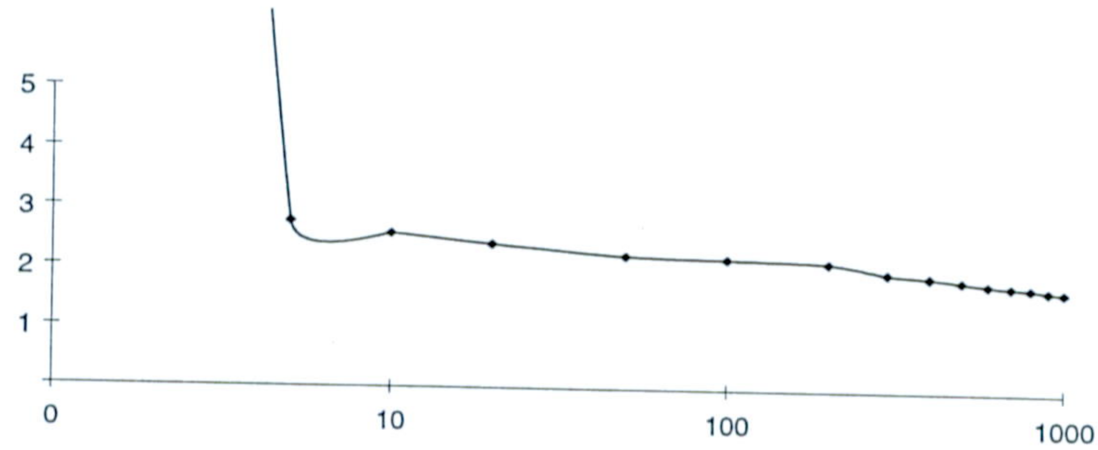


Figure A-17.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

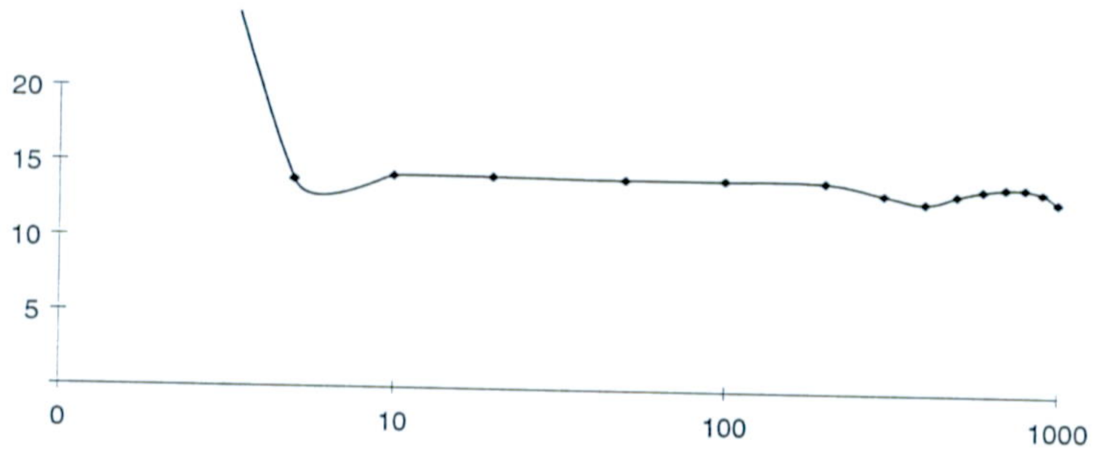


Figure A-17.3
Maximum Error Vs Number of Iterations for *Training Set*

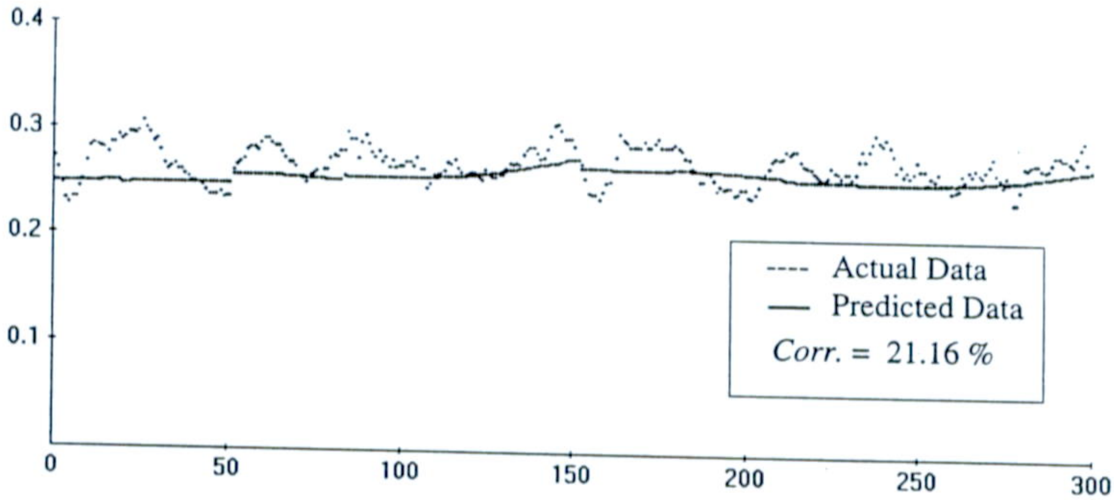


Figure A-17.4
Test Result after 1,000 Iterations for *First 300 Data*

Experiment Definition

No.	: 18
Directory	: MBPN\TEST18
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 6721
Training Data	: T02.DAC
Sawtooth File	: ON
Training Method[1]	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 1
Hidden Layer Size	: 50
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)	: 0.5
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

---

[1] Each one of ten sub-sets was presented to the network as in Experiment 13.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error (x10^{-2})	Maximum Error (x10^{-2})
0	9.69	25.43	55.87
5	14.30	27.29	39.89
10	14.21	27.29	39.89
20	13.72	27.29	39.89
50	36.03	2.28	13.42
100	45.32	2.16	13.87
200	46.72	2.14	14.16
300	48.94	2.03	13.66
400	50.24	2.02	13.14
500	51.07	2.09	13.03
600	51.35	1.93	13.01
700	51.79	1.88	13.41
800	52.44	1.83	13.99
900	53.56	1.78	14.94
1000	55.56	1.58	15.06
2000	66.64	1.49	11.74

Table A-18.1
Experiment Results on *Training Set*

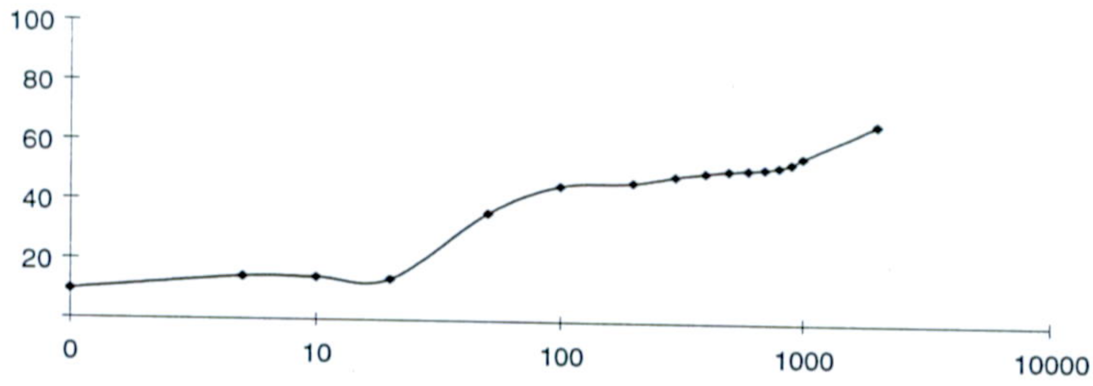


Figure A-18.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

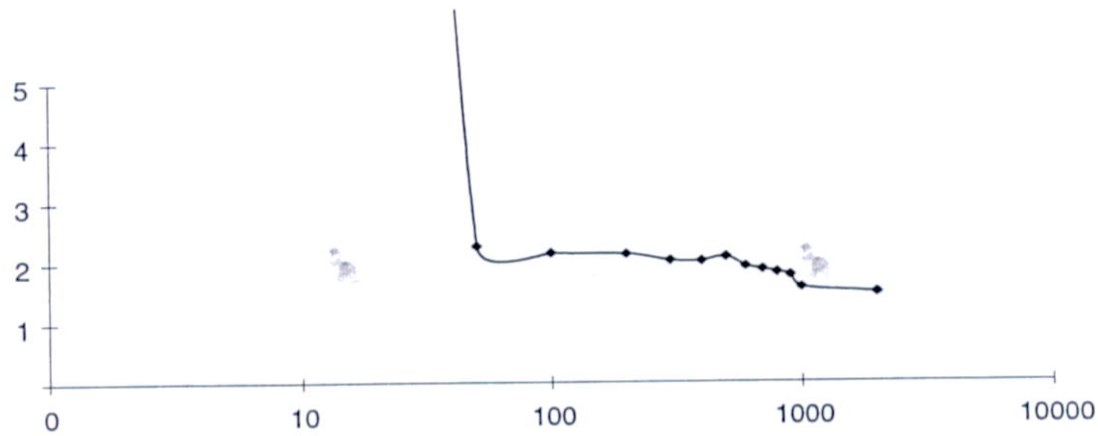


Figure A-18.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

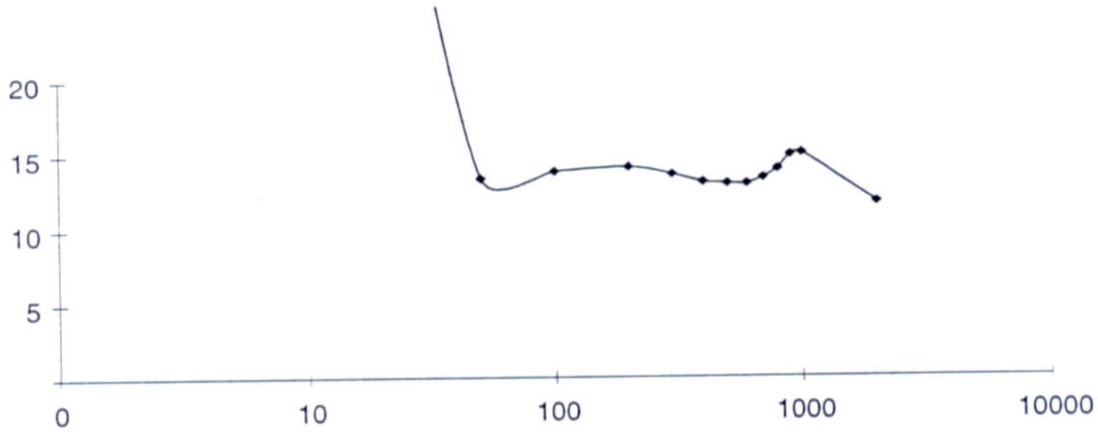


Figure A-18.3
Maximum Error Vs Number of Iterations for *Training Set*

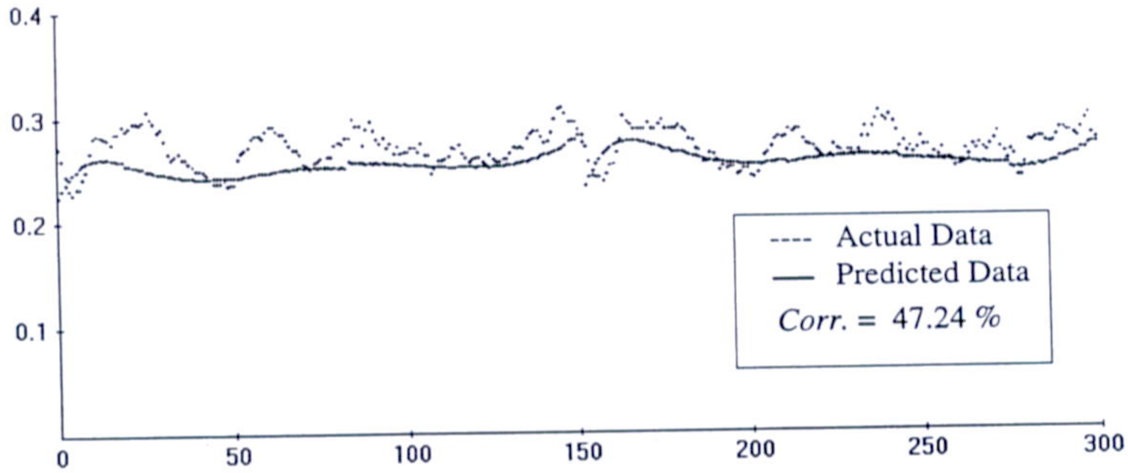


Figure A-18.4
Test Result after 2,000 Iterations for *First 300 Data*

Experiment Definition

No. : 19

Directory : MBPN\TEST19

Tool : HNC Explore Net

Applications : *MTrain* for training
MTest for testing

Application Parameters

Size of Input Set : 6721

Training Data : T02.DAC

Sawtooth File : ON

Training Method[1] : Fast training

Network Structure

Paradigm : Multi-Layer Backpropagation (*MBPN*)

Input Layer Size : 21

Output Layer Size : 1

No. of Hidden Layer : 1

Hidden Layer Size : 60

Input-Output Conn. : OFF

Training Parameters

Learning Method : Normal

Learning Rate (α) : 0.5

Activation Function : Gaussian
(Lower Limit = 0, Upper Limit =1, Steepness = 1)

Random Seed : 5

Initial Weight Max. : 0.5

Error Tolerance : 0

Lin.Act.Func.on Ou. : OFF

---

[1] Each one of ten sub-sets was presented to the network as in Experiment 13.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
0	10.71	27.08	61.16
5	3.88	27.29	39.89
10	3.88	27.29	39.89
20	3.88	27.29	39.89
50	3.88	27.29	39.89
100	3.88	27.29	39.89
200	3.88	27.29	39.89
300	3.88	27.29	39.89
400	3.88	27.29	39.89
500	3.88	27.29	39.89
600	3.88	27.29	39.89
700	3.88	27.29	39.89
800	3.88	27.29	39.89
900	3.88	27.29	39.89
1000	3.88	27.29	39.89

