IMPLEMENTATION OF PREISACH-KRASNOSELSKII HYSTERESIS MODEL WITH THE USE OF ARTIFICIAL NEURAL NETWORKS

By

Zhao Wang

A project

Presented to Ryerson University

In partial fulfillment of the

Requirements for the degree of

Master of Engineering

In the program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2004

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this project.

I authorize Ryerson University to lend this project to other institutions or individuals for the purpose of scholarly research.

Signature

I further authorize Ryerson University to reproduce this project by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

BORROWER'S PAGE

Ryerson University requires the signatures of all persons using or photocopying this project. Please sign below, and give address and date.

IMPLEMENTATION OF PREISACH-KRASNOSELSKII HYSTERESIS MODEL WITH THE USE OF ARTIFICIAL NEURAL NETWORKS

ABSTRACT

Accurate modeling of hysteresis is essential for both the design and performance evaluation of electromagnetic devices. This project proposes the use of feedforward neural networks to implement an accurate magnetic hysteresis model based on the mathematical definition provided by the Preisach-Krasnoselskii (P-K) model. Feedforward neural networks are a linear association networks that relate the output patterns to input patterns. By introducing the multi-layer feedforward neural networks to Hysteresis modeling, the proposed multi-layer perceptron neural network and radial basis function neural network both are capable of modeling hysteresis accurately. The feedforward neural networks make the hysteresis modeling accurate without estimation of double integrals. Simulation results provide the detailed illustrations. The comparisons with the experiments show that the proposed approach is able to satisfactorily reproduce many features of observed hysteresis phenomena and in turn can be used for many applications of interest.

ACKNOWLEDGMENTS

The author would like to give my academic supervisor Dr. Alireza Sadeghian sincere gratitude and appreciation for his guidance towards my academic success. The project would also not be successful without other members of committee.

Specially thanks to my husband and daughter for their full support.

Table of Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1. Introduction

1.1. Motivation

The accurate and effective modeling of hysteresis is crucial to the prediction of the performance of electromagnetic devices. Ferro-magnets, such as iron, nickel, and cobalt etc, will tend to stay magnetized to some extent after being subjected to an external magnetic field; this tendency to "remember their magnetic history" is called hysteresis.

Magnetic materials are usually described in terms of a single valued B-H curve (magnetic flux density – magnetic field) in finite element analysis.

The industrial applications range from modeling of recording heads, to modeling the behavior of magnetic cores, a single-valued B-H curve may adequately characterize the magnetic material. In many instances, it is important to have a good model of hysteresis in order for the core loss to be accurately predicted since hysteresis loss can be a significant component of the total loss.

In many electronic devices, such as motors and generators, the behavior of the device is strongly influenced by the behavior of the magnetic components (e.g. inductors, transformers). Hysteresis then can have a powerful influence on both

devices performance and losses. The use of phenomenological mathematical models of hysteresis is a well-established approach for describing magnetization phenomena. [1] A typical example of such models is the classical Preisach model that is a scalar model of magnetic hysteresis based on the physical mechanisms of magnetization. [2] Attempts to separate the physical meaning of magnetism from the Preisach model have resulted in the Preisach-Krasnoselskii model. [3] The P-K model is a mathematical tool that can be used for the mathematical description of hysteresis of any physical nature. This purely mathematical model takes a macroscopic point of view to hysteresis phenomena and provides no insights into the physical causes of hysteresis. Owing to this characteristic, the P-K model becomes a very powerful and effective tool for designing devices with hysteresis features.

Recently, artificial neural networks have been widely used to model a wide range of systems for which mathematical models either cannot be defined, or are ill defined. It has been successfully used in those applications involving pattern matching [4] and systems modeling [5]. This success was the main motivation behind some attempts to utilize ANNs in the area of magnetic hysteresis. Innovative techniques for the accurate modeling of hysteresis phenomena will have very important practical significance.

1.2. Feasibility

This project proposes the use of artificial neural networks as functional approximation tools with local memory capabilities that will be suitable for the construction of a P-K model. The exploration of the working hypothesis in this porject is that both P-K model and the ANN model have local memory, and

both exhibit fault tolerance capabilities. Furthermore, they both are mathematical tools, which take a macroscopic approach to the problem.

Specifically, this paper considers the applications of multi-layer perceptron (MLP) and radial basis function (RBF) networks that are already proven to be universal approximators, and will show that there is a natural correlation between; (i) the coefficients of the P-K model and the connection weights in ANN model and, (ii) the Preisach operators and ANN's activation functions. In this respect, this project treats the magnetic hysteresis system as a black box and the mathematical tools as means for the identification of the unknown input/output relation governing the system.

1.3. Summary

In summary, P-K model research can bring up alternative mathematical tools to ensure the accurate and effective modeling of hysteresis.

In this research paper, a review of P-K model and neural networks in Chapter 2 and Chapter 3 will uncover that multi-layer perceptron and radial basis function networks as universal approximators can be used to reconstruct the P-K model. Next, my new MLP and RBF neural network implementations using Square Permalloy 80 at different frequencies including DC, 1 kHz, 3 kHz, and 6 kHz will be described in detail followed by modeling and analysis of the implementations in Chapter 4. The simulation experiments are also provided for illustration purposes. Finally, the conclusion section will show the research contributions of this work by presenting a new methodology for accurate and effective modeling of hysteresis.

Chapter 2

HYSTERESIS & PREISACH-KRASNOSELSKII MODEL

2 Hysteresis & Preisach-Krasnoselskii Model

2.1. Hysteresis

The process of magnetic hysteresis is basically a sequence of Barkhausen jump. The Barkhausen effect consists of discontinuous changes in flux density during smoothly changing magnetic field. They are known as Barkhausen jumps and are caused by the sudden irreversible motion of magnetic domain walls as they break away from pinning sites.

When we detect a jump, the system is leaving a metastable state in favor of some other state of lower energy. Two energy terms are involved in the event, the free energy change ΔF from the initial state to the final state and the amount of energy ΔE dissipated as heat during the jump. If we can keep track of the energy changes while the jump sequence proceeds, we would have at hand quite a general description of hysteresis. The main difficulty behind this picture is that we do not know the general principles controlling the statistics of Barkhausen jump and those determining in what proportion the work performed by external sources is subdivided into stored and dissipated energy in the jump sequence. This brought to us F. Preisach; the German physicist gave the idea of describing a generic hysteresis system as the superposition of many bistable units and gave this idea an elegant and illuminating graphical representation.

In the 1970's, the Russian mathematician Krasnoselskii came across Preisach's model and understood that it contained a new general mathematical idea. As a result, a new mathematical model was revealed which can be used for the mathematical description of hysteresis of any physical nature.

2.2. The Preisach Approach

We shall first discuss how the Preisach approach fits in with the general interpretation of the magnetization process as a sequence of Barkhausen instabilities, and on this basis we shall introduce the Preisach-Krasnoselskii Model.

2.2.1. Preisach Units

The simplest energy profile giving a clear representation of a Barkhausen instability is shown in Fig 2.1. The term *preisach unit* is refers to a system characterized by an energy profile of this kind. The state variable m represents some characteristic magnetic moment involved in the problem, to be identified case by case. The system is confined to the interval $-\Delta m \le m \le +\Delta m$ $(\Delta m > 0)$. The free energy of the unit, $F_p(m)$ is determined by its values at the points $-\Delta m, 0, +\Delta m$. It is convenient to express these values in the form $-\mu_0 h_u \Delta m, \mu_0 h_c \Delta m, \mu_0 h_u \Delta m$, where h_u and h_c are appropriate fields. The three points are connected by straight segments, which give the two characteristic energy gradients, $h_u - h_c$ for $m > 0$, and $h_u + h_c$ for $m < 0$. At last, we assume that the unit is coupled linearly to the magnetic field H, in the sense that the stability of the unit is controlled by the Gibbs energy

$$G_p(m;\mathrm{H}) = F_p(m) - \mu_0 \mathrm{H} m \qquad (2.1)$$

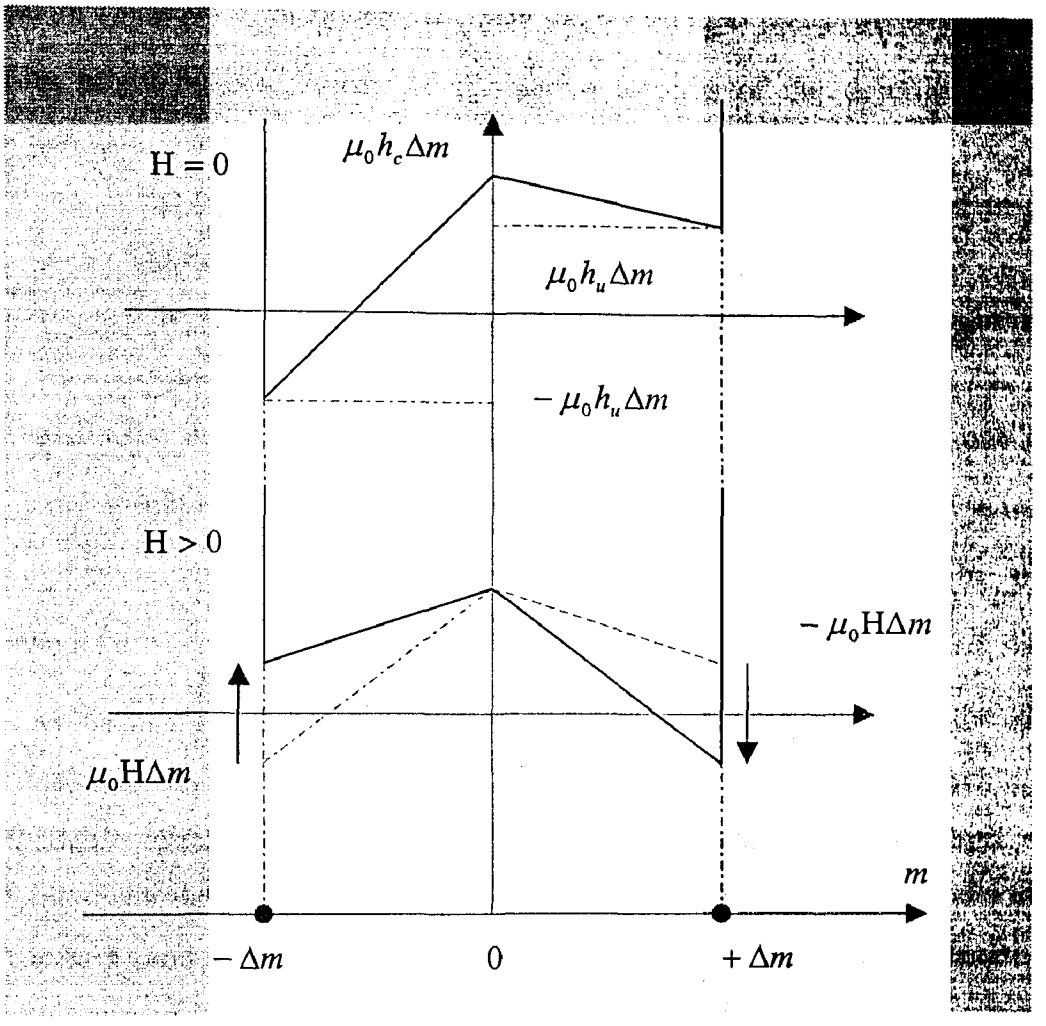

Fig 2.1. Representation of A Barkhausen Instability

Top: Energy Profile $F_p(m)$ Characterizing a Preisach Unit.

Bottom: Corresponding Gibbs Energy $G_p(m;\mathrm{H})$, as defined by Equation (2.1)

The presence of the field gives rise to a G_p profile of the same form as the F_p one, with $(h_u - H)$ in the place of h_u (see Fig 2.1). Other more realistic energy profiles might be considered, the distinctive features of the description are provided as:

i. The position, $\pm \Delta m$, of the two energy minima is independent of the field H;

ii. The height of the energy barriers separating the two minima depends linearly on H.

Depend on the relative values of h_u, h_c, and H, $G_p(m;H)$ may exhibit one or two local minima. When $H \geq h_u + h_c$, only one minima exists, at $m = +\Delta m$. Analogously, when $H \leq h_u - h_c$, only $m = -\Delta m$ is a minima. In the intermediate case, $h_u - h_c < H < h_u + h_c$, two minima, one stable and one metastable, exist if $h_c > 0$. We will not consider the case $h_c < 0$, because the potential then has one minimum under any arbitrary field, and the unit exhibits a reversible response to the field, which can be separately treated with no difficulty. The stability condition just listed can be expressed in graphical form with axes h_c and h_u. A given Preisach unit is identified by the representative point of coordinates (h_c, h_u) in this plane; Let us subdivide the half-plane $h_c > 0$ into the three regions show in Fig 2.2. The boundary of region II is the bifurcation set where Barkhausen jumps may occur.

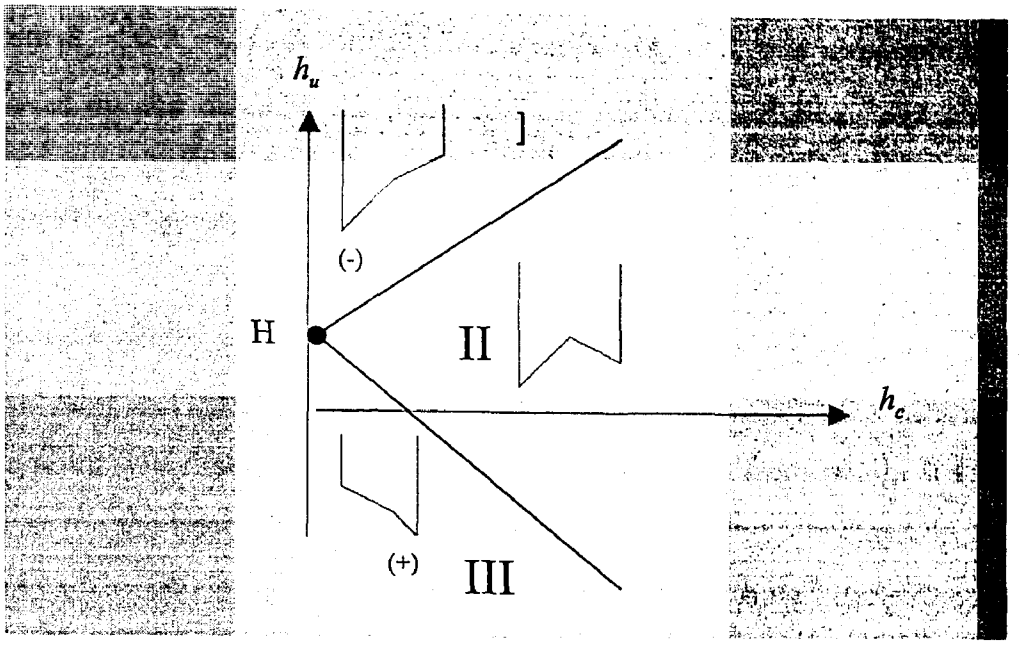

Fig 2.2. Preisach Plane with indication of regions where the energy $G_p(m;H)$ has one or two minima.

