

NEURAL NETWORKS FOR IDENTIFICATION OF DYNAMIC SYSTEMS

A DISSERTATION

*submitted in partial fulfilment of the
requirements for the award of the degree*

of

MASTER OF ENGINEERING

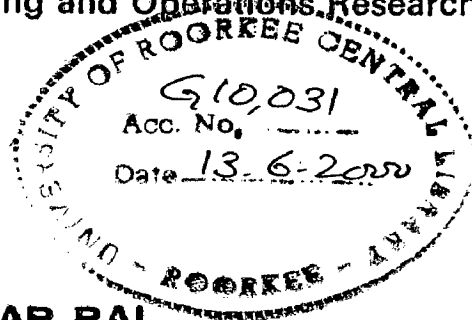
in

ELECTRICAL ENGINEERING

(With Specialization in System Engineering and Operations Research)

By

JITENDRA KUMAR RAI



DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247 667 (INDIA)

MARCH, 1999

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in this dissertation entitled "**NEURAL NETWORKS FOR IDENTIFICATION OF ~~NON~~ DYNAMIC SYSTEMS**" in the partial fulfilment of the requirements for the award of degree of Master of Engineering in **System Engineering and Operations Research**, submitted in the department of Electrical Engineering, University of Roorkee, Roorkee is an authentic record of my own work, carried out with the effect from **August 1998** to **March 1999**, under the guidance of **Dr. H.O. Gupta**, Professor and **Dr. Surendra Kumar**, Reader, Department of Electrical Engineering, University of Roorkee, Roorkee.


I have not submitted the matter embodied in this dissertation for the award of any other degree.

Dated: 30-3-1999

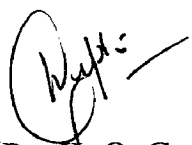
Jitendra Kumar Rai
(Jitendra Kumar Rai)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of our knowledge and belief.


(Dr. Surendra Kumar)

Reader,
Department of Electrical Engg,
University of Roorkee,
Roorkee-247 667


(Dr. H. O. Gupta)

Professor,
Department of Electrical Engg,
University of Roorkee,
Roorkee-247 667

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude to my guides **Dr.H.O.Gupta & Dr.Surendra Kumar** for encouraging me to undertake this dissertation as well as providing me all the necessary guidance and inspirational support throughout this work.

Special thanks to **Prof.M.K.Vasantha, & Dr.(Ms.)I.Gupta**, for encouraging, and providing the excellent facilities of the lab, at all times, whole-heartedly, without which it was impossible for me to complete the work.

I also take this opportunity to thank Mr. Sanjay Kumar Jain, Research scholar, for helping me in handling my computer with his extensive experience and knowledge which gave momentum and made the work roll smoothly. I also thank Mr. Belachew, Research Scholar for giving good suggestions throughout the work.

Special thanks are due to my friends, Priyaranjan, Srikanth, Harish, Ujjwal and Mihir, whose company I enjoyed and enabled me to ride some tempestuous incidents during my work..

Last, but not the least, I thank all the members of Electrical Engineering Department for their help and encouragement at the hour of need.

Jitendra Kumar Rai

ABSTRACT

Identification is the determination of a system within a specified class of system, on the basis of inputs and outputs. Such determination means that some variables, characterizing the given system are chosen and some relation are defined in the form of formula or graphs.

In this dissertation, a single layer ANN has been developed to identify the parameters of the linear dynamic system whose states and derivatives of states are given. Gradient descent algorithm has been used to learn the network. This algorithm made the learning very fast and provides global results. By this method, a non-linear system has also been identified in the form of a linear system about its operating point. Further, the effect of change in learning rate has been studied. This method has been successfully implemented on three sample systems and the results of identification of system parameter are reported

LIST OF FIGURES

- Fig. 2.1 The neural network computational model
- Fig. 2.2 Step and sigmoid function
- Fig. 2.3 Back-propagation network
- Fig. 3.1 State space system representation
- Fig. 4.1 Modular structure of ANNs using GD algorithm
- Fig. 4.2 A single layer feed forward ANN module
- Fig. 4.3 Flow-chart for normalising the data
- Fig. 4.4 Steepest descent directions
- Fig. 4.5 Flow-chart of gradient descent algorithm
- Fig. 5.1 Trajectory of w_{11} , w_{12} , b_{11} , and b_{12} .
- Fig. 5.2 Trajectory of w_{21} , w_{22} , b_{21} , and b_{22} .
- Fig. 5.3 Trajectories of w_{11} for different value of λ .
- Fig. 5.4 Trajectories of w_{12} for different value of λ .
- Fig. 5.5 Trajectories of a_{21} for different value of λ .
- Fig. 5.6 Trajectories of w_{22} for different value of λ .
- Fig. 5.7 Batch Reactor.
- Fig. 5.8 Trajectory of w_{11} , w_{12} , w_{13} , w_{14} , and w_{15} .
- Fig. 5.9 Trajectory of w_{21} , w_{22} , w_{23} , w_{24} , and w_{25}
- Fig. 5.10 Trajectory of w_{31} , w_{32} , w_{33} , w_{34} , and w_{35}
- Fig. 5.11 Trajectory of w_{41} , w_{42} , w_{43} , w_{44} , and w_{45}
- Fig. 5.12 Trajectory of w_{51} , w_{52} , w_{53} , w_{54} , and w_{55}

CONTENTS

CANDIDATE'S DECLARATION	i
ACKNOWLEDGEMENTS	ii
ABSTRACT	iii
LIST OF FIGURES	iv
Chapter 1 INTRODUCTION	1
1.1 IDENTIFICATION	1
1.2 PURPOSE OF IDENTIFICATION	2
1.3 RELATION BETWEEN IDENTIFICATION AND CONTROL	4
1.4 CLASSICAL IDENTIFICATION TECHNIQUES	5
1.5 ANN BASED IDENTIFICATION METHODS	8
1.6 ORGANISATION OF THE DISSERTATION	11
Chapter 2 NEURAL NETWORKS: AN OVERVIEW	12
2.1 HISTORICAL BACKGROUND	13
2.2 INTRODUCTION	13
2.2.1 NETWORK PROPERTIES.	15
2.2.2 NODE PROPERTIES	15
2.2.3 SYSTEM DYNAMICS	16
2.2.4 FUNCTION CLASSIFICATION	16
2.3 BACK-PROPAGATION	17
2.4 BACK-PROPAGATION ALGORITHM	17
Chapter 3 IDENTIFICATION OF LINEAR SYSTEM	20
3.1 INTRODUCTION	20
3.2 PERFORMANCE INDEX	21
3.3 PROBLEM FORMULATION	22
3.4 DYNAMIC SYSTEM PARAMETER ESTIMATION	25
Chapter 4 ANN IMPLEMENTATION FOR LINEAR SYSTEM IDENTIFICATION	27
4.1 DEVELOPMENT OF ANN MODULE	27
4.2 ARCHITECTURE OF THE MODEL	27

	4.3	TRAINING SET FOR NETWORK	30
	4.4	NORMALISATION	30
	4.5	GRADIENT OF A FUNCTION	32
		4.5.1 GRADIENT DESCENT LEARNING	32
	4.6	ALGORITHM	34
Chapter 5		RESULTS AND DISCUSSIONS	39
	5.1	INTRODUCTION	39
	5.2	TWO INPUT SYSTEM	39
		5.2.1 PROBLEM FORMULATION	39
		5.2.2 EXPERIMENTAL DATA	40
		5.2.3 RESULTS	40
	5.3	BLOOD GLUCOSE REGULATION	45
		5.3.1 PROBLEM FORMULATION	45
		5.3.2 EXPERIMENTAL DATA	46
		5.3.3 RESULTS	46
	5.4	BATCH REACTOR	53
		5.4.1 PROBLEM FORMULATION	55
		5.4.2 EXPERIMENTAL DATA	56
		5.4.3 RESULTS	58
Chapter 6		CONCLUSION AND FUTURE SCOPE	64
		REFERENCES	66
		APPENDIX A	68

INTRODUCTION

1.1 IDENTIFICATION

System identification problems are found everywhere in the science, medicine, and engineering. In such a problem, fundamental properties of a system are to be determined from observed behaviour of that system.

A wide class of identification problems may be numerically resolved by the use of modern mathematical and conceptual methods and high-speed computers. Many direct problems may be formulated in terms of systems of differential equations of the form

$$\dot{X}(t) = f(x, \alpha)$$

Here, t is independent variable, x is an n -dimensional vector whose components are the dependent variables, and α is the parameter of the system. When the parameter in $f(\cdot)$ and complete set initial conditions,

$$X(0) = C,$$

are known, a numerical integration produces the solution $x(t)$ on the interval $0 < t < T$. This initial value problem can be readily solved with a digital or an analog computer [2].

On the other hand, in identification problems, the quantity $x(t)$ or some function of $x(t)$ is approximately known at various times, while the parameters are not directly determinable. It is desirable to estimate the structure of the system as expressed by the

parameter vector α , and a complete set of initial conditions C . This may be regarded as the non linear boundary value problem in which the unknowns are some or all of the C 's and α 's.

Problems, which do not naturally occur in the form of systems of ordinary differential equations, may be expressed in that form in an approximate representation.

1.2 PURPOSE OF IDENTIFICATION

When formulating and solving an identification problem it is important to have the purpose of the identification. In control problems the final goal is often to design control strategies for a particular system. There are situations where the primary interest is to analyse the properties of system. Determination of rate coefficients in chemical reactions, heat transfer coefficients of industrial processes and reactivity coefficient in nuclear reactors are typical examples of such a “diagnostic” situation. In such a cases determination of specific parameter values might be the final goal of the identification. Many problems of this type are also found in biology, economy, and medicine [1].

Though the purpose of the identification is to design a control system the character of the problem might vary widely depending on the nature of the control problem. A few examples are given below:

Design a stable regulator.

Design a control program for optimal transitions form one state to another.

Design a regulator that minimises the variations in process variables due to disturbances.

In the first case it might be sufficient to have a fairly crude model of the system dynamics. The second control problem might require a fairly accurate model of the system dynamics. In the third problem it is also necessary to have a model of the environment of the system. Assuming that the ultimate aim of the identification is to design a control strategy for a system, what would constitute a satisfactory solution from a practical point of view?

In most practical problems there is seldom-sufficient a priori information about a system and its environment to design a control system from a priori data only. It is often necessary to make some kind of experiment, and observe the process while using perturbations as input signals and observe the corresponding changes in process variables. In practice there are severe limitations on the experiments that can be performed. In order to get realistic models it is often necessary to carry out the experiments during normal operation. This means that if the system is perturbed, the perturbations must be small so that the production is hardly disturbed. It might be necessary to have several regulators in operation during the experiment in order to keep the process fluctuations within acceptable limits. This may have an important influence on the estimation results.

When carrying out identification experiments of this type, there are many questions, which arise naturally:

How should the experiment be planned? Should a sequential design be used, i.e. plan an experiment using the available a priori information, perform that experiment, plan a new experiment based on the results obtained, etc. When should the experimentation stop?

What kind of analysis should be applied to the results of the experiment in order to arrive at control strategies with desired properties? What confidence can be given to the results?

What type of perturbation signal should be used to get as good results as possible within the limits given by the experimental conditions?

If a digital computer is used what is suitable choice of the sampling interval? In spite of the large amount of work that has been carried out in the area of system identification we have at present practically no general answers to the problems raised above. Since the general problems discussed above are very difficult to formalise one may wonder if there are rational answers to them. Nevertheless, it is worthwhile to recognise the fact that the final purpose of identification is often the design of a control system, since this simple observation may resolve many of the ambiguities of an identification problem. A typical example is the discussion whether the accuracy of identification should be judged on the basis of deviations in the model parameters or in time response. If the ultimate purpose is to design control systems then it seems logical that the accuracy of identification should be judged on the basis of the performance of the control system designed from the results of the identification.

1.3 RELATIONS BETWEEN IDENTIFICATION AND CONTROL

Whenever the design of a control system for a practically known process is approached via identification it is an a priori assumption that the design can be divided into two steps: identification and control [1]. In analogy with the theory of stochastic control we refer to this assumption as the separation hypothesis. The approach is very natural, in particular if we consider the multitude of techniques which have been

developed for the design of systems with known process dynamics and known environments. It is seldom true that optimum solutions are obtained if a process is identified and the results of the identification are used in a design procedure, developed under the assumption that the process and its environment are known precisely. It can be necessary to modify the control strategy to take into account the fact that the identification is not precise. Conceptually it is known how these problems should be handled. In the extreme case when identification and control are done simultaneously for a system with time-varying parameters the dual control concept can be applied. This approach lead to more computation even for simple cases.

It can also be argued that the problem of control process with unknown parameters can be approached without making reference to identification at all. As a typical example online tuning of PID regulators. In any case it seems to be a worthwhile problem to investigate rigorously under what conditions the separation hypothesis is valid.

1.4 CLASSICAL IDENTIFICATION METHODS

A number of methods have been proposed in the process identification literature. The conventional methods that are widely used in the industries are following [3]:

Direct Sine-Wave Testing

In this testing the input of the plant, which is usually a control -value position or a flow-controller set point, is varied sinusoidal at a fixed frequency w . After waiting for all transients to die out and for a steady state oscillation in the output to be established, the amplitude ratio and phase angle are found by recording input and output data. The data

point at this frequency is plotted on Nyquist, Bode, or Nichols plot. Then the frequency is changed to another value and a new amplitude ratio and phase angle is determined. Thus, the complete frequency-response curves are found experimentally by varying frequency over the range of interest. Approximate transfer functions can be fitted to the experimental curves.

Pulse-Testing

One of the most useful and practical methods for obtaining experimental dynamic data in many chemical engineering processes is pulse testing. It gives reasonably accurate frequency response curves and requires only a fraction of time that directs sine-wave-testing takes.

In this testing, the input pulse $m(t)$ of arbitrary shape is applied to the process. This pulse starts and ends at the same value and is often just a square pulse. The response of the output is recorded. It yields reasonably accurate frequency - response curve and requires only a fraction of the time that direct sine wave testing takes. If $m(t)$ is input and $x(t)$ is output. By definition, the transfer function of the process is

$$G(s) = X(s)/M(s)$$
$$= \int_0^{\infty} x(t)e^{-st} dt / \int_0^{\infty} m(t)e^{-st} dt$$

This testing does not work well on process that are highly non-linear because pulse tends to drive the process away from the steady state into a non-linear region unless the pulse height is made very small. If the load disturbances occur at the same time as the

pulse is being performed, then it creates problems. These other disturbances can effect the shape of the output response and produced poor results.

Step Testing

A plant operator makes changes from time to time in various input variables such as feed flow rate, steam flow rate etc. from one operating level to a new operating level. This changes can be instantaneous or gradual. The step-response data are obtained by recording the variables of interest for a few hours or days of plant operation.

These data are converted into frequency - response curves, by federating both input and output curves in the frequency domain. The process transfer function $G(s)=X(s)/M(s)$. Since the numerical differentiation is used to get frequency response from step-test data, the results are less reliable than pulse test data.

Least-Squares Method

In this method, basic idea is to use a difference-equation model for the process in which the current output x_n is related to previous values of the output (x_1, x_2, \dots) and present and past values of input (m_n, m_{n-1}, \dots). The relationship is linear, so classical least squares is used to solve for the best values of the unknown coefficients.

$$\bar{x}_{nm} = (a_0 m_n + a_1 m_{n-1} + \dots + a_M m_{n-M}) - (b_1 x_{n-1} + b_2 x_{n-2} + \dots + b_N x_{n-N})$$

State Estimators

The main idea behind this method is to solve mathematical models of the system that are solved on-line. These models usually assume linear ODEs, but non-linear

equations can be incorporated. The actual measured inputs to the process are fed into the model equations, and the model equations are integrated. Then the variable measured output variables are compared with the predictions of the model. The differences between the actual measured output variables and the predictions of the model for these same variables are used to change the model estimates through some sort of feedback. When differences between the predicted and measured variables are zero, the model predictions of all the state variable are changed.

1.5 ANN BASED IDENTIFICATION

In these methods physical system is identified in the form of a transfer function. Though the transfer function model provides us with simple and powerful analysis and design techniques, it suffers from certain drawbacks, e.g., a transfer function is only defined under zero initial conditions. Further, it has certain limitations due to the fact that the transfer function model is only applicable to linear time-invariant systems and it is generally restricted to SISO (single-input-single output) systems, as this approach becomes highly cumbersome for use in MIMO (multi-input multiple-output) systems. Another limitation of the transfer function technique is that it reveals only the system output for a given input and provides no information regarding the internal state of the system. There may be situations where the output of systems is stable and yet some of the system elements may have a tendency to exceed their specified ratings. It is observed that the classical design methods (root locus and frequency domain methods) based on the transfer function model are essentially trial and error procedures. Such procedures are difficult to visualise and organise even in moderately complex systems and may not lead to a control system, which yields an optimum performance in some defined sense [8].

Thus, the need of a more general mathematical representation of a system which, along with the output, yields information about the state of the system variables at some predetermined points along the flow of signals. Such considerations have led to the development of the state variable approach. It is a very powerful technique for the analysis and design of linear and non-linear, time-invariant or time-varying multi-input multi-output systems.

But, this method is local in scope i.e., optimum point is in the neighbourhood of the initial conditions. From the above discussion, a somewhat unsettling conclusion can be drawn "*conventional methods are not robust*". The implications of robustness for artificial systems are manifold. If artificial systems can be made more robust, costly redesigns can be eliminated, If higher levels of adaptation can be achieved, existing system can perform their functions longer and better.

Thus the desire for robust methods led the birth of non-traditional methods (one of them is, ANN), which have theoretically and practically proved to provide robust solution in the complex process identification.

Narendra and Parthasarthy [5] have focused on the applications of neural networks for linear system identification. They proposed feed forward and recurrent network for identification of a non-linear time-invariant dynamic system, by measuring the inputs, states and derivatives of states. However, for some problems, a globally optimal solution is not guaranteed by the network, the network computes locally optimum solution. In such cases it is not used, as it gives quite inaccurate results and is computationally complex.

Wang and Lin [11] have proposed Runge-Kutta neural network for identification of unknown dynamical systems described by ordinary differential equations. The proposed network precisely estimates the changing rates of the system states directly in their subnetworks based on space-domain interpolation instead of time-domain interpolation

Elias and Marios [7] have proposed a discrete-time recurrent neural network models which stability depends on the modelling errors. In the case of no modelling error, the state error between the system and RHONN model converges to zero.

Baldi and Hornik [9] have surveyed several learning algorithms for feed forward as well as recurrent neural networks. Many doubts have been answered regarding properties of error function, local connectivity and bias.

Bhama and Singh [6] have proposed a gradient descent learning algorithm, which is also known as modified back propagation to train a single layer neural network (SLNN) for identifying the parameters of linear dynamic systems. An appropriately formulated least mean square error (LMSE) is used as a performance measure for the network proposed.

The proposed network is simpler in structure and therefore less expensive in implementation. The structure is suitable for on-line computation because of the parallel nature.