Table A-19.1
Experiment Results on *Training Set*

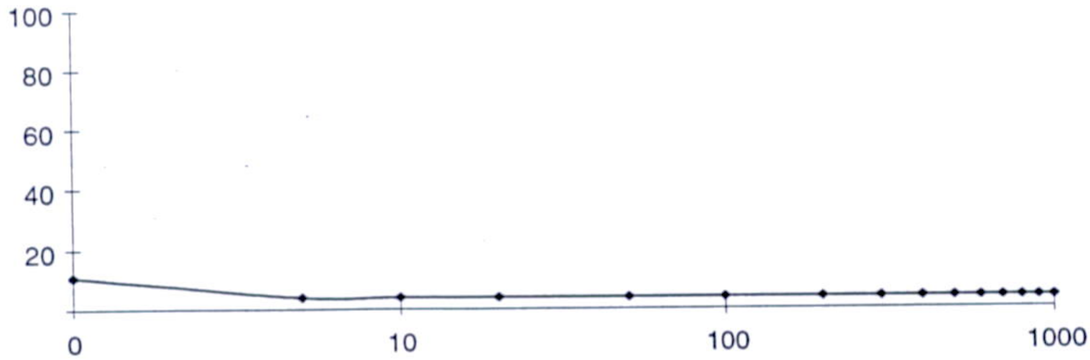


Figure A-19.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

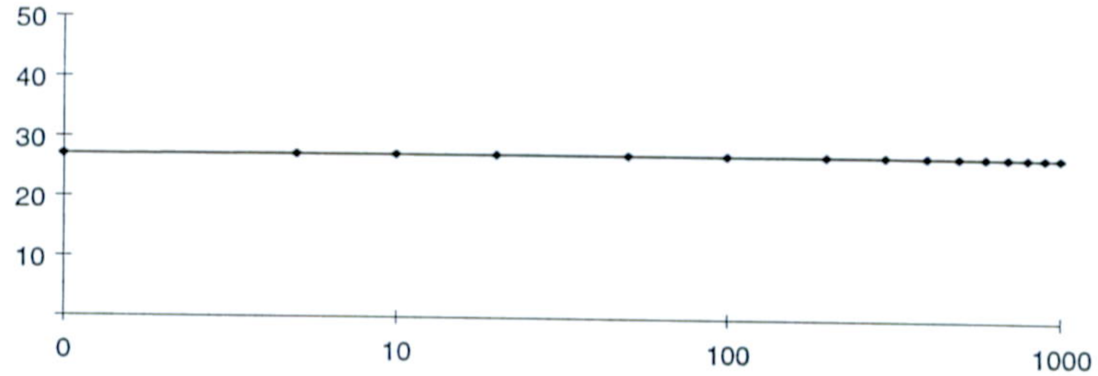


Figure A-19.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

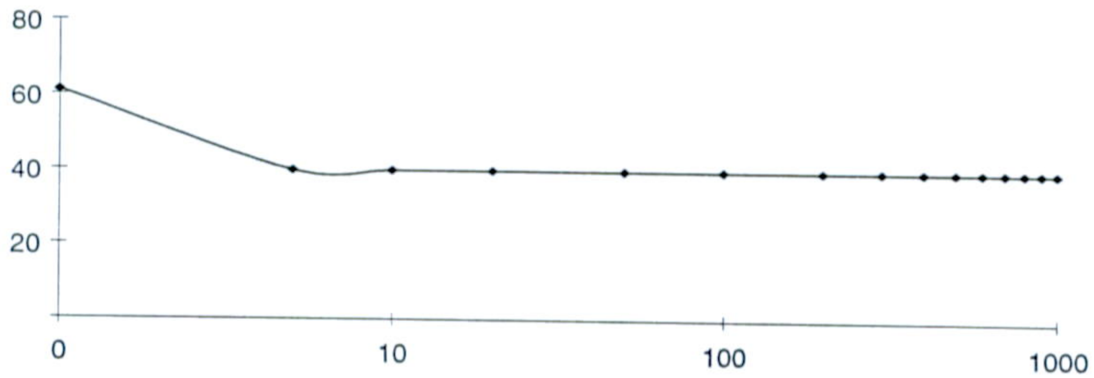


Figure A-19.3
Maximum Error Vs Number of Iterations for *Training Set*

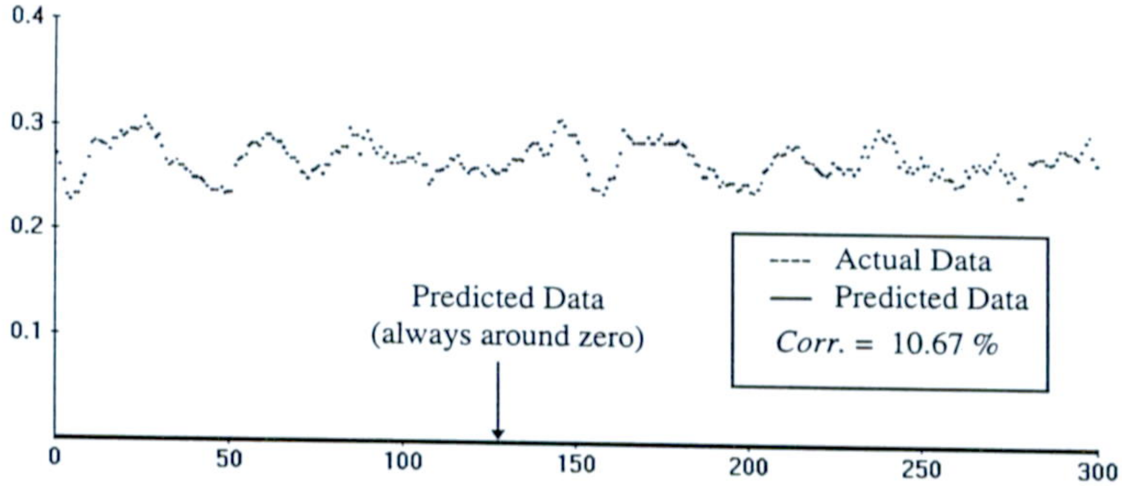


Figure A-19.4
Test Result after 1,000 Iterations for *First 300 Data*

Experiment Definition

No. : 20

Directory : MBPN\TEST20

Tool : HNC Explore Net

Applications : *MTrain* for training
MTest for testing

Application Parameters

Size of Input Set : 6721

Training Data : T02.DAC

Sawtooth File : ON

Training Method[1] : Fast training

Network Structure

Paradigm : Multi-Layer Backpropagation (*MBPN*)

Input Layer Size : 21

Output Layer Size : 1

No. of Hidden Layer : 1

Hidden Layer Size : 60

Input-Output Conn. : OFF

Training Parameters

Learning Method : Normal

Learning Rate (α) : 0.1

Activation Function : Gaussian
(Lower Limit = 0, Upper Limit =1, Steepness = 1)

Random Seed : 5

Initial Weight Max. : 0.5

Error Tolerance : 0

Lin.Act.Func.on Ou. : OFF

---

[1] Each one of ten sub-sets was presented to the network as in Experiment 13.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($x10^{-2}$)	Maximum Error ($x10^{-2}$)
0	10.71	27.08	61.16
5	30.11	5.48	16.41
10	32.57	5.15	16.23
20	35.58	4.58	15.95
50	40.54	3.64	15.08
100	44.60	2.81	14.26
200	46.77	2.25	14.12
300	47.99	2.07	13.68
400	49.13	1.98	13.13
500	49.99	1.91	12.82
600	50.70	1.86	12.56
700	51.38	1.82	12.31
800	52.01	1.78	12.02
900	52.54	1.74	11.68
1000	53.08	1.70	11.65
2000	55.74	1.55	13.99

Table A-20.1
Experiment Results on *Training Set*

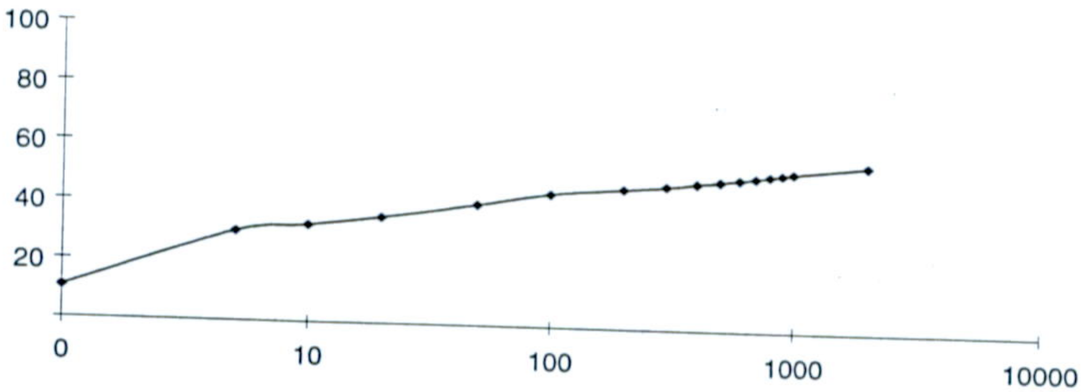


Figure A-20.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

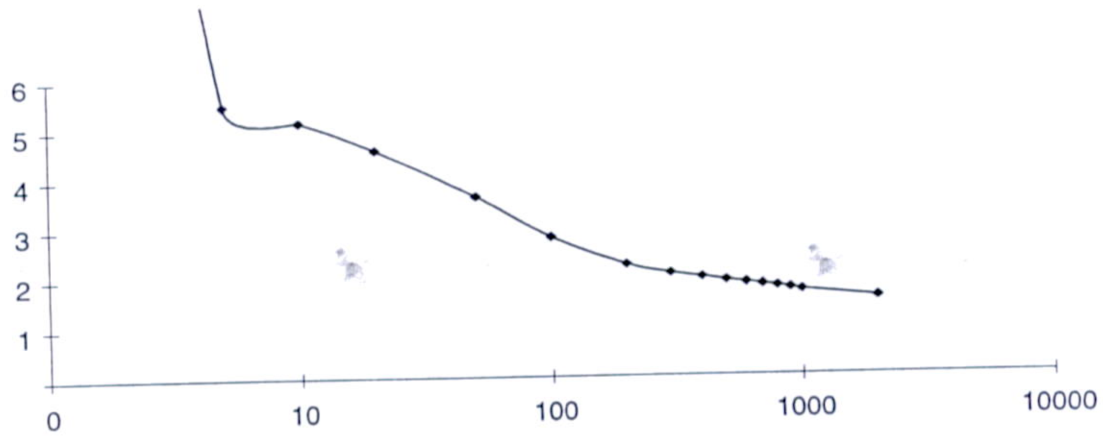


Figure A-20.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

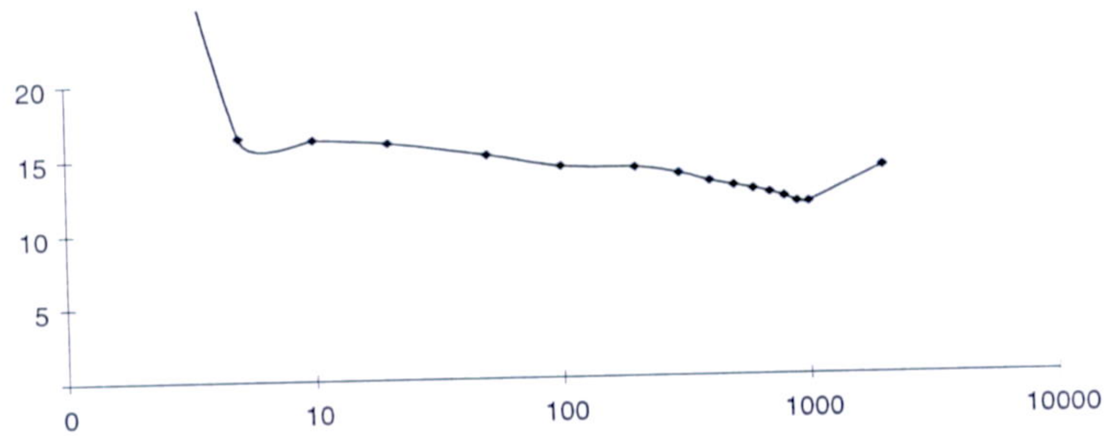


Figure A-20.3
Maximum Error Vs Number of Iterations for *Training Set*

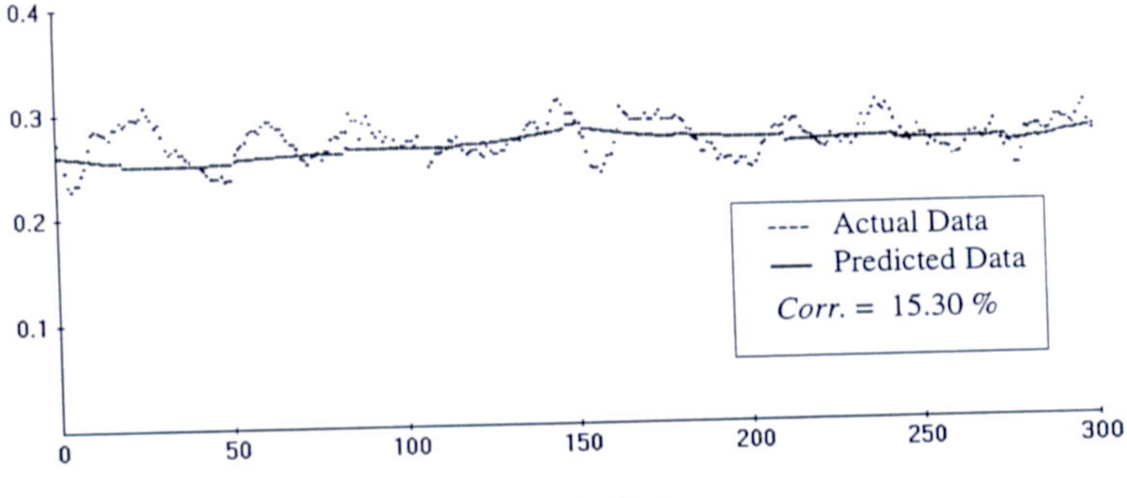


Figure A-20.4
Test Result after 2,000 Iterations for *First 300 Data*

Experiment Definition

No.	: 21
Directory	: MBPN\TEST21
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 6721
Training Data	: T02.DAC
Sawtooth File	: ON
Training Method[1]	: Fast training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 1
Hidden Layer Size	: 60
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate (α)[2]	: 0.5 and 0.1
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

[1] Each one of ten sub-sets was presented to the network as in Experiment 13.
[2] 0.5 is for hidden layer, and 0.1 for output layer.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($x10^{-2}$)	Maximum Error ($x10^{-2}$)
0	10.71	27.08	61.16
5	36.11	3.16	13.89
10	39.37	2.69	13.59
20	41.54	2.73	13.97
50	44.21	2.83	14.78
100	45.81	2.78	15.03
200	49.34	2.72	14.26
300	51.36	2.56	13.29
400	52.42	2.15	12.93
500	53.57	1.90	13.68
600	54.97	1.68	13.43
700	55.41	1.61	13.02
800	56.04	1.59	12.74
900	56.54	1.58	12.20
1000	57.10	1.58	11.46
2000	71.27	1.49	8.19
3000	73.38	1.36	8.16
4000	74.80	1.30	7.77
5000	76.08	1.25	8.55
6000	77.07	1.21	8.96
7000	77.75	1.19	9.17
8000	78.19	1.18	9.28
9000	78.40	1.18	9.33
10000	78.64	1.17	9.31

