Magnetic Material Property Identification Using Neural Networks

Hamadou H. Saliah and David A. Lowther
CADLab
Electrical Engineering Department
McGill University
Montreal, Quebec, CANADA

Abstract- **In the numerical solution of field problems for many devices, the modeling of the hysteresis properties of a material is extremely important. While considerable work has been published on models for handling this behavior, it is still difficult to obtain the parameters for these models from measured data. This paper proposes the use a neural network for this operation.**

I. Introduction

The design of an electromagnetic device invariably uses the properties of magnetic materials in order to guide and shape the field so 2 that the desired output effects are obtained. It is obvious that the materials and their properties are critical to the correct operation of the system and thus any attempt to analyze the behavior of such a system must contain an accurate model of the material properties. Until recently, most low frequency devices have been constructed using soft magnetic materials, i.e. ones in which the relationship between the magnetic flux density (B) and the magnetic field (H) is largely single valued. In such devices, the material is used largely to provide an easy flux path for thus both containing the field and reducing the cost of generating it. However, advances in material technologies coupled with a need to create sources of magnetic field have resulted in more and more practical devices using hard magnetic materials, i.e. ones in which there is a deliberate attempt to retain magnetization information. Such materials exhibit hysteresis and have a relationship between B and H (or M and H) which is multivalued and dependent on previous history. Thus it is becoming important that computationally efficient systems are developed for modeling such properties.

Currently, the Preisach Model seems to be one of the most practical for solving this problem based on its requirement for computer memory and time [1],[2]. The model, as are most of the others, is basically phenomenological and does not claim to be accurately representing the physics at a micro level. Rather, it can be used to develop material models which appear to have the same macro responses to a magnetic field stimulus as the real material However, before such a model can be used, the necessary parameters must be identified from the real material behavior. Currently, the approaches to solving this problem are largely empirical and, thus, problem dependent. In this paper, the use of the neural network paradigm is proposed in order to overcome these difficulties. Two different forms of artificial neural networks are proposed; the first uses Radial Basis Functions, while the second employs a CMAC (a form of associative memory) [3]. The combination of these two structures appears to provide both the generalization properties which are expected from a neural network and an improvement in the speed of learning. The use of Radial Basis Functions has a significant advantage over more conventional summing operators in that they provide a method by which the learnt knowledge (or parameters) may be extracted from the network after it is trained. The goal of the work is to model the parameters such that both the major and minor loop behavior of the material is characterized effectively. However, using the Preisach model in an actual analysis situation based around a finite element mesh is likely to be expensive in both time and memory since each element in the system (and there may be hundreds of thousands of them in a 3-d system) requires that the local magnetization history be maintained. Thus this paper proposes a method for training a second neural network by using the Preisach model. This network can then be used in a computational system and, it is hoped, will reduce the time and memory requirements for a non-linear analysis involving hard magnetic materials.

Manuscript received September 23, 1996
The work was supported by the Natural Sciences and Engineering Research Council of Canada.
D.A.Lowther, e-mail lowther@cadlab.ee.mcgill.ca, fax (+1)-514-398-4470

II. Radial Basis Functions

The most common form of neural network currently being used is based around a set of simple neurons (processors) capable of generating an output based on the weighted sum of the inputs:

$$O_j = \sum_{i=1}^{n} W_i I_i \qquad (1)$$

However, a single layer of such neurons cannot handle problems which are inherently non-linear. In essence, a single layer, when the weights have been determined from a training set, performs a separation operation on the input data. That is, in a binary system, the output neuron goes to state 1 if the weighted sum of the inputs exceeds some prespecified threshold and is zero otherwise. In order to solve this problem, hidden layers of neurons are added and this generates the classical feed-forward neural network as shown in Fig. 1.

The problem with the hidden layers is that the system becomes difficult and expensive to train - this is referred to as the 'hard-learning' problem and is related to the fact that the only error measurements which can be made on the network are at the outputs but the weights of the hidden layers must be adjusted based on these. The general strategy for training relies on the back propagation of the errors to the input and the training problem can then be stated in terms of a more conventional fitting or optimization process.