Region I: $h_u \geq H + h_c$. If the representative point lies in this region, the unit is certainly in the state $m = -\Delta m$, which we call the $(-)$ state.

Region II: $H - h_c < h_u < H + h_c$. If the representative point lies in this region, both the states $m = -\Delta m$ and $m = +\Delta m$ are locally stable. Which state the unit occupies will depend on the past history.

Region III: $h_u \leq H - h_c$. If the representative point lies in this region, the unit is certainly in the state $m = +\Delta m$, which we call the $(+)$ state.

The metastability region, region II, has the form of a core whose vertex is at the point $h_c = 0, h_u = H$. When the field changes in time, this metastability cone moves up and down in the plane and a given representative point may pass from one region to another. The boundary of cone represents the set of points where Barkhausen jumps may take place.

Let us consider a certain point of the Preisach plane, of coordinates (h_c, h_u). The point will certainly belong to region I when $H \to -\infty$. With increasing H, the point will first enter region II and then it will pass from region II to region III at the moment when $H = h_u + h_c$. At this point, a jump occurs and the unit suddenly passes from $m = -\Delta m$ to $m = +\Delta m$. A decreasing field will cause the same thing except that the jump from $m = +\Delta m$ to $m = -\Delta m$ takes place at $H = h_u - h_c$. The unit describes the square hysteresis loop is shown in Fig 2.3.

Units associated with different points of the Preisach Plane will describe loops differing in their width as well as in their shift along the H axis. Also, the loop height, $\Delta m(h_c, h_u)$, may be, in principle, different from unit to unit.

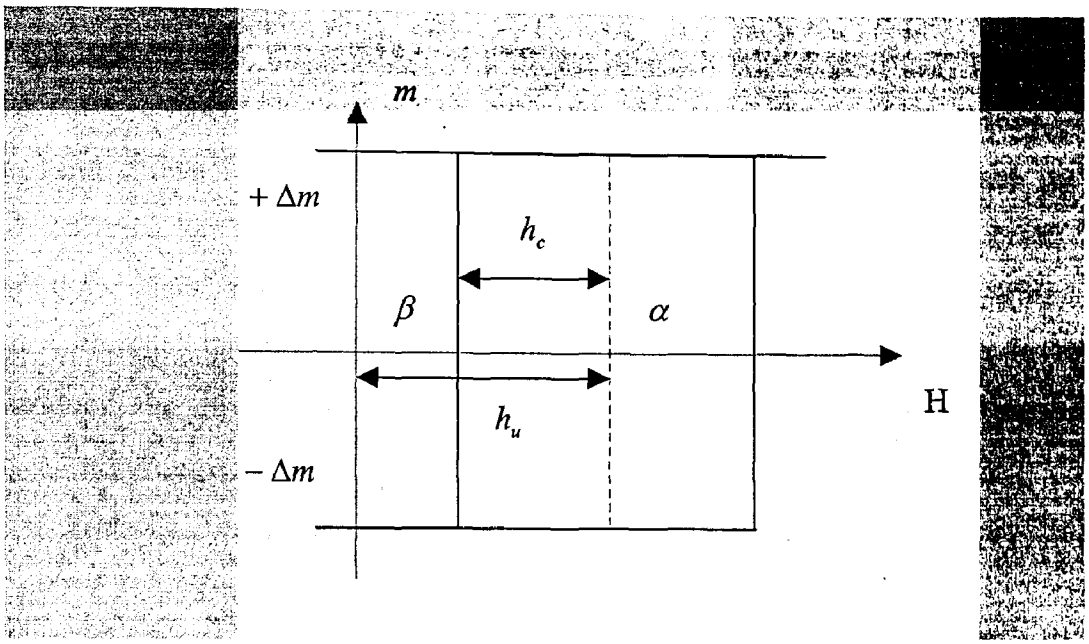

Fig 2.3 the Square Hysteresis Loop

The energy representation given in Fig 2.1 shows that, when a Barkhausen jump occurs, the energy $G_p(m;\mathrm{H})$ suddenly decreases by $2\mu_0 h_c \Delta m$. This is the amount of energy dissipated as heat in the jump. At the same time the free energy $F_p(m)$ changes by the amount $\Delta F_p = 2\mu_0 h_u \Delta m$ at the jump. We see that h_c and h_u are appropriate coordinates to keep track of dissipated and stored energy. In subsequent sections, we sometimes use the square loop switching fields,

$$\alpha = h_u + h_c \qquad (2.2)$$
$$\beta = h_u - h_c$$

instead of h_c and h_u, as alternative coordinates in the Preisach plane.

In the next section, the starting point of the Preisach model is usually the idea that a system should be described by a collection of elementary square loops of the kind shown in Fig 2.3. This is mathematically appropriate because the elementary loops are actually a pictorial representation of elementary hysteresis operators that can be superposed, to build up hysteresis nonlinearities with a more complex structure. The basic physical aspect of the approach is to introduce the two variables, h_c and h_u, describing stored energy versus dissipated energy.

2.3. The Preisach-Krasnoselskii Model

The phenomenon of hysteresis is encountered in many areas of science. But the interpretation of hysteresis differs from one area to another and from paper to paper. This led us to rigorous mathematical definitions of hysteresis in order to avoid confusion and ambiguity. The definition provided in [3] is of particular interest. According to this definition, a function is called a static hysteresis function if its input-output relationship is a multi-branch non-linearity for which a branch-to-branch transition occurs after each input reaches a minimum or maximum (Fig 2.4). The term "static" implies that branches of hysteresis are determined by the past extremum values of the inputs, while the speed of input variation between extremum values have no influence on branching. This definition highlights two key attributes of the hysteresis phenomenon. One is that hysteresis behaves sequentially and can be specified by following a predetermined sequence of states which the output is a function of input and internal states. Another is that hysteresis is inevitably associated with memory.

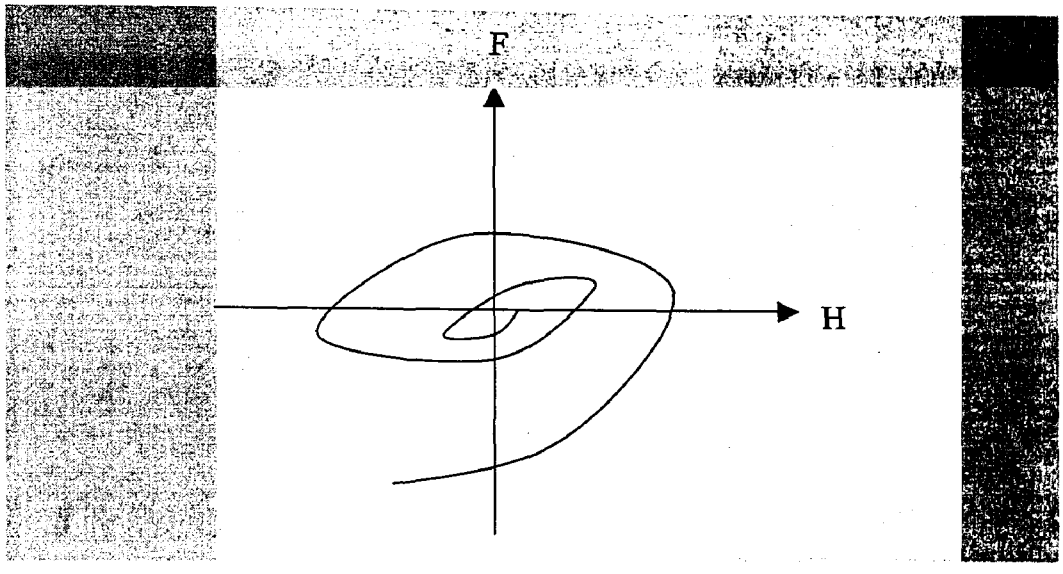

Fig 2.4. A hysteresis multi-branch non-linearity

The P-K model can be used as a mathematical tool to represent description of magnetic hysteresis [1]. This model can be represented by using a finite set of hysteresis binary operators, $\hat{\gamma}_{\alpha\beta}$ (with α and β corresponding to up and down switching values of the inputs), serves as a local memory (Fig 2.5).

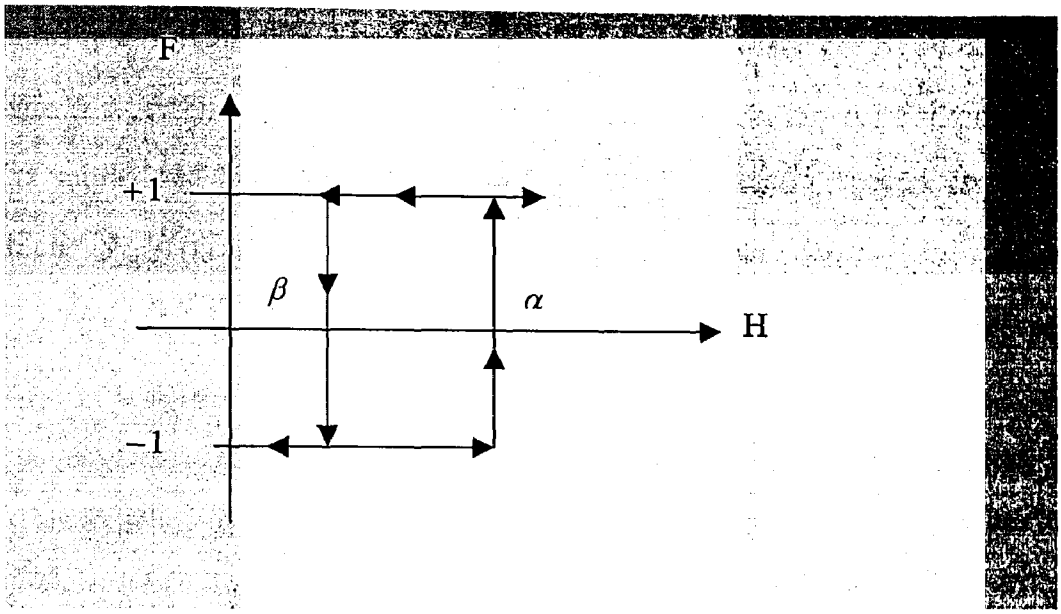

Fig 2.5. An elementary hysteresis operator $\hat{\gamma}_{\alpha\beta}$

The P-K model (2.3) expresses that given the output $F(t)$ and the input $H(t), t \le t_0$, then the output $F(t)$ can be uniquely determined for all $t \le t_0$. This captures the essence of the hysteresis phenomenon that is the past history of the input/output influences the present instantaneous output:

$$F(t) = \iint \mu(\alpha, \beta)\hat{\gamma}_{\alpha\beta} H(t) d\alpha d\beta \qquad (2.3)$$

Where $\mu(\alpha, \beta)$ is a set of an arbitrary weight function with respect to hysteresis operator $\hat{\gamma}_{\alpha\beta}$, and the hysteresis operator can only assume the values

+1 or -1. In (2.3), function $\mu(\alpha,\beta)$ represents the only model unknown and has to be determined from some experimental data.

The geometric interpretation of model (2.3) is based on the fact that there is one-to-one relationship between operator $\hat{\gamma}_{\alpha\beta}$ and points (α,β) of the half plane $\alpha > \beta$. Using this fact, it can be concluded that at any time the triangle T is subdivided into two sets (Figure 2.6). $S^+(t)$ consists of points (α,β) for which $\hat{\gamma}_{\alpha\beta}H(t)=1$ and $S^-(t)$ consists of points (α,β) for which $\hat{\gamma}_{\alpha\beta}H(t)=-1$. It can be shown that the interface $L(t)$ between $S^+(t)$ and $S^-(t)$ is a staircase line whose vertices have α and β coordinates coinciding with local maxima and minima of input at a previous time. The final link of $L(t)$ is attached to the line $\alpha=\beta$ and moves when the input changes. When the inputs increase, this link is a horizontal one and moves up, and it is a vertical one, and moves from right to left when the input decreases.

Using this interpretation, the model (2.3) can be presented as:

$$F(t)=\iint_{S^+(t)}\mu(\alpha,\beta)H(t)d\alpha d\beta-\iint_{S^-(t)}\mu(\alpha,\beta)H(t)d\alpha d\beta, \qquad (2.4)$$

From the expression (2.4), it follows that an instantaneous value of output depends on the shapes of the interface $L(t)$, which is determined by the extreme values of input at previous instants of time. As a result, the past extremum value of inputs shape the interface $L(t)$, and with which they leave their mark upon the future.