Table A-21.1
Experiment Results on *Training Set*

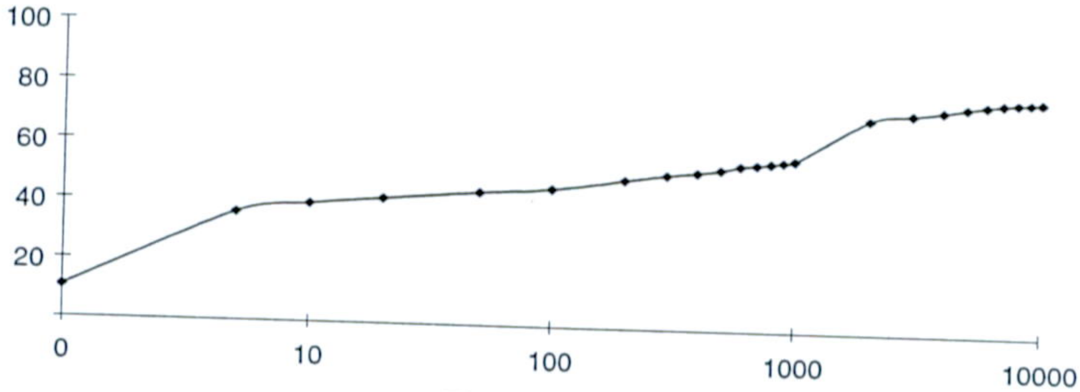


Figure A-21.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

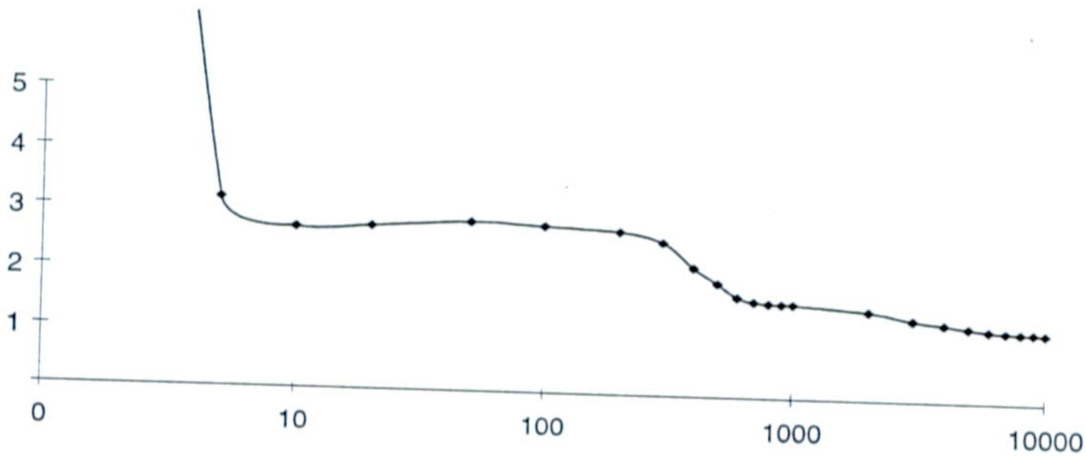


Figure A-21.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

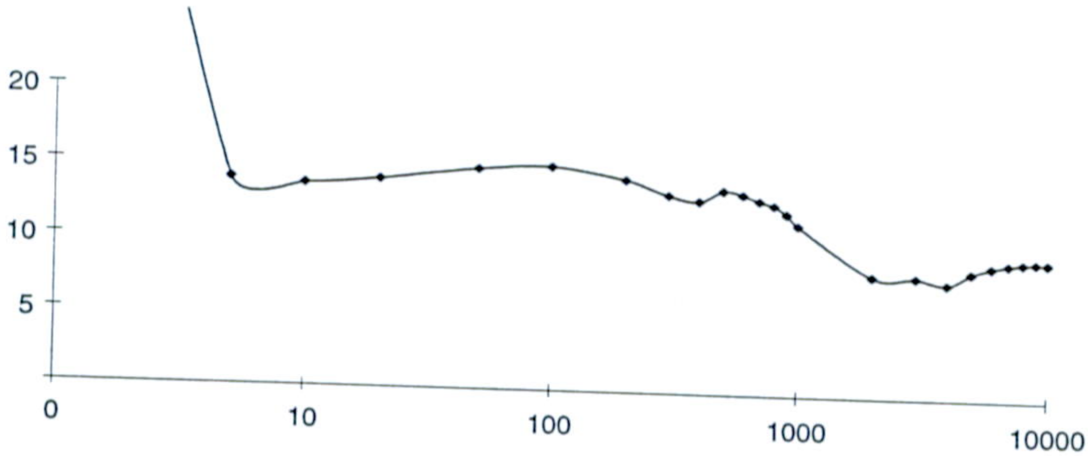


Figure A-21.3
Maximum Error Vs Number of Iterations for *Training Set*

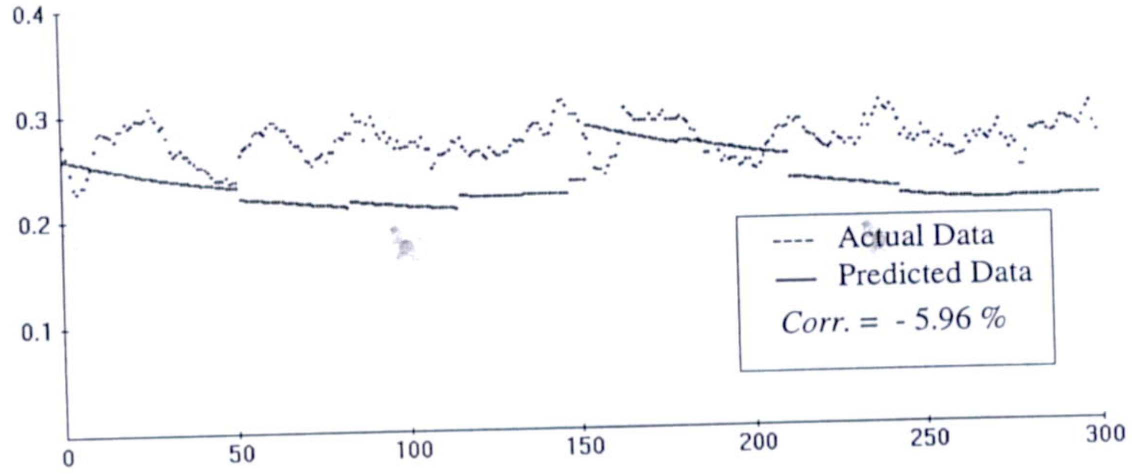


Figure A-21.4
Test Result after 10 Iterations for *First 300 Data*

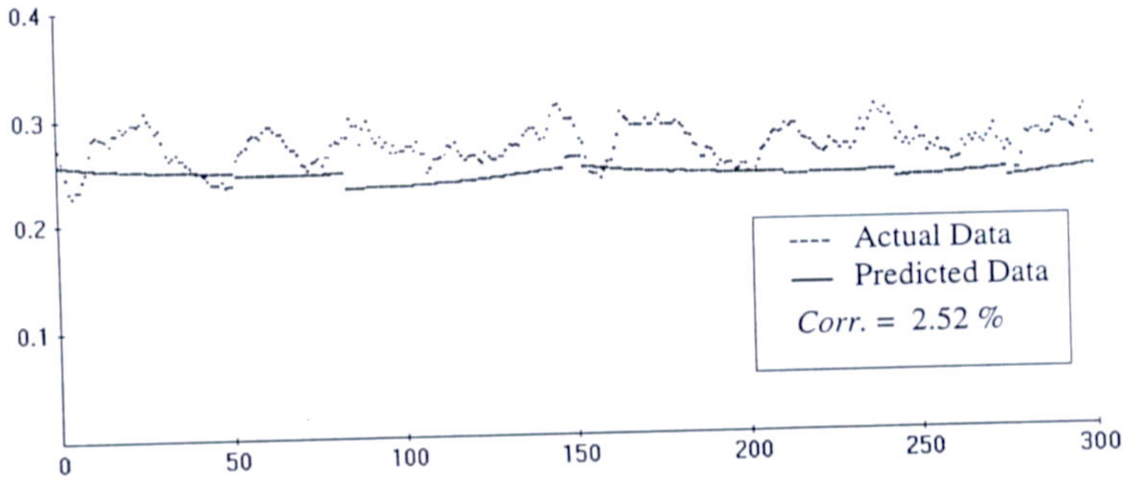


Figure A-21.5
Test Result after 100 Iterations for *First 300 Data*

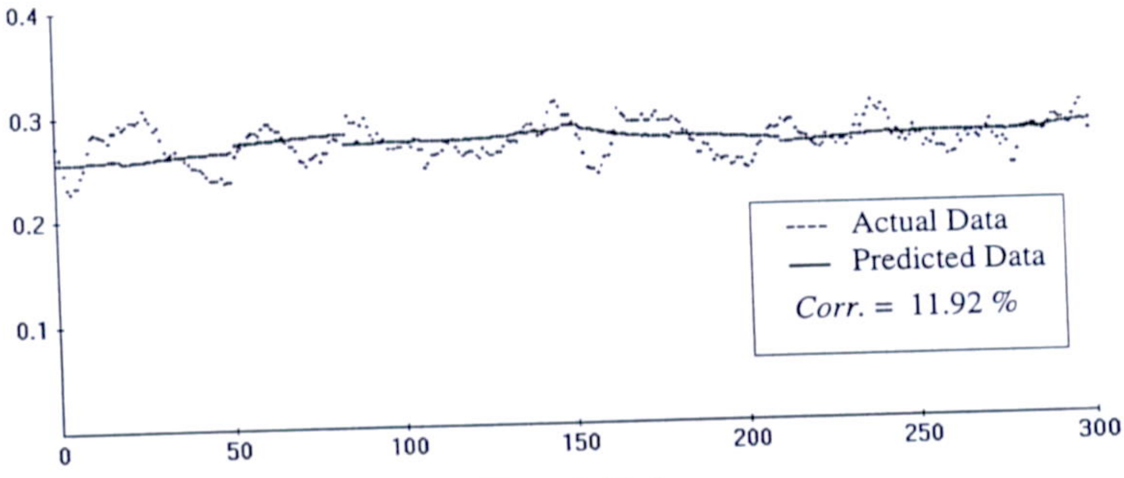


Figure A-21.6
Test Result after 1,000 Iterations for *First 300 Data*

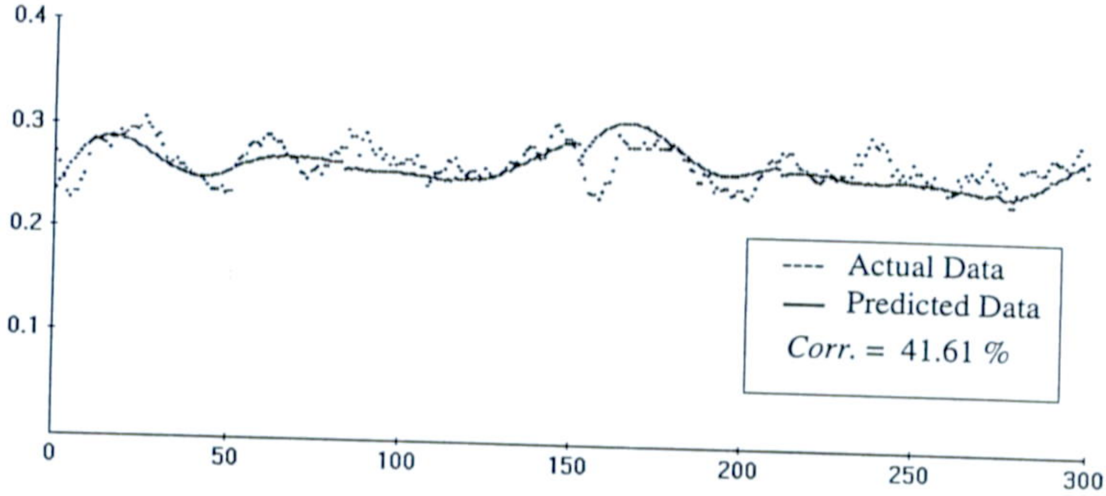


Figure A-21.7
Test Result after 5,000 Iterations for *First 300 Data*

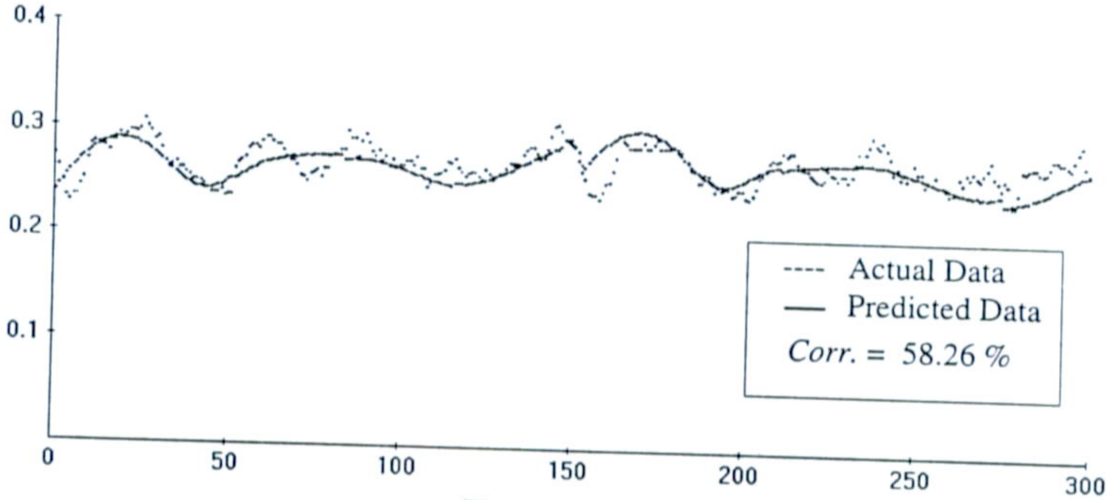


Figure A-21.8
Test Result after 10,000 Iterations for *First 300 Data*

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
10	26.75	3.47	17.67
100	38.87	3.15	15.62
1000	48.10	1.87	13.93
5000	68.92	1.44	16.14
10000	70.50	1.40	15.77