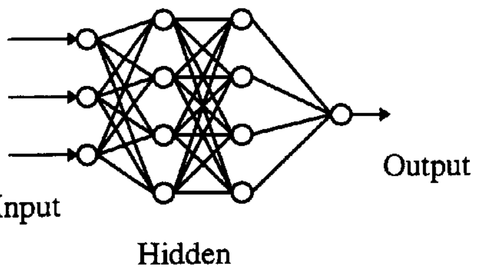

Figure 1. Basic Feed Forward Network.

An alternate approach to handling non-linear problems, which can also lead to more efficient (and faster) training systems is based around the use of Radial basis functions (RBF) [4]. Such artificial neural networks have been shown to be practical in a number of applications. A typical radial basis function neural network consists of three layers of neurons; input, hidden and output. It is a fully connected perceptron feedforward-like architecture in which the output units have a linear activation function, Fig 2.

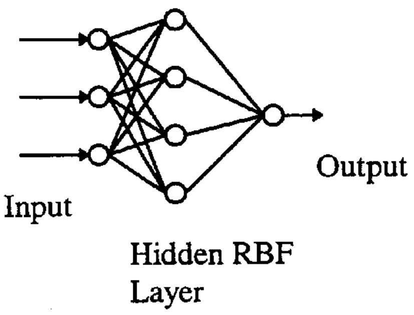

Fig 2. RBF Based Feed Forward Network.

As for the basic feed forward network described above, the network paradigm is based on the simple intuitive idea that an arbitrary function $\hat{y}(x)$ can be approximated as the linear superposition of a set of localized basis functions, $\varphi_i(x)$, by the following equation,

$$\hat{y}_i = \sum_j w_{ij}\varphi_j(x) \tag{2}$$

where $\varphi_i(x)$ is a radially symmetric function, called a "kernel function", centered on the ith data point and x is the corresponding input. A common basis function is usually the bell-shaped gaussian function given by

$$\varphi_j(x) = e^{D_j^2/2\sigma_j^2} \tag{3}$$

Other kernel functions that have good theoretical backing are the thin plate splines, $\varphi(x) = x^2 \log x$, the multiquadric and inverse multiquadric that are expressed respectively by $\varphi(x) = (x^2 + c^2)^{½}$ and $\varphi(x) = (x^2 + c^2)^{-½}$. The euclidian distance $\mathbf{D}_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}_j\|$ between the input vector **x** and $\mathbf{c}_j$, is determined by

$$\mathbf{D}_j^2 = (\mathbf{x} - \mathbf{c}_j)^T(\mathbf{x} - \mathbf{c}_j) \tag{4}$$

where the vector $\mathbf{c}_j$ is the center; σ_j, the standard deviation, describes the width or the spreading factor of the gaussian basis function at node j, and w_{ij} are the second-layer weights. The network is entirely defined when the parameter set $\{\mathbf{c}_j, \sigma_j, \{w_{ij}\}\}$ is determined. Each hidden unit has a localized response, that is, valid responses range within a limited zone, generally a circular shape, named the receptive field, σ is the size of the receptive field. The implementation should be adaptive (weights, center location, widths are all tuned to the data).

In effect, the use of the radial basis functions maps the input data from the parameter space in which it is presented to a new space in which the different characteristics are clearly separated. The separation can now be determined at the output by a simple linear summation. Training takes place in much the same way as it would with a classical multi-layer fed forward network.

III. Cerebellar Model Articulation Controller (CMAC)

The main aim of using neural network architectures, i.e. systems which can be trained from a set of examples and can generalize these examples to solve problems which have not been encountered before, is to construct a structure that might be seen as equivalent to a Response Surface. Thus in modeling magnetic material properties, the system would be trying to determine the output response (i.e. H) given a time sequence of B values. Thus the question to be answered is "what will the next value of H be given the next value of B". Clearly, as with all computing systems, there are two solutions. The first involves storing the B-H pairs representing all possible tracks through the system - this would require unacceptably large amounts of memory. The second requires computing the tracks when needed - this requires considerable amounts of processing and does not eliminate the need for memory entirely because past history has to be stored somewhere. The neural network tries to provide a compromise between these two extremes by modeling the response surface implicitly with a limited amount of data stored and then using an interpolation approach to finding the new outputs to new inputs. The CMAC, first developed by Albus [5], attempts to do much the same thing. The idea is to minimize the amount of data to be stored while still being able to retrieve all the points on the surface. This is achieved by designing a structure which provides a "contents addressable" memory. This is closely related to hashing schemes and associative memories in general. A simple CMAC architecture is shown in Fig. 3.