1.6 ORGANISATION OF THE DISSERTATION

The present chapter I introduces the definition of identification problem and its goal. The classical identification techniques and brief literature review is introduced

Chapter II presents the concepts of neural network. Back propagation algorithm for multi layer neural networks has been developed.

In chapter III, The dynamic system identification, problem formulation, and performance index has been discussed. An algorithm for the dynamical system parameter estimation has been developed.

Chapter IV presents the structure of the neural networks, which has been developed in this dissertation. The gradient descent technique algorithm has been developed to train the Ann.

In chapter V some selected practical examples has been solved using gradient descent algorithm, which has been developed in this dissertation.

Chapter VI presents some conclusions and further scope of the dissertation.

NEURAL NETWORKS: AN OVERVIEW

2.1 HISTORICAL BACKGROUND

The information processing principles of biological neural networks have been applied to building a computer system for solving difficult problems whose solutions normally require human intelligence. Hebb (1949) proposed a learning law that explained how a network of neurons learned from the data. Other researchers pursued this notion through many years and Rosenblatt is credited with the perceptron learning algorithm.

Later, Minsky and Papert (1969) pointed out theoretical limitations of single-layer neural network model in their book "Perceptrons". Due to this negative projection, research on ANNs lapsed into an eclipse. Despite the negative atmosphere, some researchers still continued their research and came out with meaningful results.

In the early 1980s, the neural network approach was resurrected. Hopfield (1982) introduced the idea of energy minimisation in physics into neural networks. His influential work endowed this technology with renewed momentum.

In 1985s, Rumelhart, Hinton, and Williams offered a powerful solution "back propagation learning algorithm" to training a multi layer neural networks and shattered the curse imposed on perceptrons.

Although the neural network approach rejects to notion of separating knowledge from the inference mechanism, it does not reject the importance of knowledge in many

tasks that require intelligence. It just was a different way store and manipulates knowledge.

2.2 INTRODUCTION

The neural network contains a large number of simple neuron like processing elements and a large number of weighted connections between the elements. The weight on the connections encodes the knowledge of the network. A simple view of the network structure and behaviour is given in Fig. 2.1. Construction of a neural network involves the following task:

- **Determine the network properties:** The network topology (connectivity), the types of connections, the order of connections, and the weight range.
- **Determine the node properties:** The activation range and the activation function.
- **Determine the system dynamics:** the weight initialisation scheme, the activation-calculating formula, and the learning rule.

2.2.1 Network Properties

The topology of a neural network refers to its framework as well as its interconnection scheme. The framework is often specified by the number of layers and the number of nodes per layer. The types of layer include:

- **The input layer:** The nodes in it are called input units, which encode the instance presented to the network for processing.

- **The hidden layer:** The nodes in it are called hidden units, which are not directly observable and hence hidden.
- **The output layer:** The node in it is called output units, which encode possible concepts to be assigned to the instance under consideration.

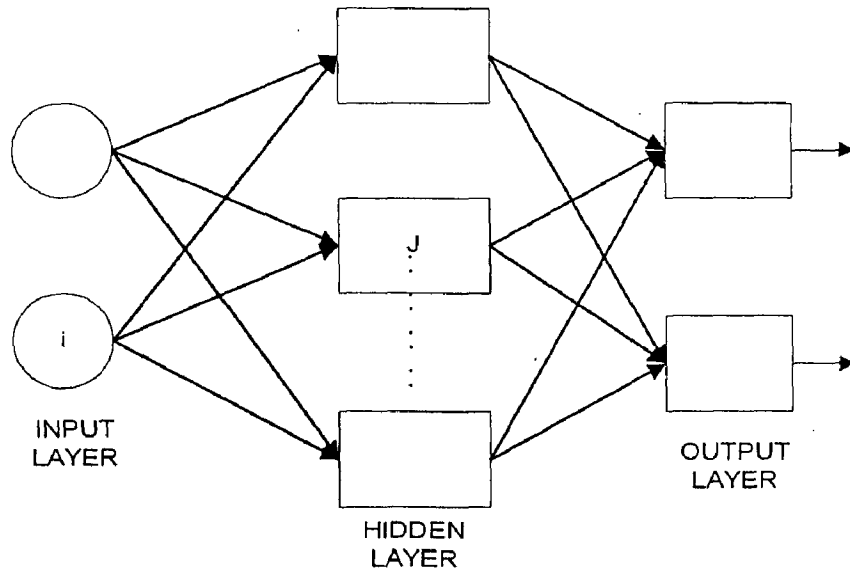


Fig. 2.1: The neural network computational model

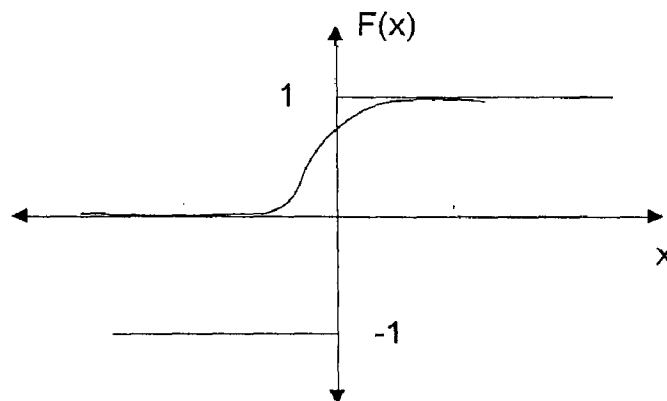


Fig. 2.2: Step and Sigmoid function

According to the interconnection scheme, a network can be either feed forward or recurrent.

- Feed forward networks: All connections point in one direction (from the input toward the output layer).
- Recurrent networks: There are feedback connections or loops.

2.2.2 Node Properties

The activation levels of nodes can be discrete (e.g. , 0 and 1) or continuous across a range (e.g. [0,1]) or unrestricted . This depends on the activation function chosen. One simple function that is appropriate for discrete neural networks is the step function, which is shown in Fig. 2.2

$$\begin{aligned}
 F(Y) &= 1 && , \text{if } Y > 0 \\
 &= F(Y) && , \text{if } Y = 0 \\
 &= -1 && , \text{if } Y < 0
 \end{aligned}$$

Where Y is the summation of the product of the incoming neurones activation and the synaptic weight of the connection:

$$Y_j = \sum_{i=0}^n x_i w_{ji} \quad \dots(2.1)$$

Where,

n is the number of incoming neurons

x_i is the activation of ith neurons.

w_{ji} is the vector of synaptic weights connecting the ith incoming neurones to the jth neurones we are examining.

Another popular class of function is the sigmoid or squashing function. An example of the sigmoid function, has been shown in fig 2.2 .

2.2.3 System Dynamics

The learning rule is one of the most important attributes to be specified for neural networks. The learning rule determines how to adopt connection weights in order to optimise the network performance. It indicates how to calculate the weight adjustment during each training cycle. However the rule is suspended after training is completed.

The activation levels of input need not be calculated since they are given. Those of hidden and output units are calculated according to the activation function used. If it is a sigmoid function, the activation level (O_j) of unit j is calculated by

$$O_j = 1/(1 + \exp(-Y_j - \theta_j)). \quad \dots(2.2)$$

Where Y_j is calculated from equation (2.1), and θ_j the threshold on unit j .

2.2.4 Function Classification

Neural computational models can be categorised in terms of their applications:

Classification: Assignment of the input data to one of a finite number of categories.

Association:

Auto association: retrieval of an object based on part of the object itself.

Hetro association: retrieval of an object (memory) in one set using another object (memory) in different set.

Optimisation: Finding the best solution - often by minimising a certain cost function.

Self-organisation: Organising received information using adaptive learning capabilities.

2.3 BACK PROPAGATION

The back-propagation network is a multi layer feed forward network with a different transfer function in the artificial neurone and a more powerful learning rule. The learning rule is known as back propagation, which is a kind of gradient descent technique with backward error propagation as shown in Fig.2.3.

The back propagation network, in essence, learns the mapping from a set of input patterns to a set of output patterns. This network can be designed and trained to accomplish a wide variety of mappings. It is capable of approximating arbitrary mappings given a set of examples. The sigmoid function grants that the output is bounded between 0 and 1.

2.3.1 Back Propagation Algorithm

The algorithm of back propagation training for three layer network is given as follows:

P training patterns are given: $\{z_1, d_1, z_2, d_2, \dots, z_p, d_p\}$

Where z_i is the input and d_i is output of the i th pattern.

Step 1: Initialisation:

The learning rate λ and error tolerance is chosen. Set all weight and node thresholds to small random numbers.

Step 2: Calculation of Activation:

Input is presented and the output is computed. The activation level of an output unit is determined by the instance presented to the network. The activation level O_j of a hidden and output unit is determined by eqn.(2.2).

Step 3: Calculation of Error:

$$E_j = 1/2(d_j - O_j)^2, \text{ for } j=1,2,\dots,k \quad \dots(2.3)$$

Step 4: Error gradient: The error gradient δ_j for both (hidden and output) layers are computed.

$$\text{For the output units } \delta_j = O_j(1 - O_j)(d_j - O_j) \quad \dots(2.4)$$

$$\text{For the hidden units: } \delta_j = O_j(1 - O_j) \sum_k \delta_k W_{kj} \quad \dots(2.5)$$

Where δ_k is the error gradient at unit k to which a connection points from the hidden unit j .

Step 5: Weight adjustment:

Start at the output units and work backward to the hidden layer recursively. Adjust weight by ,

$$W_{ji}(n+1) = W_{ji}(n) + \lambda \delta_j Y_i \quad \dots(2.6)$$

Where $W_{ji}(n)$ is the weight from unit i to j at time n .

Step 6: Repeat steps 2 to 5 for all patterns.

Step 7: Repeat iteration until convergence in terms of the selected error criteria satisfied.

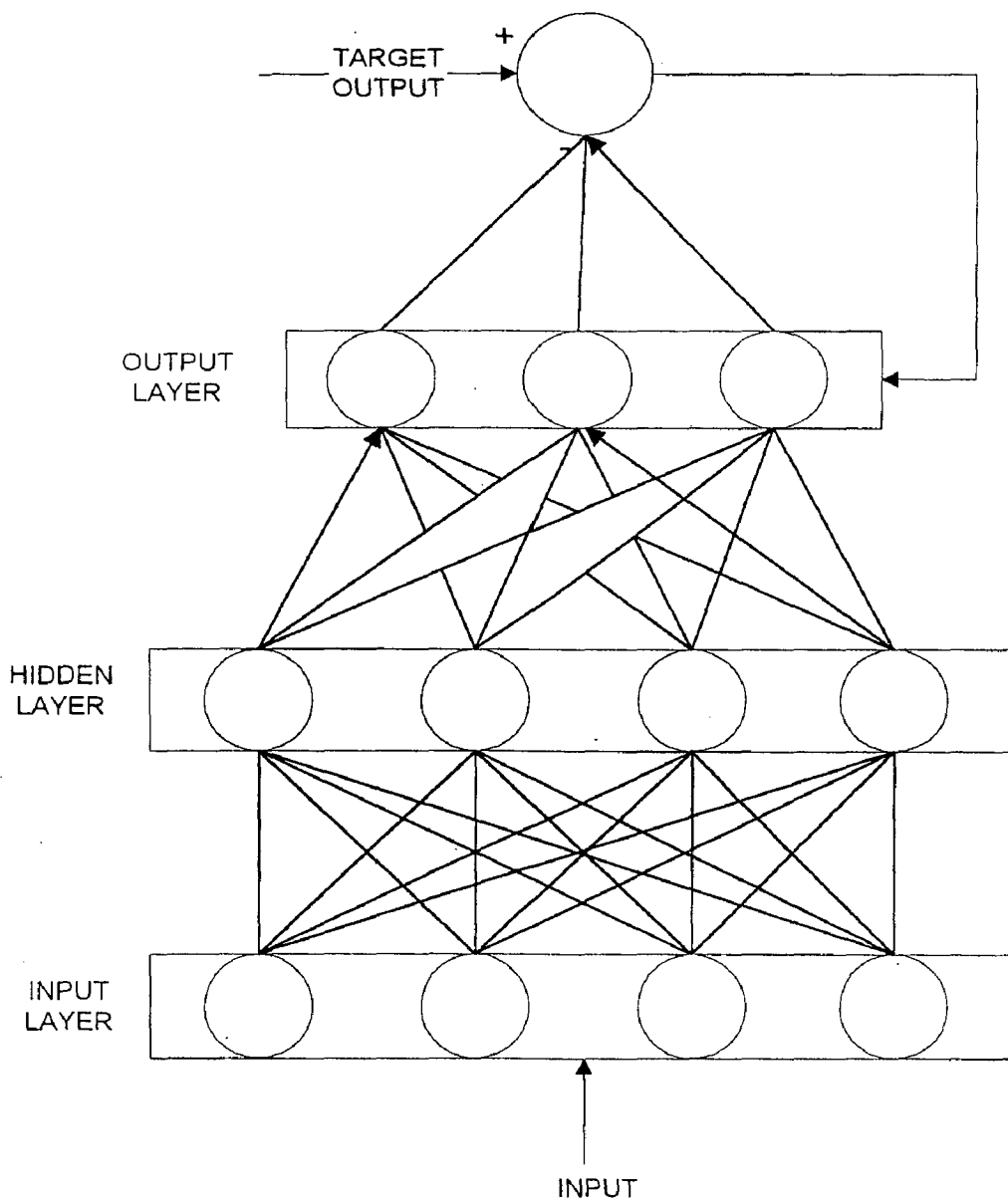


Fig. 2.3: Back Propagation network

IDENTIFICATION OF LINEAR SYSTEM

3.1 INTRODUCTION

The term identification of a system implies in fact building of its mathematical model. This consists of writing a set of mathematical equations along with a set of boundary conditions and limitations posed on a system itself. If the system to be identified is a linear dynamic system as shown in Fig. 3.1, then its mathematical model will be described by an ordinary differential equation (ODEs) of the type.

$$y(t)^{(n)} + a_{(n-1)}y(t)^{(n-1)} + \dots + a_1y(t)^1 + a_0y(t)^0 = u(t) \quad \dots(3.1)$$

Where,

$y(t)$ is system output.

$u(t)$ is system input and

a_i is system parameter, which is identified.

Dynamic system whose time-behaviour is described by an ordinary differential equations of the order n is a linear dynamic system of the same order n . It is called time-invariant systems if its parameters (i.e. the coefficients a_0, a_1, \dots, a_{n-1}) are constant, otherwise it is a time variant system. In the modern system theory it is usual to describe the system in the state space form :

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

$$Y(t) = C(t) X(t)$$

Where,

$X(t)$ is the state vector of order $(k, 1)$ of the system,

$A(t)$ is system matrix of order (k, k)

$b(t)$ is control vector of order (k, p) ,

$u(t)$ is the control units of order $(p, 1)$,

$Y(t)$ is the output vector of order $(1, m)$, and

$C(t)$ is observability or measurement vector of order $(m, 1)$.

Our objective is to develop a neural network that an model on ODE system precisely whose right hand side function $f(\dots)$ is unknown such that it can estimate system matrix $A(t)$ and control vector $b(t)$.

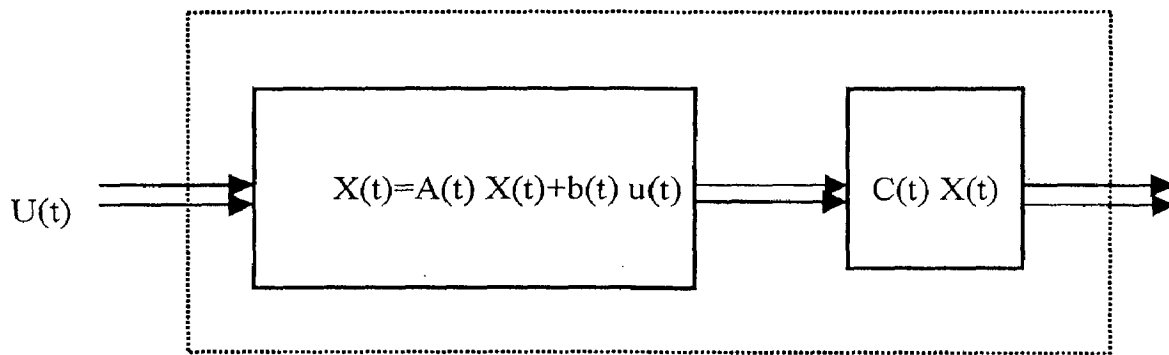


Fig. 3.1: State space system representation

3.2 PERFORMANCE INDEX

The analytical identification in which some criterion based on integrated history of the response of the entire system is used as a measure of performance. For example, either the minimisation of Mean-Square-Error (MSE) for a noisy system or the Integral-Square-Error (ISE) for a noise-free case can be used as a performance criterion. Use of classical methods of the calculus of variations is made to minimise the criterion and to obtain consequently the compensating network. The system that minimises the

performance index is then said to be the 'best' or 'optimal'. A number of such performance indices are used in practice, the most common being the ISE, given by

$$ISE = \int_0^{\infty} E^2(t) dt \quad \dots(3.2)$$

For higher order system, ISE is computed numerically. Since it is not practicable to integrate up to infinity, the limit infinity is replaced by T, which is chosen sufficiently large so that e (t) for t>T is negligible.

3.3 PROBLEM FORMULATION

Zadeh gave the following definition of the identification problem. "Identification is the determination, on the basis of input and output, of a system within a specified class of systems, to which the system under test is equivalent." Using Zadeh's formulation it is necessary to specify a class of system, a class of input signals, and the meaning of "equivalent", the system under test is called the process and the element of the system class is called models. Equivalence is often defined in terms of a criterion or a loss function that is a functional of the process output y and the model output y_m i.e.,

$$V = V(y, y_m)$$

Two model m₁ and m₂ are said to be equivalent if the value of the loss function is the same for both models i.e.

$$V(y, y_{m_1}) = V(y, y_{m_2})$$

There is a large freedom in the problem formulation, which is reflected in the literature of identification problems. The selection of the class models, the class of input signals and the criterion is largely influenced by the a priori knowledge of the process as well as by the purpose of the identification.

When equivalence is defined by means of a loss function the identification problem is simply an optimisation problem: find a model such that the loss function is as small as possible. In such a case it is natural to ask several questions:

Is the minimum achieved?

Is there a unique solution?

Is the uniqueness of the solution influenced by the choice of input signals?

If the solution is not unique, what is the character of the models that give the same value of the loss function and how should be restricted in order to ensure uniqueness?

The formulation of an identification problem as an optimisation problem also makes it clear that there are connections between identification theory and approximation theory.

Many direct problems may be formulated in the terms of systems ordinary differential equations of the form [6]:

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

Or
$$\dot{X}(t) = f(A(t), b(t), X(t), u(t)) \quad \dots(3.3)$$

Our objective is to develop a neural network that can model an ODEs system precisely whose right hand side function $f(\dots)$ is unknown, such that it can estimate system matrix $A(t)$ and control vector $b(t)$, where derivative of the states and states are given.