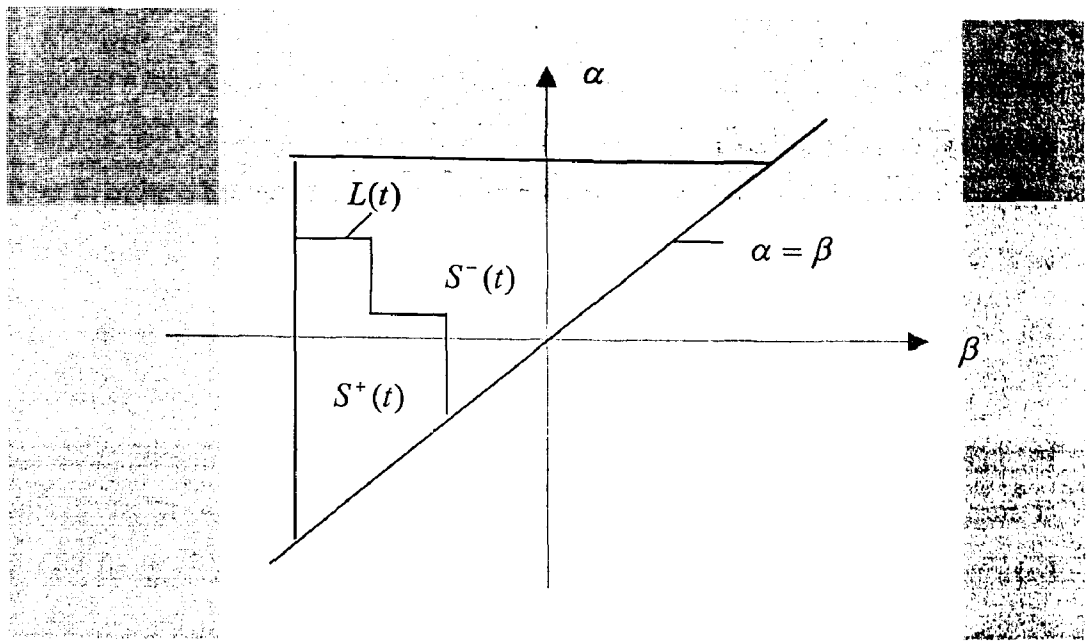

Fig 2.6 A geometric Interpretation of P-K model

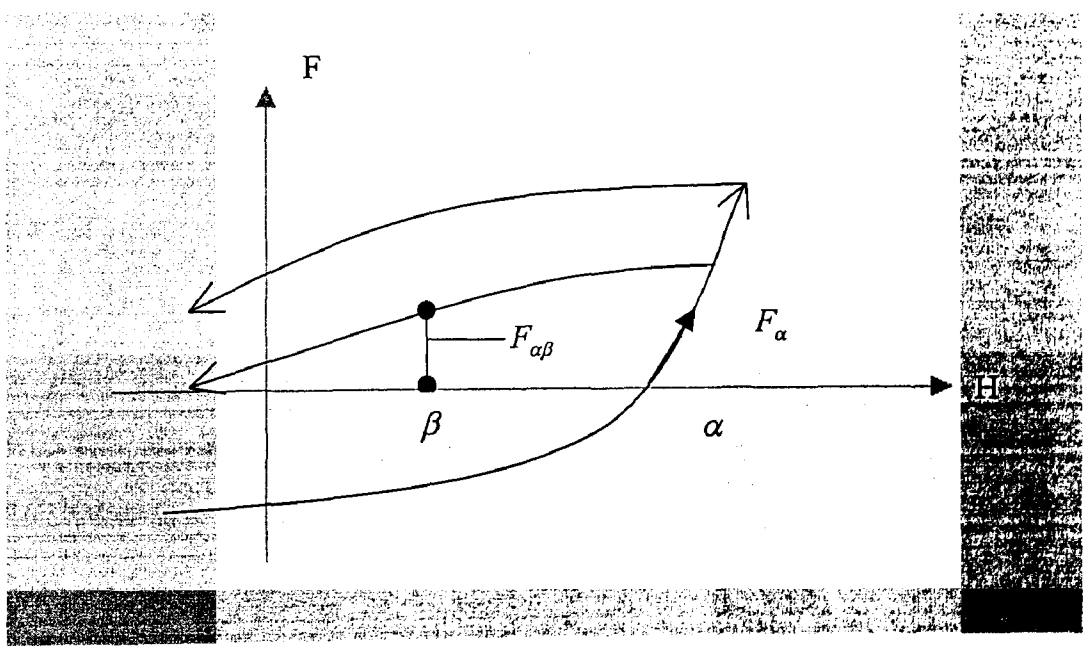

Fig 2.7 First Order Reversal Curves

The unknowns of this model may be systematically found using f-o reversal curves (Fig 2.7). This can only lead to good prediction results. This can be done by first bringing the input value to states such that the outputs of all operators $\hat{\gamma}_{\alpha\beta}$ are equal to -1. If we now gradually increase the input value, it will follow along a limiting ascending branch (Fig 2.7). The notation F_α will be used for the output value on this branch corresponding to the input $H = \alpha$. The first-order reversal curves are attached to the limiting ascending branch. Each of these curves is formed when the above monotonic increase of the input is followed by a subsequent monotonic decrease. The notation $F_{\alpha\beta}$ will be used for the output value on the transition curve attached to the limiting ascending branch at the point F_α. This output value corresponds to the input $H = \beta$. Now we can define the function $F(\alpha, \beta)$:

$$F(\alpha, \beta) = (F_\alpha - F_{\alpha\beta})/2 \qquad (2.5)$$

Using the geometric interpretation of the model (2.3), it is easy to prove that:

$$F(\alpha, \beta) = \iint_{T(\alpha,\beta)} H(x, y)dxdy = \int_\beta^\alpha \left(\int_\beta^y H(x, y)dx \right) dy, \qquad (2.6)$$

where $T(\alpha, \beta)$ is the triangle formed by the intersection of the lines $x = \beta, y = \alpha$ and $y = x$. (Fig 2.8)

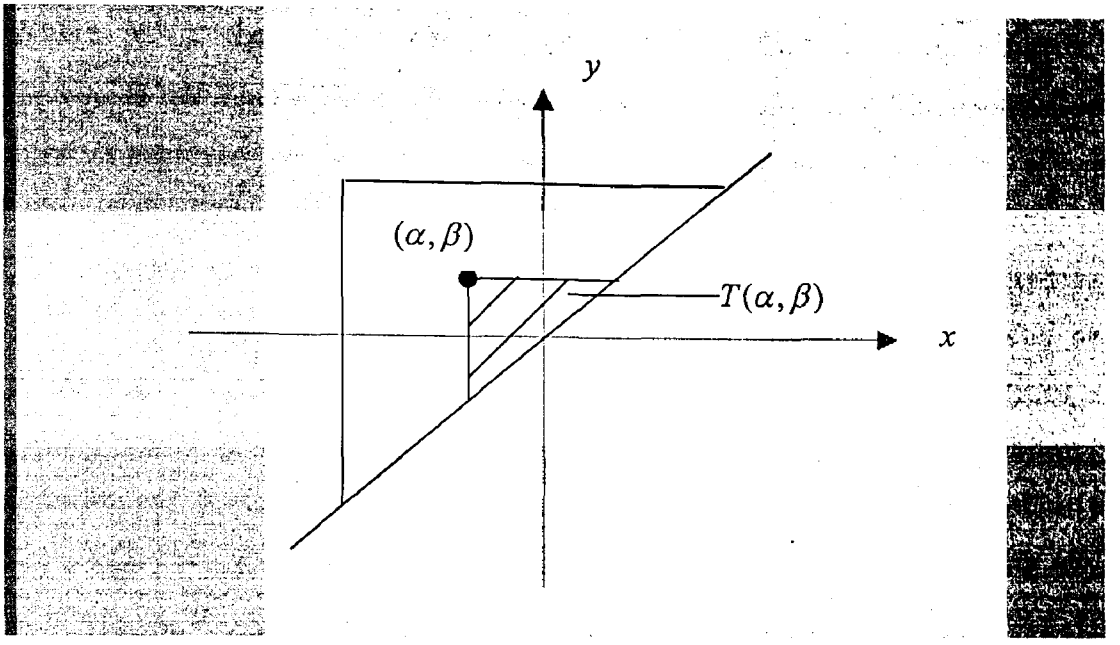

Fig 2.8 Graphical Representation of expression (2.8)

From (2.6), we find:

$$\mu(\alpha,\beta) = -\frac{\partial^2 F(\alpha,\beta)}{\partial\beta\partial\alpha} \tag{2.7}$$

To avoid numerical evaluation of double integrals, It is shown in [4] that the determination of the coefficients of the hysteresis operator can be simplified by approximating (2.3) using a finite superposition of hysteresis operators

$$F(t) \approx \sum_{i=1}^{N}\sum_{j=1}^{N} \mu(\alpha_i,\beta_j)\hat{\gamma}_{\alpha_i\beta_j} H(t) \tag{2.8}$$

$$\alpha_i = \beta_i = \alpha_1 - 2\frac{(i-1)}{(N-1)}\alpha_1 \tag{2.9}$$

Where N^2 is the total number of hysteresis operators involved while α_1 represents the input at which the positive saturation of the actual magnetization curve is achieved.

The Preisach- Krasnoselskii model, as given by (2.8), can be illustrated as shown in Fig 2.9. According to this figure, the model identification can be simplified to finding out the different branch of weights. It then becomes a typical ANN problem which is to find different weights using arbitrary input/output training data. This means if the P-K model can be explicitly realized by an ANN, then the model unknowns may be determined using any arbitrary experimental data.

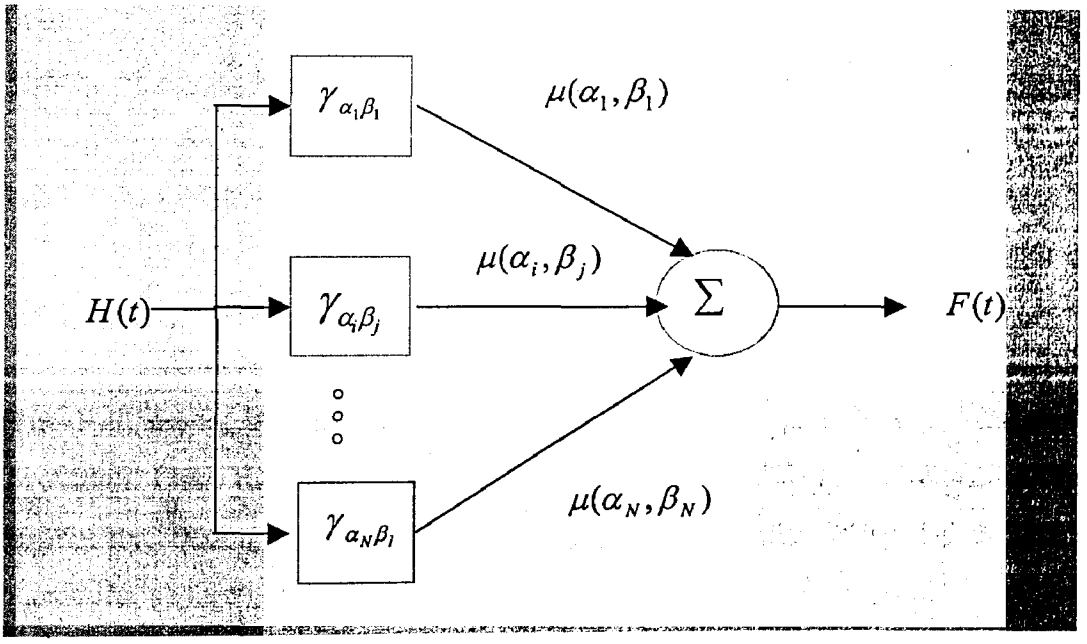

Fig 2.9 Block Representation of the Classical Preisach-Krasnoselskii Model

It turns out that model (2.9) may be able to be realized by the ANN's shown in Fig 2.10 if the hysteresis operators are represented by elementary rectangular loops.

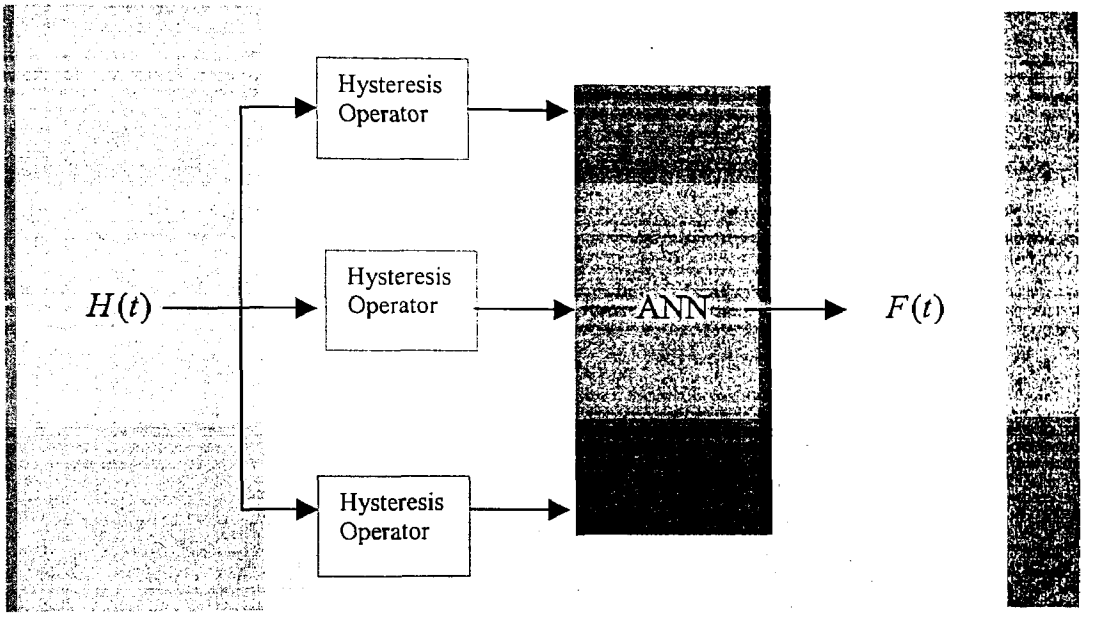

Fig 2.10 Operator ANN Realization of the P-K model

2.4. Summary

The P-K model above is shown to satisfy the formal definition of hysteresis provided. First, the P-K model uses the binary units to form the local memory capability. Secondly, it is shown that this model stores information not in particular separate units, but some groups of the units which together keep each piece of information related to hysteresis. As a result, if some of the units are destroyed, the stored information still might be preserved [1]. This, so called,

fault tolerance property is similar to that of the memories in biological systems, and to the memory capability of an ANN.