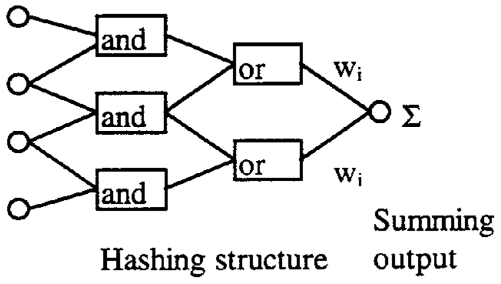

Fig. 3. Simple CMAC Layout.

Generally, Fig. 3. is a simplification of the process. In order to provide a real hashing function, the internal layers consist of randomly connected sets of AND and OR operators. The circuit shown above hashes a 4 bit number into a bit representation [6]. The single layer summing neuron at the output is then trained to adjust the weights such that the desired function for the given 4 bit input sets is achieved.

Basically, the core of a CMAC is an associative memory representing, in this case, the hidden layer units, which has the ability to realize complex nonlinear functions. This achievement is carried out by a series of sequential mappings from the set of real values at the input into an N-dimensional binary associate vector - a set of integers in the associative memory. The output of the associative memory is connected to a simple summing circuit to produce the final output. Once the system has been trained, an input to the network will cause one or more entries in the associative memory to fire. The number of entries firing depends on the distribution of data in the input training set; similar network inputs activate overlapping neighborhoods inside the network. As a result of this, the output of a CMAC is dependent on the previous history of its inputs and the generalization is due to the overlapping cells in the input region. This dependence on previous history is a property which can be made use of in modeling hysteresis.

Among the numerous benefits of the properties offered by a CMAC are the following:

1) Local generalization capability;
2) Fast learning without fixation problems, in fact, a CMAC network is considered as an alternative to backpropagation multi-layer networks;
3) Incremental training and output superposition capability ; A CMAC network is an adaptive system that uses local learning and also permits incremental training. This enables the net to be retrained on-line to produce the correct signals for locally changed conditions;

A. Architecture

Three-layers feed forward network using locally tuned neurons to achieve a series of sequential mappings:

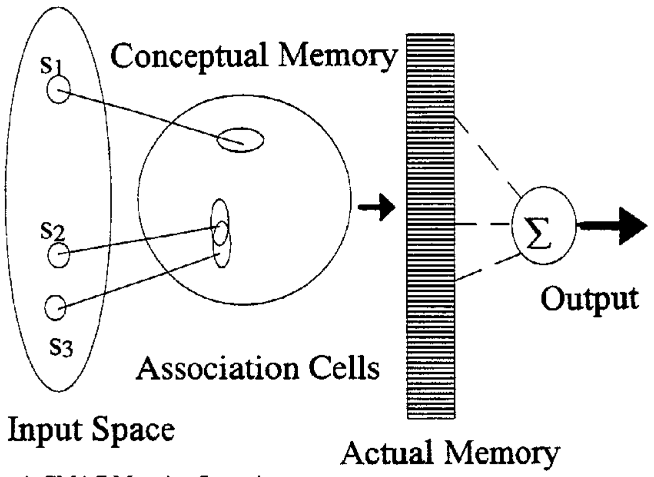

Fig. 4. CMAC Mapping Operations.

$S \rightarrow M \rightarrow A \rightarrow p$

Where $S = \{\text{set of all input vectors}\} = \{s_1, s_2, \ldots, s_n\}$; $S_i = \{s_{i1}, s_{i2}, \ldots, s_{in}\}$

p is the output value;

$A = \{\text{the association cell vectors}\}$

When a set of S input vectors are presented to the CMAC, the transformation $M \rightarrow A$ is an encoding scheme where each R-ary variable is converted into a binary variable according to a specific conversion rule. An output value p is computed according.

B. Training CMAC Networks

The training process is similar to that used for a conventional feed forward network, i.e. it is a supervised process and uses error back-propagation to adjust the weights. In this case, the process uses a learning algorithm, basically a variant of LMS method, which involves an adjustable mixture of fixed basis functions and a linear update rule. This computationally efficient algorithm can be treated as Gauss-Seidel iteration of a linear system.

D. Data Representation

The main goal of the process is the transformation of D-dimensional input vector space (state variables) to K-dimensional address space (K association coefficients) with a good generalization or interpolation capability.