Table A-21.2
Validation Results on *Whole Set*

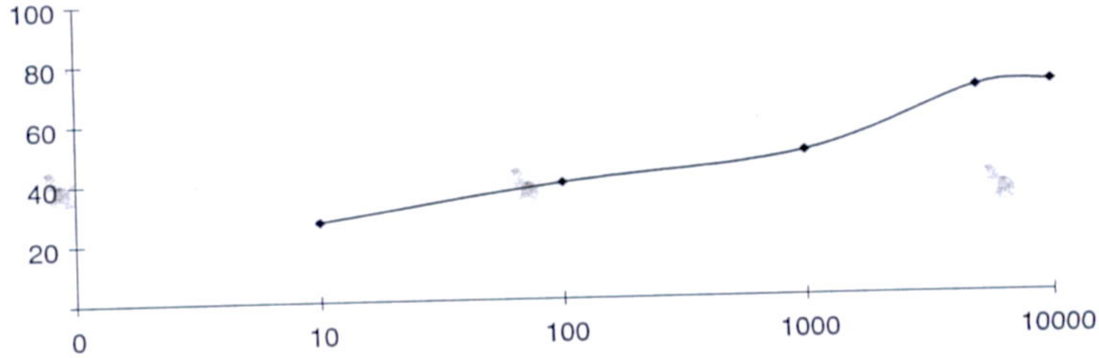


Figure A-21.9
Correlation Coefficient Vs Number of Iterations for *Whole Set*

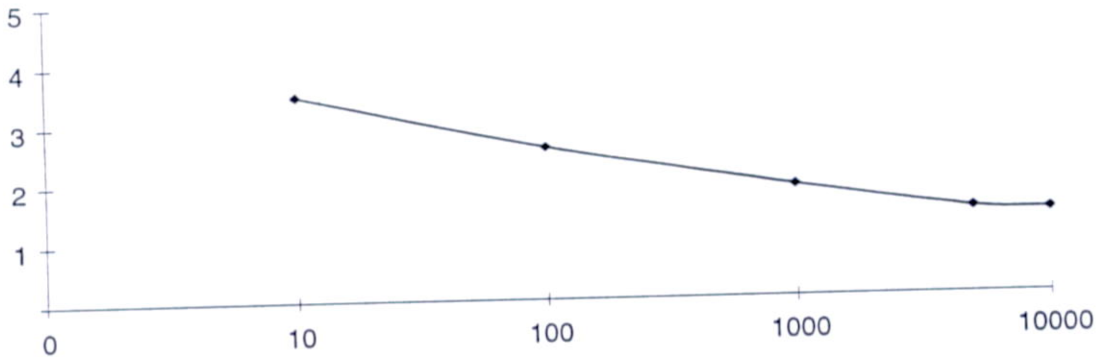


Figure A-21.10
Mean Absolute Error Vs Number of Iterations for *Whole Set*

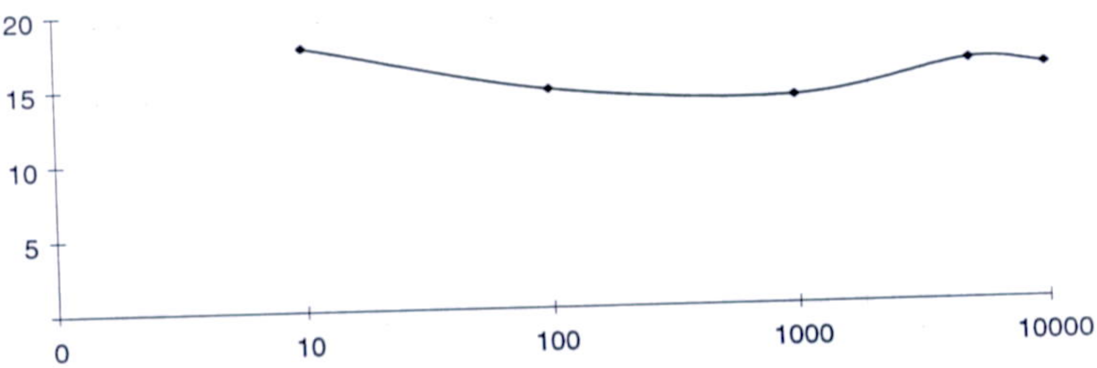


Figure A-21.11
Maximum Error Vs Number of Iterations for *Whole Set*

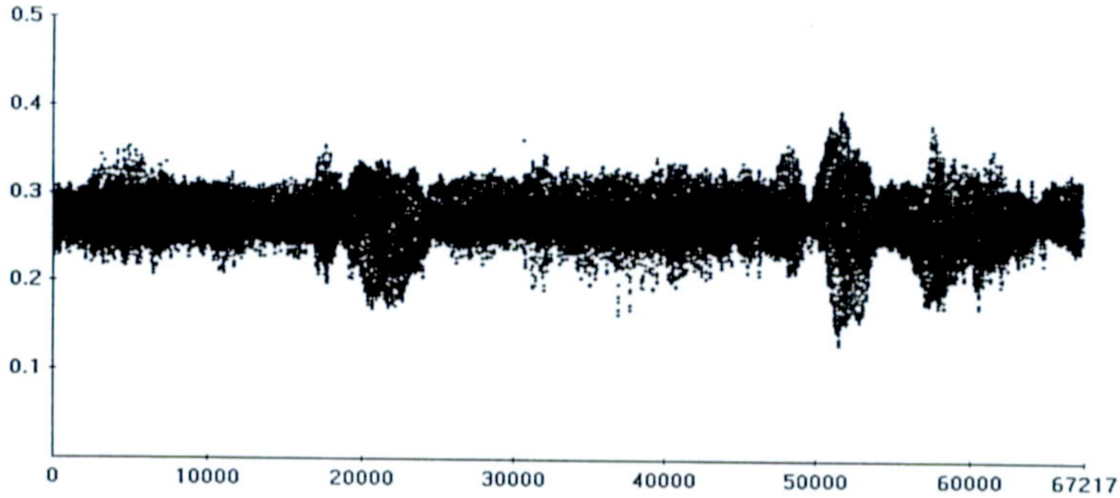


Figure A-21.12
Actual Strain Signal

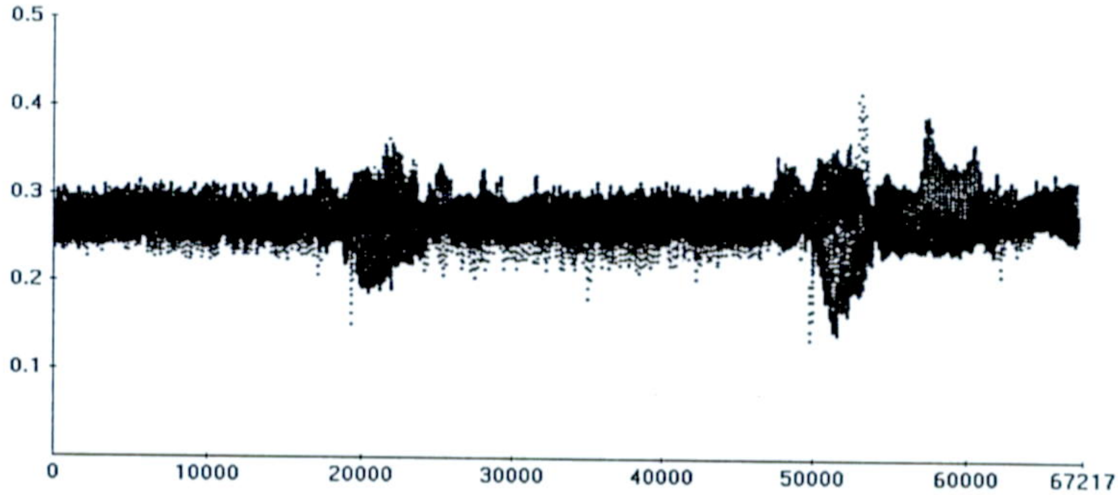


Figure A-21.13
Predicted Signal after *10,000 Iterations*

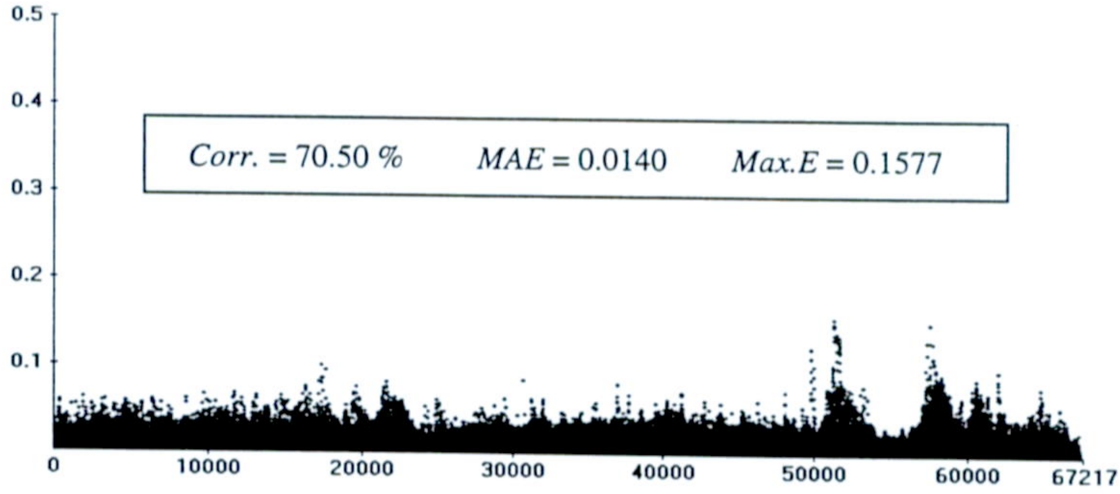


Figure A-21.14
Absolute Difference Between Actual and Predicted Signals

Experiment Definition

No.[1]	: 21a
Directory	: MBPN\TEST21
Tool	: HNC Explore Net
Applications	: *MTrain* for training
	MTest for testing

Application Parameters

Size of Input Set	: 6721
Training Data	: T02.DAC
Sawtooth File	: ON
Training Method	: Normal training

Network Structure

Paradigm	: Multi-Layer Backpropagation (*MBPN*)
Input Layer Size	: 21
Output Layer Size	: 1
No. of Hidden Layer	: 1
Hidden Layer Size	: 60
Input-Output Conn.	: OFF

Training Parameters

Learning Method	: Normal
Learning Rate[2] (α)	: 0.1 (and 0.5 for hidden layer)
Activation Function	: Gaussian
	(Lower Limit = 0, Upper Limit =1, Steepness = 1)
Random Seed	: 5
Initial Weight Max.[3]	: 0.5
Error Tolerance	: 0
Lin.Act.Func.on Ou.	: OFF

[1] The status of 10,000 iterations of Experiment 21 was used as initial state in this experiment

[2] After 500 iterations, learning rate for hidden layer was 0.5 like in Experiment 21.

[3] Initial weight maximum had no function here as previous weights were used.

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
	$\alpha = 0.1$		
Initial	78.64	1.17	9.31
100	80.29	1.15	7.35
200	80.38	1.15	7.19
300	80.40	1.15	7.13
400	80.43	1.15	7.10
500	80.44	1.15	7.07
	$\alpha^{*} = 0.5$		
600	80.02	1.17	7.36
700	80.03	1.17	7.35

Table A-21a.1
Experiment Results on *Training Set*

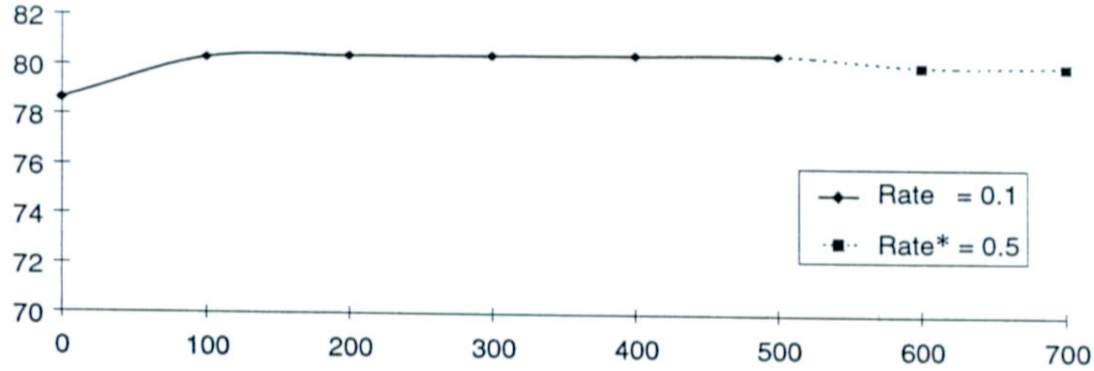


Figure A-21a.1
Correlation Coefficient Vs Number of Iterations for *Training Set*

---

* It is 0.5 for hidden layer, not for output layer.