Let $A_e(t)$ and $b_e(t)$ be the estimated value of unknown parameters. Then estimated value of the derivative is given by

$$\dot{X}_e(t) = A_e(t)X(t) + b_{e(t)}u(t) \quad \dots(3.4)$$

or

$$\begin{bmatrix} \dot{x}_{e1}(t) \\ \dot{x}_{e2}(t) \\ \dots \\ \dot{x}_{ek}(t) \end{bmatrix} = \begin{bmatrix} a_{e11} & a_{e12} & \dots & a_{e1k} \\ a_{e21} & a_{e22} & \dots & a_{e2k} \\ \dots & \dots & \dots & \dots \\ a_{ek1} & a_{ek2} & \dots & a_{ekk} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \dots \\ x_k(t) \end{bmatrix} + \begin{bmatrix} b_{e11} & b_{e12} & \dots & b_{e1p} \\ b_{e21} & b_{e22} & \dots & b_{e2p} \\ \dots & \dots & \dots & \dots \\ b_{ek1} & b_{ek2} & \dots & b_{ekp} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ \dots \\ u_p(t) \end{bmatrix} \quad \dots(3.5)$$

Let

$$E(t) = X(t) - X_e(t) \quad \dots(3.6)$$

Where, $E(t)$ is error vector of an order $(k,1)$.

Differentiating eqn. (3.5) w. r. t t , and writing in matrix form, we have

$$\begin{bmatrix} \dot{e}_1(t) \\ \dot{e}_2(t) \\ \dots \\ \dot{e}_k(t) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dots \\ \dot{x}_k(t) \end{bmatrix} - \begin{bmatrix} \dot{x}_{e1}(t) \\ \dot{x}_{e2}(t) \\ \dots \\ \dot{x}_{ek}(t) \end{bmatrix} \quad \dots(3.7)$$

From eqns. (3.4) and (3.6), we have

$$\begin{bmatrix} \dot{e}_1(t) \\ \dot{e}_2(t) \\ \dots \\ \dot{e}_k(t) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dots \\ \dot{x}_k(t) \end{bmatrix} - \begin{bmatrix} a_{e11} & a_{e12} & \dots & a_{e1k} \\ a_{e21} & a_{e22} & \dots & a_{e2k} \\ \dots & \dots & \dots & \dots \\ a_{ek1} & a_{ek2} & \dots & a_{ekk} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \dots \\ x_k(t) \end{bmatrix} - \begin{bmatrix} b_{e11} & b_{e12} & \dots & b_{e1p} \\ b_{e21} & b_{e22} & \dots & b_{e2p} \\ \dots & \dots & \dots & \dots \\ b_{ek1} & b_{ek2} & \dots & b_{ekp} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ \dots \\ u_p(t) \end{bmatrix} \quad \dots(3.8)$$

Considering the i th component of error derivative

$$\dot{e}_i(t) = \dot{x}_i(t) - [a_{ei1} \quad a_{ei2} \quad \dots \quad a_{eik}] X(t) - [b_{ei1} \quad b_{ei2} \quad \dots \quad b_{eip}] u(t). \quad \dots(3.9)$$

or

$$\dot{e}_i(t) = \dot{x}_i(t) - \sum_{j=1}^k (a_{ej} x_j(t)) - \sum_{m=1}^p (b_{eim} u_m(t)) \quad \dots(3.10)$$

or

$$\dot{e}_i(t) = \dot{x}_i(t) - W_i^T Z \quad \dots(3.11)$$

Where,

$$Z^T = [x_1(t) \quad x_2(t) \quad \dots \quad x_k(t) \quad u_1(t) \quad u_2(t) \quad \dots \quad u_p(t)] \quad \dots(3.12)$$

$$W_i^T = [a_{ei1}(t) \quad a_{ei2}(t) \quad \dots \quad a_{eik}(t) \quad b_{ei1}(t) \quad b_{ei2}(t) \quad \dots \quad b_{eip}(t)] \quad \dots(3.13)$$

3.4 DYNAMIC SYSTEM PARAMETER ESTIMATION

In the section 3.3 the error function is calculated and given in eqn.(3.11). Performance index is defined as

$$ISE = E[e_i(t)] \quad , \text{ average value of error function.}$$

or

$$E[e_i(t)] = 1/2T \int_0^T (\dot{x}_i(t) - \dot{x}_{ei}(t))^2 dt \quad \dots(3.14)$$

$$= 1/2T \int_0^T (\dot{e}_i(t))^2 dt \quad \dots(3.15)$$

$$= 1/2T \int_0^T [\dot{e}_i(t)]^T [\dot{e}_i(t)] dt \quad \dots(3.16)$$

$$= \text{Trace} \{ 1/2T \int_0^T \dot{e}_i(t) [\dot{e}_i(t)]^T dt \} \quad \dots(3.17)$$

Substituting the value of $\dot{e}_i(t)$ from equation (3.10), we have

$$E[e_i(t)] = \text{Trace} \{ 1/2T \int_0^T [\dot{x}_i(t) - W_i^T Z][\dot{x}_i(t) - W_i^T Z]^T dt \} \quad \dots(3.18)$$

$$= \text{Trace} \left\{ \frac{1}{2T} \int_0^T (\dot{x}_i(t) \dot{x}_i^T(t) + W_i^T Z Z^T W_i - 2\dot{x}_i(t) Z^T W_i) dt \right\} \quad \dots(3.19)$$

The gradient of error function $E[e_i(t)]$ can be found by differentiating eqn. (3.19) w. r. t W_i .

$$\nabla_i = \partial E / \partial W_i \quad \dots(3.20)$$

$$= \frac{1}{2T} \int_0^T (2Z Z^T W_i - 2Z \dot{x}_i^T(t)) dt \quad \dots(3.21)$$

In discrete form the equation is expressed as

$$\nabla_i = 1/N \sum_N (Z Z^T W_i - Z \dot{x}_i^T) \quad \dots(3.22)$$

Where N is total no. of samples considered for the purpose of simulation. Setting gradient of error function equal to zero, i.e. satisfies the condition of optimisation.

$$\nabla_i = 0 \quad \dots(3.23)$$

Under this condition, the obtained value of W_i will be the optimum. The updated weight is given as

$$W_i(n+1) = W_i(n) - \lambda \nabla_i \quad \dots(3.24)$$

Where W_i are weight of the Nith network.

ANN IMPLIMENTATION FOR LINEAR SYSTEM IDENTIFICATION

Neural networks are rich in applications across large and growing number of discipline. Neural systems gained popularity over other methods as they are efficient in discovering similarities among large bodies of data and in synthesising distributed/fault tolerance models for non-linear, partly unknown, and noisy corrupted systems.

4.1 DEVELOPMENT OF ANN MODEL

A modular neural network has been developed to identify the system parameters. In this model, each module is a **single layer** feed forward neural network. Gradient descent algorithm along with adaptive learning rate λ , is used for training the neural network. The number of input nodes is equal to the number of state vector plus number of inputs variable. The number of output node is one. In modular network, even when the total number of parameters in all the modules exceeds the number of parameters in the non-modular network, the modular approach yields faster convergence and generalises equally well and sometimes better than a non-modular network. The modules of the modular neural network can be trained independently and in parallel. The block-diagram of the modular neural network is given in Fig. (4.1). The number of module in the modular neural network is equal to the number of the state vectors.

4.2.1 Architecture of the Module

As shown in Fig. 4.2, the module has $(k+p)$ input nodes and one output node. If the target is 0.9, outputs greater than 0.9 are clamped to 0.9, Similarly, outputs smaller

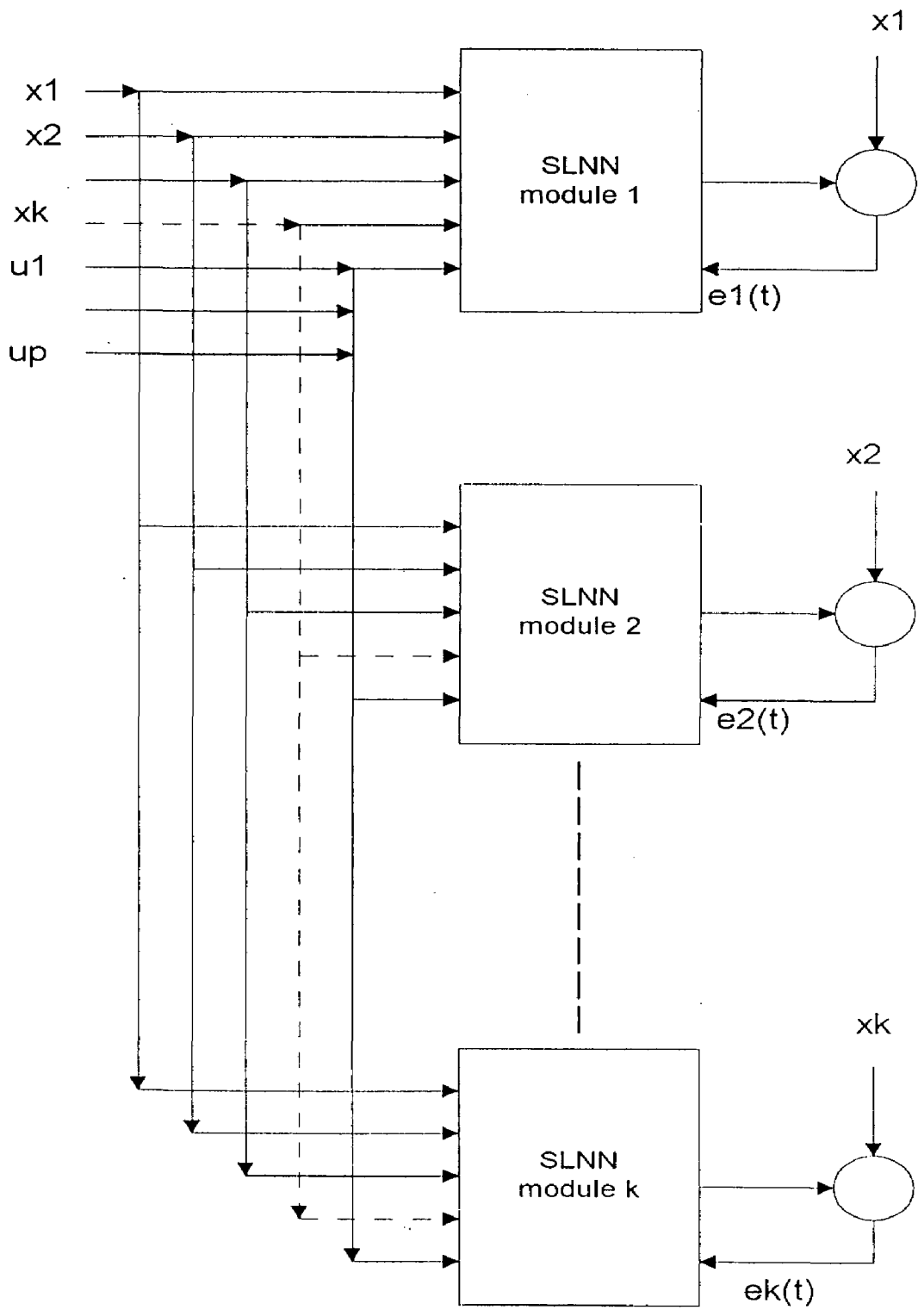


Fig 4.1: Modular structure of ANNs using BP algorithm

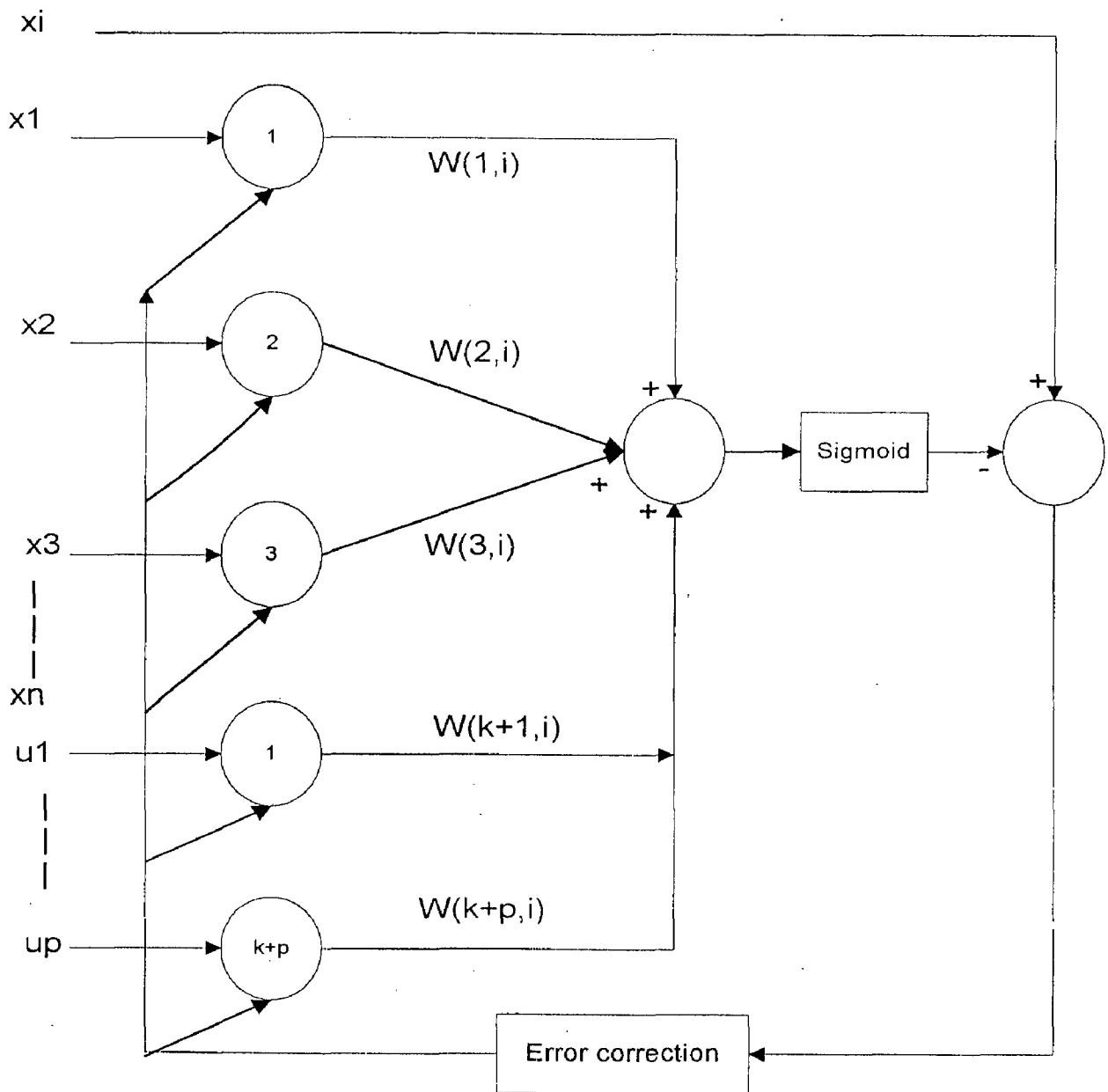


Fig. 4.2: A Single layer feed forward ANN module

than 0.1 are clamped to 0.1 to reduce likelihood of the network getting stuck in local minimum.

The weights on the connection from the i th-input node of the module to the j th output nodes are w_{ij} . The collection of all weights of j th module is denoted as $W_j = (w)$.

4.3 TRAINING SET FOR A MODULE

The training set T has IP number of patterns in wide range of variation.

$$T = \{ (x_j, t_j) ;$$

$j=1,2,\dots,IP\}$. where x_j is the input vector for j th patterns.

4.4 NORMALISATION

During training of a neural network, the higher valued input variables may tend to suppress the influence of smaller once. To overcome this problem, the neural networks are trained with normalised input data, leaving it to the network to learn weight associated without he connections emanating from these inputs. The raw data are scaled in the range (0.1-0.9) for use by neural networks to minimise the effects of magnitude between inputs. In case of output variables, if it assumes value close to unity or zero, it cause difficulty in training as the value unity or zero are practically never realised by the activation or threshold function. A way to overcome this difficulty is to normalise the variable (x) to keep its value between some suitable range (0.1 and 0.9). In present application, each input or output parameter x_n is normalised as before being applied to the neural network according to eqn.

$$x_n = (0.8 * (x - x_{min}) / (x_{max} - x_{min})) + 0.1 \quad \dots(4.1)$$

Where x_{max} and x_{min} are the maximum and minimum values of data parameter x . The input data are normalised between value 0.9 and 0.1. Similarly, output data (t) are

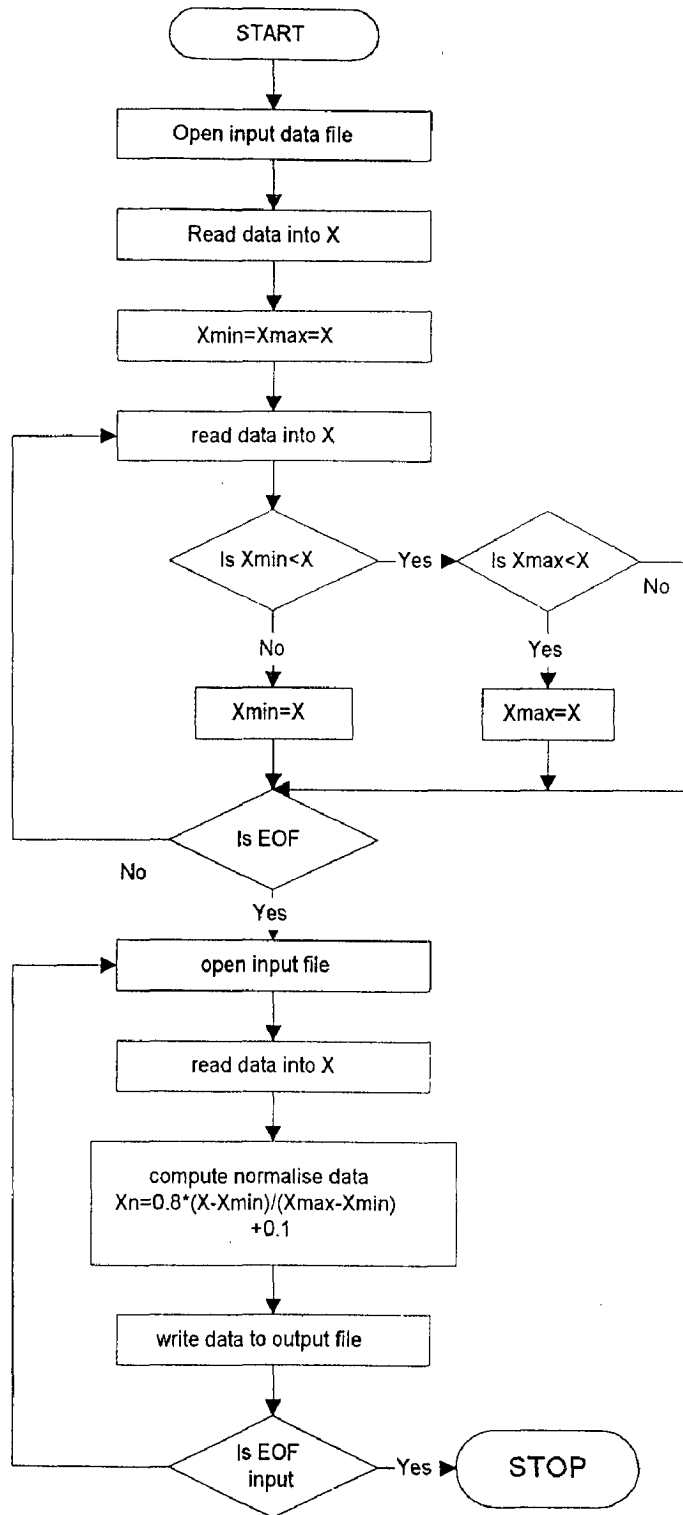


Fig. 4.3 Flow chart for normalising the data

can depend only on the local electrochemical environment of a given synapse. Obviously a single synapse does not know anything about the global tasks the organism is trying to learn.