A coarse coding scheme is used to represent each state variable input. In this manner, the latter is quantized into several discrete regions, called blocks. Areas formed by quantized regions are called hypercubes. and the quantization for each variable is shifted by one interval.

For information storage and recovery, each hypercube is assigned a physical memory address.

The generalization width

The generalization width, *c*, is the width, in one input dimension, of a single receptive field. It could be thought of as a quantization resolution. CMAC makes use of a number of overlapping sets of receptive fields of this width. Hence the CMAC, by this way, has an input resolution of the generalization width divided by the number of sets of receptive fields. This latter number is the number of memory locations accessed by each input, since each receptive field corresponds to a virtual memory location and a memory location is accessed if a receptive field is activated.

E. Memory locations

The number of memory locations required depends on the degree of nonlinearity of the system being modeled. The total number of memory locations is a design parameter under our control. It is the number of memory locations used by the network. Choosing the size of the physical memory is a simple task, but choosing the size of the generalization parameter affects a lot of different things. The overlay displacement parameters vastly improve the quality of the CMAC [7]. This influences the shape and size of the additive modeling region whose "width" is determined by the generalization parameter.

IV. RBF-CMAC COMBINED MODEL

The techniques of Radial Basis Functions and the CMAC can be combined in order to take the merits from both. An RBF is used to preprocess the input data. Each basis function consists of hidden units covering a region in the input space. A radial basis function training procedure reduces the number of typical data and also enhances the generalization capability of the hybrid RBF-CMAC network.

V. MODELING HYSTERESIS

Having described an architecture consisting of a RBF neural network coupled with a CMAC which appears to have properties which match those needed for modeling magnetic hysteresis effectively, the remainder of this paper will discuss some initial results.

While the ultimate intention of this work is to be able to perform the parameter identification necessary for constructing a Preisach model from experimental measurements (for which a similar network architecture is proposed), the work described here concentrates on demonstrating the effectiveness of the architecture in generating magnetization characteristics after being trained on the output of a Preisach model.

The initial hysteresis loops are generated by using a Preisach model based on the parameters discussed in [8]. The set of nested characteristics of M plotted against H are shown in Fig. 5.

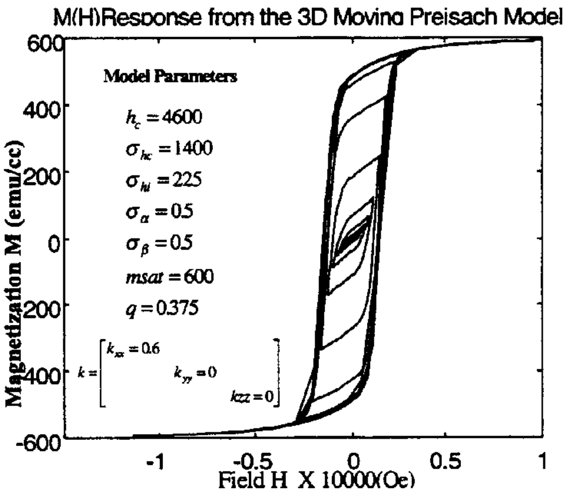

Fig. 5. M-H curves generated from a Preisach model.

The Preisach model was then excited with a damped sine wave and the M-H curves were generated to be the training data for the neural network system. The H waveform and the corresponding M waveform is shown in Fig. 6.

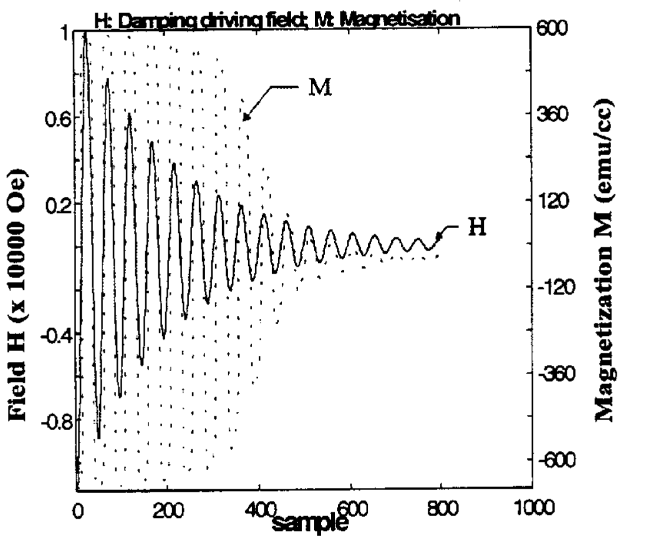

Fig. 6 . M and H training data.