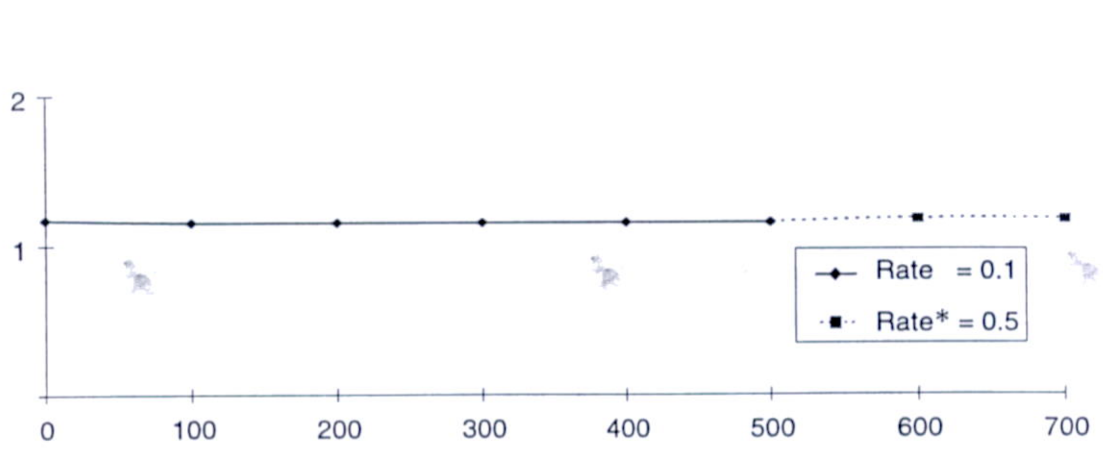


Figure A-21a.2
Mean Absolute Error Vs Number of Iterations for *Training Set*

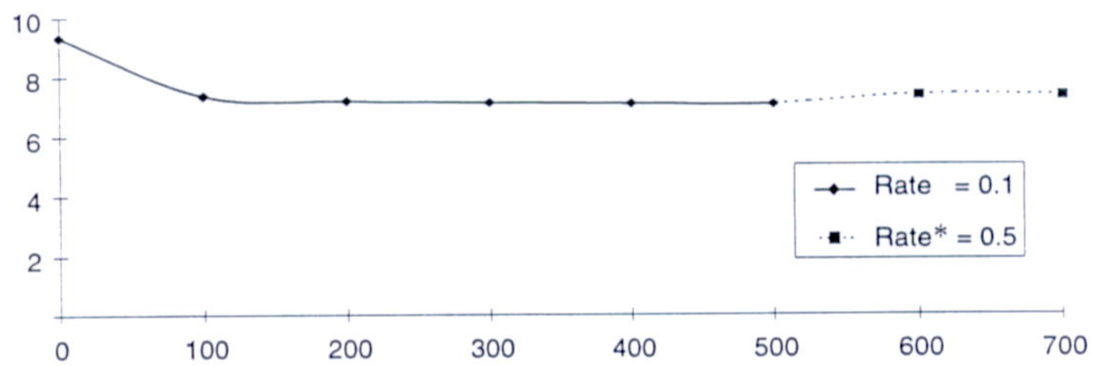


Figure A-21a.3
Maximum Error Vs Number of Iterations for *Training Set*

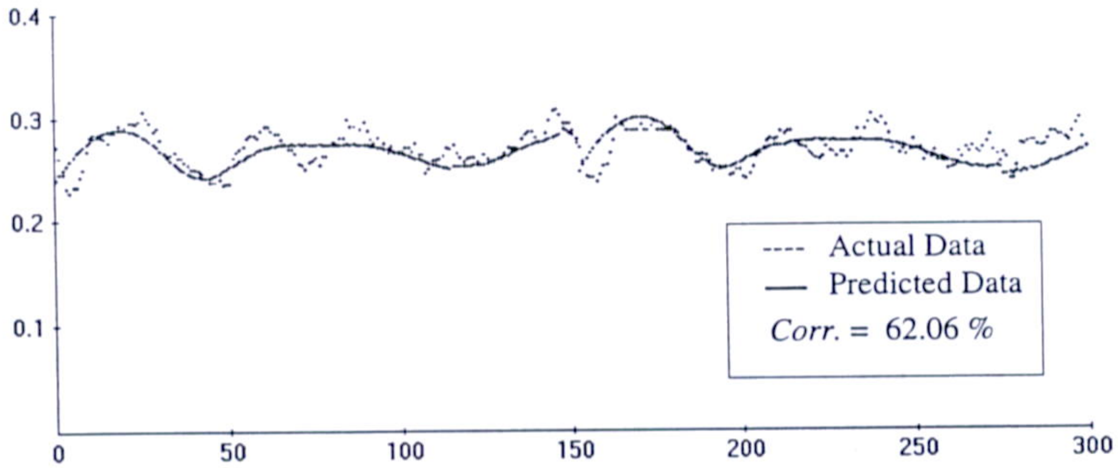


Figure A-21a.4
Test Result after 100 Iterations for *First 300 Data*

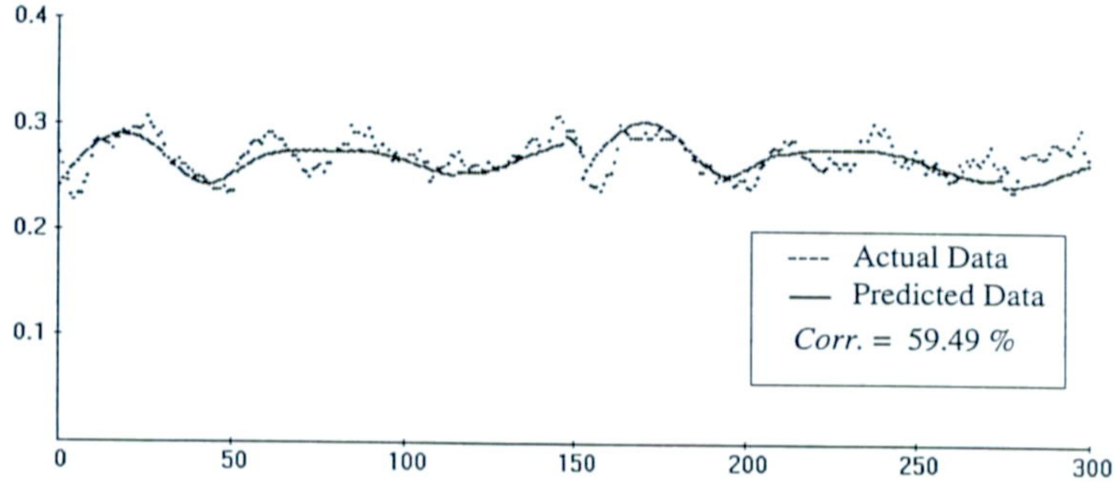


Figure A-21a.5
Test Result after 500 Iterations for *First 300 Data*

Number of Iterations	Correlation Coefficient (%)	Mean Absolute Error ($\times 10^{-2}$)	Maximum Error ($\times 10^{-2}$)
	$\alpha = 0.1$		
Initial	70.50	1.40	15.77
100	72.37	1.35	14.95
200	72.34	1.35	14.70
300	72.24	1.36	14.76
400	72.31	1.36	14.81
500	72.24	1.37	14.83
	$\alpha^{*} = 0.5$		
600	72.18	1.38	15.29
700	72.16	1.39	15.31

Table A-21a.2
Validation Results on *Whole Set*

* It is 0.5 for hidden layer, not for output layer.