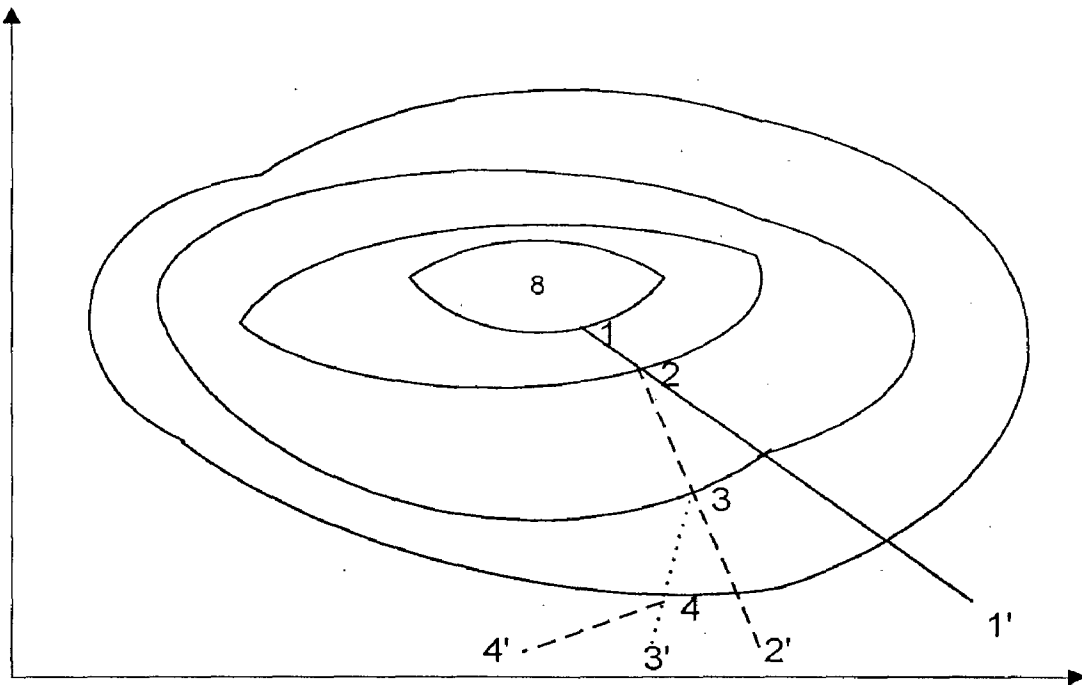


Fig 4.4 Steepest descent directions

Synaptic modifications and global tasks belong to two very distinct levels of brain hierarchy. Therefore, the fundamental question of learning is what are the principles according to which local synaptic changes are organised in order to yield global learning of complex behaviours for the organism? This puzzling question remains largely unanswered. There are many basic ideas in the theoretical literature, which have shed some light on this question.

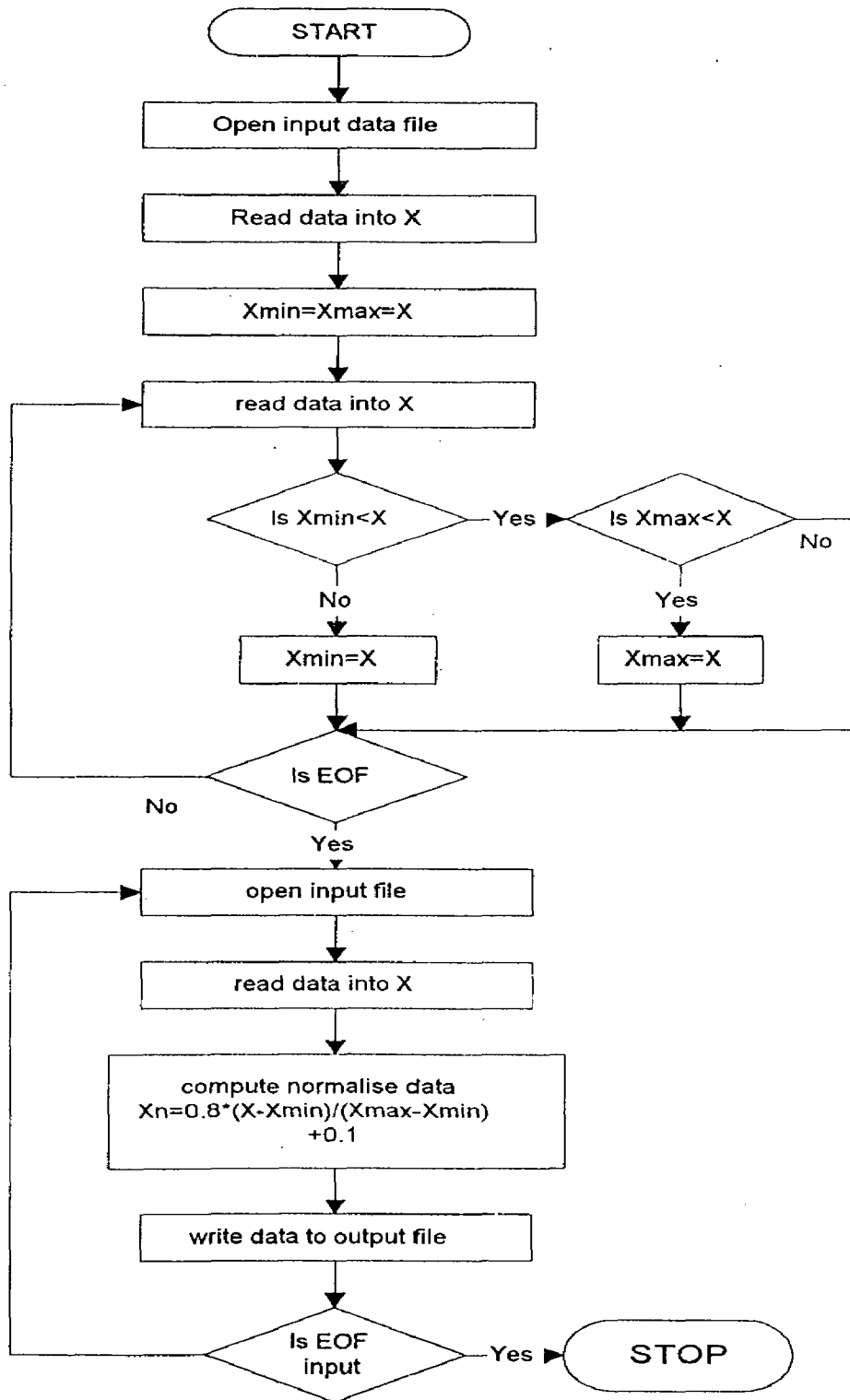


Fig. 4.3 Flow chart for normalising the data

normalised for each pattern between 0.9 and 0.1. Its flowchart implementation is shown in fig (4.3).

4.5 GRADIENT OF A FUNCTION

The partial derivatives of function $f(x)$, with respect to each of the n variables are collectively called the gradient of the function and is denoted by :

$$\nabla f = \left\{ \begin{array}{l} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \dots\dots \\ \dots\dots \\ \frac{\partial f}{\partial x_n} \end{array} \right\}$$

The gradient is a n -component vector and it has a very important property. If we move along the negative of gradient direction from any point in n -dimensional space, the function value decreases at the fastest rate, which has been proved in appendix A. Hence this gradient direction is called the direction of steepest descent. Unfortunately, the direction of steepest descent is a local property and not a global one. 4 line along the directions 11, 22, 33, and 44 respectively. Thus the function value decreases at the fastest rate in direction 11' at point 1, but not at point 2. Similarly, the function value at point 3(4). In other words, the direction of steepest descent generally varies from point, and if we make infinitely small moves along the direction of steepest descent, the path will be a curved line like the curve 1 2 3 4 as shown in the Fig.(4.4)

Since the gradient vector represents the direction of steepest ascent, the negative of the gradient vector denotes the direction of steepest descent.

4.5.1 Gradient Descent Learning

Learning in biological systems is widely believed to result from progressive modifications occurring at the level of synapses connecting neurones. Such modifications

can depend only on the local electrochemical environment of a given synapse. Obviously a single synapse does not know anything about the global tasks the organism is trying to learn.

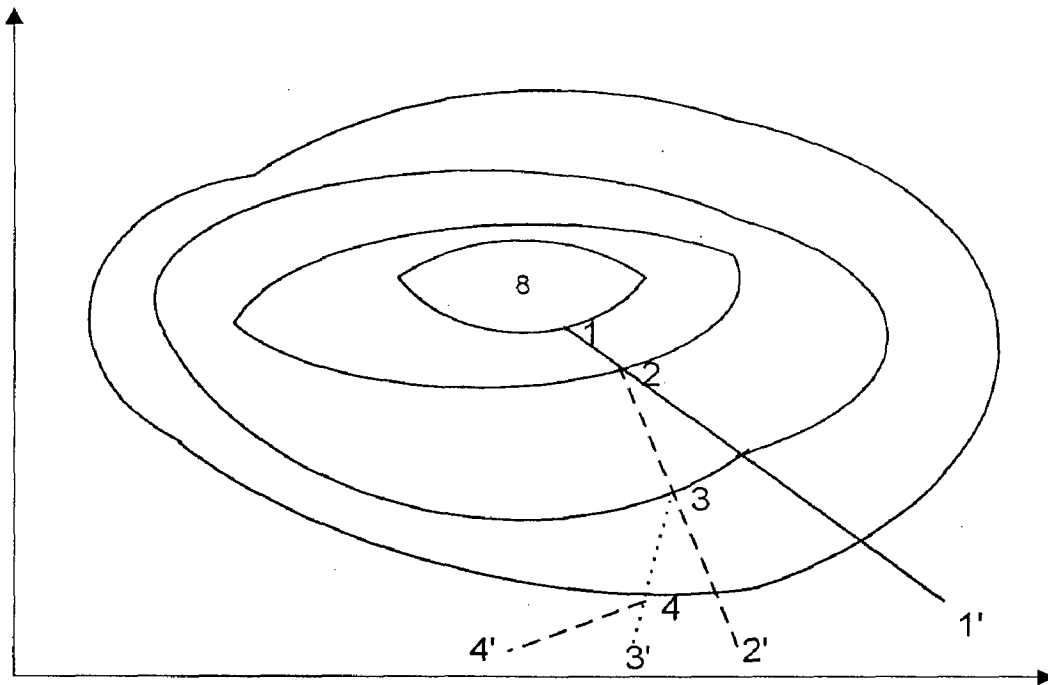


Fig 4.4 Steepest descent directions

Synaptic modifications and global tasks belong to two very distinct levels of brain hierarchy. Therefore, the fundamental question of learning is what are the principles according to which local synaptic changes are organised in order to yield global learning of complex behaviours for the organism? This puzzling question remains largely unanswered. There are many basic ideas in the theoretical literature, which have shed some light on this question.

Gradient descent is the one idea in the theory of learning, which precisely attempts to offer a simple guiding principle for the overall organisation of synaptic changes across networks.

Considering the learning problem for N-dimensional dynamical system of the form

$$\dot{x} = f(x, u, w)$$

with initial condition $u(t_0)$. In this relation, the k-dimensional vector x represents the state variable of the system, w an array of adjustable parameters, and u a vector of external inputs, the vector x is used to represent particular target values for some of the x variables.

The learning problem consists of adjusting the parameter w so the trajectories of f have certain specified properties. Gradient descent learning requires that, at any time, the performance of the dynamical system be assessable through a certain error function E that measures the discrepancy between the trajectories of the dynamical system and the desired behaviour.

5.6 ALGORITHM

Stepwise procedure for identify the dynamic system is as follows

Step 1: Pattern selection:

p training patterns are given: $\{t_1, d_1, t_2, d_2, \dots, t_p, d_p\}$

Where t_i is the input and d_i is output of the i th pattern.

Step 2. Normalisation:

Normalise the input data, as higher valued input variables tend to surpass the influence of smaller ones. Scale each input parameter Z in the range of 0.1-0.9 for

use by neural networks to minimise the effects of magnitude between inputs, according to eqn.(4.1).

Step 3: Initialisation:

The learning parameter λ , number of states, number of inputs, maximum number of cycles to train the networks and error tolerance is chosen. Set all weight and node thresholds to small random numbers.

Step 4: Reset:

Set $ERROR=0$, and

$GRAD_ERROR=0$.

Step 5: Select one pattern from the patterns.

$z_i \leftarrow t_i$

Step 6: Calculation of Activation:

The activation level of an output unit is determined by the instance presented to the network. The activation level Y_j of the output unit is determined by eqn.

$$Y_j = \sum_{i=0}^n z_i w_{ji} \quad \dots(4.2)$$

Step 7: Error function:

$$E_j = 1/2(d_j - Y_j)^2 \quad \dots(4.3)$$

Step 8: Error gradient:

Calculate the gradient of the error function E_j w.r. to weights,

$$\nabla E_j = \partial E / \partial w = Y_j(1 - Y_j)(d_j - Y_j) \quad \dots(4.4)$$

Step 9: Sum of error and error gradient:

$$(ERROR)_j = (ERROR)_j + E_j. \quad \dots(4.5)$$

$$(\text{GRAD_ERROR})_j = (\text{GRAD_ERROR})_j + \nabla E_j \quad \dots(4.6)$$

Step 10: Repeat steps 5 to 9 for all patterns.

Step 11: Mean Square of error.

$$E_j = \text{square root of } ((\text{ERROR})_j/p), \text{ and} \quad \dots(4.7)$$

Average error gradient

$$\delta_j = ((\text{GRAD_ERROR})_j/p). \quad \dots(4.8)$$

Step 12: Search direction:

Calculate the direction of the descent vector S_j that points in a downhill direction for using equation;

$$S_j = -\delta_j / \sum_{i=1}^n \delta_i^2 \quad \dots(4.9)$$

Step 13: Weight adjustment:

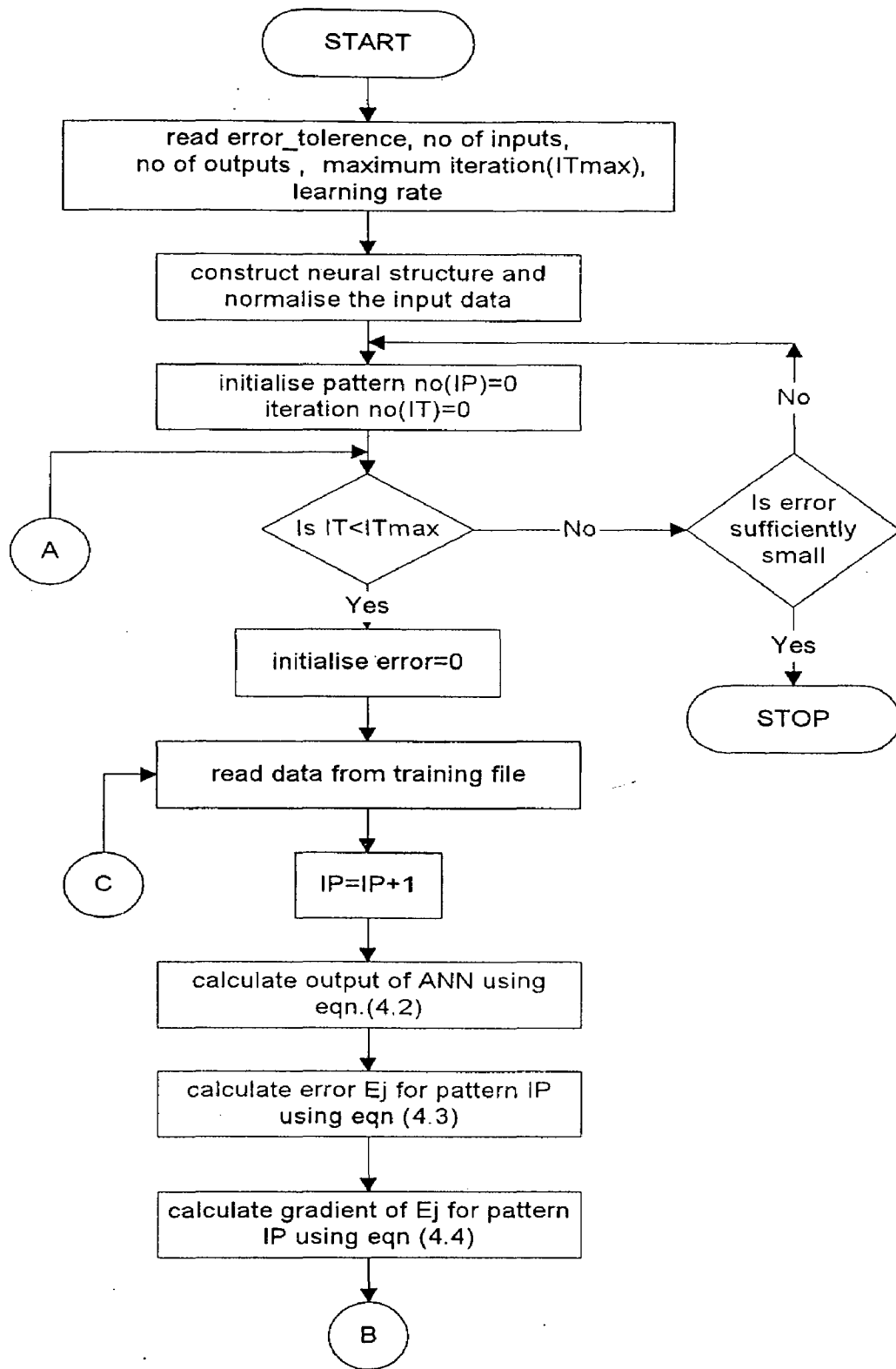
Calculate the new weights by equation;

$$W_{ji}(n+1) = W_{ji}(n) + \lambda \delta_j S_j \quad \dots(4.10)$$

Where $W_{ji}(n)$ is the weight from unit i to j at time n .

Step 14: Error Criteria

Repeat steps 4 to 13 until convergence in terms of the selected error criteria satisfied.