Part of this data was then used as the training set for the network. Once the network was trained, it was tested using the test set. Fig. 7 shows the response to an input to the trained net of a damped sine wave for H - the "output" is the expected output from the Preisach model, the "predicted" output is from the network. The figure also shows the error expressed as the difference between the M value predicted by the network and the M value output by the Preisach model. As can be seen the error is generally considerably less than 10%. Figs. 8 and 9 are zoomed up views of sections of Fig. 7. A pruning task where one weight is removed at a time has been carried out to improve the network generalization performance. Fig. 12 shows that the final prediction error reaches its minimum when the number of parameters decreases from 70 to 38. This result give the corresponding optimal network structure.

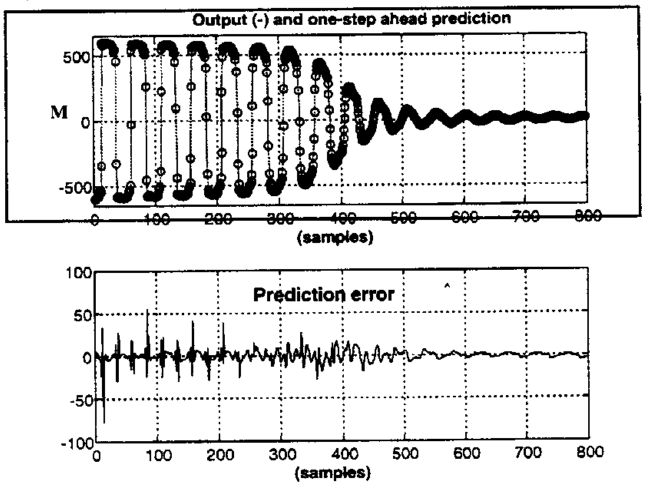

Fig. 7. Output of feed forward network for damped H waveform.

Fig. 10 shows the equivalent predictions from the RBF network and Fig. 11 gives the predicted nested M-H curves produced and these may be compared with those generated directly from the Preisach model.

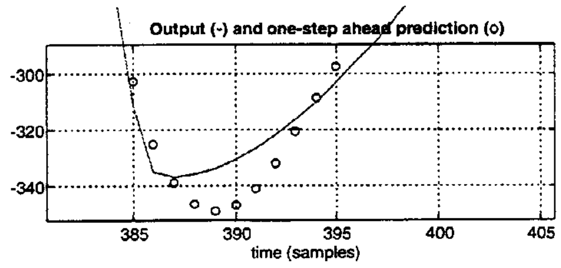

Fig. 8. Zoom of Fig. 7 around time 390.

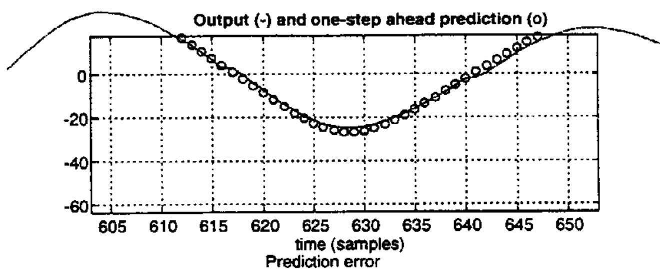

Figure 9. Zoom of Fig. 7. around time 630.

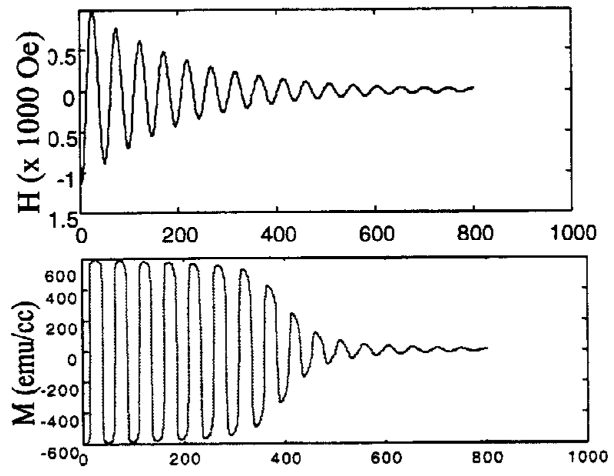

Fig. 10. Output of RBF network for damped H waveform.