Chapter 3

NEURAL NETWORKS

3 Neural Networks

3.1. Neural Networks Structure

Artificial Neural networks, commonly known as "neural networks", can be defined as a model of reasoning inspired by biological neural interaction in the natural world. The human brain consists of a densely interconnected set of nerve cells, neurons. Similarly, a neural network is a computer system made up of a number of very simple and highly interconnected processors, also called neurons, that process information in parallel by that dynamic state response to external inputs [5].

A neuron is the basic information-processing unit of a neural network. A neuron k, from input to output can be described using the following two equations:

$$v_k = \sum_{i=1}^{r} w_{ki} x_i \qquad (3.1)$$

and

$$y_k = \varphi(v_k) \qquad (3.2)$$

where $x_1, x_2, ..., x_r$ are the inputs; $w_{k1}, w_{k2}, ..., w_{kr}$ are the synaptic weights of neuron k ; v_k is the neuron internal activity index; $\varphi(\bullet)$ is the activation function and y_k is the output of the neuron k .

A neuron model can be illustrated as Fig 3.1. The threshold θ_k of neuron k here is represented by a synaptic link with an input. This signifies the scenario where a neuron generates an output if its input is beyond a threshold.

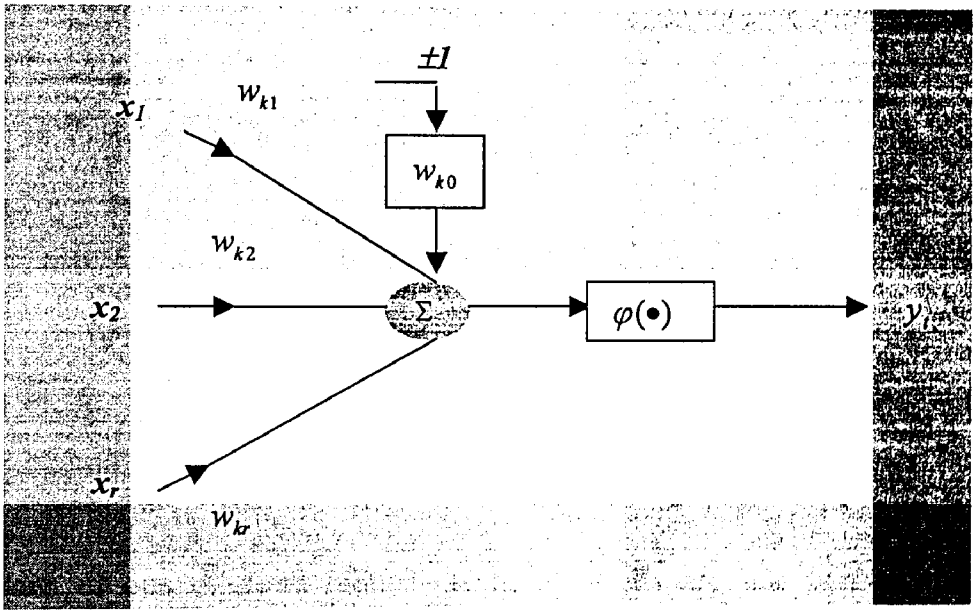

Fig 3.1 Neuron Model

This model is a simple, yet useful, approximation of a biological neuron and can be used to develop different neural structures including feedforward networks.

A typical network consists of an input layer, one or more hidden layer and an output layer of neurons. A multilayer perceptron with one hidden layer is shown in Fig 3.2.

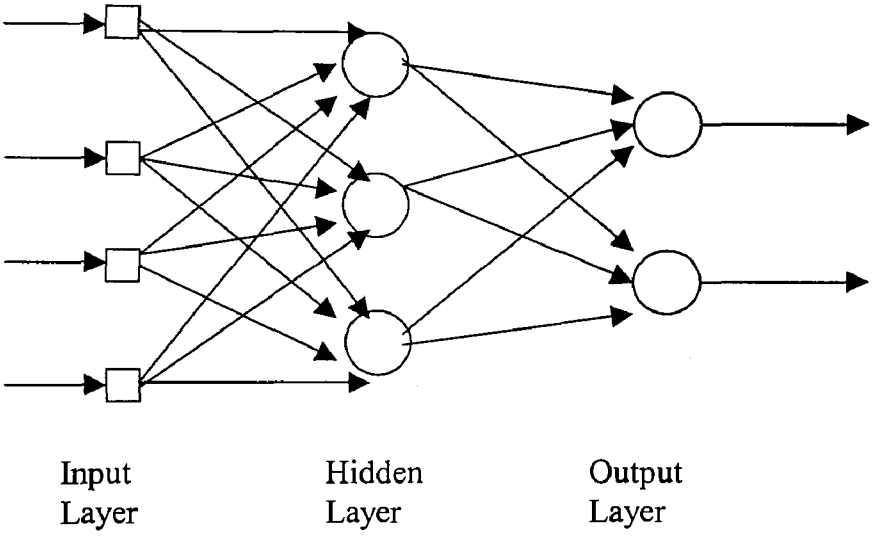

Fig 3.2 Multi-Layer Perceptron with One Hidden Layer

The neurons of one or more hidden layers of MLPs are not part of the input/output of the networks. These hidden neurons provide the learning capability for the networks. It enables the network to learn complex tasks by extracting more meaningful features from the input vectors. The procedure of this method is referred as the error-correction learning law.

3.2. Error-correction Learning Law

To derive the error-correction learning law, let us consider the three-layer network shown in Fig 3.3. The indices i, j and k refer to neurons in the input, hidden and output layers, respectively.

Input signals, $x_1, x_2, ..., x_n$, are propagated through the network from the left to right, and error signals, $e_1, e_2, ..., e_l$, from right to left. The symbol w_{ij} denotes the weight for the connection between neuron i in the input layer and neuron j in the hidden layer, and the symbol w_{jk} the weight between neuron j in the hidden layer and neuron k in the output layer.

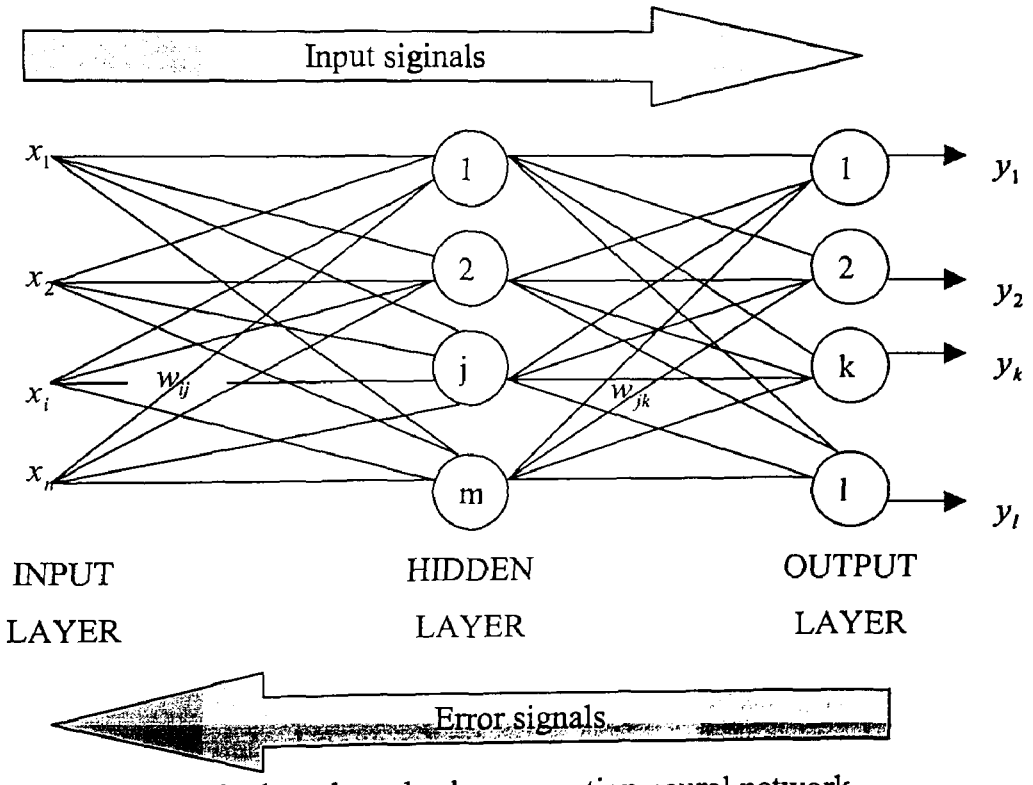

Fig 3.3 Three-layer back-propagation neural network

Suppose the learning procedure of the neural network selects (3.2.1) as the cost function to minimize.

$$\xi = \frac{1}{2}\sum_{k} e_k^2 \qquad (3.2.1)$$

To propagate error signals, we start at the output layer and work backward to the hidden layer. The error signal at the output of neuron k is defined as:

$$e_k = d_k - y_k \qquad (3.2.2)$$

where d_k denotes desired output for neuron k, y_k denotes the actual output of neuron k when the input vector x is presented.

The purpose of error-correction learning is to minimize the cost function with respect to the synaptic weights W of the network so as to make the network closer to the desired output.

This can be achieved by employing the method of gradient descent, also known as the steepest descent method. We can adjust the weight values as following :

$$W^{new} = W^{old} + \eta D \qquad (3.2.3)$$

where η is a positive constant known as rate of learning, which determines the length of step; D is a search guiding direction, determined by the gradient of cost function ξ evaluated at the weight value W^{old} :

$$D = -\frac{\partial \xi}{\partial W^{old}} \qquad (3.2.4)$$

According to the steepest descent method, the adjustment to the weight Δw should be along the negative gradient so we have:

$$\Delta w = -\eta \frac{\partial \xi}{\partial w} \qquad (3.2.5)$$

The relationship is shown in Fig 3.4.

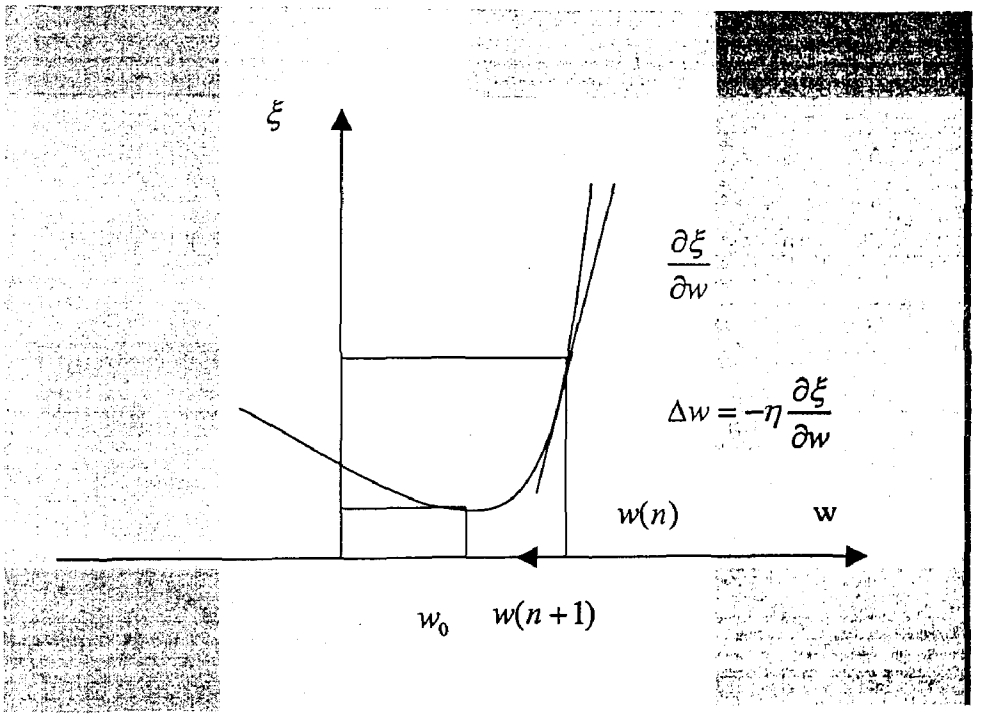

Fig 3.4 Weight Adjustments Using Steepest Descent Method

To calculate $\frac{\partial \xi}{\partial w}$ for (3.2.5), let us consider a sample neuron relation between hidden layer and output layer as Fig 3.5.