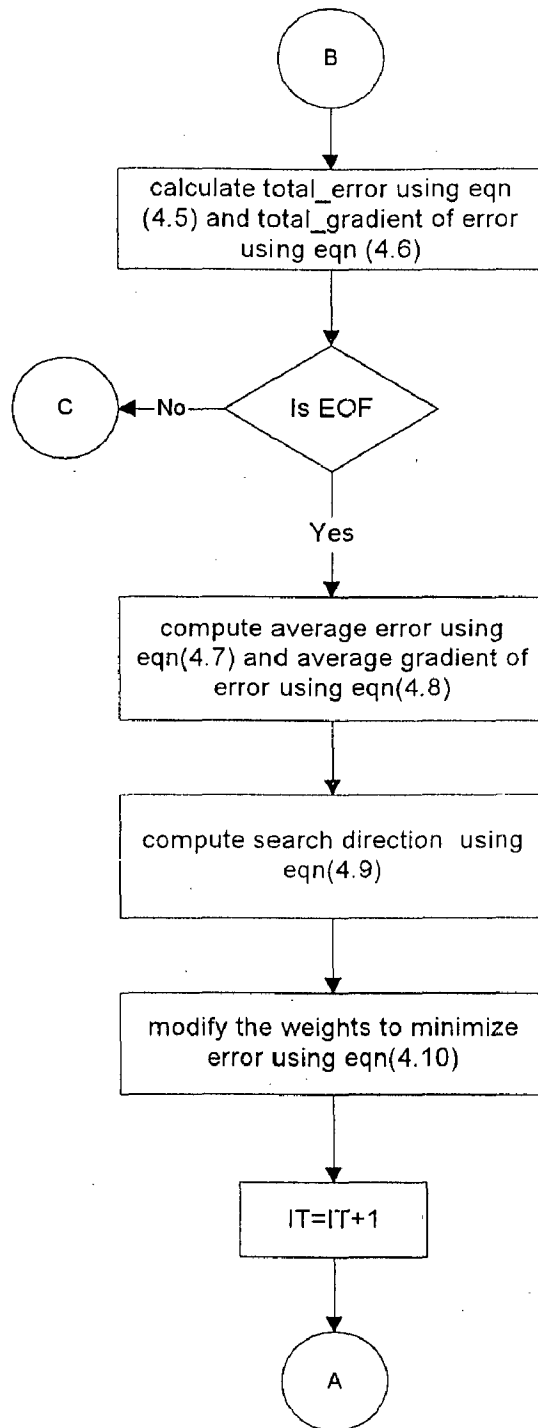


Fig. 4.5 Flow chart of gradient descent algorithm

RESULTS AND DISCUSSION

5.1 INTRODUCTION

The method discussed in the previous chapter for identification of dynamical systems has to be applied to identify the parameters of the problems. The chemical and biological problems have been considered from practical applications. In this section, the examples and their response behaviour are described. An important by-product resulted during the work was the ability to solve system of non-linear differential equation.

5.2 TWO INPUT SYSTEM

The two input systems are commonly considered in the control-plant problem.

5.2.1 Problem Formulation

The example considered has no physical significant, it is purely numerical and the system model is [4] by

$$\dot{x}_1(t) = w_{11}x_1 + w_{12}x_2 + b_{11}u_1 + b_{12}u_2$$

$$\dot{x}_2(t) = w_{21}x_1 + w_{22}x_2 + b_{21}u_1 + b_{22}u_2$$

Where,

x_1, x_2 are system states.

u_1, u_2 are the inputs to the system.

$w_{11}, w_{12}, w_{21},$ and w_{22} are elements of system matrix.

$b_{11}, b_{12}, b_{21},$ and b_{22} are elements of control matrix.

5.2.1 Experimental data:

The input u_1 is a step unit, and u_2 is a ramp of slope 0.2 i.e., $u_1=1.0$, and $u_2=0.2*t$.

Where time t is an independent variable. Initially, the constants are assumed. The initial values of constants are follows:

$$\begin{array}{cccc} w_{11}=0.0 & w_{12}=1.0 & b_{11}=0.0 & b_{12}=1.0 \\ w_{21}=-0.14 & w_{22}=-1.0 & b_{21}=1.0 & b_{22}=0.70 \end{array}$$

System eqns. are simulated for 5 seconds on digital computer to obtain the values of states and derivative of states. In this way 50 patterns are generated for the learning of neural network, which has been given in **table 5.1**.

5.2.2 Results

Gradient descent algorithm (modified-back-propagation), which has been discussed in the chapter 4, is used to identify the system parameters. The number of input and output to the ANN is 4 and 2 respectively i.e., the structure of the network is (4-2). The value of learning rate λ has been taken as 0.2. The trajectories of the various parameters as obtained by the neural network training are shown in fig.5.1 and fig. 5.2.

From the graphs, the values of the constants are

$$\begin{array}{cccc} w_{11}=0.0140 & w_{12}=1.014 & b_{11}=0.0019 & b_{12}=0.937 \\ w_{21}=-0.142 & w_{22}=-0.980 & b_{21}=0.9913 & b_{22}=0.687 \end{array}$$

These values are very close to the initial values, which we have assumed.

TABLE 5.1

x_1	x_2	u_1	u_2	\dot{x}_1	\dot{x}_2
0.0114	0.1324	1.0000	0.0280	0.1604	0.8937
0.0131	0.1413	1.0000	0.0300	0.1713	0.8866
0.0148	0.1502	1.0000	0.0320	0.1822	0.8795
0.0167	0.1589	1.0000	0.0340	0.1929	0.8724
0.0187	0.1676	1.0000	0.0360	0.2036	0.8654
0.0208	0.1762	1.0000	0.0380	0.2142	0.8584
0.0230	0.1848	1.0000	0.0400	0.2248	0.8515
0.0253	0.1933	1.0000	0.0420	0.2353	0.8447
0.0277	0.2017	1.0000	0.0440	0.2457	0.8379
0.0302	0.2100	1.0000	0.0460	0.2560	0.8312
0.0328	0.2183	1.0000	0.0480	0.2663	0.8245
0.0355	0.2265	1.0000	0.0500	0.2765	0.8178
0.0383	0.2347	1.0000	0.0520	0.2867	0.8112
0.0412	0.2427	1.0000	0.0540	0.2967	0.8047
0.0443	0.2507	1.0000	0.0560	0.3067	0.7982
0.0474	0.2587	1.0000	0.0580	0.3167	0.7917
0.0506	0.2666	1.0000	0.0600	0.3266	0.7853
0.0539	0.2744	1.0000	0.0620	0.3364	0.7790
0.0573	0.2822	1.0000	0.0640	0.3462	0.7727
0.0608	0.2899	1.0000	0.0660	0.3559	0.7664
0.0644	0.2975	1.0000	0.0680	0.3655	0.7602
0.0681	0.3051	1.0000	0.0700	0.3751	0.7540
0.0719	0.3126	1.0000	0.0720	0.3846	0.7479
0.0758	0.3200	1.0000	0.0740	0.3940	0.7418
0.0798	0.3274	1.0000	0.0760	0.4034	0.7358
0.0839	0.3347	1.0000	0.0780	0.4127	0.7298

Continue

x_1	x_2	u_1	u_2	\dot{x}_1	\dot{x}_2
0.0881	0.3420	1.0000	0.0800	0.4220	0.7239
0.0923	0.3492	1.0000	0.0820	0.4312	0.7180
0.0967	0.3564	1.0000	0.0840	0.4404	0.7122
0.1012	0.3635	1.0000	0.0860	0.4495	0.7064
0.1057	0.3705	1.0000	0.0880	0.4585	0.7006
0.1103	0.3775	1.0000	0.0900	0.4675	0.6949
0.1150	0.3844	1.0000	0.0920	0.4764	0.6892
0.1198	0.3913	1.0000	0.0940	0.4853	0.6836
0.1247	0.3981	1.0000	0.0960	0.4941	0.6780
0.1297	0.4048	1.0000	0.0980	0.5028	0.6724
0.1348	0.4115	1.0000	0.1000	0.5115	0.6669
0.1400	0.4182	1.0000	0.1020	0.5202	0.6615
0.1452	0.4247	1.0000	0.1040	0.5287	0.6560
0.1505	0.4313	1.0000	0.1060	0.5373	0.6506
0.1559	0.4378	1.0000	0.1080	0.5458	0.6453
0.1614	0.4442	1.0000	0.1100	0.5542	0.6400
0.1670	0.4506	1.0000	0.1120	0.5626	0.6347
0.1727	0.4569	1.0000	0.1140	0.5709	0.6295
0.1784	0.4631	1.0000	0.1160	0.5791	0.6243
0.1843	0.4694	1.0000	0.1180	0.5874	0.6192
0.1902	0.4755	1.0000	0.1200	0.5955	0.6140
0.1962	0.4816	1.0000	0.1220	0.6036	0.6090
0.2023	0.4877	1.0000	0.1240	0.6117	0.6039
0.2084	0.4937	1.0000	0.1260	0.6197	0.5989

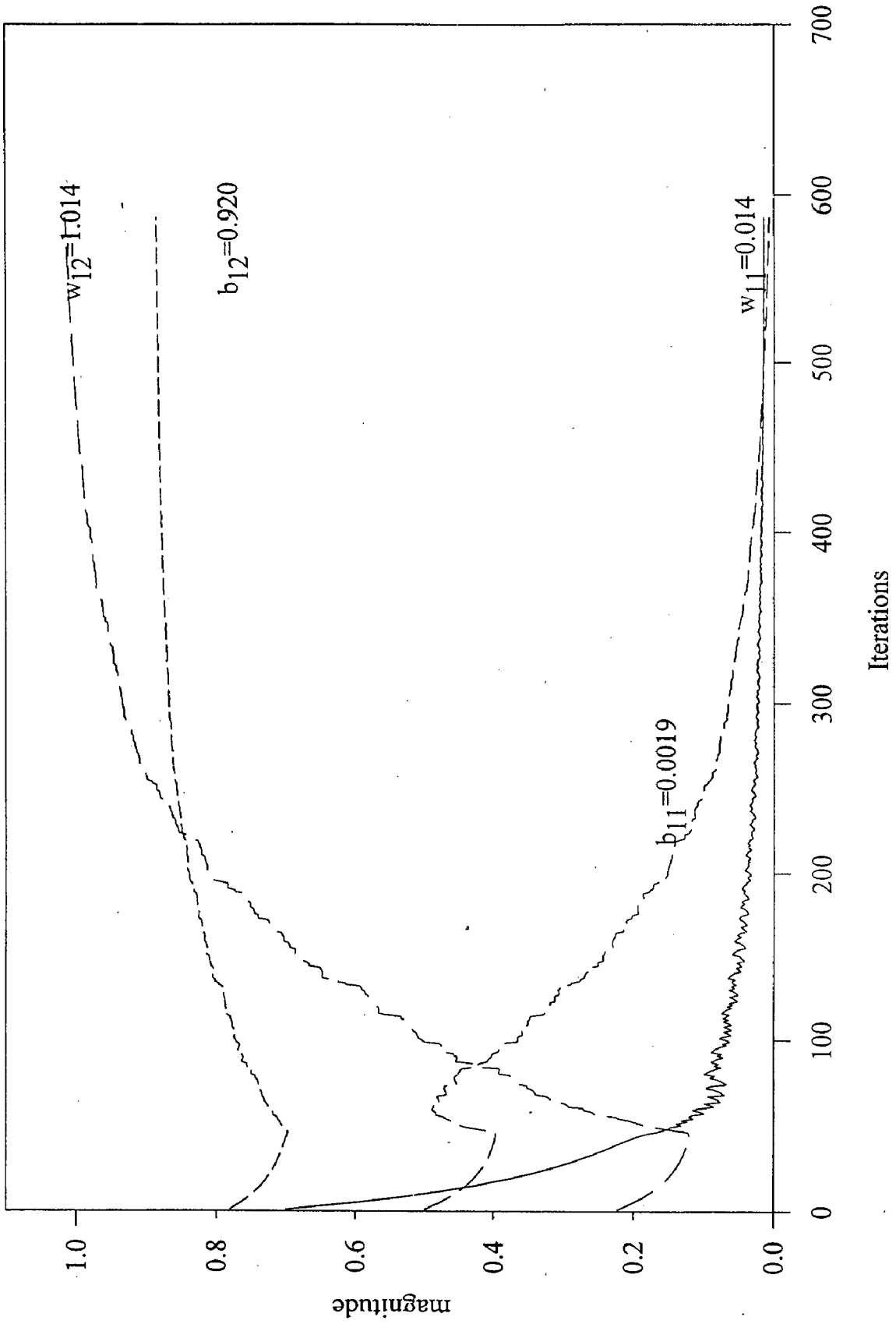


Fig. 5.1 Trajectory of w_{11} , w_{12} , b_{11} , and b_{12}

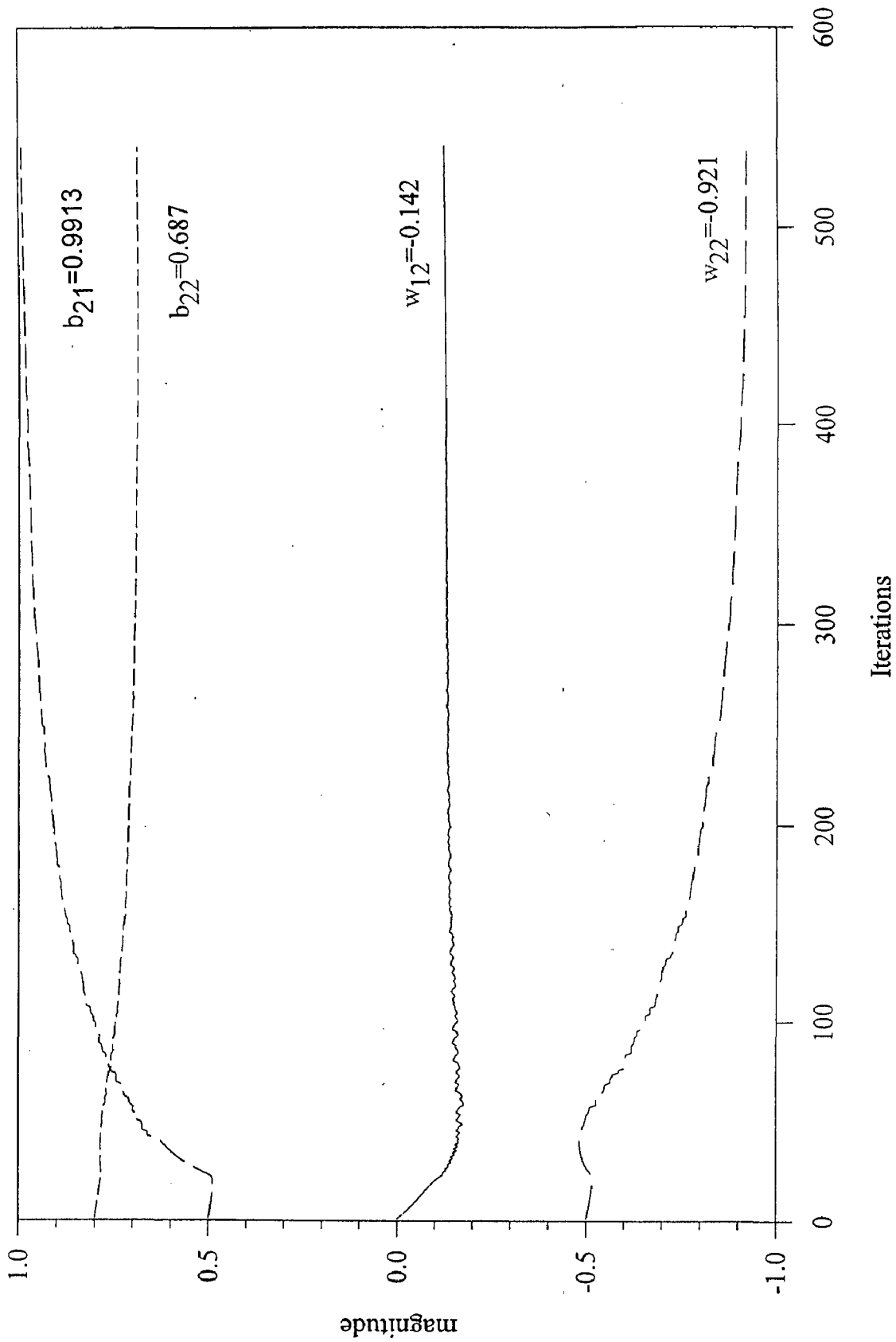


Fig. 5.2 Trajectory of $w_{21}, w_{22}, b_{21}, b_{22}$

5.3 BLOOD GLUCOSE REGULATION

The physiology of the blood glucose regulation process is extremely complex. As the model of the process increase in complexity, the estimation of the parameters becomes more difficult

The blood glucose concentration in normal mammals is finely regulated. The administration of glucose, orally or in vein, results in a brisk, precipitous rise in the plasma insulin concentration due to a direct effect of the rising blood sugar upon the beta cells of the pancreas. The insulin, in turn, accelerates the rate of disappearance of glucose from the plasma compartment and the blood sugar quickly returns to a normal value of 80-100 mg/100 ml.

In certain metabolic disease, e.g., juvenile diabetes mellitus, the responsiveness of the beta cells of the pancreas may be severely diminished, resulting in a failure of the blood glucose to return quickly to its normal value. In an alternative form of the disease, i.e., maturity onset diabetes, the pancreatic sensitivity to glucose may be normal, or supernormal, but the sensitivity of glucose uptake by peripheral tissues to the elevated insulin concentration may be obtained. The altered time course of the blood glucose concentration following a glucose load may be similar, but traditional clinical tests (such as the glucose tolerance test) lack the ability to discriminate effectively between various forms of diabetes.

5.3.1 Problem formulation

The system model is the linear two-compartment model [2] introduced by Bolie and is given by

$$\dot{x}_1(t) = w_{11}x_1 + w_{12}x_2$$

$$\dot{x}_2(t) = w_{21}x_1 + w_{22}x_2$$

Where, x_1 is the deviation of the extracellular insulin concentration from the mean, mU/ml, and x_2 is the deviation of the extracellular glucose concentration from the mean, mg/100 ml. The parameter w_{12} is most interesting because it determines the rate of insulin production. For diabetic dogs it is expected that this parameter will be considerably different from that for the average normal dog.

5.3.2 Experimental data

A glucose tolerance test was performed on two different conscious, intact dogs. 9gm of glucose were injected intravenously into a 66-lb dog over a time interval of 1-min. Measurements of the plasma glucose and insulin concentrations were made every 2.5-min for the first 20 min and every 5 min thereafter for an additional 70 min. The first measurements were made 2.5 min after the start of the glucose injection. The measurement deviation from the mean levels has been given in Table 5.2 The insulin response appears to be somewhat oscillatory for approximately the first 15 min. This is to be expected since it is well known that a step input of glucose causes a biphasic insulin response.

Glucose was measured by the glucose oxidase technique using a Beckman glucose analyser. Insulin was determined by radiomunoassay using the dextran charcoal separation technique.