IX. Conclusions

The accurate modeling of hysteresis phenomena in magnetic devices is critical if analysis systems such as those based around finite elements are to be effective in the design process. This paper has discussed the use of a hybrid neural network strategy based on the combination of a Radial Basis Function feed forward network and a CMAC. The tests so far seem to show that such a system may be able of providing an accurate model of the hysteresis property. It also is both fast and memory economic - two features which are necessary if realistic analysis systems are to be considered. The next set of tests will embed this system within a realistic finite element model where the input-output behavior is an arbitrary waveform different from the training ones. The systems performance will then be further examined.

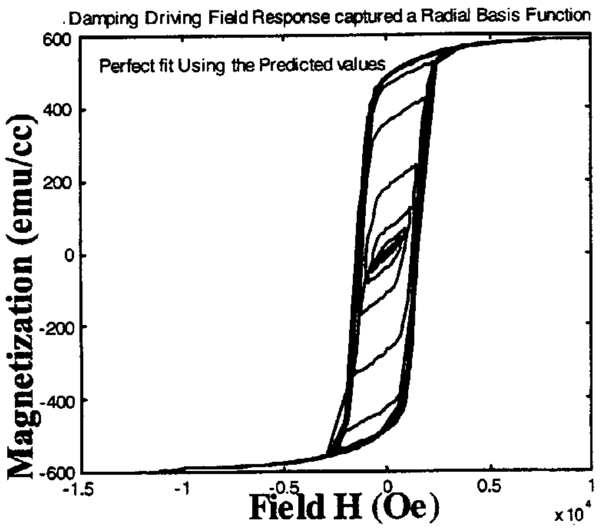

Fig. 11. M-H Curves from RBF network.

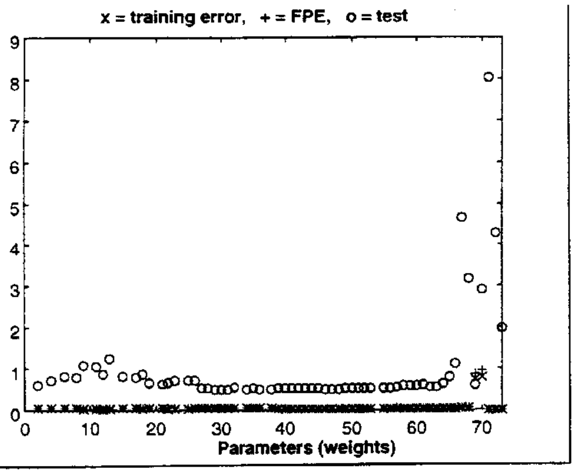

Fig. 12. M-H Curves from RBF network

REFERENCES

[1] F.Vajda, E.Della Torre,. "Hierarchy of Scalar Hysteresis Models for Magnetic Recording Media", *IEEE Trans. on Magnetics*, 32, 1996, pp. 1112-1115.

[2] F.Ossart, R.Davidson, S.H.Charap, "A 3D Moving Vector Preisach Hysteresis Model", *IEEE Trans. on Magnetics*, 31, 1995, pp. 1785-1788.

[3] W.T.Miller, F.F.Glanz, G.Kraft, "CMAC: An Associative Neural Network Alternative to Backpropagation", *Proc. IEEE*, 78, 1990, pp. 1561-1567.

[4] J.-S.Jang,C.-T.Sun, " Functional equivalence between radial basis function networks and fuzzy inference systems ", *IEEE Transactions on Neural Networks*. v 4 n 1 Jan 1993. p 156-159

[5] J.S.Albus, "A new approach to manipulator control: the cerebellar model articulation controller (CMAC)", *Trans. ASME J. Dyn. Syst. Mes. Control.*, 93, 1975, pp.220-233.

[6] I.Aleksander, H.Morton, "An Introduction to Neural Computing", 2nd Edition, Thomson Computer Press, London, 1995.

[7] M.Brown, C.Harris, "Neurofuzzy adaptive modelling and control", Prentice Hall, New York, 1994.

[8] R.Davidson, "Vector Preisach Hysteresis Models for Simulation of Recording Media", Ph.D. Thesis, Carnegie Mellon University, May 1996.