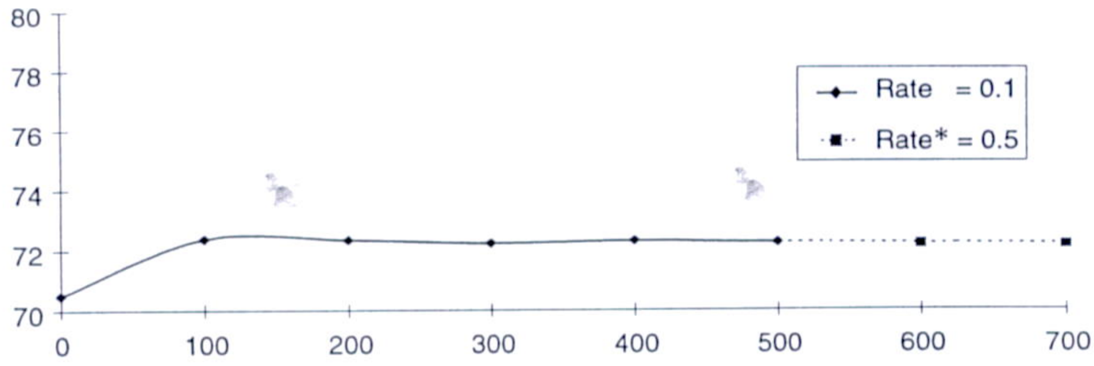


Figure A-21a.6
Correlation Coefficient Vs Number of Iterations for *Whole Set*

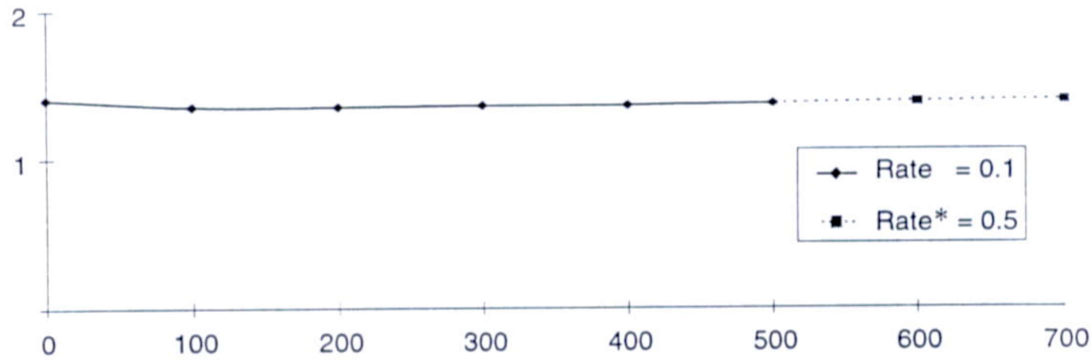


Figure A-21a.7
Mean Absolute Error Vs Number of Iterations for *Whole Set*

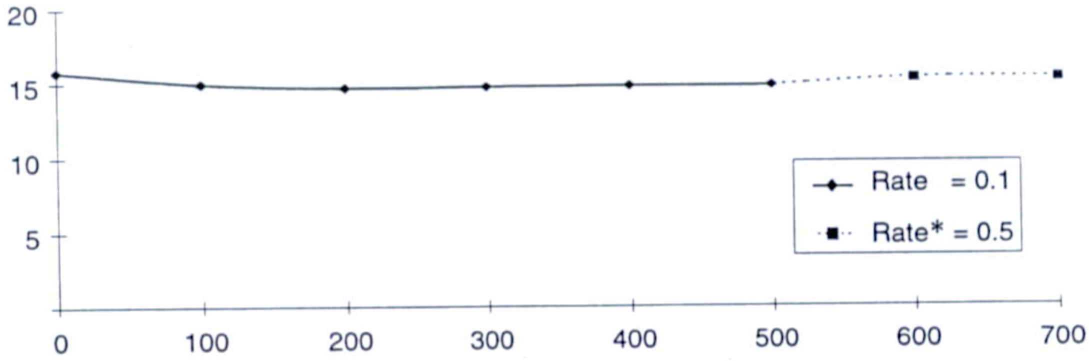


Figure A-21a.8
Maximum Error Vs Number of Iterations for *Whole Set*

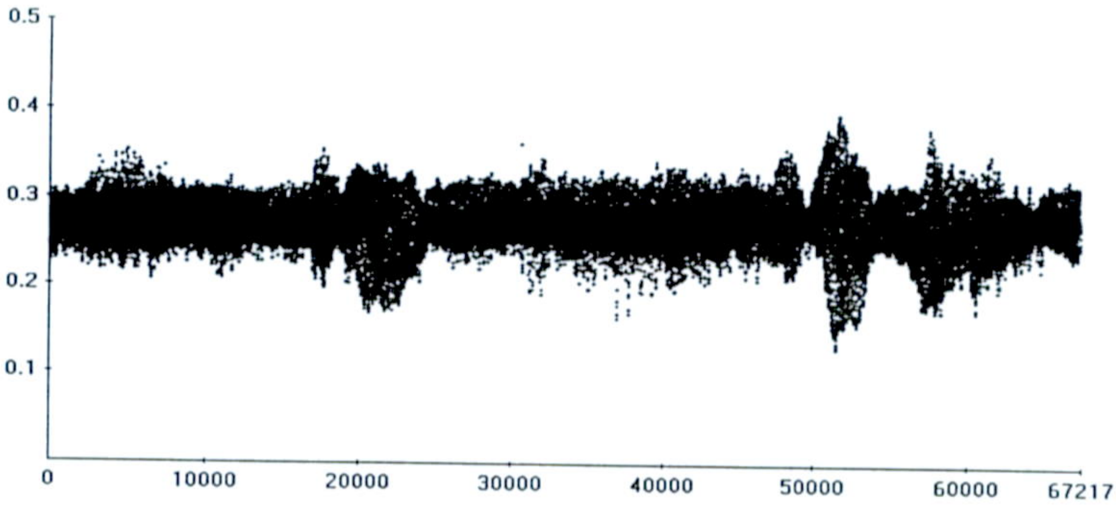


Figure A-21a.9
Actual Strain Signal

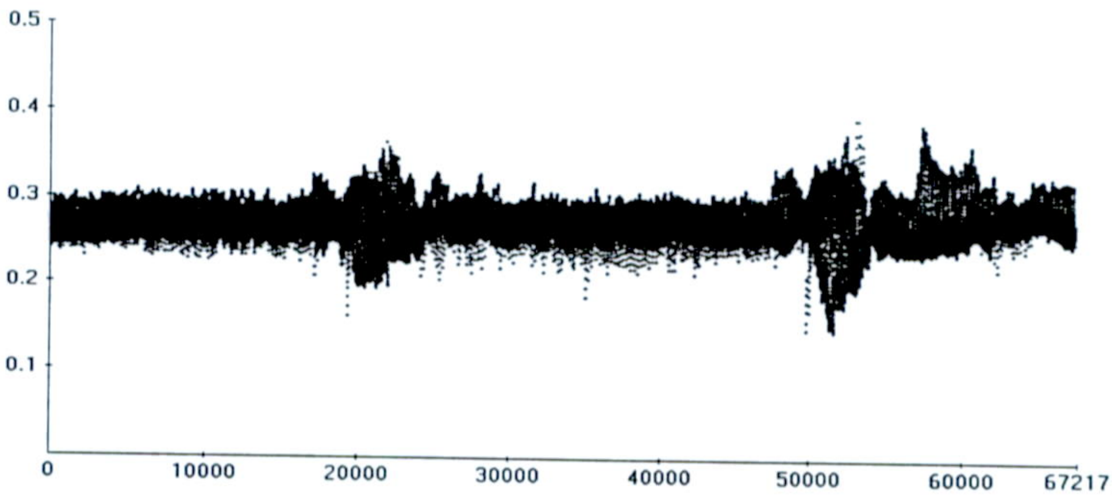


Figure A-21a.10
Predicted Signal after *100 Iterations (Total epochs = 10,100)*

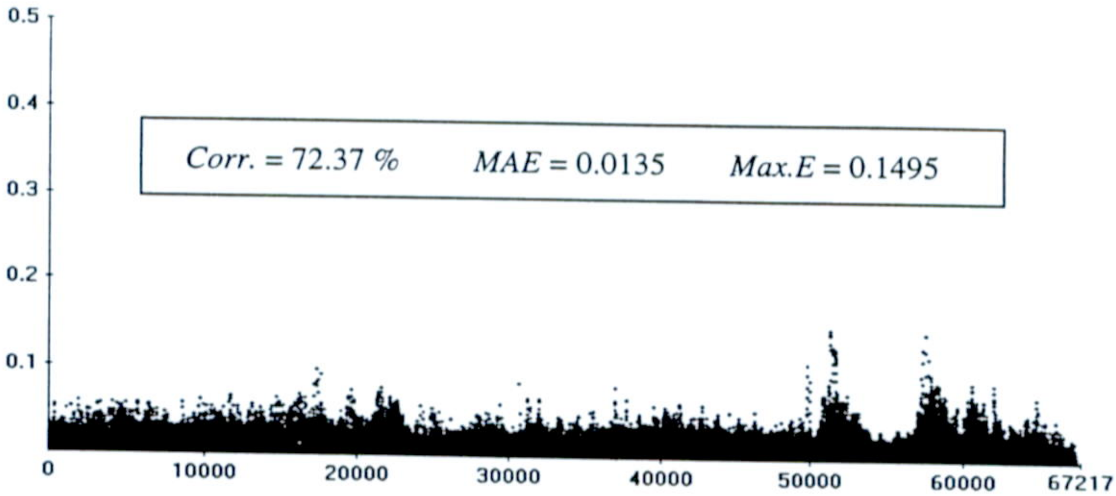


Figure A-21a.11
Absolute Difference Between Actual and Predicted Signals

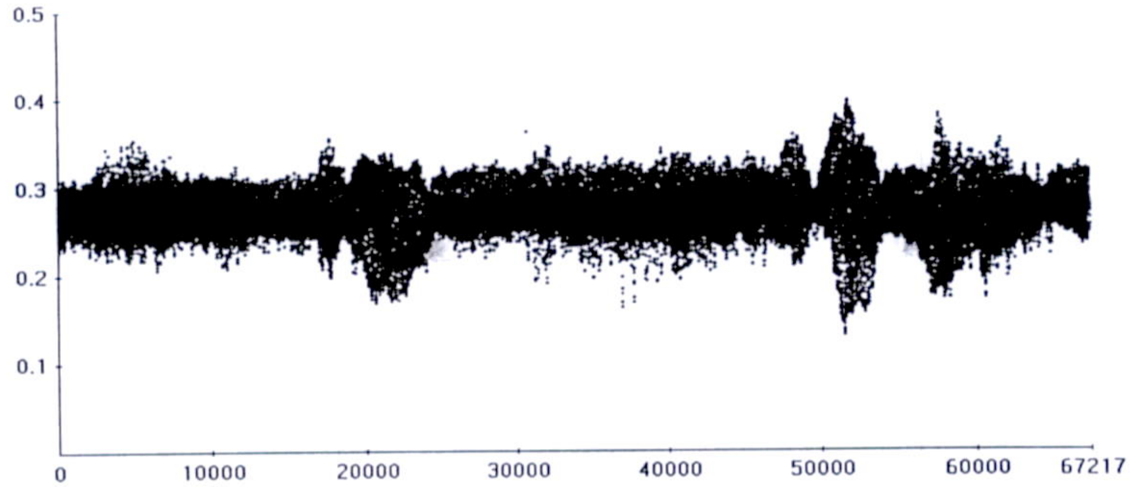


Figure A-21a.12
Actual Strain Signal

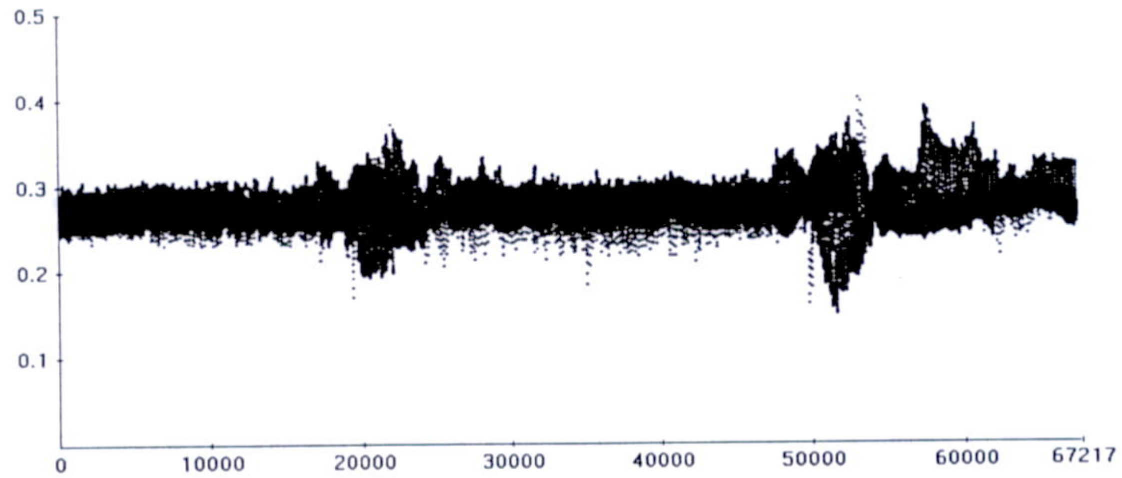


Figure A-21a.13
Predicted Signal after *500 Iterations (Total epochs = 10,500)*