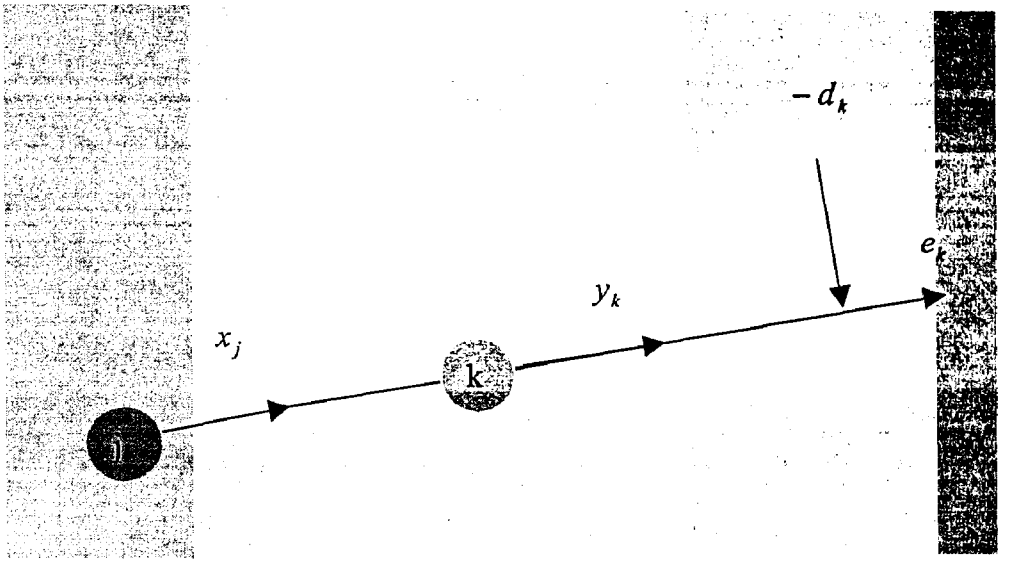

Fig 3.5 A Sample Neuron's Relation Between Hidden Layer And Output Layer

Based on (3.2.5), we set

$$\Delta w_{kj} \cong -\eta \frac{\partial \xi}{\partial w_{kj}} \qquad (3.2.6)$$

The chain rule can be used to calculate the partial derivative and the above partial derivative can be written in the equivalent form:

$$\frac{\partial \xi}{\partial w_{kj}} = \frac{\partial \xi}{\partial v_k}\frac{\partial v_k}{\partial w_{kj}} = \frac{\partial y_k}{\partial v_k}\frac{\partial \xi}{\partial y_k}\frac{\partial v_k}{\partial w_{kj}} \qquad (3.2.7)$$

According to (3.1), we have

$$\frac{\partial v_k}{\partial w_{kj}} = x_j \qquad (3.2.8)$$

According to (3.2), we have

$$\varphi'(v_k) = \frac{\partial y_k}{\partial v_k} \qquad (3.2.9)$$

where $\varphi'(v_k)$ is the derivative of the activation function $\varphi(\bullet)$.

To calculate $\frac{\partial \xi}{\partial y_k}$, two cases have to be considered.

1. k is an output neuron in the output layer of the network:

$$\frac{\partial \xi}{\partial y_k} = -(d_k - y_k) \qquad (3.2.10)$$

then (3.2.6) can be solved as:

$$\Delta w_{kj} \cong -\eta \varphi'(v_k)(d_k - y_k)x_j \qquad (3.2.11)$$

2. k is not an output neuron in the output layer of the network:

This is the case where the output error e_k cannot be used directly in calculating the error signal. The total output error is:

$$\frac{\partial \xi}{\partial y_k} = \sum_l e_l^2 = \sum_l \frac{\partial \xi}{\partial v_l}\frac{\partial v_l}{\partial y_k} = \sum_l \frac{\partial \xi}{\partial v_l}\left(\frac{\partial}{\partial y_k}\left(\sum_l w_{li} y_i\right)\right) = \sum_l \frac{\partial \xi}{\partial v_l} w_{lk} = \delta_k \qquad (3.2.12)$$

where l is the corresponding output layer that neuron k is not in.

then (3.2.6) will be

$$\Delta w_{kj} \cong -\eta \varphi'(v_k) \sum_l \frac{\partial \xi}{\partial v_l} w_{lk} x_j \qquad (3.2.13)$$

and (3.2.7) can be expressed as:

$$\Delta w_{kj} \cong -\eta \delta_k x_j \qquad (3.2.14)$$

According to (3.2.11), when the output layer is lth layer, we can calculate δ_l as:

$$\delta_l = -\varphi'(v_l)(d_l - y_l) \qquad (3.2.15)$$

otherwise, calculating the δ_k recursively as (3.2.12).

The back-propagation learning algorithm for error-correction learning explained above is applicable to any network with a differentiable activation function $\varphi(\bullet)$. According to (3.2.12) and (3.2.15), we can back-propagate the errors layer by layer; according to (3.2.14), we can update the weight values.

Now we can derive the back-propagation training algorithm.

3.3. Error Back-propagation Training

The best known supervised learning algorithm. The learning algorithm was first developed by Werbos [5] and latter rediscovered by Rumelhart et al. [8]. The learning is done on the basis of direct comparison of the output of the network with known correct answers. The back propagation algorithm is an efficient

method of computing the change in each connection weight in a multi-layer network so as to reduce the error in the outputs. The training algorithm consists of the following steps:

Step 1: Initialization

Set all the weights and thereshold levels of the network to random numbers uniformly distributed inside a small range [5]:

$$\left(-\frac{2.4}{F_i},+\frac{2.4}{F_i}\right),$$

where F_i is the total number of inputs of neuron i in the network. The weight initialization is done on a neuron-by-neuron basis.

Step 2: Activation

Activate the back-propagation neural network by applying inputs $x_1, x_2, ..., x_n$ and desired outputs $d_1, d_2, ..., d_n$.

a) Calculate the actual outputs of the neurons in the hidden layer:

$$y_j = sigmoid\left[\sum_{i=1}^{n} x_i \times w_{ij} - \theta_j\right],$$

where n is the number of inputs of neuron j in the hidden layer, and *sigmoid* is the sigmoid activation function.

b) Calculate the actual outputs of the neurons in the output layer:

$$y_k = sigmoid\left[\sum_{j=1}^{m} x_{jk} \times w_{jk} - \theta_k\right],$$

where m is the number of inputs of neuron k in the output layer.

Step 3: Weight Training

Update the weights in the back-propagation network propagating backward the errors associated with output neurons.

a) Calculate the error gradient δ_k for the neurons in the output layer using (3.2.11) or (3.2.12):

Calculate the weight corrections Δw_{kj} using (3.2.14):

Update the weights at the output neurons:

$$w_{kj} = w_{kj} + \Delta w_{kj}$$

b) Calculate the error gradient for the neurons in the hidden layer:

$$\delta_j = y_j \times (1 - y_j) \times \sum_{k=1}^{l} \delta_k \times w_{kj}$$

Calculate the weight corrections:

$$\Delta w_{ij} = -\eta \times x_i \times \delta_j$$

Update the weights at the hidden neurons:

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

Step 4: Iteration

Increase the iteration by one, go back to Step 2 and repeat the process until the selected error criterion is satisfied (Fig 3.6)

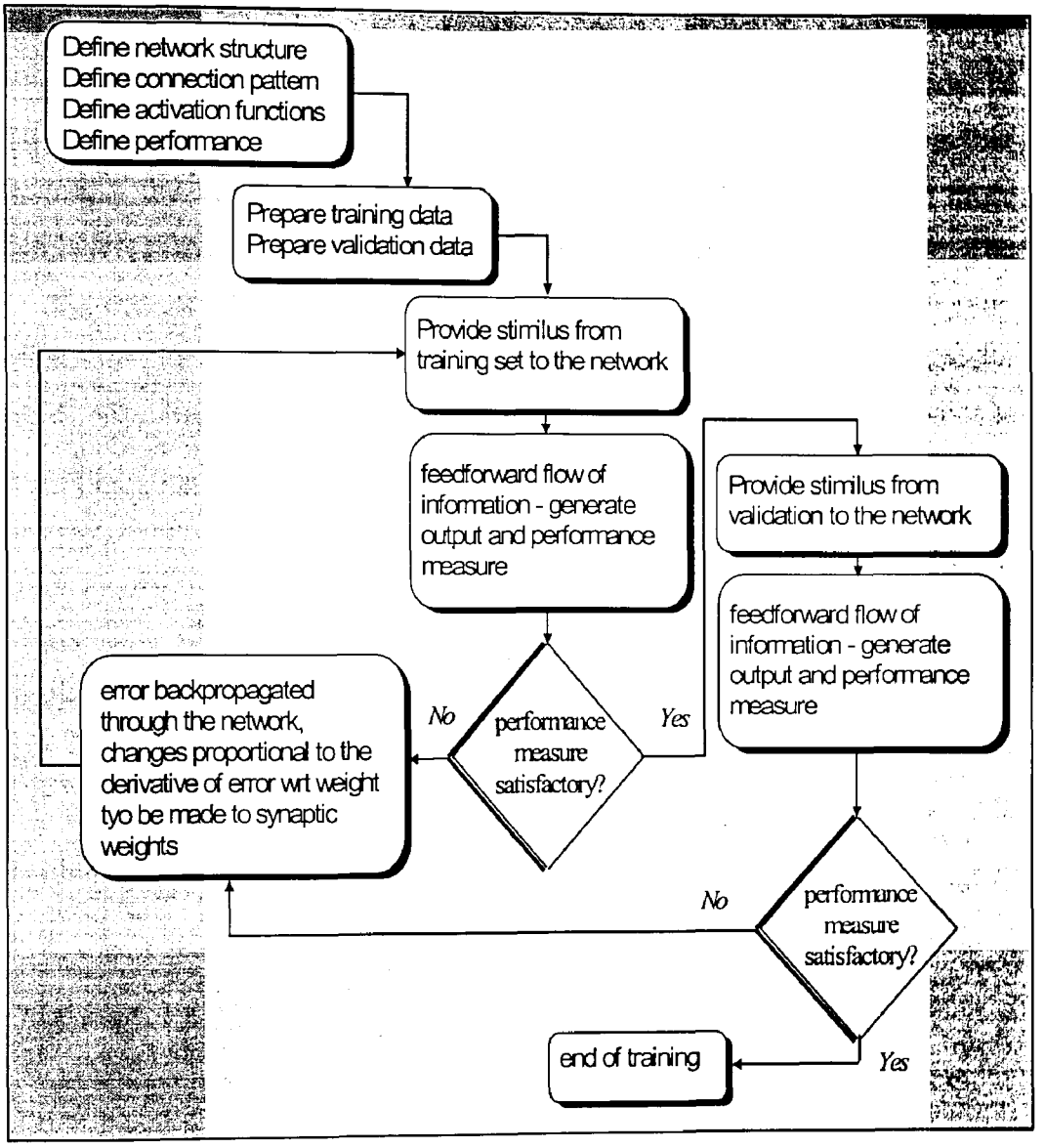

Fig 3.6 Basic Flowchart Presentation of Back-propagation Training Algorithm

3.4. Multi-Layer Perceptron Neural Networks

A multi-layer perceptron network (MLP) represents a class of feedforword neural networks that contain one input layer with one output layer, together representing the system inputs and outputs, respectively, and one or more hidden layers providing the learning capability for the network [7](Fig 3.7). The basic element of a MLP network is an artificial neuron whose activation function, for the hidden layer, is a smooth, differentiable function (usually sigmoid). The neurons in the output layer have a linear activation function. The back-propagation algorithms based on error-correction learning rule can be considered as an extension of the "mean least square" algorithms [10]. This learning algorithm works by continuously presenting batches of input vectors and their associated desired outputs and then iteratively changing the values of the network weights and biases in the direction of the steepest decent with respect to the relative error. One way of determining when to stop is the average error over the entire training set reaches a defined error or any other convergence criterion. A MLP network function can be formulated as:

$$f(x_1,\ldots,x_n)=\sum_{i=1}^{m}\omega_i\cdot g\left(\sum_{j=1}^{n}w_{ij}x_j-\theta_i\right) \qquad (3.2.16)$$

$$g(x)=\frac{1}{1+e^{-x}} \qquad (3.2.17)$$

Where $f(\bullet)$ is the network output; $g(\bullet)$ is the activation function; w_{ij} and θ_i represent weights and biases in the hidden layer; m denotes the number of hidden neurons; and ω_i represents the output layer connection weights which, in effect, serve as coefficients to the linear output function.

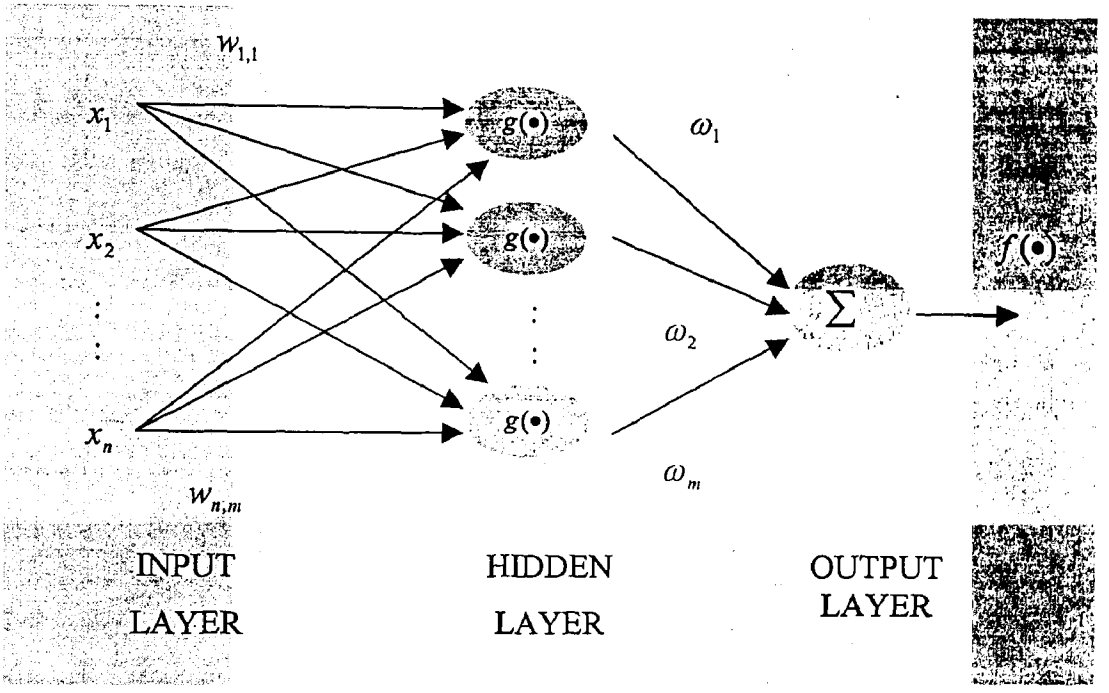

Fig 3.7 Block Representation of the Multi-Layer Perceptron Network

3.5. Radial Basis Function Neural networks

In general, Radial basis function (RBF) network exhibits similar properties as MLP networks, such as generalization ability, fault tolerance and robustness. It therefore can be treated as an alternative tool for learning in neural networks.