5.2.3 Results

Gradient descent algorithm (modified back algorithm) has been discussed in chapter 4, is used to identify the system parameters. The number of input and output to

the ANN is 2 and 2 respectively i.e., the structure of the network is (2-2). Total number of patterns is 24. The value of learning rate λ has been varied from 0.1 to 0.7 with an increment of 0.2. The trajectory of various parameters is shown in figs 5.3, 5.4, 5.5, and 5.6. By viewing the graphs, it is analysed that convergence is achieved for all value of λ . However, the number of iterations has increased with the decrease in value of learning rate λ , which is expected.

Learning rate	Cycles	w ₁₁	w ₁₂	w ₂₁	w ₂₂
0.1	70	0.1852	0.3417	0.0349	0.0263
0.3	36	0.1852	0.3417	0.0263	0.0349
0.5	21	0.1851	0.3420	0.0263	0.0349
0.7	19	0.1851	0.3419	0.0263	0.0349

Table 5.2

Insulin (x1)	Glucose(x2)	\dot{x}_1	\dot{x}_2
34.0000	143.0000	55.1960	4.9475
166.0000	118.0000	71.0660	8.8968
123.0000	100.0000	56.9550	6.9227
149.0000	75.0000	53.2150	7.1726
89.0000	72.0000	41.0890	4.9997
202.0000	61.0000	58.2320	8.6541
170.0000	46.0000	47.1820	7.1428
83.0000	32.0000	26.2990	3.7383
59.0000	22.5000	18.6100	2.6509
35.0000	13.0000	10.9210	1.5634
26.0000	6.0000	6.8620	1.0652
17.0000	-1.0000	2.8030	0.5670
7.0000	-1.5000	0.7820	0.2049
0.0000	-11.0000	-3.7620	-0.2893
10.0000	-7.0000	-0.5440	0.1649
0.0000	-7.0000	2.3940	-0.1841
10.0000	0.0000	1.8500	0.3490
-3.0000	1.0000	-0.2130	-0.0784
-3.0000	0.0000	-0.5550	-0.1047
0.0000	5.0000	1.7100	0.1315
12.0000	6.0000	4.2720	0.5766
0.0000	1.0000	0.3420	0.0263
2.0000	1.0000	0.7120	0.0961

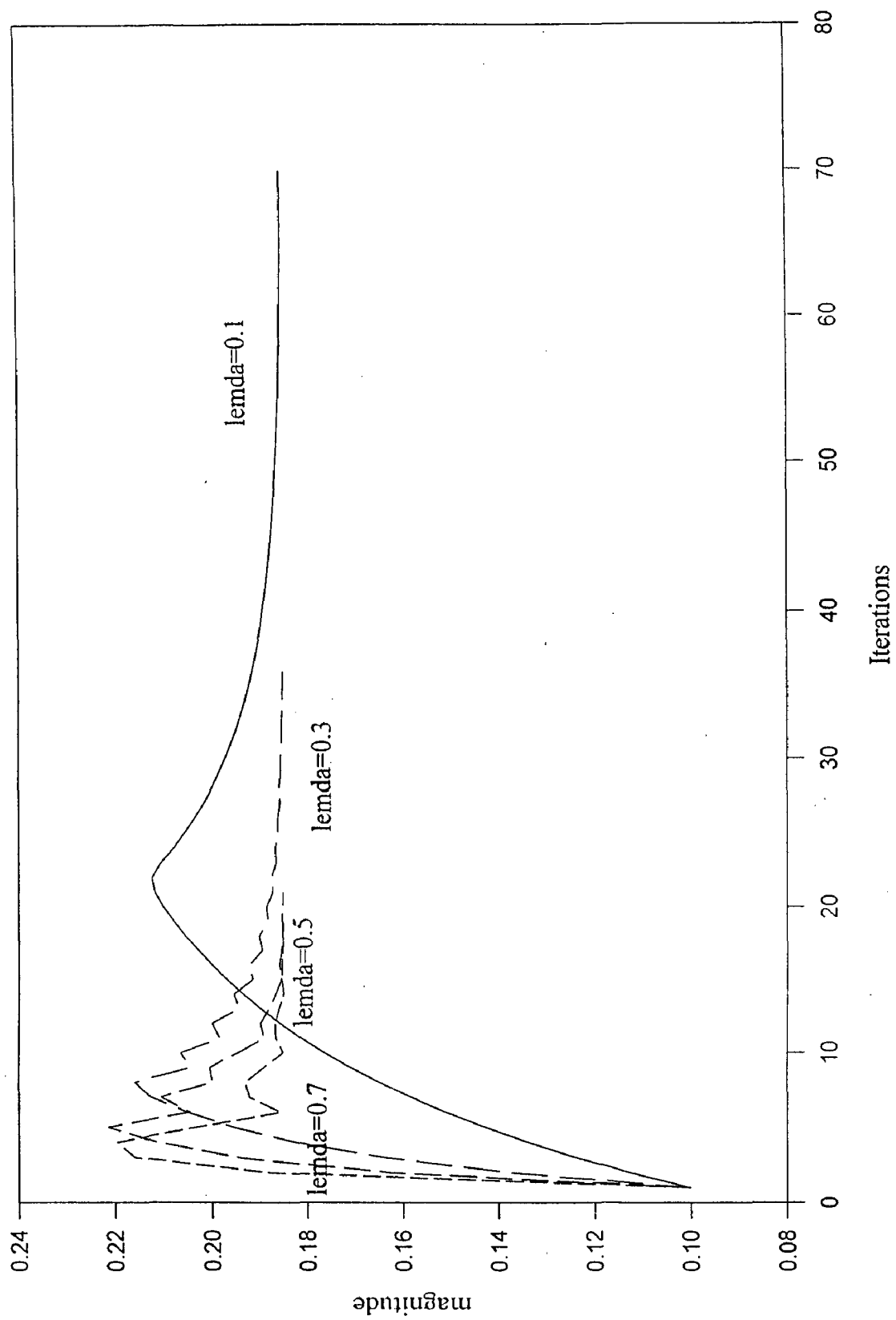


Fig. 5.3 Trajectories of w_{11}

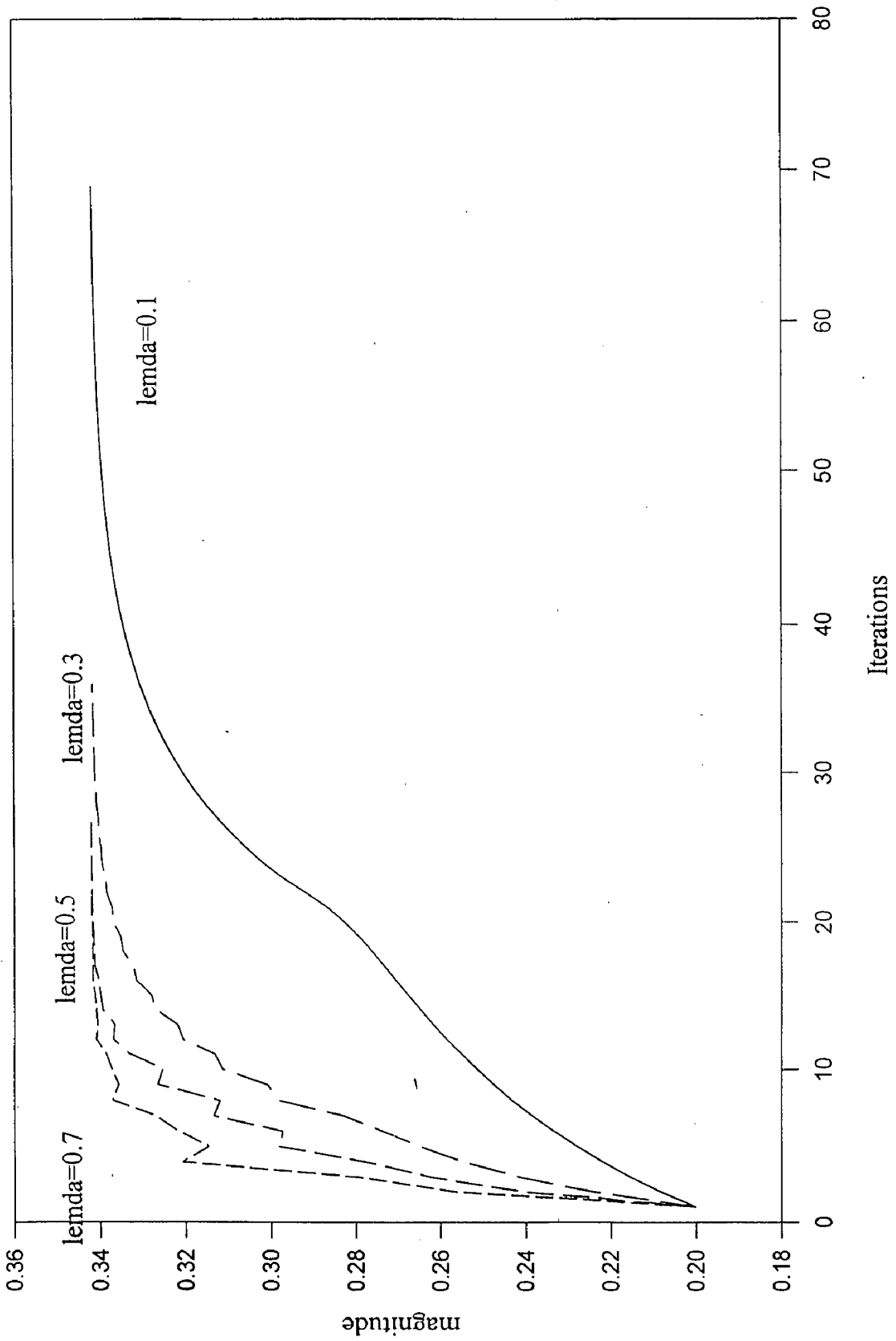


Fig. 5.4 Trajectories of w_{12}



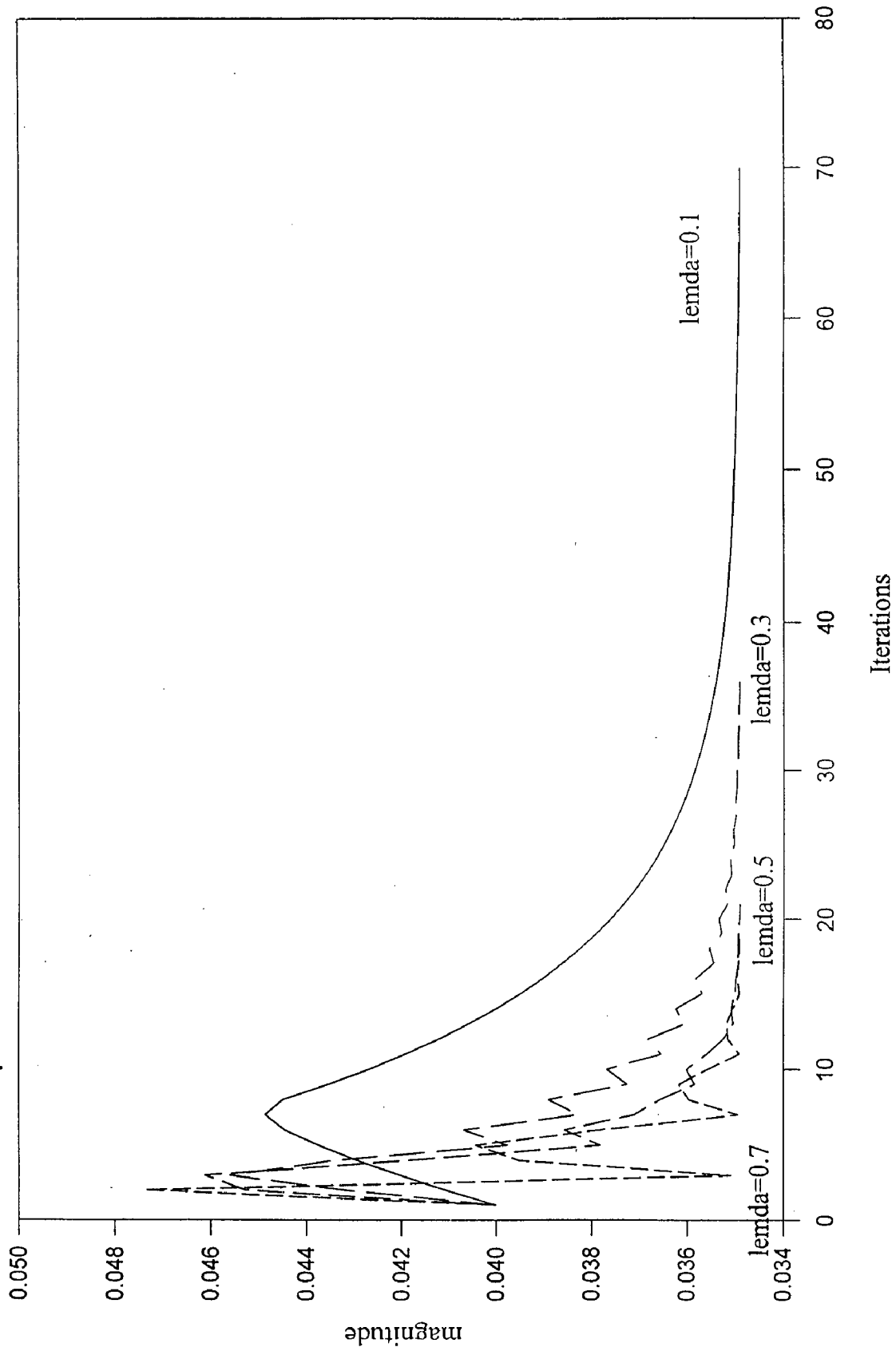


Fig. 5.5 Trajectories of w_{21}

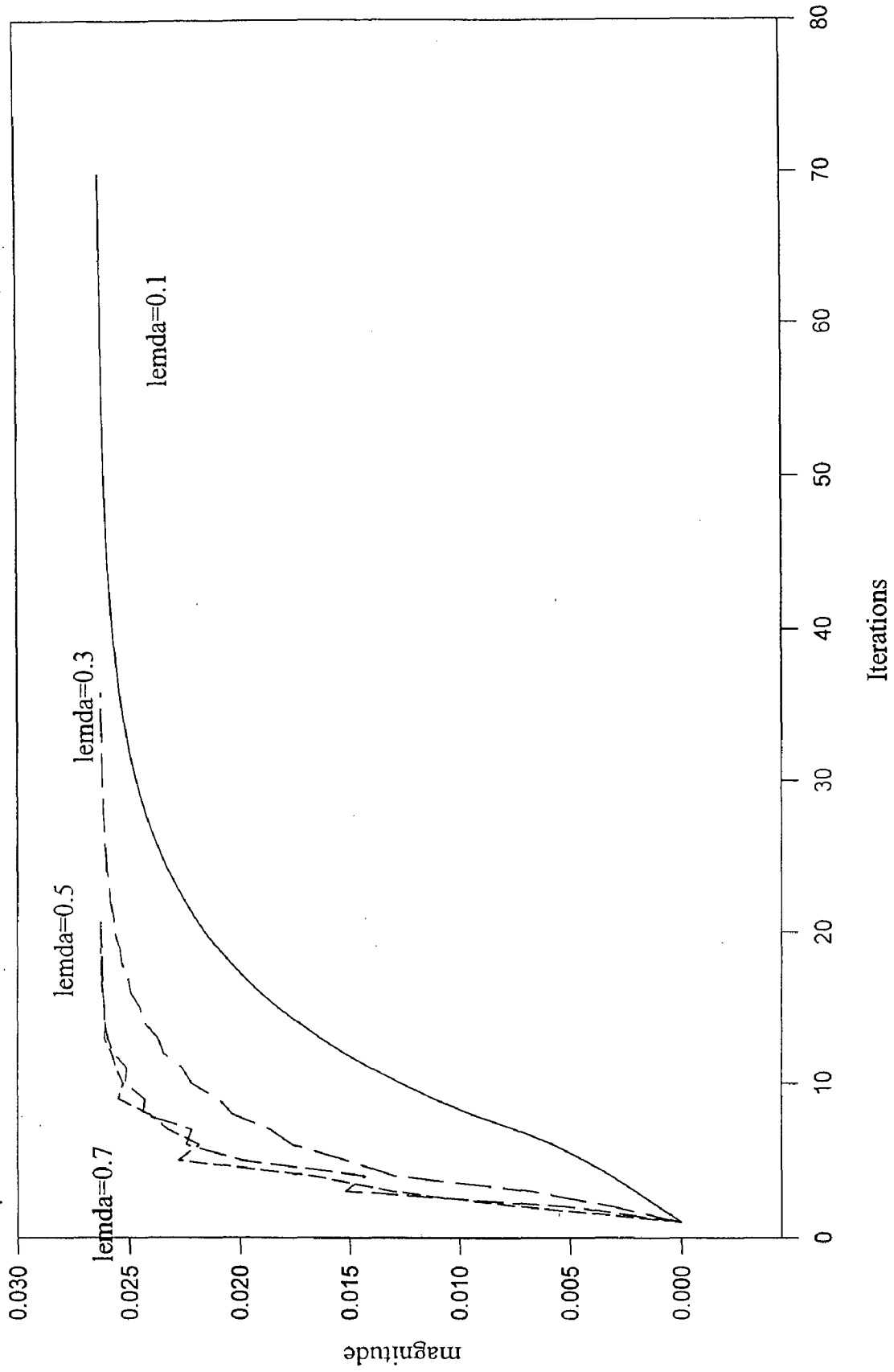
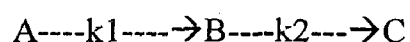


Fig. 5.6 Trajectories of w_{22}

5.4 BATCH REACTOR

Batch processes offer the most interesting and challenging problems in chemical systems modelling and control because of their inherent dynamic nature [1]. Although most large-scale chemical processes have been operated in continuous fashion, but many batch processes are still used in smaller volume production. The wide use of digital process control computers has permitted automation and optimisation of batch processes and made them more efficient and less labour intensive.

The batch reactor is sketched in fig 5.7. Reactant is charged into the vessel. Steam is fed into the jacket to bring the reaction mass up to a desired temperature then cooling water is added to the jacket to remove the exothermic heat of reaction. First-order consecutive reactions take place in the reactor as time proceeds



Let the component B is the desired product. If we let the reaction go on too long, too much of B will react to form undesired C; that is the yield is low. If we stop the reaction too early, too little A will have reacted; i.e., the conversion and yield will be low. Therefore, there is an optimum batch time when reaction is stopped. For this the control of batch reactor is necessary.

If the temperature dependence of the specific reaction rates k_1 and k_2 are same, then reaction run at the highest possible temperature to minimise the batch time.

If k_1 is more temperature-dependent than k_2 , then reaction run at the highest temperature to favour the reaction B.