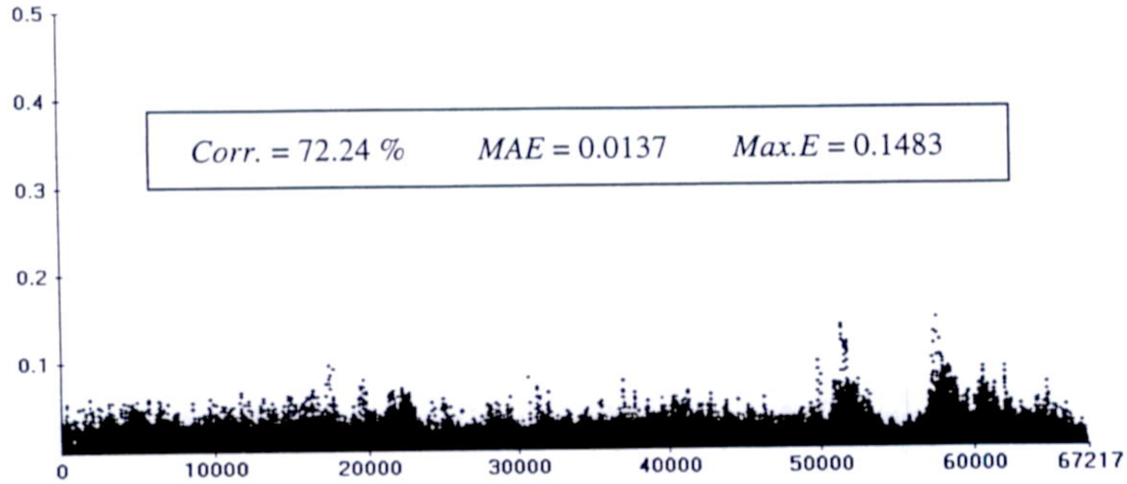


Figure A-21a.14
Absolute Difference Between Actual and Predicted Signals

Appendix B

FLIGHT PARAMETER DATA

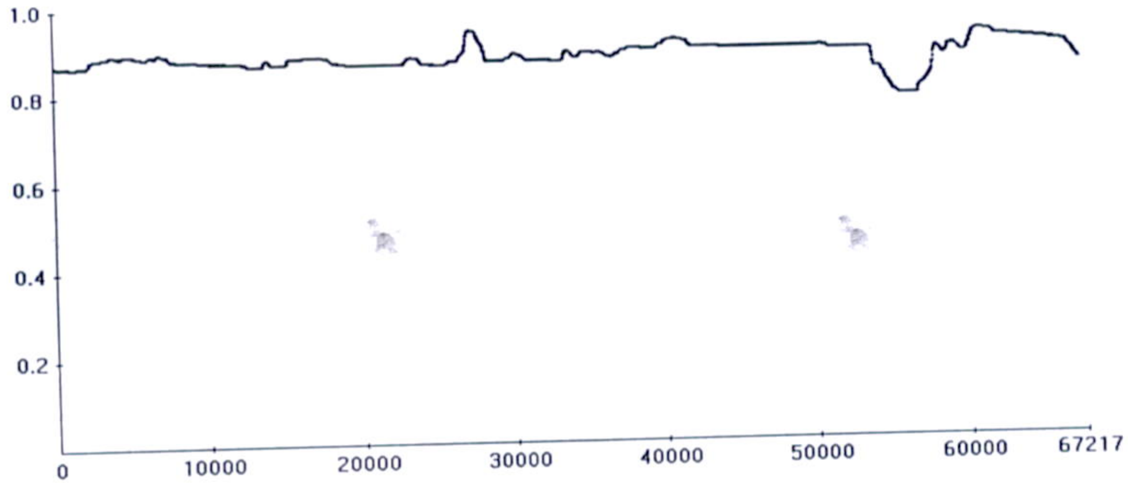


Figure B-1
Collective Lever Position

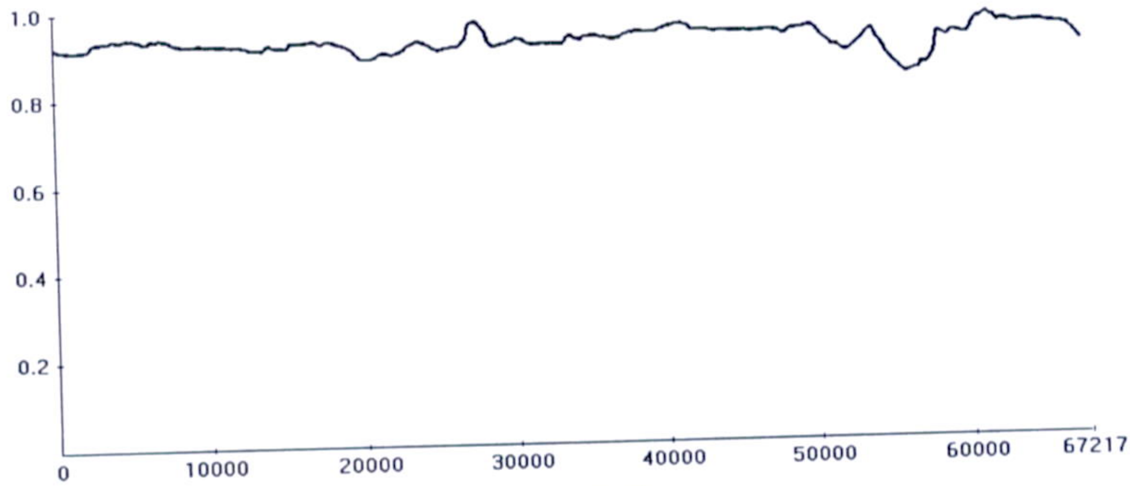


Figure B-2
Collective Servo Position

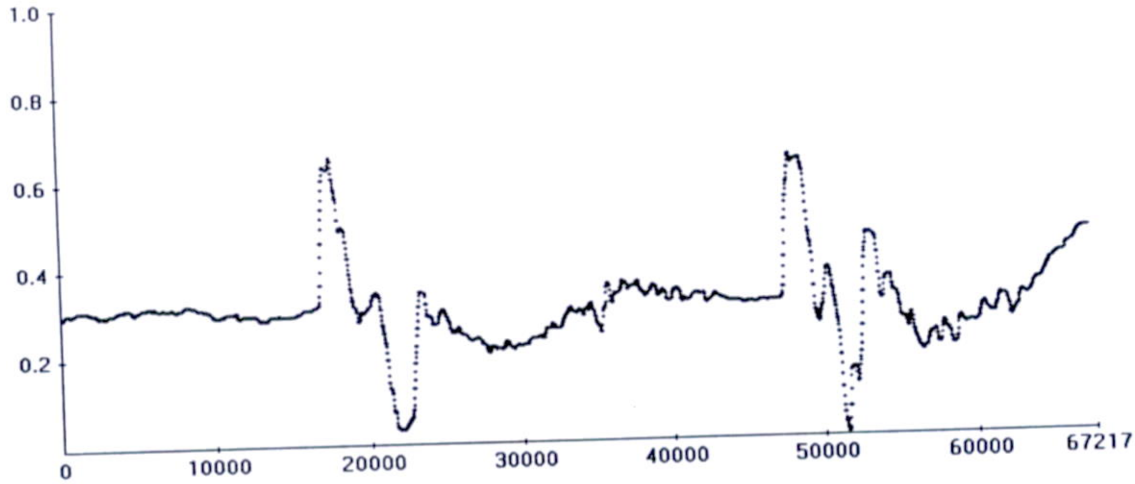


Figure B-3
F/A Cyclic Stick Position

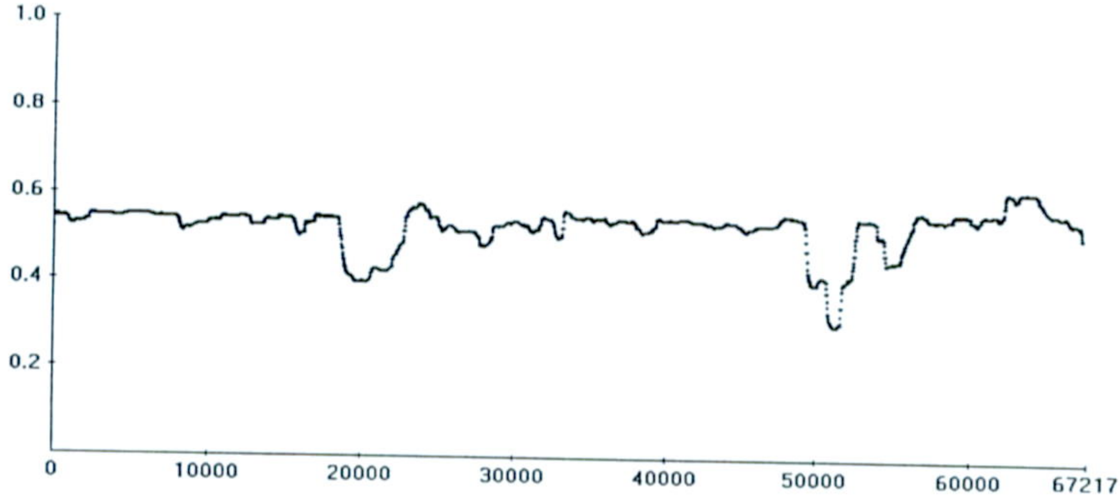


Figure B-4
Control Cyclic Stick Position

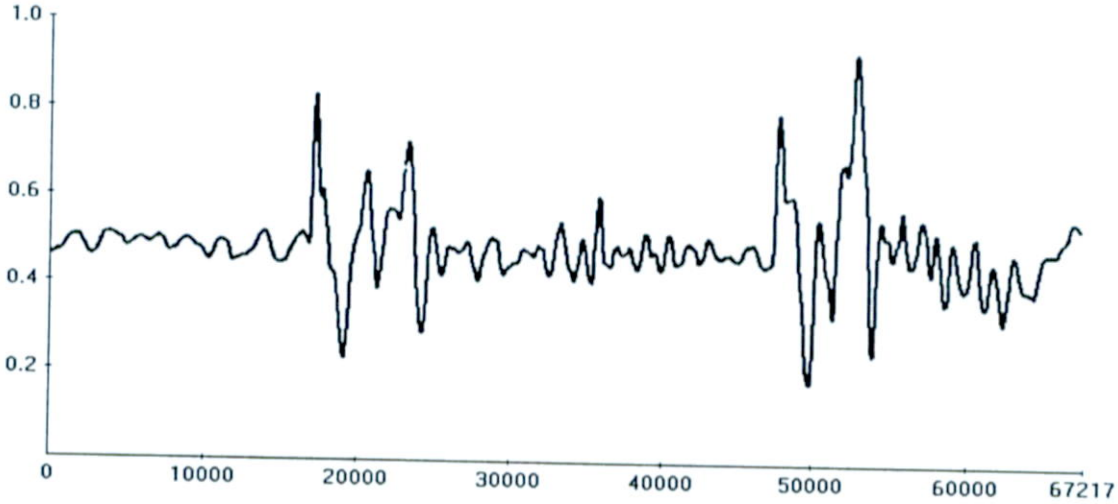


Figure B-5
F/A Cyclic Servo Position

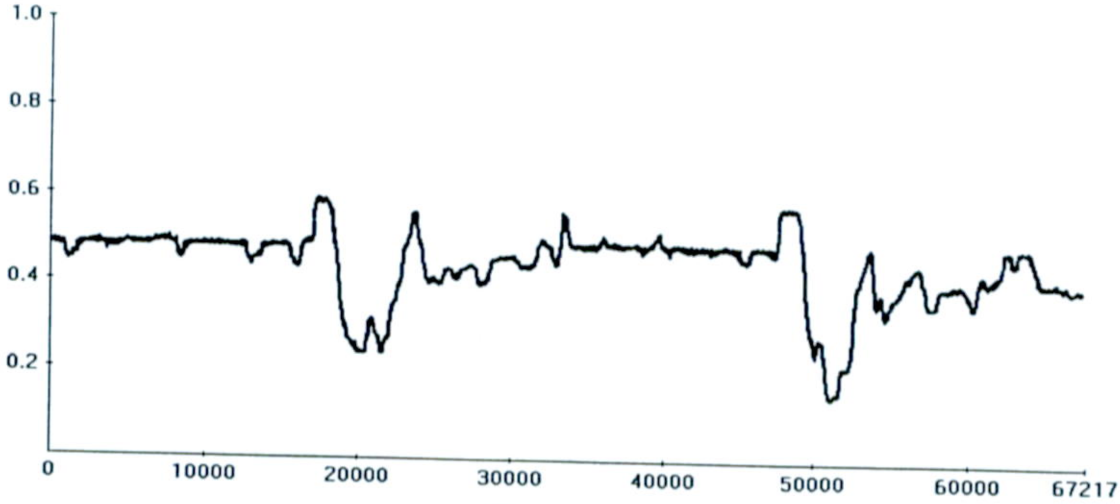


Figure B-6
Control Cyclic Servo Position

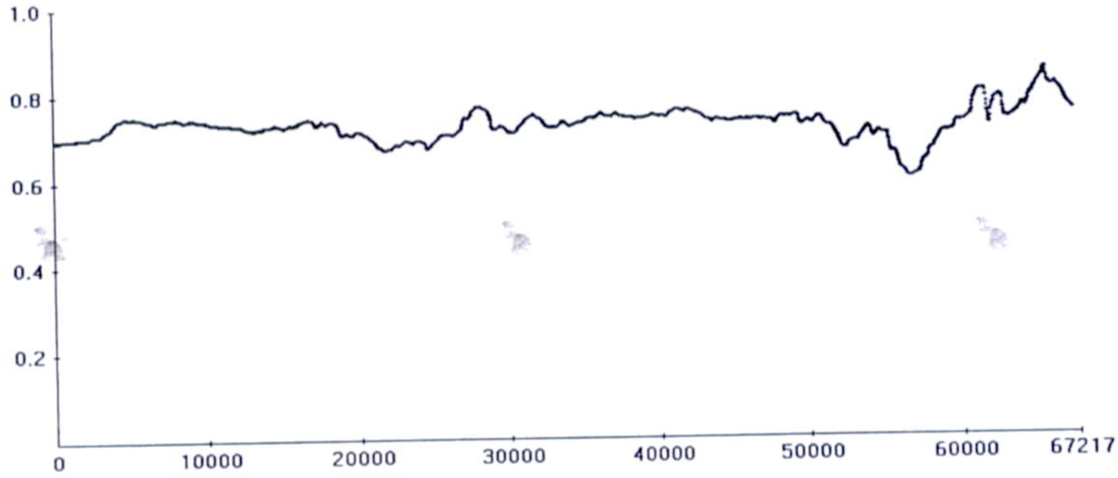


Figure B-7
Tail Rotor Pitch Angle

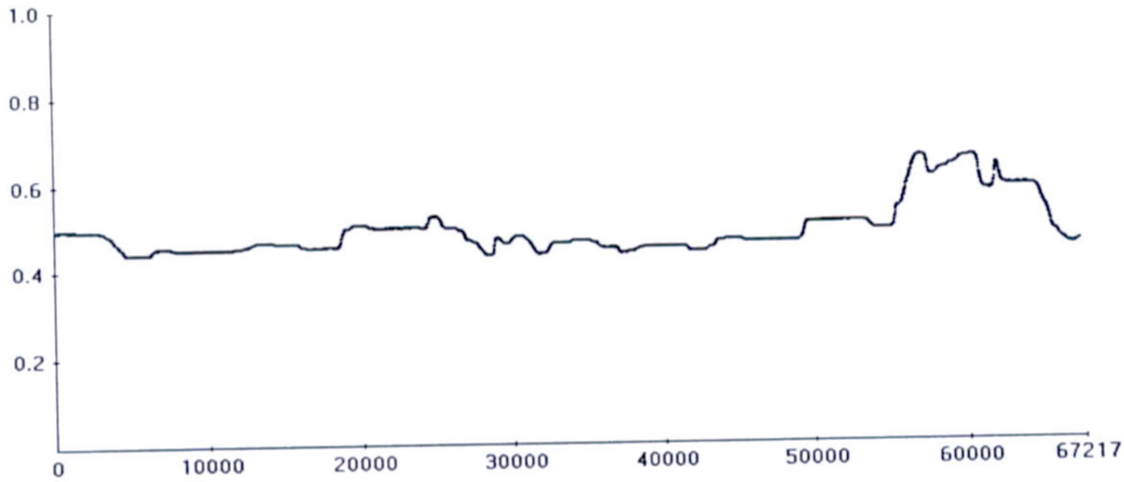


Figure B-8
Rudder Pedal Position

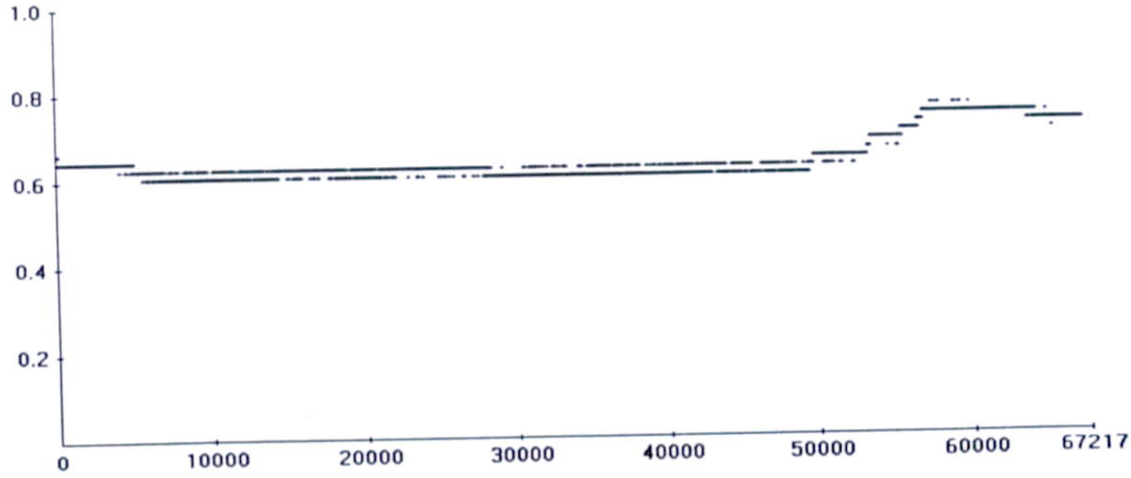


Figure B-9
Barometric Altitude