RBF networks also have a fast learning capability comparing with other feedforward neural networks [9]. The architecture of the RBF network is similar to a MLP network and consists of a fully connected two-layer network; the hidden layer and the output layer. The hidden layer is a collection of neurons with radial basis activation functions, typically one of "Thin Plate Spline", Gaussian, Multi-Quadratic or Inverse Multi-Quadratic functions [9]. There are two parameters, the center c and the width, r, work with RBF neurons. They are similar to the weights and biases of the hidden layer in a MLP network. The output layer is just an "adder" that sums a weighted inputs using the output connection weight, w, that represent the strength of these connections (Fig 3.8) A RBF network function can be formulated as:

$$f(x_1,...,x_n) = \sum_{i=1}^{m} w_i \cdot \phi\left(\left\|x - c_i\right\|\right) \qquad (3.2.18)$$

$$\phi(x-c) = e^{\left(-\frac{(x-c)^2}{r^2}\right)} \qquad (3.2.19)$$

where $\phi(\bullet)$ is the activation function; w_i represents the weights in the output layer.

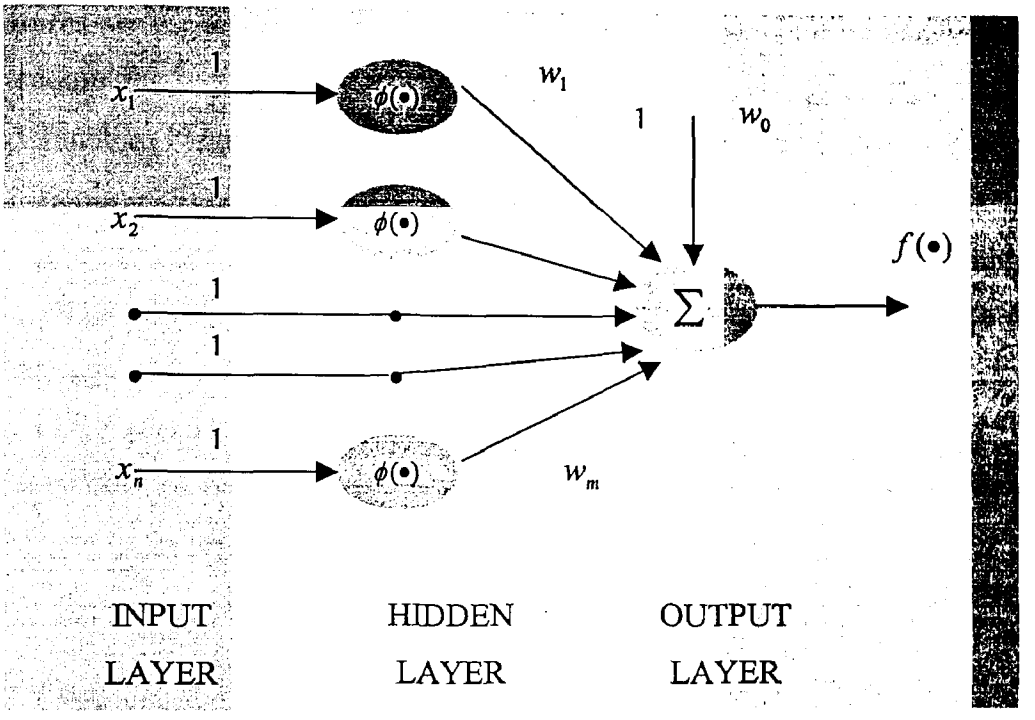

Fig 3.8 Block Representation of the Radial Basis Function Network

3.6. Summary

According to the review, a neural network is massively parallel & distributed processor that works in a fashion similar to the human brain. It resembles a biological brain in two respects; first, the knowledge is acquired through a learning process, & second, inter-neuron connection strengths known as weights are used to store the knowledge. The learning process involves modification of the connection weights to obtain a desired objective. Two kinds

of neural networks that we specifically mentioned are MLP and RBF networks. They are all feedforward neural networks. Feedforward networks can be considered as linear association networks that relate output patterns to input patterns. Among the advantage of feedforward neural networks, the following features are especially important to the proposed implementation of Hysteresis phenomenon:

1) function approximation (I/O mapping): ability to approximate any nonlinear function to the desired degree
2) Learning and generalization: ability to learn I/O patterns, extract the hidden relationship among presented data, & provide acceptable response to new data that the network has not yet experienced.
3) Fault tolerance: due to their highly parallel/distributed structure, failure of a number of neurons to generate the correct response does not necessarily lead to failure of the overall performance of the system.

Chapter 4

RECONSTRUCTION OF P-K HYSTERESIS MODEL

4 Reconstruction of P-K Hysteresis Model

In this paper, two major kinds of neural network architectures, namely multi-Layer Perceptron (MLP) and the Radial Basis Function (RBF) networks, are proposed to model the Electric Arc Furnace (EAF). The reconstruction of the P-K Model is achieved using the normalized data representing the hysteresis loops for Square Permalloy 80 at different frequency including DC, 1 kHz, 3 kHz, and 6 kHz. Square Permalloy 80 is materials consists of 80/20 alloy of nickel and iron that can be easily magnetized and demagnetized. The performance criteria by which neural network-based models can usually be measured including the training time, number of epochs, size of the network and accuracy. The first three criteria measurements are straightforward except the accuracy. In order to investigate the accuracy of implemented neural networks, comparisons between experimental data and the output of the networks are made. The error index used for the comparison is non-dimensional index error (NDIE) that is the mean square error (MSE) of hysteresis modeling normalized by the standard deviation of the experimental data.

4.1. Simulation and Experimental Results of MLP Neural Networks

4.1.1. Simulation set up

The EAF sample used for simulation experiments is illustrated as Fig 4.1.

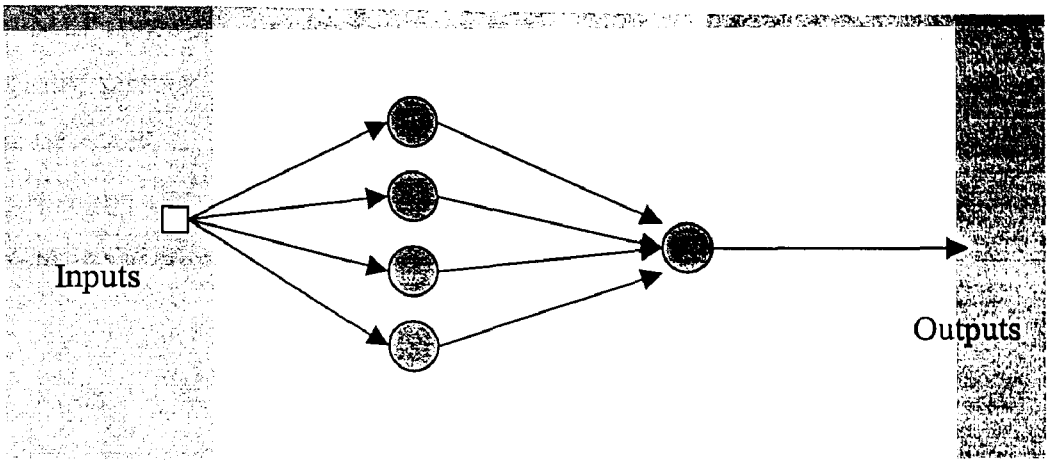

Fig 4.1 Hysteresis Loop Sample for Simulation Experiments

The dimension of the input vector and output vector is 1. The numbers of hidden layer neurons will vary. The corresponding simulation software program is coded in Matlab 6.5. The computer system used for the implementation is Pentium® 4 CPU 2.4 GHz, 512 MB of RAM.

The configuration of the hysteresis MLP is controlled by a script file. The configuration script file controls the parameters such as the dimension of the input and output vectors, the dimension of the hidden layer, the learning stop condition and learning rate etc. The activation function is a sigmoid function. The training algorithm is the Levenberg-Marquardt algorithm that was designed to approach second-order training speed without having to compute the Hessian matrix [17]. The weight and bias are all set to an identical initial value before

training. At the same time the inputs are normalized to between 0 and 1 so that the neural network will be sensitive to the input pattern.

The MLP network implementation is the matter of finding the following matching factors that attributed to failure in learning: 1) adequate numbers of training cases; 2) sufficient numbers of hidden neurons; 3) non-deterministic relationships between inputs and outputs. The second one is of particular interest in the simulation.

4.1.2. Simulation Experiments of MLP networks

Simulation experiments are carried out to show the effect of different numbers of hidden neurons to the speed of learning and accuracy.
The configuration parameters for the rest of the experiments in this section are listed as Table 4.1.

Table 4.1 Configuration Parameters for the MLP Experiments

Weight Initial Value	Dimension of Input Vector	Dimension of Output Vector	Learning Rate	Learning Stop Condition	Maximum No. of Epochs
1	1	1	0.05	.0001	300

The first experiment is to try to use a set of data in single frequency with 51 training cases and 51 validation data at the frequency of 3 kHz. Numbers of MLP networks with different numbers of hidden neurons have been

implemented and their NDIE, MSE, training time, and No of Epochs have been recorded in the table 4.2. The experiment shows that the error indices (NDIE) vary from as low as 0.33% to as high as 0.64% with an average NDIE of 0.2327%, and the computational results compared well with existing measurements. Based on the simulation results, we can see that the optimum number of neurons in this experiment is 20. Fig 4.2(b) depicts the hysteresis loops at frequency of 3 kHz generated by the modified MLP networks.

Table 4.2. MLP Network Results Using Square Permalloy 80 at Frequency of 3 kHz

Size of Networks	Training Time (Second)	No of Epochs	NDIE (%)	MSE (10^{-4})
10	2.1560	75	0.34	0.998305
15	0.8750	66	0.33	0.92339
20	0.4380	14	0.32	0.86197
21	0.5780	27	0.28	0.669531
22	0.5470	23	0.34	0.999437
23	0.4690	16	0.25	0.563349
24	0.5000	17	0.31	0.815577
25	0.5630	21	0.31	0.860202
26	0.7350	23	0.33	0.930848
30	0.7030	17	0.28	0.788511
35	0.5620	12	0.19	0.322733
40	0.7500	12	0.32	0.693552

45	0.7180	10	0.24	0.480072
50	0.9370	11	0.24	0.469641
55	0.9840	11	0.55	0.297932
60	1.0150	8	0.33	0.97101
65	1.0470	7	0.16	0.23044
70	1.0320	7	0.27	0.647826
75	0.8120	6	0.16	0.231463
80	1.2500	7	0.29	0.746818
85	0.9220	5	0.092153	0.0736879
90	1.4060	6	0.065053	0.0367206
95	0.9850	5	0.074806	0.048557
100	0.9220	4	0.095903	0.0098066
105	1.0000	5	0.018202	0.00287469
110	1.0930	6	0.057863	0.0290517
115	1.3440	5	0.037592	0.0122623

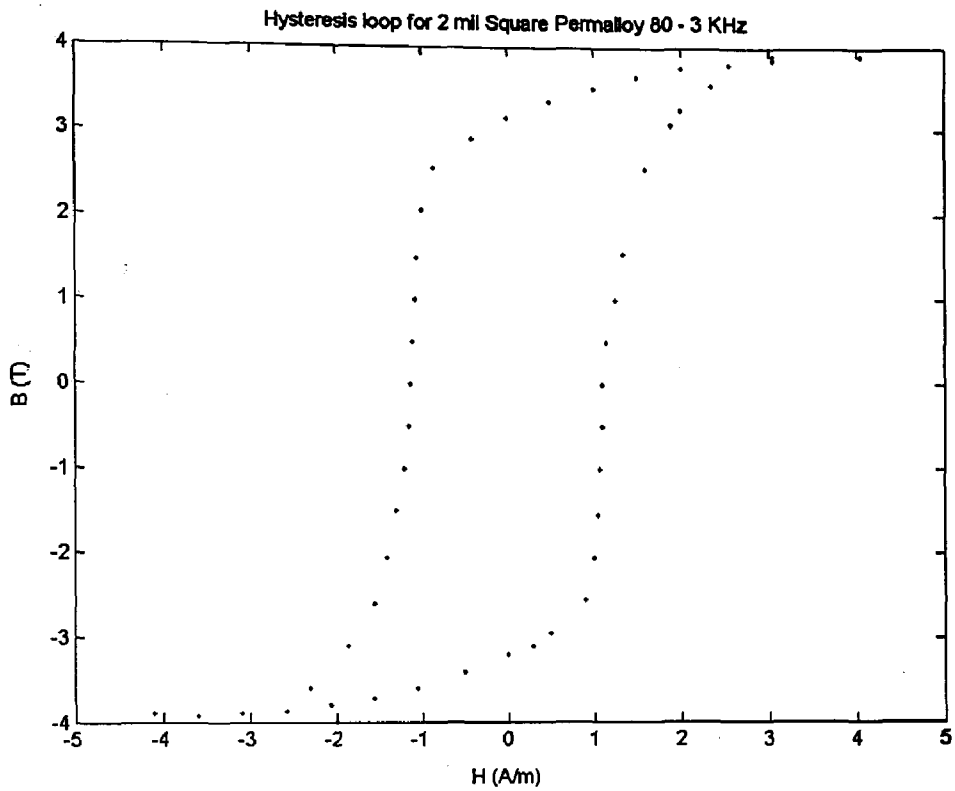

Fig 4.2(a) Experimental data illustrating normalized hysteresis loops for Square Permalloy 80 at frequency of 3 kHz