If k_1 is less temperature-dependent than k_2 , then reaction run at the highest temperature to favour the reaction B, but then drops to prevent the loss of too much B

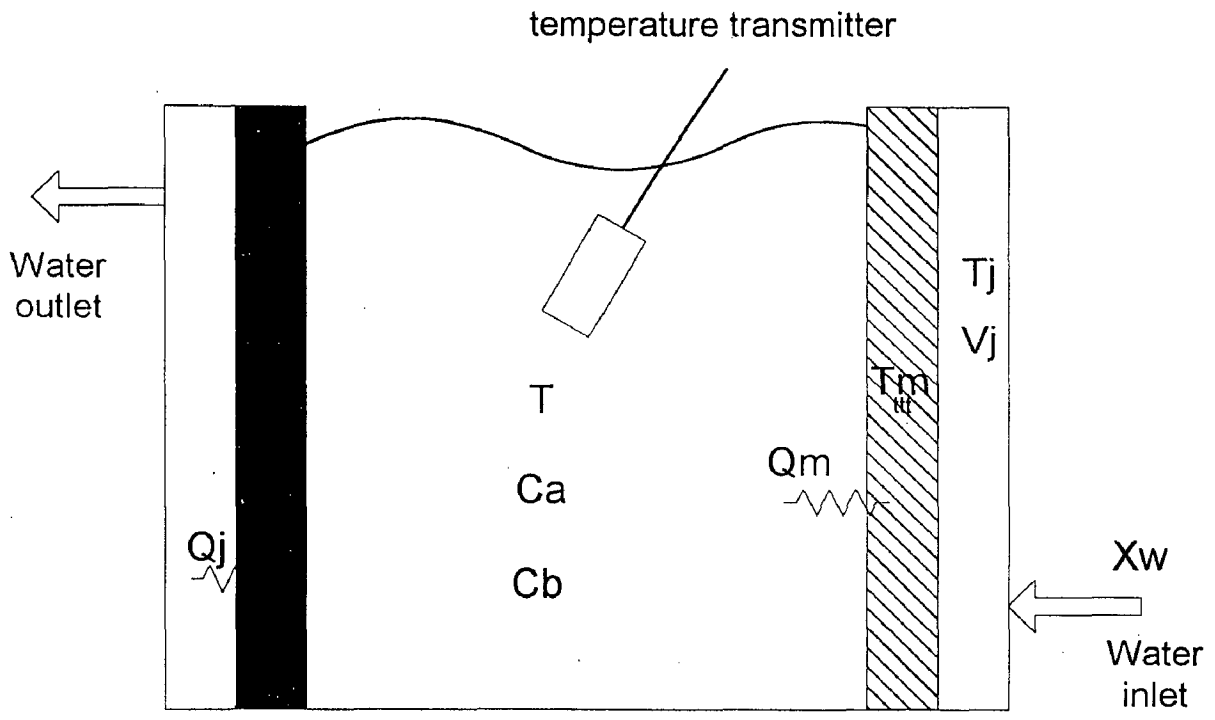


Fig. 5.7: Batch Reactor

5.4.1 Problem formulation

The equations for the reaction liquid inside the tank are given as

$$\dot{C}_A = -k_1 C_A$$

$$\dot{C}_B = k_1 C_A - k_2 C_B$$

$$\dot{T} = (-\lambda_1 / \ell C_p) k_1 C_A + (\lambda_2 / \ell C_p) k_2 C_B - Q_M / V \ell C_p$$

$$Q_M = h_i A_i (T - T_M)$$

$$\dot{T}_M = (Q_M - Q_J) / \ell_M C_M V_M$$

Where, C_A the concentration of component A

C_B the concentration of component B

T the temperature inside the reactor.

T_M the maximum temperature, and

T_J the temperature of water-jacket.

The output signal of the temperature controller goes to two split-ranged valves, a steam valve and a water valve. The instrumentation is all-pneumatic, so the controller output pressure goes from 3 to 15 psig. The valve will be adjusted so that the steam valves is wide open when the controller output pressure is at 15 psig and closed at 9 psig. The water valve will be closed at 9 psig and wide open at 3 psig. we have identified the batch reactor in the cooling mode. The equations of water-jacket during the cooling are:

$$\dot{V}_J = F_{W0}$$

$$V_J \dot{T}_J = F_{W0} T_{J0} - F_{W0} T_J + Q_J / \ell_J C_J$$

$$Q_J = h_{0W} A_0 (T_M - T_J)$$

$$F_{W0} = C_{VW} X_W (20)^{1/2}$$

5.4.2 Experimental data

The required data are given in table 5.3,

TABLE 5.3

C_A	C_B	T	T_M	T_J	X_w	\dot{C}_A	\dot{C}_B	\dot{T}	\dot{T}_M	\dot{T}_J
0.8518	0.1537	0.6986	0.9000	0.8731	0.1525	0.4502	0.5583	0.7657	0.1826	0.8920
0.8402	0.1641	0.7490	0.8715	0.8059	0.3623	0.3679	0.6412	0.7387	0.1710	0.8920
0.8287	0.1765	0.7881	0.8240	0.6669	0.5459	0.2977	0.7112	0.6960	0.1159	0.8920
0.8152	0.1910	0.8273	0.7765	0.6221	0.7951	0.2220	0.7864	0.6567	0.1423	0.8920
0.7863	0.2220	0.8832	0.4724	0.1538	0.8738	0.1079	0.8973	0.3692	0.1000	0.8920
0.7535	0.2571	0.8944	0.3489	0.1090	0.9000	0.1000	0.9000	0.2437	0.2055	0.8920
0.7207	0.2902	0.9000	0.3204	0.1018	0.9000	0.1064	0.8885	0.2102	0.2334	0.8920
0.6899	0.3233	0.9000	0.3157	0.1000	0.8869	0.1255	0.8644	0.1983	0.2373	0.8920
0.6590	0.3543	0.9000	0.3157	0.1000	0.8738	0.1446	0.8406	0.1909	0.2373	0.8920
0.6282	0.3853	0.8944	0.3109	0.1009	0.8607	0.1765	0.8046	0.1764	0.2420	0.8920
0.5993	0.4142	0.8888	0.3109	0.1018	0.8213	0.2063	0.7711	0.1673	0.2412	0.8920
0.5723	0.4432	0.8832	0.3109	0.1027	0.7951	0.2342	0.7396	0.1591	0.2404	0.8920
0.5453	0.4680	0.8720	0.3109	0.1063	0.7426	0.2728	0.6990	0.1491	0.2404	0.8920
0.5202	0.4948	0.8608	0.3062	0.1090	0.6902	0.3086	0.6611	0.1358	0.2451	0.8920
0.4971	0.5176	0.8497	0.3062	0.1125	0.6377	0.3419	0.6263	0.1281	0.2451	0.8920
0.4740	0.5403	0.8385	0.3062	0.1175	0.5721	0.3738	0.5932	0.1211	0.2462	0.8920
0.4528	0.5610	0.8217	0.3062	0.1242	0.4934	0.4125	0.5544	0.1144	0.2474	0.8920
0.4335	0.5796	0.8049	0.3062	0.1336	0.4279	0.4482	0.5190	0.1090	0.2509	0.8920
0.4161	0.5982	0.7881	0.3014	0.1480	0.3492	0.4811	0.4867	0.1004	0.2641	0.8920
0.3988	0.6147	0.7713	0.3062	0.1695	0.2705	0.5121	0.4564	0.1021	0.2726	0.8920
0.3814	0.6313	0.7545	0.3062	0.2008	0.1918	0.5414	0.4280	0.1000	0.2951	0.8920

Continued

C_A	C_B	T	T_M	T_J	X_w	\dot{C}_A	\dot{C}_B	\dot{T}	\dot{T}_M	\dot{T}_J
0.3660	0.6457	0.7378	0.3109	0.2501	0.1393	0.5682	0.4020	0.1038	0.3277	0.8920
0.3525	0.6581	0.7210	0.3204	0.2905	0.1000	0.5930	0.3785	0.1134	0.3471	0.8920
0.3390	0.6726	0.7098	0.3347	0.3263	0.1000	0.6102	0.3616	0.1276	0.3587	0.8920
0.3275	0.6829	0.7042	0.3632	0.3577	0.1000	0.6202	0.3516	0.1554	0.3517	0.8920
0.3140	0.6953	0.7042	0.3964	0.3801	0.1000	0.6248	0.3460	0.1867	0.3331	0.8920
0.3024	0.7078	0.7098	0.4249	0.3577	0.1131	0.6230	0.3464	0.2125	0.2827	0.8920
0.2889	0.7202	0.7154	0.4487	0.3084	0.1656	0.6218	0.3458	0.2332	0.2144	0.8920
0.2773	0.7305	0.7210	0.4724	0.2681	0.1918	0.6200	0.3459	0.2542	0.1539	0.8920
0.2639	0.7429	0.7322	0.4724	0.2457	0.2049	0.6131	0.3502	0.2502	0.1376	0.8920
0.2523	0.7553	0.7322	0.4487	0.2501	0.1918	0.6175	0.3449	0.2247	0.1687	0.8920
0.2388	0.7656	0.7266	0.4154	0.2681	0.1525	0.6282	0.3343	0.1903	0.2206	0.8920
0.2272	0.7780	0.7210	0.3917	0.3039	0.1262	0.6381	0.3243	0.1659	0.2773	0.8920
0.2157	0.7884	0.7098	0.3822	0.3398	0.1000	0.6532	0.3104	0.1571	0.3160	0.8920
0.2041	0.7987	0.7042	0.3869	0.3667	0.1000	0.6623	0.3013	0.1615	0.3323	0.8920
0.1945	0.8090	0.7042	0.4059	0.3667	0.1000	0.6657	0.2972	0.1792	0.3106	0.8920
0.1829	0.8194	0.7042	0.4297	0.3443	0.1262	0.6696	0.2925	0.2013	0.2641	0.8920
0.1713	0.8276	0.7098	0.4534	0.3084	0.1525	0.6685	0.2921	0.2219	0.2074	0.8920
0.1617	0.8380	0.7154	0.4677	0.2860	0.1787	0.6669	0.2921	0.2334	0.1733	0.8920
0.1501	0.8483	0.7154	0.4629	0.2815	0.1656	0.6710	0.2872	0.2269	0.1749	0.8920
0.1405	0.8669	0.7154	0.4392	0.2950	0.1525	0.6743	0.2825	0.2021	0.2136	0.8920
0.1308	0.8752	0.7098	0.4202	0.3174	0.1262	0.6827	0.2747	0.1831	0.2532	0.8920
0.1270	0.8835	0.7042	0.4059	0.3487	0.1000	0.6887	0.2690	0.1700	0.2951	0.8920
0.1096	0.8917	0.6986	0.4107	0.3711	0.1000	0.6992	0.2589	0.1736	0.3075	0.8920
0.1000	0.9000	0.6930	0.4202	0.3756	0.1131	0.7069	0.2517	0.1834	0.2990	0.8920
0.1000	0.9000	0.6930	0.4202	0.3756	0.1131	0.7069	0.2517	0.1834	0.2990	0.8920

5.4.3 RESULTS

Gradient descent algorithm (modified back algorithm) has been discussed in chapter 4, is used to identify the system parameters. The number of input and output to the ANN is 6 and 5 respectively i.e., the structure of the network is (6-5). Total number of patterns is 46. The value of learning rate λ has been taken as 0.1. The trajectory of various parameters is shown in figs 5.8, 5.9, 5.10, 5.11 and 5.12. By viewing the graphs, it is analysed that convergence is achieved. The values of the parameters are:

i	w_{i1}	w_{i2}	w_{i3}	w_{i4}	w_{i5}	w_{i6}
1	11.7054	5.2651	-26.9313	-3.5183	-7.6111	18.5117
2	1.7212	0.4900	-2.9491	-0.5031	-0.8376	2.4826
3	0.1566	-0.0420	-0.2260	0.8156	0.0164	0.2413
4	0.4003	0.3896	0.2593	-2.5433	2.4736	-0.0690
5	0.1662	0.1891	0.0253	-0.0210	0.0483	0.0004