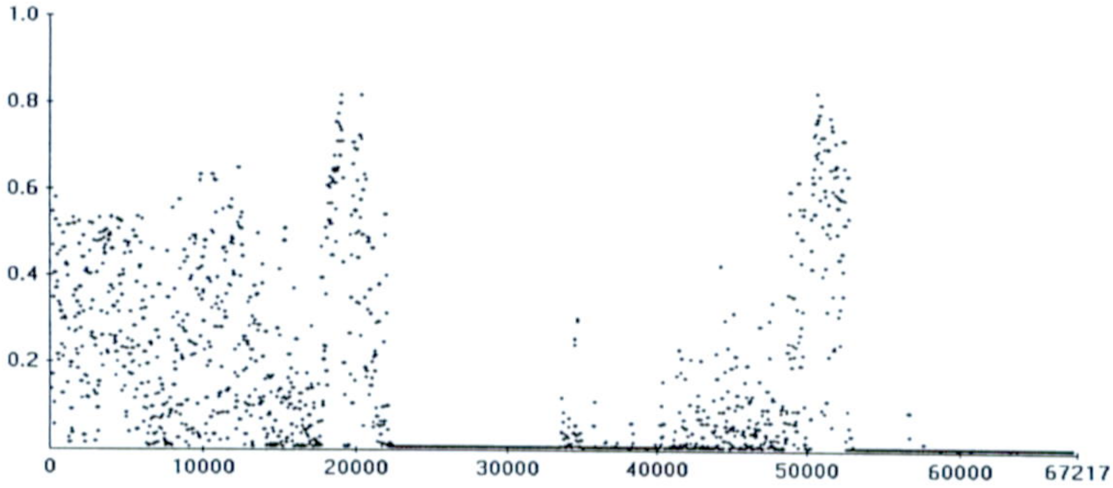


Figure B-10
Barometric Airspeed

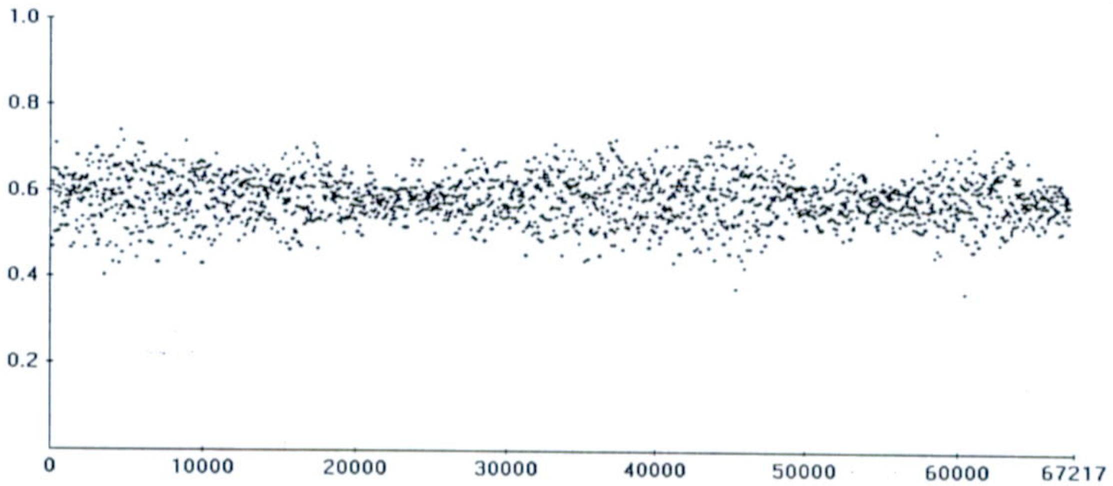


Figure B-11
Lateral Acceleration

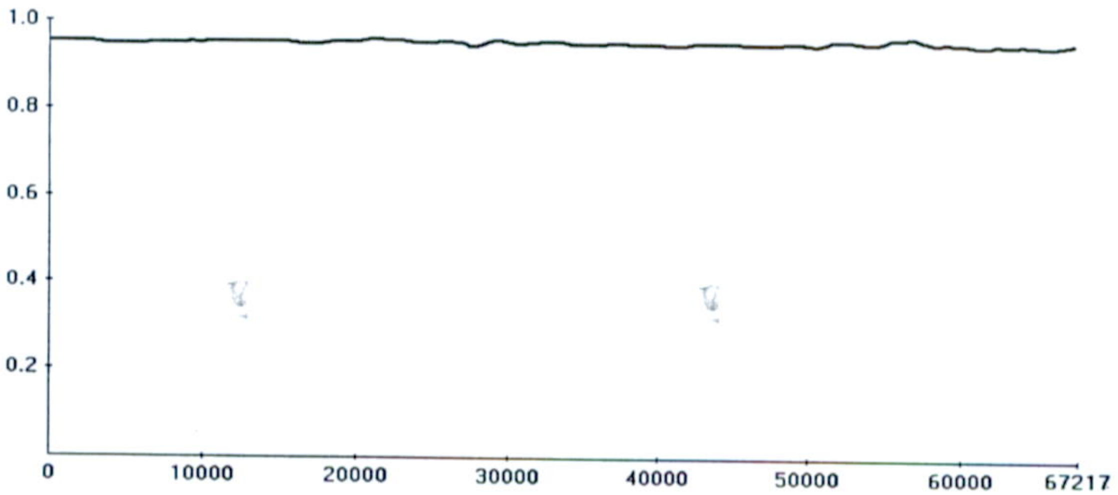


Figure B-12
Main Rotor Speed

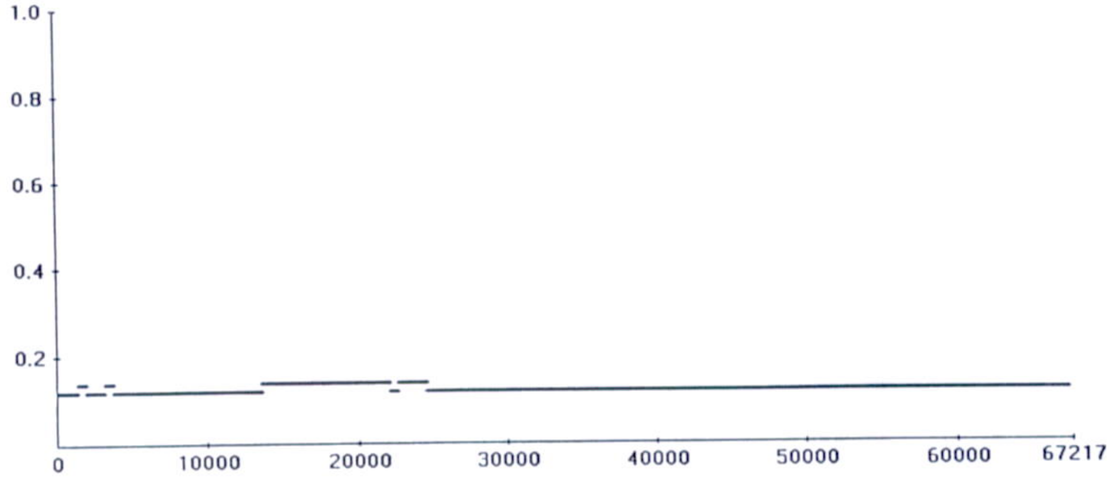


Figure B-13
Outside Air Temperature

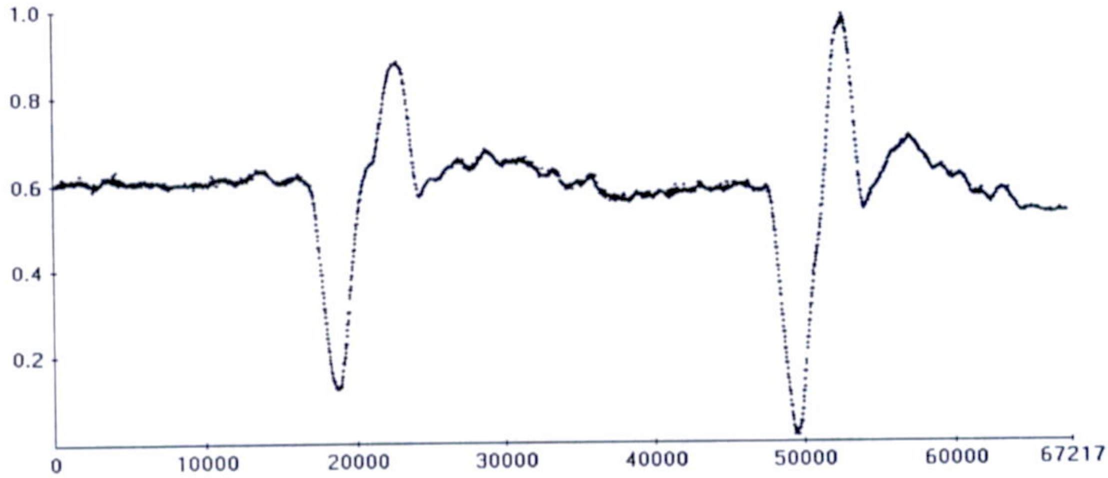


Figure B-14
Pitch Attitude

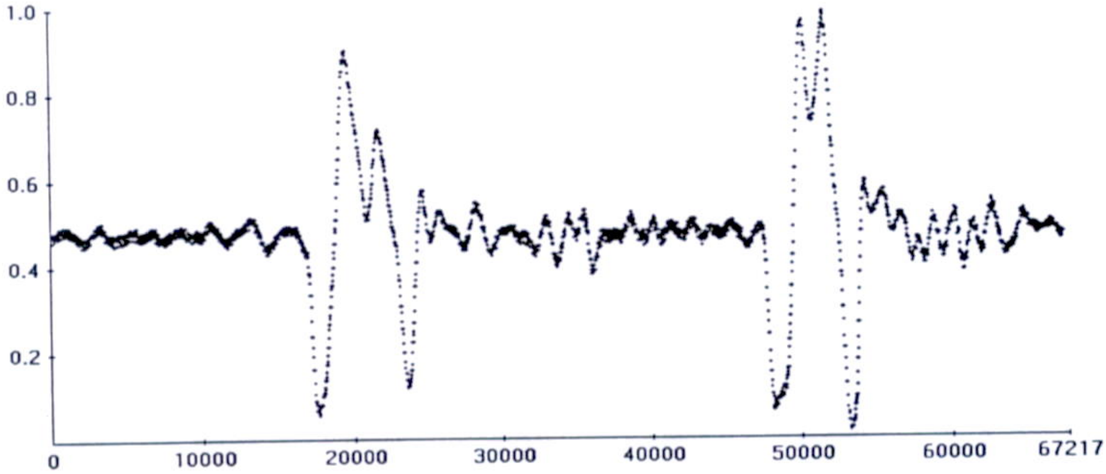


Figure B-15
Pitch Rate

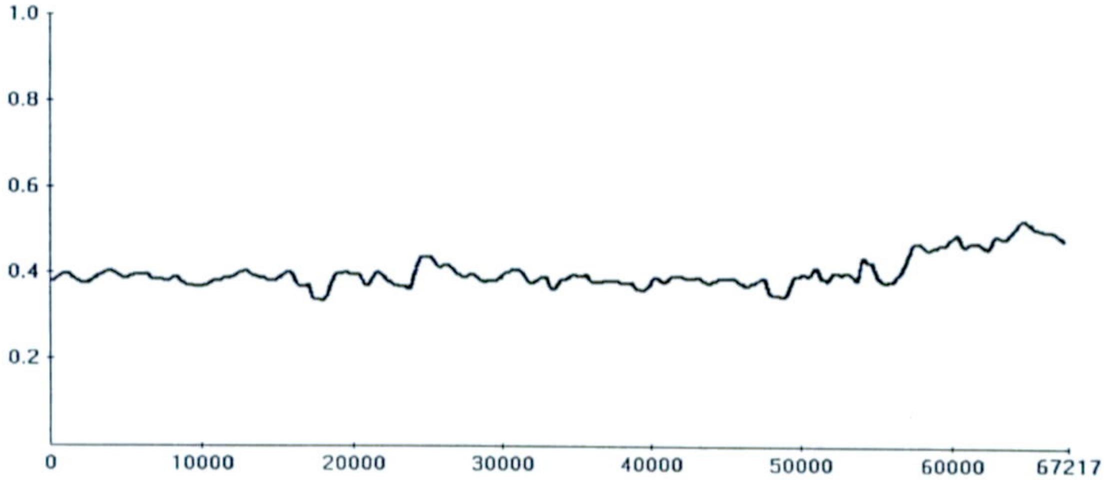


Figure B-16
Roll Attitude

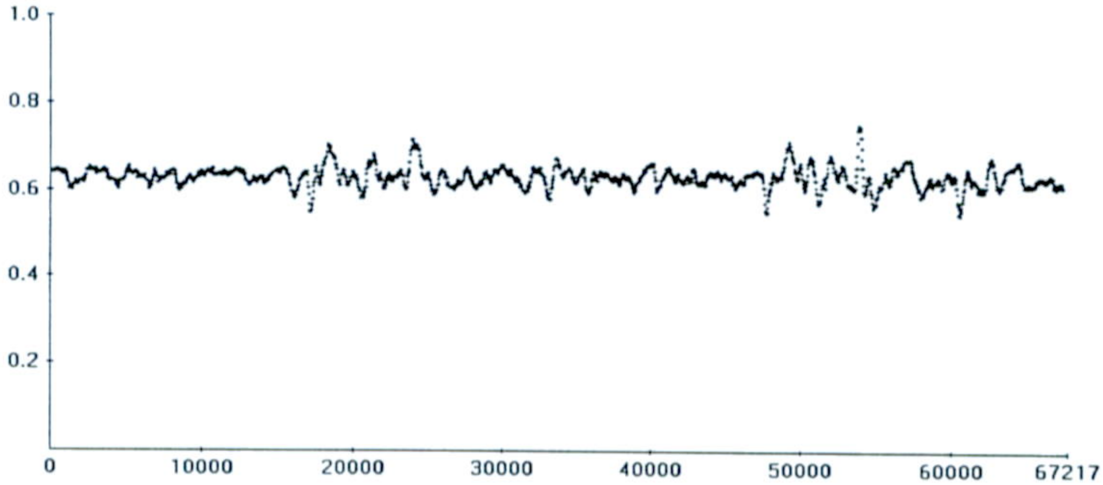


Figure B-17
Roll Rate

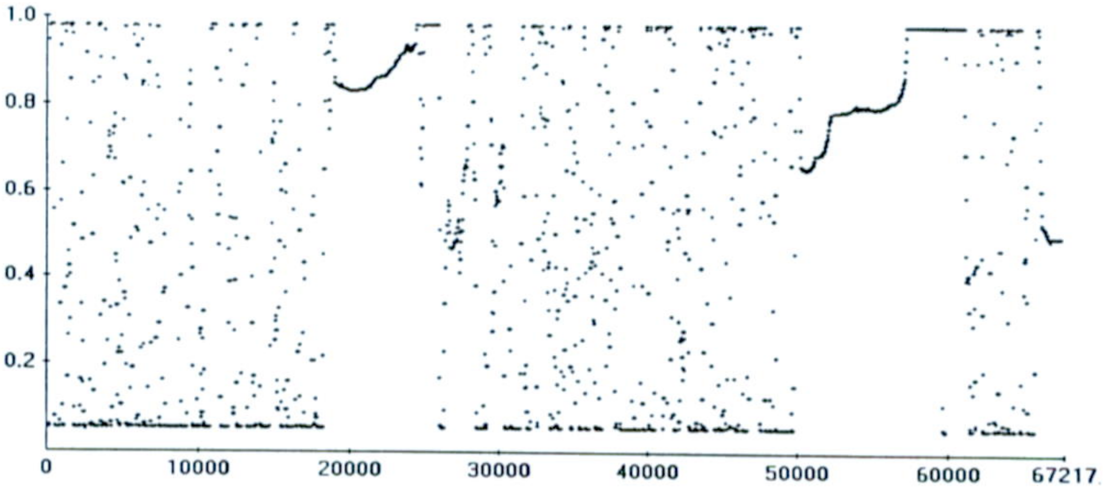


Figure B-18
Aerodynamic Yaw

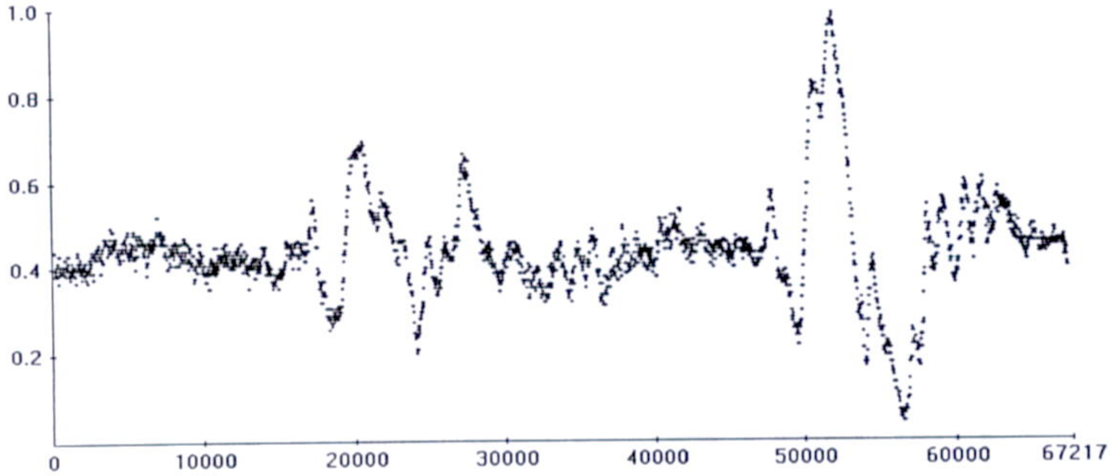


Figure B-19
Normal Acceleration Centre of Gravity

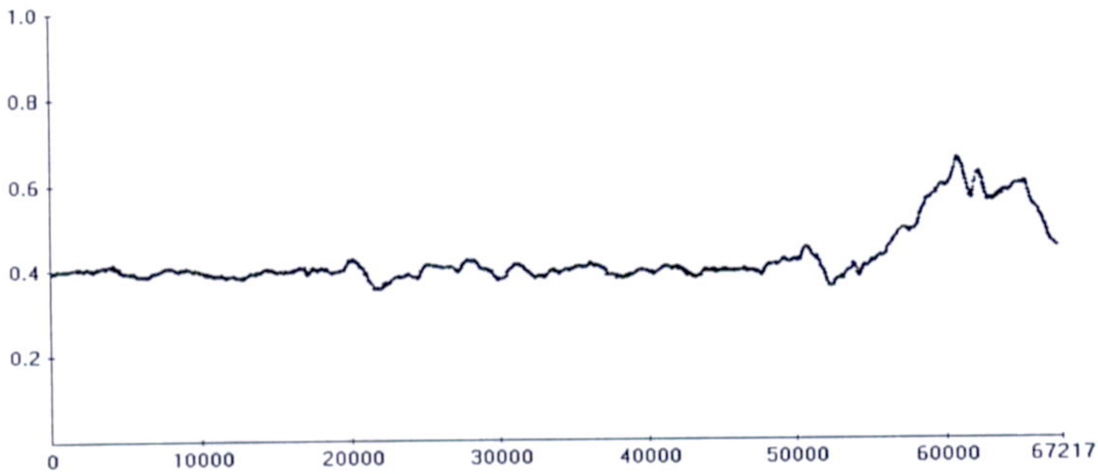


Figure B-20
Yaw Rate

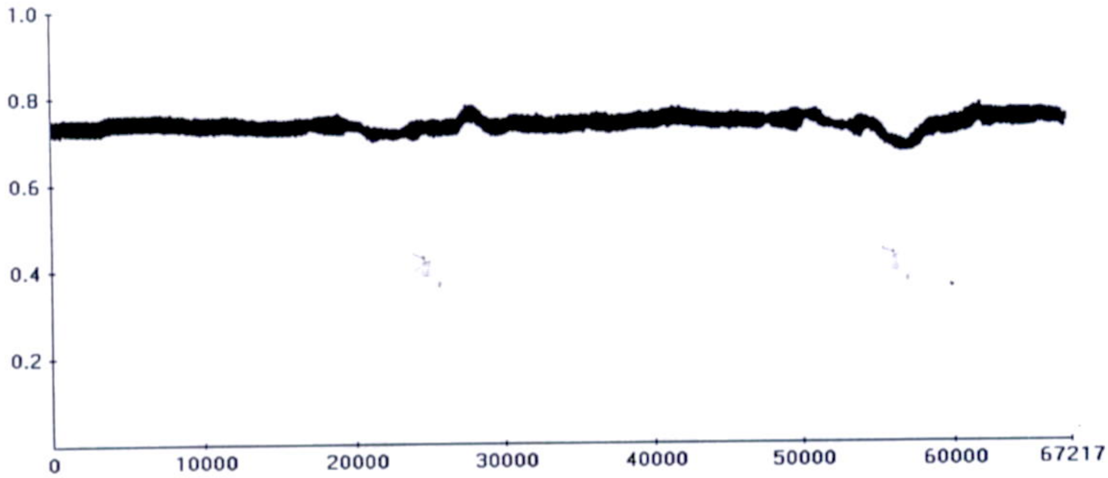


Figure B-21
Main Rotor Hub Torque