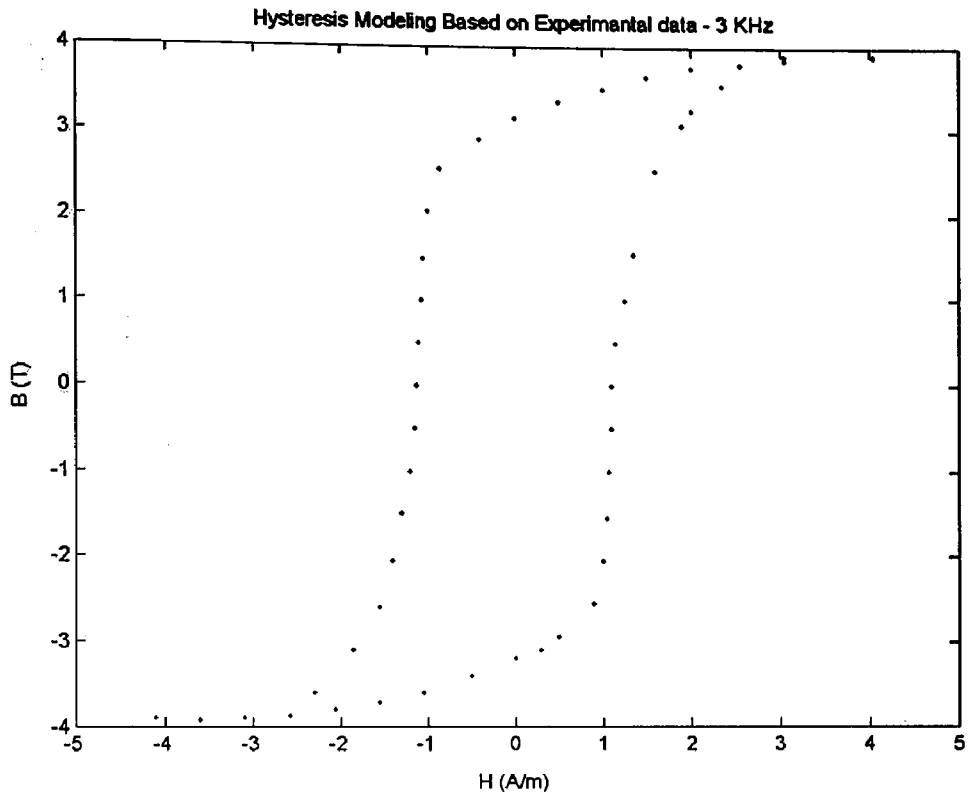

Fig 4.2(b) Results from MLP Neural network-based hysteresis model built based on P-K Model

The Second experiment is to try to use the data of different frequency with 204 training cases and 204 validation data at different frequency specifically DC, 1 kHz, 3 kHz, and 6 kHz. Numbers of MLP networks with different numbers of hidden neurons have been implemented and their NDIE, MSE, training time, and No of Epochs have been recorded in the Table 4.3. The experiment results shows that the error indices (NDIE) vary from as low as 0.29% to as high as 0.34% with an average NDIE of 0.3615%, and the computational results compared well with existing measurements. Based on the simulation results, we can see that the optimum number of neurons in this experiment is 49. Fig 4.3(b)

depicts the hysteresis loops at different frequency (DC, 1 kHz, 3 kHz, 6 kHz) generated by the modified MLP networks.

Table 4.3. MLP Network Results Using Square Permalloy 80 at Different Frequency (DC, 1 kHz, 3 kHz, 6 kHz)

No. of Hidden Neurons	Training Time (s)	No. of Epochs	NDEI (%)	MSE (10^{-4})
40	15.8280	300	0.64	3.50616
45	19.3130	300	0.35	1.00509
46	12.4680	184	0.33	0.926991
47	11.125	158	0.34	0.988855
48	9.5310	132	0.33	0.944316
49	9.0310	122	0.34	0.962414
50	12.9840	167	0.34	0.988385
55	16.7970	166	0.34	0.872004
60	20.8750	172	0.34	0.990783
70	35.0470	211	0.34	0.966097
80	38.7810	180	0.33	0.930218
90	34.9370	127	0.34	0.956249
100	61.9220	180	0.34	0.998783

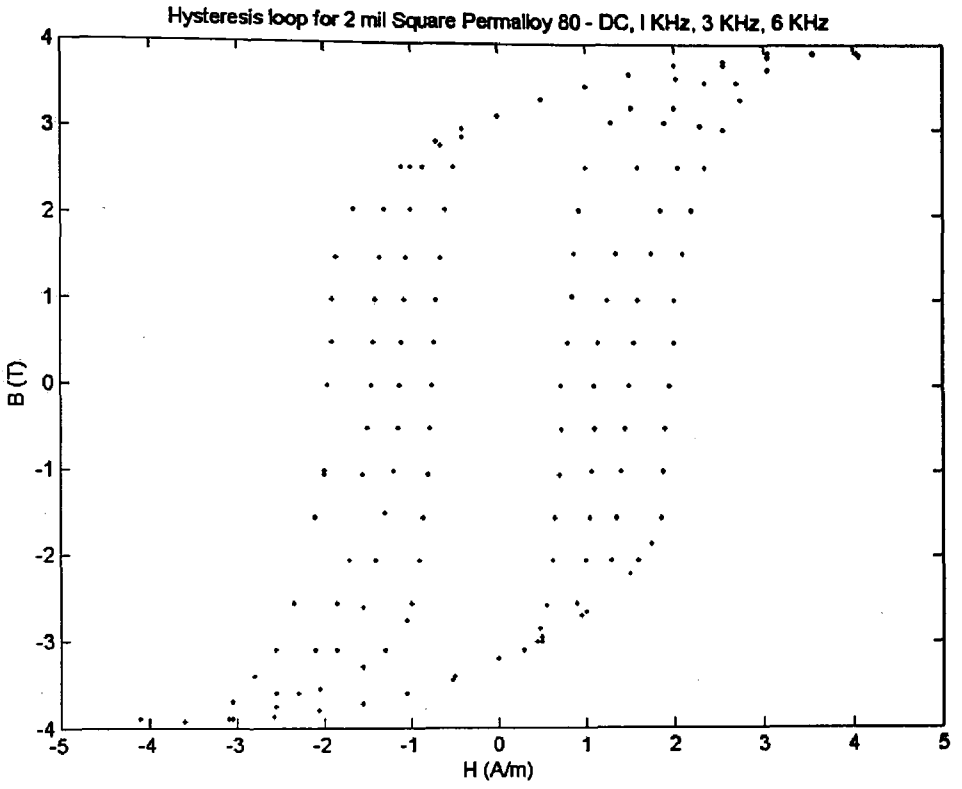

Fig 4.3(a) Experimental data illustrating normalized hysteresis loops for Square Permalloy 80 at different frequency

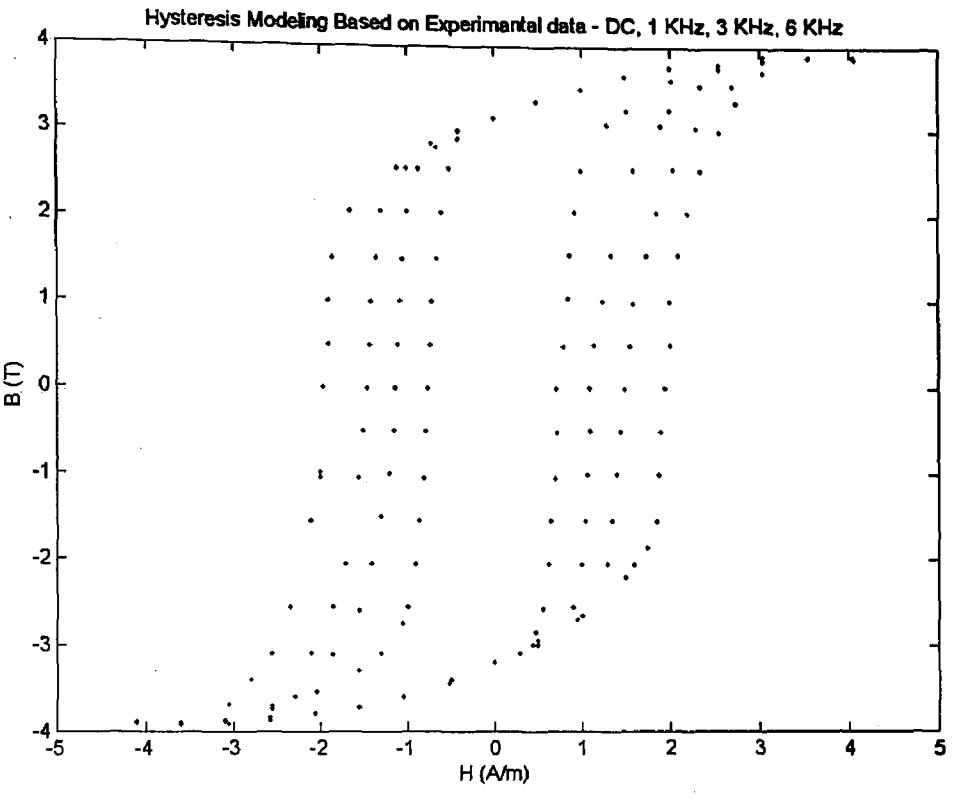

Fig 4.3(b) Results from MLP Neural network-based hysteresis model built based on P-K Model

4.2.Simulation and Experimental Results of RBF Neural Networks

4.2.1. Simulation Set up

The EAF sample used for simulation experiments is illustrated as Figure 4.1.

The dimension of the input vector and output vector is 1. The configuration of the hysteresis RBF is controlled by a script file. The configuration script file controls the parameters such as the dimension of the input and output vectors,

maximum number of neurons, the learning stop condition and spread constant etc. The weight and bias are all set to an identical initial value before training. At the same time the inputs are normalized to between 0 and 1 so that the neural network will be sensitive to the input pattern.

The RBF network implementation is a matter of finding the proper radial basis function and the appropriate learning algorithm. It has been shown theoretically and practically, that the performance of the RBF network does not depend on the choice of the function [8] and hence, Gaussian functions are usually chosen. Therefore, the learning problem is to select adequate number of RBF neurons, function centres and widths. In this paper, the learning was implemented using the orthogonal least squares algorithm that provides a systematic approach to the selection of RBF centres while using forward regression procedure [10].

4.2.2. Simulation Experiments of RBF networks

Simulation experiments are carried out to show the effect of different spread constants to the speed of learning and accuracy. The configuration parameters for the rest of experiments in this section are listed as Table 4.4.

Table 4.4 Configuration Parameters for the RBF experiments

Weight Initial Value	Dimension of Input Vector	Dimension of Output Vector	Learning Stop Condition	Maximum No. of Neurons
0.5	1	1	.0001	300

The first experiment is to try to use a set of data in single frequency with 51 training cases and 51 validation data at the frequency of 3 kHz. A Number of RBF networks have been implemented and their NDEI have been recorded in Table 4.5. The error indices vary from as low as 1.9048×10^{-16} to as high as 4.4714×10^{-4} with an average NDIE of 1.5592×10^{-4} and the computational results compared well with existing measurements. Based on the simulation results, we can see that the optimum number of neurons in this experiment is 49. Fig 4.4(b) depicts the hysteresis loops at frequency of 3 kHz generated by the modified RBF networks.

Table 4.5. RBF Network Results Using Square Permalloy 80 at Frequency of 3 kHz

Spread Constant	No of Neurons	Training Time (s)	NDIE (10^{-4})	MSE (10^{-4})
0.1	49	1.2810	2.6724×10^{-6}	3.1446×10^{-13}
0.5	50	1.0940	1.9048×10^{-12}	1.59758×10^{-25}
0.8	49	1.0780	0.5070	0.0113751
1	48	0.9370	3.9509	0.69077

1.2	48	1.0620	2.8253	0.35325
1.5	48	1.0150	4.4714	0.884753
2	49	1.0780	2.2781	0.229666
2.5	50	1.0150	5.3465×10^{-8}	1.265×10^{-16}
3	50	1	3.4471×10^{-7}	5.25825×10^{-15}

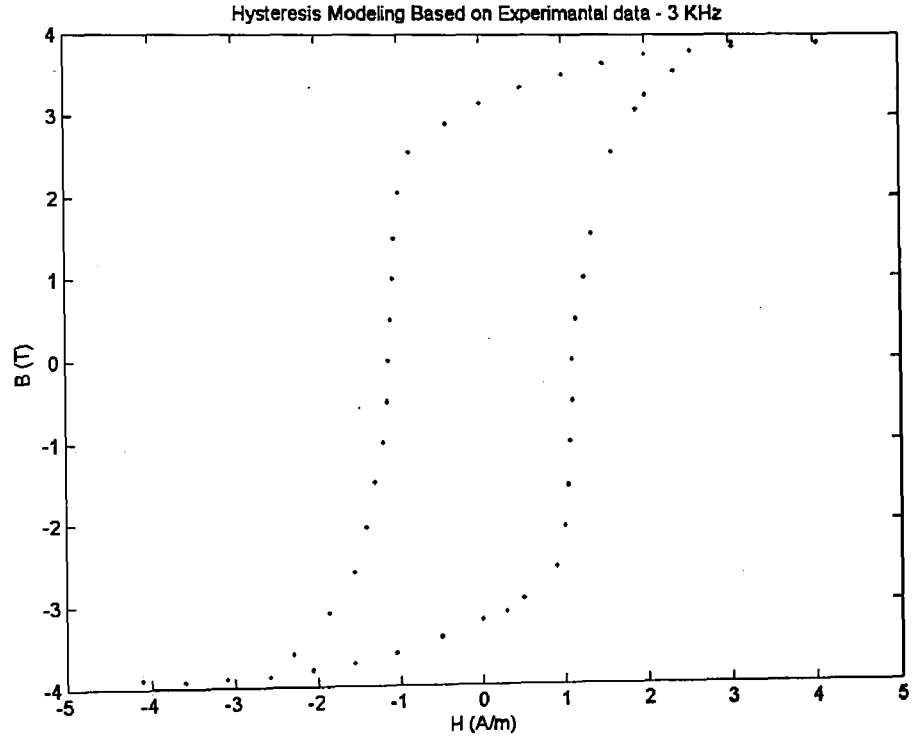

Fig 4.4(a) Experimental data illustrating normalized hysteresis loops for Square Permalloy 80 at frequency of 3 kHz