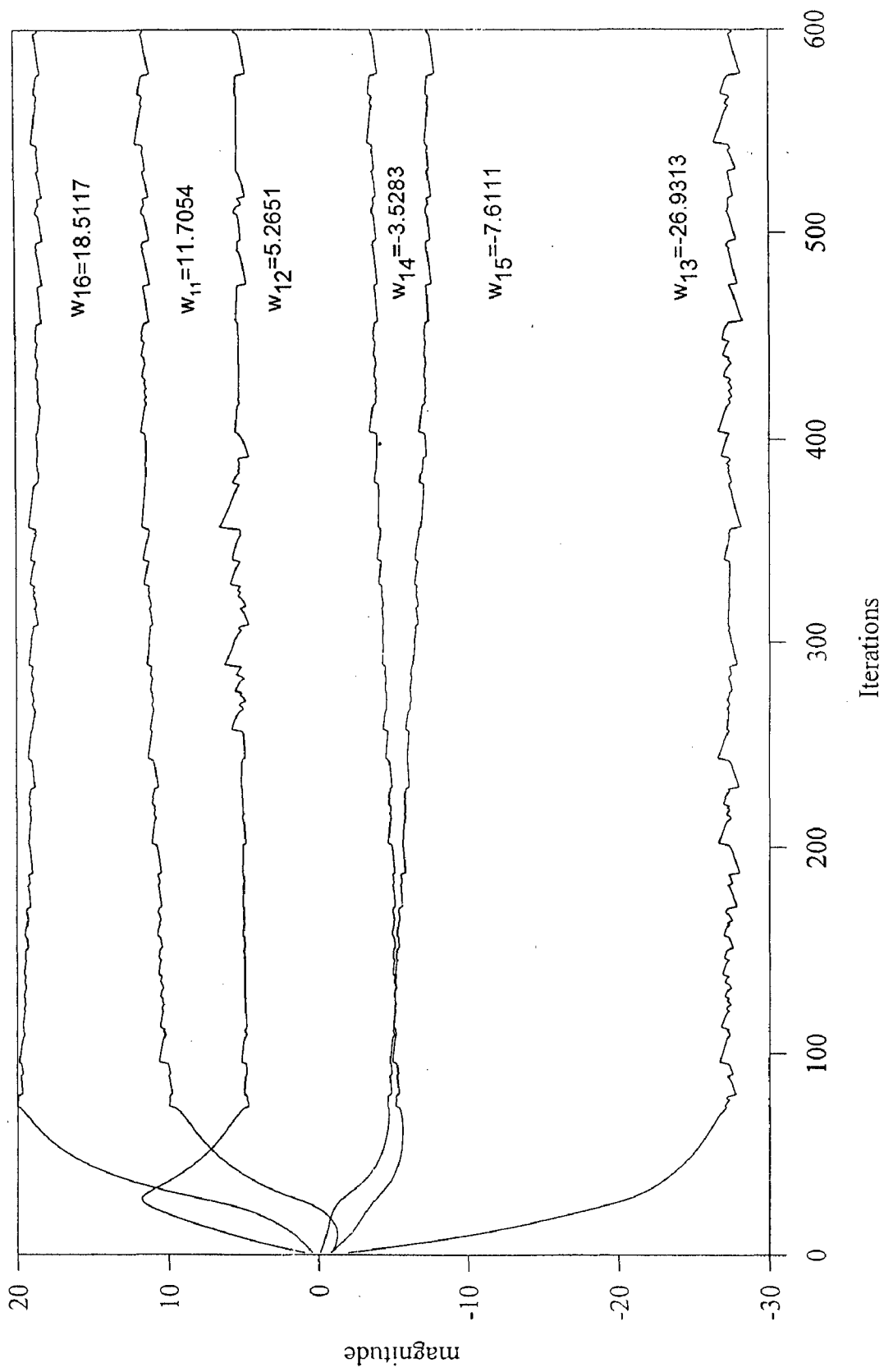


Fig.5.8 Trajectory of $w_{11}, w_{12}, w_{13}, w_{14}, w_{15}$, and w_{16}

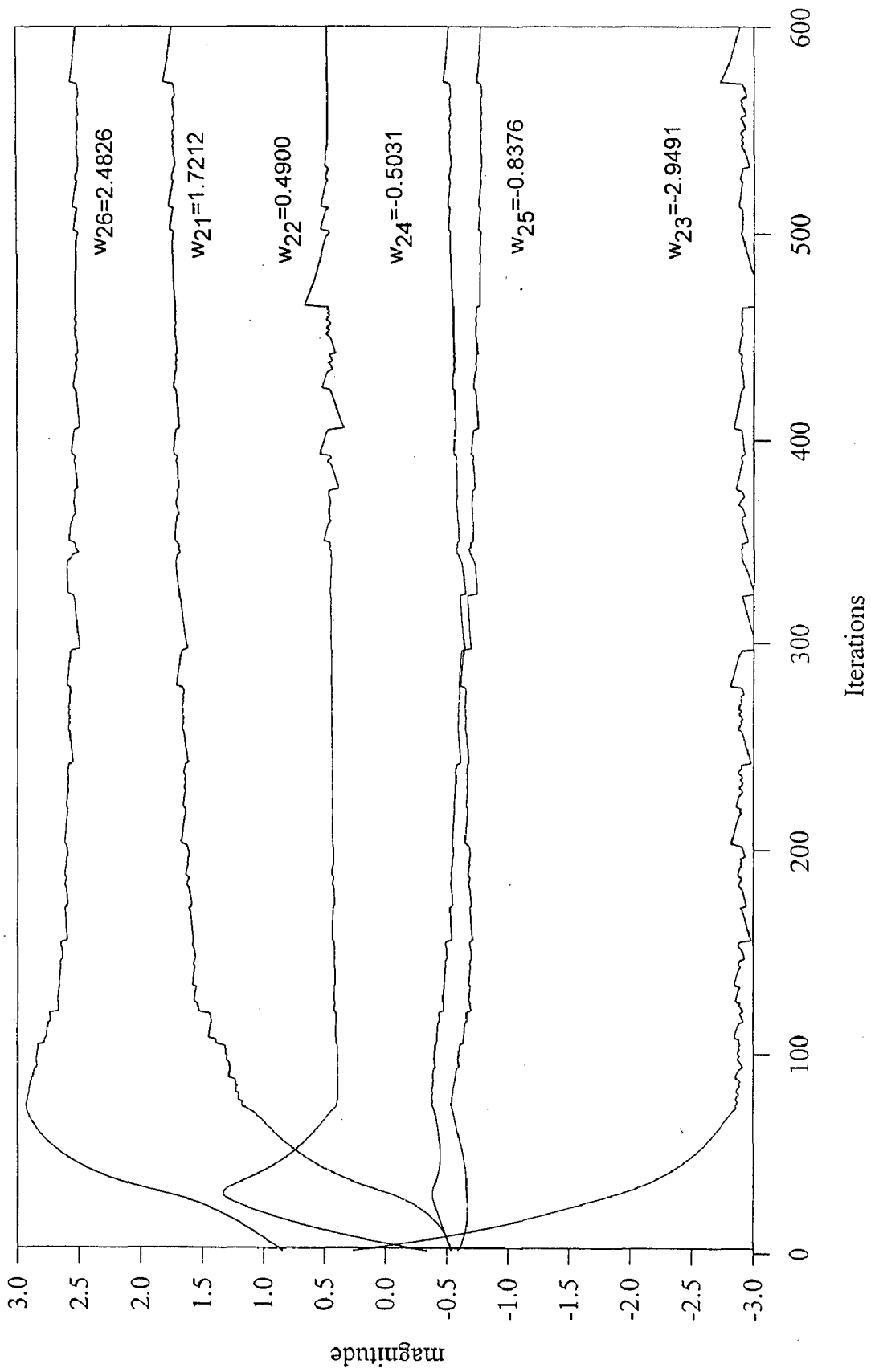


Fig.5.9 Trajectory of w_{21} , w_{22} , w_{23} , w_{24} , w_{25} , and w_{26}

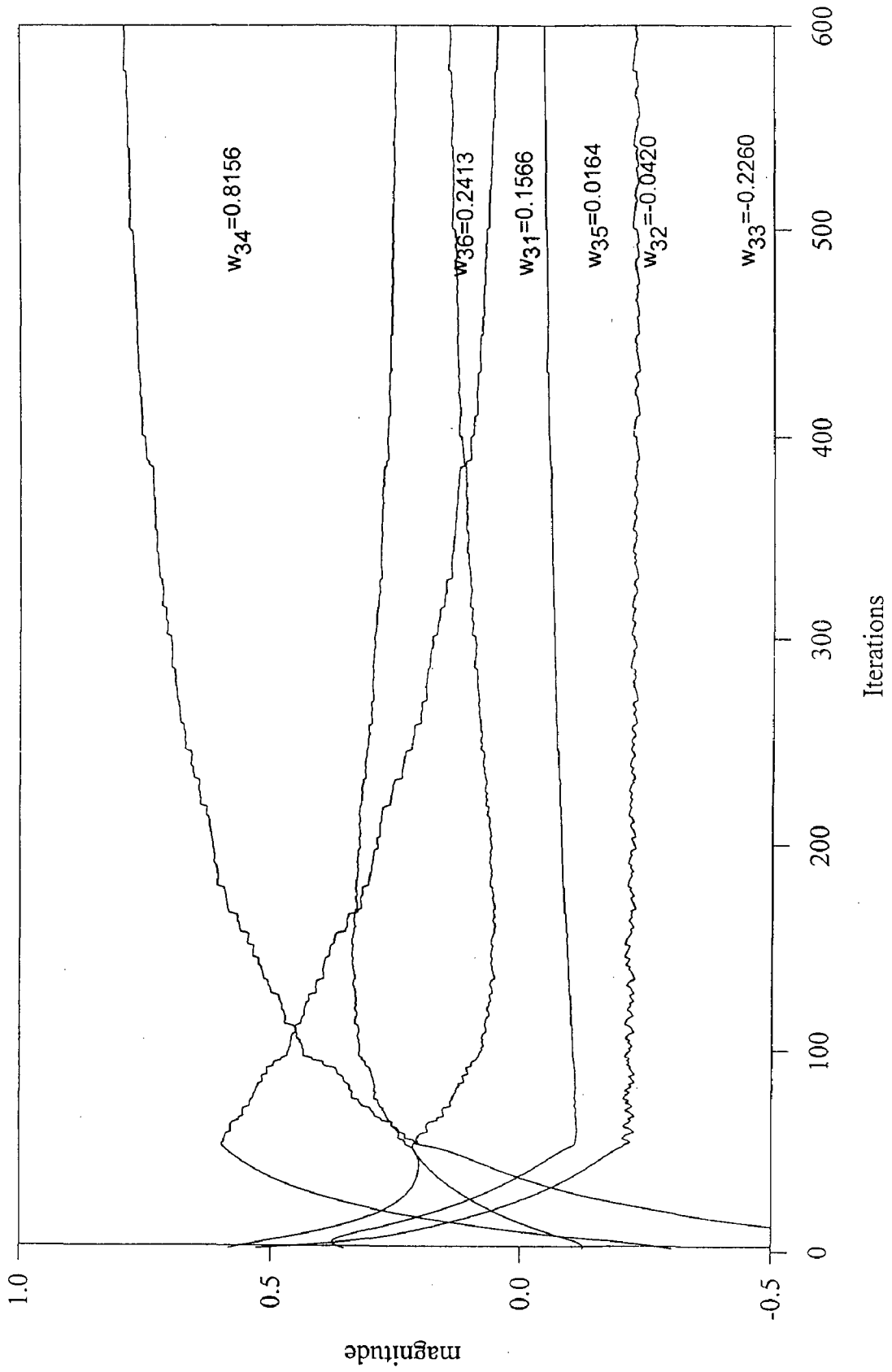


Fig.5.10 Trajectory of w_{31} , w_{32} , w_{33} , w_{34} , w_{35} , and w_{36}

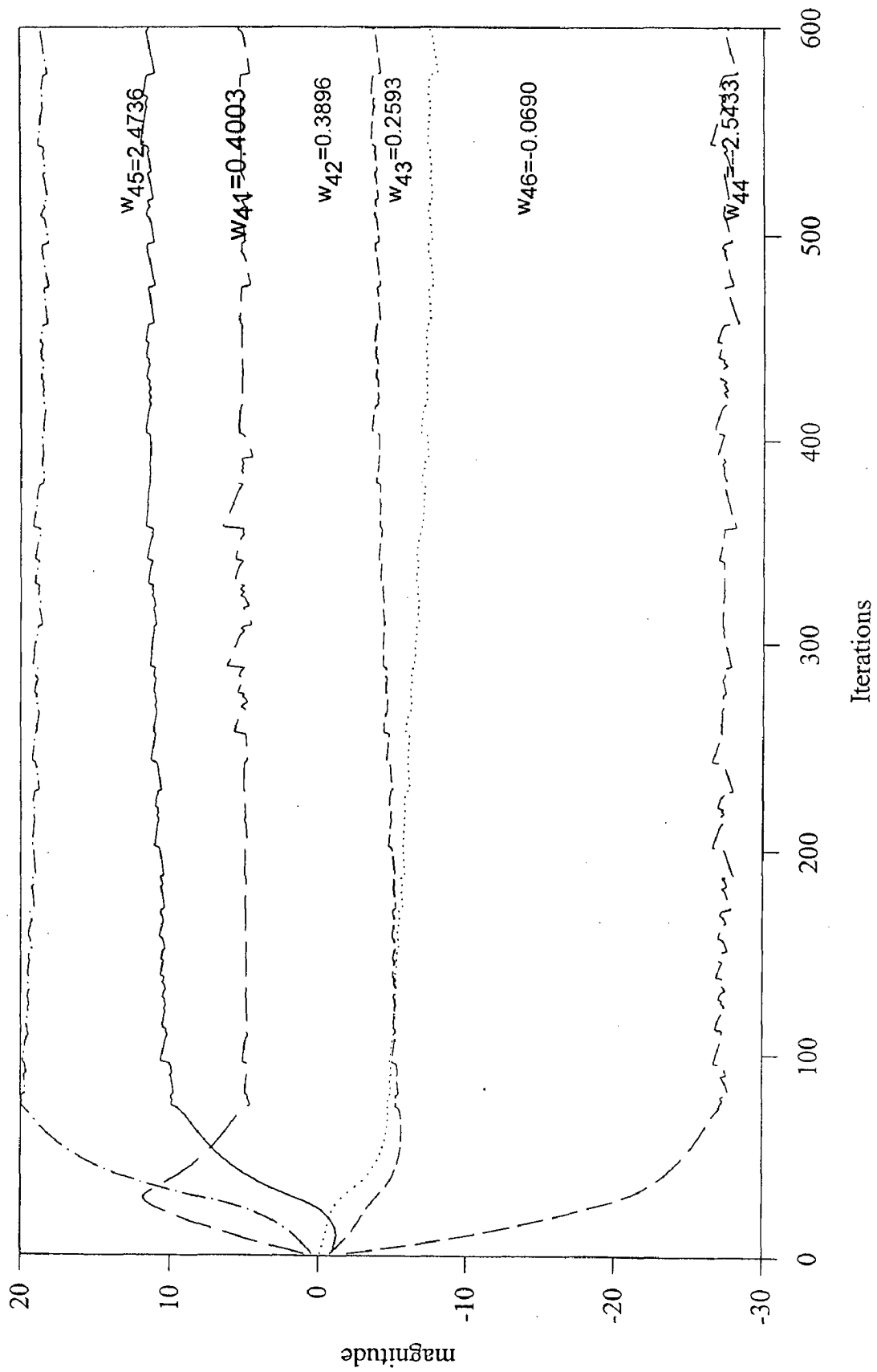


Fig. 5.11 Trajectory of w_{41} , w_{42} , w_{43} , w_{44} , w_{45} , and w_{46}

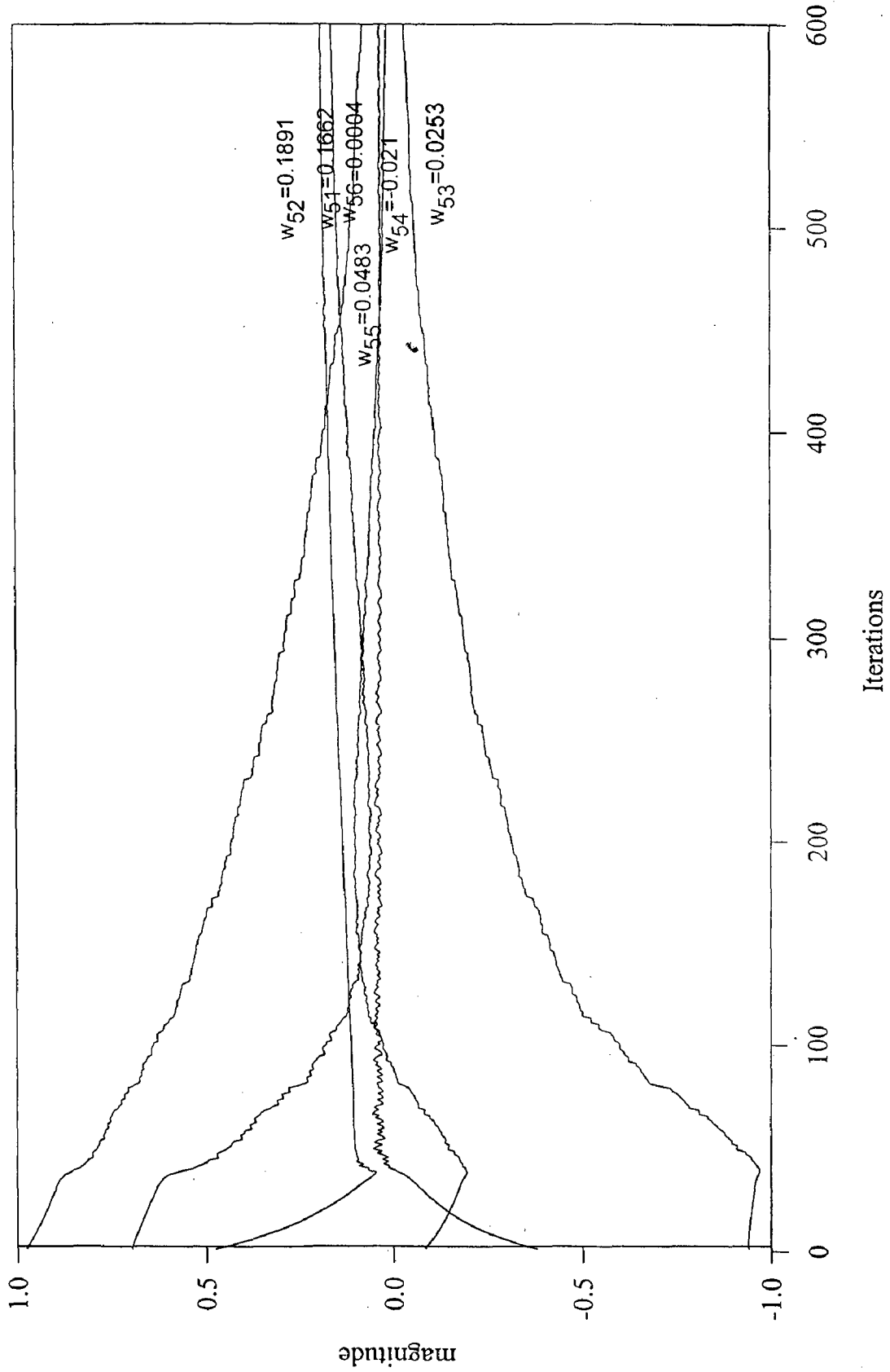


Fig.5.12 Trajectory of w_{51} , w_{52} , w_{53} , w_{54} , w_{55} , and w_{56}

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSIONS

The solution of the problems suggests the effectiveness of the ANN implementation for the system identification problems. Some of the important conclusions derived in the dissertation work are given below.

The identification of system by analytical methods is quite complex. Substantial benefits are gained by using neural network in such cases. In the neural network approach, the parameters are adjusted dynamically which are not possible using conventional techniques. A neural network is able to identify the linear dynamic system very accurately and quickly.

From example one (section 5.2) it is observed that SLNN using the gradient descent technique (Modified back-propagation) as learning algorithm provides more accurate estimation than SLNN using back error-propagation, and learning is observed to be much more faster as compared to single-layer feed-forward ANN for the same error performance. The learning of SLNN (using GD algorithm) is found to be more than 50 times faster as compared to SLNN (back-error-propagation).

From example two (section 5.3) it is observed that the large learning rate $[0,1]$ in weight modification equation resulted slow convergence of error performance. The value of learning constant should be appropriately selected in accordance to the error value.

From the example three (section 5.4) it is observed that a non-linear system can be identified in the form of a linear system at the operating conditions

6.2 FUTURE SCOPE

The field of identification is at the moment bewildering even for experts. Many different methods and techniques are being analysed and treated, “new methods” are suggested en masse, and on the surface, the field appears to look more like a bag of tricks, rather than a unified subject. I would like to point out a few facts that have struck us.

- The learning rate should be determined dynamically.
- More powerful optimisation technique should be used for learning the neural networks.
- Learning procedure should be developed for noisy inputs.

REFERENCES

1. K.J. Astrom and P. Eykhoff, "System Identification-A Survey," *Automatica*, Vol. 7, pp. 123-162, Pergaman Press, 1971, Printed in Great Britain.
2. Robert Kalaba and Karl Spingarn," *Control, Identification, and Input optimisation*," Plenum Press, 1982.
3. William L. Luyben, "Process modelling and control for chemical engineering". McGraw-Hill International editions, 1989.
4. Katsuhiko Ogata, " *Time Control System*," Prentice-Hall, Inc.
5. K.S. Narendra and K. Parthasarathy, "Identification and control of dynamical system using neural networks," *IEEE Transactions on neural networks*. Vol. 1, pp. 4-27, 1990.
6. S. Bhamra and H. Singh," Single layer neural networks for linear system identification. Using gradient descent technique," *IEEE Transactions on neural networks*. Vol. 4, No. 5, 1993.
7. Elias B. Kosmatopoulos and Masios M. Polycarpou, "High-order neural network structures for idenification of dynamical systems," *IEEE Transactions on Neural networks*. Vol. 6, No. 2, March 1998.
8. I. J. Nagrath & M Gopal "Control System Engineering," Wiley Eastern Limited, May 1995.
9. P. F. Baldi and Kurt Hornik, "Learning in linear neural networks: A Survey," *IEEE Transactions on Neural networks*. Vol. 6, No. 4, July 1995

10. Dr. Valluru B. Rao & Hayagria V. Rao " C++ Neural networks and Fuzzy logic," BPB Publications, 1996
11. Wang and Ling., "Runge-kutta neural network for identification of dynamical system in high accuracy," IEEE Transactions on Neural networks. Vol. 9, No. 2, March 1998.
12. Mohamed Ibnkahla. " Neural networks modelling and identification of non-linear channels with memory: Algorithm, Application, and Analytical Models" IEEE Transactions on signal processing. Vol. 46, No. 5, May 1998
13. Zhou and Jannie, "Advanced neural networks training algorithm with reduced complexity based on Jacobean Deficiency," IEEE Transactions on neural networks. Vol. 9, No. 3, pp. 448-453, 1998.
14. Mrs. Laxmi Srivastava, "Artificial neural network to power system voltage security assessment," A Ph.D. thesis submitted in the department of electrical engineering, university of Roorkee, Roorkee, July 1998

APPENDIX A

THEOREM 1: *The gradient vector represents the direction of steepest ascent.*

Proof: From any arbitrary point X in n -dimensional space, take a small step dr such that

$$dr^T = \{dx_1, dx_2, \dots, dx_n\}$$

And

$$dr^T dr = (ds)^2 = \sum (dx_i^2)$$

Where dx_1, dx_2, \dots, dx_n represent the components of the vector dx , and ds denotes the magnitude of the vector dr . If f is the value of the objective function at the base point, the change in f (df) associated with the change in the coordinates dx_i is given by

$$df = \sum_{i=1}^n (\partial f / \partial x_i) dx_i = \nabla f^T dr$$

If u denotes the unit vector along the direction dr , we can write

$$dr = u ds$$

The rate of change of the function with respect to the step length ds is given by Eq. (6.56)

as

$$d/ds = \sum_{i=1}^n (\partial f / \partial x_i) (dx_i / ds) = \nabla f^T (d/ds) = \nabla f^T u$$

The value of (df / ds) will be different for different directions and we are interested in finding the particular step dr along which the value of (df / ds) will be maximum. This will give the direction of steepest ascent.

By denoting the i th component of f and u as b_i and u_i respectively, Eq. can be written as

$$df / ds = \sum_{i=1}^n b_i u_i$$

In this equation, u_i are the variables whose magnitudes will depend on the direction of dr . Selecting the set u_1, u_2, \dots, u_n which makes (df/ds) as maximum subject to the condition.

$$|u|^2 = \sum_{i=1}^n u_i^2 = 1$$

Thus the problem of finding the direction of steepest ascent can be posed as a maximization problem subject to an equality constraint. This problem can be solved by the LaGrange multiplier method where the LaGrange function is given by

$$L(u, \lambda) = df / ds + \lambda(1 - \sum_{i=1}^n u_i^2)$$

Where λ is the LaGrange multiplier to be determined. By setting the partial derivatives of L with respect to u_i and λ equal to zero, we obtain the necessary conditions as

$$\begin{aligned} \partial L / \partial u_i &= 0 \\ \partial L / \partial \lambda &= 0 \end{aligned}$$

Or

$$\begin{aligned} b_i - 2\lambda u_i &= 0 \\ \sum_{i=1}^n u_i^2 &= 1 \end{aligned}$$

These equations give us

$$u_i = b_i / 2\lambda$$

And

$$\sum_{i=1}^n (b_i / 2\lambda)^2 = 1$$

From which the value of λ can be obtained as

$$2\lambda = \left(\sum_{i=1}^n b_i^2 \right)^{1/2} = |\nabla f|$$

With the help of Eq. (6.66) Eq. (6.64) can be written as

$$u_i = (b_i / |\nabla f|) = (\partial f / \partial x_i) / |\nabla f|$$

This result shows that the direction of steepest ascent is nothing but a vector pointing in the direction of the gradient.