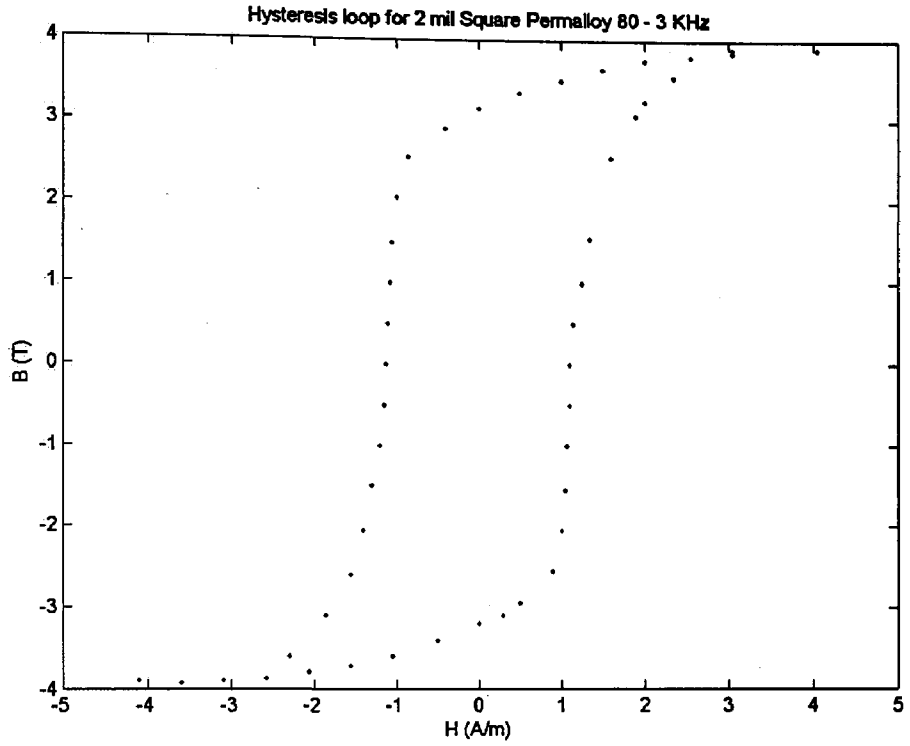

Fig 4.4(b) Results from RBF Neural network-based hysteresis model built based on P-K Model

The Second experiment is try to use the data in different frequency with 204 training cases and 204 validation data at different frequency specifically DC, 1 kHz, 3 kHz, and 6 kHz. Numbers of RBF networks have been implemented and their NDIE, MSE, training time, and No of Epochs have been recorded in the table 4.6. The experiment shows that the error indices (NDIE) vary from as low as 0.00019686 to as high as 0.00022638 with an average NDIE of 0.00020456, and the computational results compared well with existing

measurements. Based on the simulation results, we can see that the optimum number of neurons in this experiment is 154. Fig 4.5(b) depicts the hysteresis loops at different frequency (DC, 1 kHz, 3 kHz, 6 kHz) generated by the modified RBF networks.

Table 4.6. RBF Network Results Using Square Permalloy 80 at Different Frequency (DC, 1 kHz, 3 kHz, 6 kHz)

Spread Constant	No of Neurons	Training Time (s)	NDIE (10^{-4})	MSE (10^{-4})
0.05	154	9.1720	1.9686	0.666667
0.1	154	9.4380	1.9686	0.666667
0.15	154	9.7810	2.1164	0.770521
0.2	157	9.3120	2.1458	0.79207
0.3	159	9.0310	2.2595	0.878263
0.4	161	10.3910	2.2638	0.881554
0.5	160	10.3290	1.9820	0.675763
1	161	10.5470	1.9686	0.666667
1.5	161	10.8130	1.9686	0.666667
2	163	10.6720	1.9686	0.666667
2.5	162	10.6570	1.9686	0.666667
3	162	10.9060	1.9686	0.666835

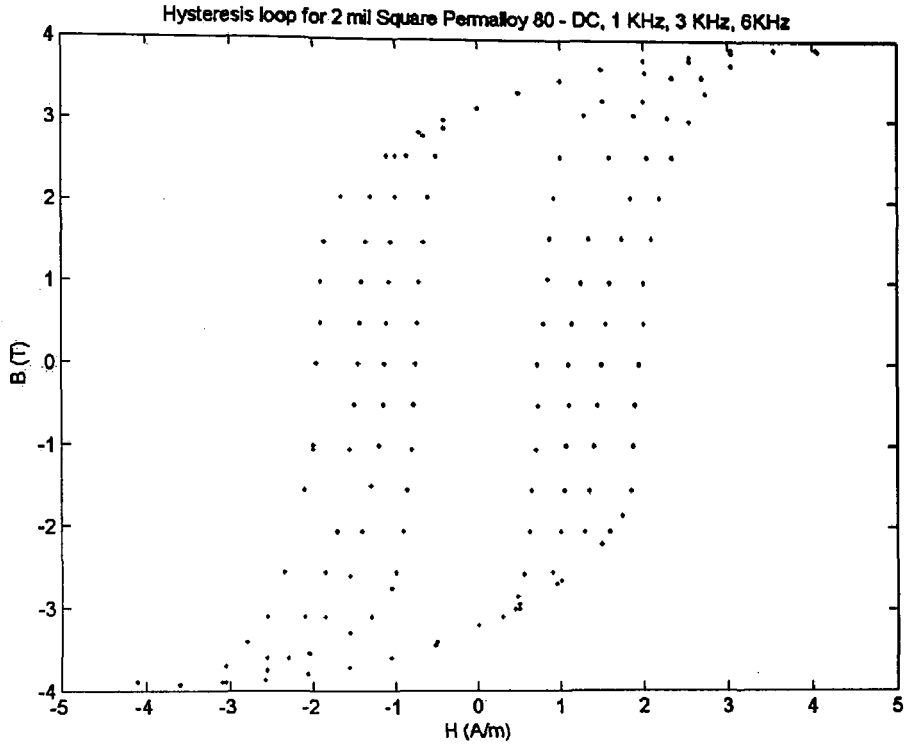

Fig 4.5(a). Experimental data illustrating normalized hysteresis loops for Square Permalloy 80 at different frequency

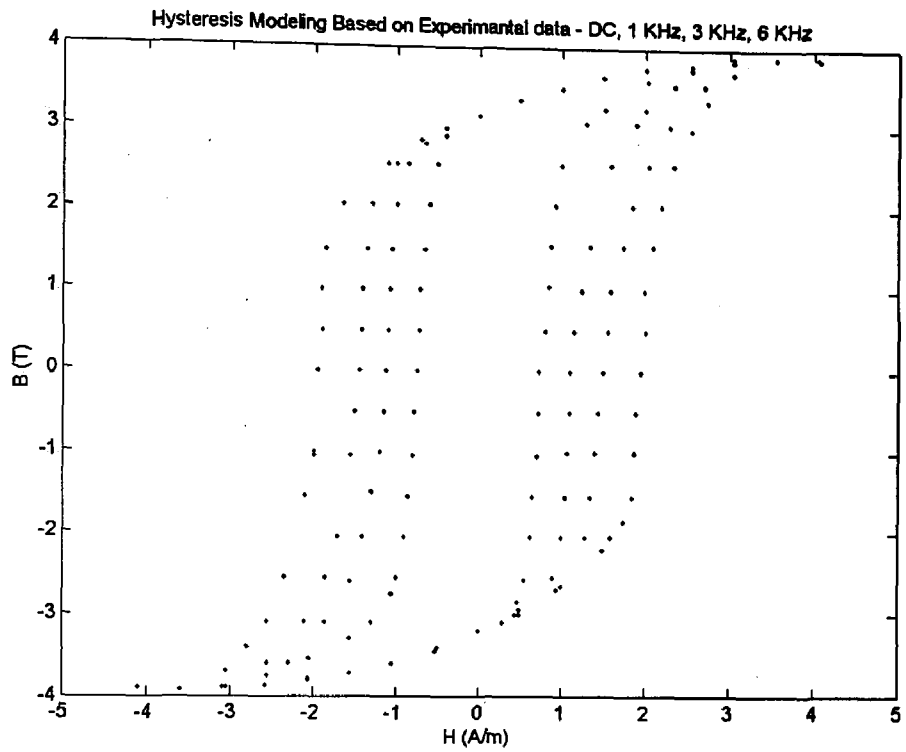

Fig 4.5(b) Results from RBF Neural network-based hysteresis model built based on P-K Model

4.3. Summary

The simulation experiments provide illustration to verify the potential possibility to apply neural networks for reconstruction of P-K based hysteresis phenomena. The project contributes to the reconstruction of P-K models based on both MLP networks and RBF networks. The results of simulations show that both MLP and RBF neural networks capable of modeling P-K based hysteresis phenomena. Based on a qualitative comparisons of the RBF-based and MLP

based results and experimental data. It was observed that the computational results compared well with existing measurements. The numbers of neurons in hidden layers helps to guarantee the training time and accuracy of modeling procedure so that design of devices with hysteresis features can be obtained accurately without numerical evaluations.

Chapter 5

CONCLUSION

In the project, the source of the problem is to consider the potential possibility of applying neural networks for the reconstruction of P-K based hysteresis phenomena. The review of both P-K based hysteresis and neural networks demonstrates the feasibility of modeling P-K based hysteresis using neural networks. **The contribution of this project** is to provide an interpretation of the Preisch-Krasnoselskii model that can be readily implemented by means of neural network methodology. The analysis of the simulation shows that the number of hidden layer neurons plays an important role in the design. It directly effects the training time and accuracy of the modeling. The results of the simulation show that MLP and RBF neural networks are both capable of modeling P-K based hysteresis phenomena.

In summary, based on the formal definition of the P-K based hysteresis provided, the P-K based hysteresis model has several similarities shared with artificial neural network characteristics for reconstruction of P-K model. First of all, the proposed MLP and RBF neural networks both have ability to approximate any nonlinear function to the desired degree, to learn I/O patterns, extract the hidden relationship among presented data, and provide acceptable response to new data that the network has not yet experienced. This enables neural networks to provide models based on imprecise information. Secondly, they both have local memory, and due to their highly parallel/distributed

structure, failure of a number of neurons to generate the correct response does not lead to failure of the overall performance of the system. In fact there are many magnetic field applications that will consider the proposed neural network practical. Every time the design of electronic devices is applied, the proposed neural networks could be helpful.

Further research work about applying different type of artificial neural networks in the magnetic related fields will provide more valuable information for the design of electronic devices with hysteresis features. Furthermore, we can consider the following approach:

1) Add noise to the validation data set as the training set remains unchanged.
2) Use Leave-one-out (Jack-Knife) method in the training and validation.

REFERENCES

[1] I.D. Mayergoyez, *Mathematical Model of.Hysteresis*, Springer-Verlag, New York, 1991

[2] G. Bertotti, *Hysteresis in Magnetism*, Academic Press, San Diego, 1998

[3] I.D. Mayergoyez, "Mathematical model of hysteresis," *IEEE Trans. On Magnetics*, vol. 22, no. 5, pp. 603-608, Sept. 1986

[4] A.A. Adly and S.K. Abd-El-Hafiz, "Using Neural Networks in the Identification of Preisach-Type Hysteresis Models," *IEEE Trans. On Magnetics*, vol. 34, no. 3, pp. 629-635, 1998

[5] Simon Haykin, *Neural Networks, A Comprehensive Foundation.* Macmillan College Publishing Company, 1998

[6] A.R. Sadeghian, "An Interpretation of Preisach-Krasnoselskii Hysteresis Model With the Use of Artificial Neural Networks," *IEEE International, Magnetic Conf. INTERMAG Europe 2002*, 28 Apr.-2 May 2002

[7] K. Hornik, M. Stinchcombe, and H. White, "Multi-layer Feedforward Networks Are Universal Approximator," *Neural Networks*, Vol. 2, pp. 359-366, 1989

[8] D.E. Rumelhart, G.E. Hinton, and R. J. Williams, *Learning Internal representations by Error Backpropagation, Parallel Data Processing*, Vol. 1, Cambridge, MA: MIT Press, 1986

[9] M.J.D. Powell, "Radial Basis Function Approximations for Polynomials," *Proc. 12th Biennial Numerical Analysis Conf.*, pp. 223-241, Dundee, 1987

[10] S Chen, C. F.N. Cowan, and P.M. Garnt, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Trans. On Neural Networks*, Vol. 2, no. 2, pp.302-309, March 1991

[11] H.H. Saliah, D.A. Lowther, and B. Forghani, "Modeling Magnetic Materials Using Artificial Neural Networks," *IEEE Tran. On Magnetics*, Vol. 34, no. 5, pp. 3056-3059, September 1998

[12] Silvano Cincotti, Michele Marchesi, and Antonino Serri, "NN Models of Hysteretic Non-linear Electromagnetic Devices Under Voltage and Current Supply," *IEEE Trans. on Magnetics*, Vol. 35, no. 3, pp.1235-1238, May1999

[13] D. Lederer, H., Igarashi, A. Kost, and T. Honma, " On the Parameter Identification and Application of the Jiles-Atherton Hysteresis Model for Numerical Modeling of Measured Characteristics," *IEEE Trans. On Magnetics*, Vol. 35, no.3, pp.1211-1214, May 1999

[14] Jun Ho Lee, Dong Deok Hyun, "Hysteresis Analysis for the Permanent Magnet Assisted Synchronous Reluctance Motor By Coupled FEM & Preisach Modeling," *IEEE Trans. On Magnetics*, Vol. 35, no. 3, pp. 1203-1206, May 1999

[15] Silvano Cincotti, Ivano Daneri, " A Non-linear Circuit Model of Hysteresis," *IEEE Trans. On Magnetics*, Vol 35, no. 3, pp. 1247-1250, May 1999

[16] D.A. Lowther , E.M. Freeman, C.R.I. Emson, and C.M. Saldanha, "Knowledge Based Systems Applied to the Design of Electrical Devices," *IEEE Trans. On Magnetics*, Vol 24, no. 6, pp. 2576-2578, November, 1988

[17] M.T. Hagan, H.B. Demuth, and M.H. Beale, *Neural Network Design*, PWS Publishing Company, Boston, MA 1996