Electromagnetic Device Performance Identification Using Knowledge Based Neural Networks

Frederic Dandurand and David A. Lowther
Electrical Engineering Department, McGill University, Montreal, Quebec, Canada

Abstract—**A knowledge based artificial neural network which represents the design equations of an electromagnetic device is described. The network architecture is based on a rule set developed from a simple algebraic model of the device. The system is then revised by using numerical solutions as training sets to remove the assumptions built in by the algebraic system.**

Index Terms—**Neural Networks, Electromagnetic Device, Performance Identification.**

I. Introduction

Understanding the characteristics and performance of an electromagnetic device is crucial to the design process. Often, a designer can gain valuable insight by using a simple algebraic model. While this usually makes major assumptions, such as linearity of material characteristics, a lack of fringing, etc., the major characteristics of a device can be seen. Moreover, the effects of changes in one parameter on the performance of the device can be evaluated quickly. It is the understanding of the "shape" of the design space which enables a designer to move rapidly towards an optimal solution. The term "shape" is used here to identify the effect on an output (i.e. increase, decrease or none) of a change in one or more inputs. Given this information, the designer can make intelligent decisions as to how to find an optimal solution and how sensitive this solution might be to small parameter changes. These simple models, however, are limited in their applications and thus, ultimately, experimental models have been traditionally used to derive performance characteristics of a class of device. This is both slow and expensive and impractical in many modern design environments.

As numerical analyses become more prevalent, they provide a method of replacing the experimental laboratory tests. The accuracy, though, is obtained at a cost of often lengthy computation times. In addition, such systems only provide descriptions of single points in the design space, much in the way that physical experiments do, and do not provide the designer with the same "feel" for the device as can be achieved with an analytical model. There are several possible solutions to this slow response time and all require an exploration of the design space "offline" followed by the construction of an appropriate model. The simplest method of doing this is to construct a look up table and use linear interpolation to estimate the output for a given input [1]. This tends to be limited in terms of the number of parameters which can be considered. A second approach is to use a response surface method where a high order surface is fitted to a set of experimental results [2]. While this can handle a larger range of parameters, it requires considerable expertise in the construction of the surface. A third approach is to use a conventional feed forward neural network to "learn" the surface. This requires many offline experiments followed by long training times. All of these approaches, however, predict output values, not the directions of changes in the outputs that is really required for effective optimization.

While the algebraic model often makes many assumptions, it can provide a form of qualitative reasoning which can guide an optimization process. An earlier proposal to make use of this information used a constraint based model to provide a fast design tool for examining the effects of parameter changes [3]. The intention of this paper is to show how the information encoded in an algebraic model can be used to develop a set of design rules that can be transferred to a neural network structure. By encoding the information from the algebraic model directly in the structure of the neural network, the training time can be considerably reduced, as can the number of training sets. In fact, the "experimental" results from the numerical analysis system are used to fine tune the network to remove some of the assumptions built into the original model. This network can then be used to guide a conventional optimization process, thus considerably reducing the computation time needed to achieve a design. As an added benefit, the system can show a designer why certain choices were made.

To gauge the effectiveness of this approach, the Knowledge Based Artificial Neural Network (KBANN) derived from the initial equations is compared with a network developed by a conventional training method and a set of tests is performed on both architectures.

II. An Algebraic Model and the Derivation of Rules

The example to be considered in the paper is that of a simple C-Core inductor (Fig. 1). The parameters shown in the figure and listed in Table I below (with the exception of *I, R* and *F*) may be considered as *structural* and these are the ones that the designer can manipulate in order to achieve the required performance. The performance of the device is given in terms of *design* parameters; in this case these are the terminal resistance, *R*, the input current, *I*, and the generated force, *F*.

Manuscript received June 3, 1998 This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.
Email: lowther@cadlab.ece.mcgill.ca

0018-9464/99$10.00 © 1999 IEEE

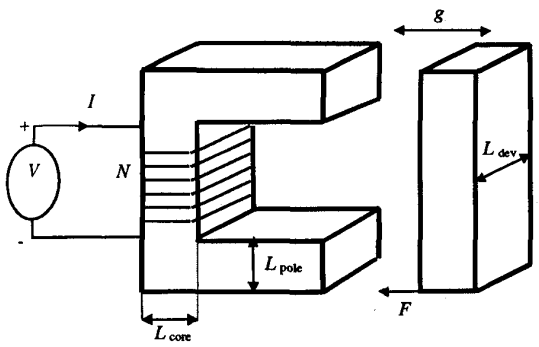

Fig. 1. A Simple C-Core Actuator.

Using a simple magnetic circuit model, a set of algebraic equations can be created which relates the input current, the geometric structure and the coil architecture to the fluxes in the system and the force generated. This ideal device model assumes that:

1) Magnetic materials are not saturated
2) There is no fringing.
3) Design parameters are all independent of each other.

The resulting set of equations is shown below (1),(2),(3):

$$R = 2\rho N(L_{core} + L_{dev})/a_{wire} \quad (1)$$

$$I = Va_{wire}/(2\rho N(L_{core} + L_{dev})) \quad (2)$$

$$F = \mu_0 V^2 a_{wire} L_{pole} L_{dev}/(32\rho^2 g^2 (L_{core} + L_{dev})^2) \quad (3)$$

From this set of equations, the derivatives of each of the design parameters with respect to the structural parameters can be obtained leading to a description of how a change in one parameter can affect the others. For example, the length of the core (L_{core}) affects R according to the following equation:

$$\frac{\partial R}{\partial L_{core}} = \frac{2\rho * N}{a_{wire}} \quad (4)$$

TABLE I
THE BASIC SET OF PARAMETERS

Parameter	Description
Structural	
g	Air gap between core and armature
N	Number of turns of the coil
V	Voltage applied to the coil
L_{core}	Length of the core
a_{wire}	Area of the wire used for the coil
L_{pole}	Length of the pole
L_{dev}	Length of the device in the Z plane.
Design	
F	Force
I	Current in the coil
R	Resistance of the coil

This process results in a set of fourteen partial differential equations describing the model. Since ρ, N and a_{wire} must be non-negative numbers, (4) implies that R and L_{core} change together - an increase in L_{core} implies and increase in R. The set of equations can be re-expressed as a set of *trend rules*.

In moving the design of a device towards the desired performance, the required inputs are whether a parameter is too large, too small or within range. Thus, the trend rules have inputs, which consider the current state of a parameter, and outputs, which are commands to change a parameter. There are four sources of knowledge are available: the input state of each structural parameter itself and the state of R, I and F. Each structural parameter can generate an output command applying to itself as follows:

- *If (X is too small) then (increase X)*
- *If (X is too large) then (decrease X)*

This set of commands is needed to allow the imposition of design constraints such as the width of the core cannot exceed a particular value.

Next, in order to relate the input states of F, I and R to output commands, the set of trend rules described above has to be analyzed. This is done by examining the sign of the equations and mapping it onto a symbolic representation of the relationships between the parameters. For instance:

- *If (R is too small) then (increase L_{core})*
- *If (R is too large) then (decrease L_{core})*

It is possible that, in the application of the trend rules to an existing design state, commands will be issued to both increase and decrease a parameter simultaneously. Such a condition might occur where there is a need to increase the force which, in turn, implies and increase in the cross-sectional area of the pole but the pole area already exceeds constraints placed on the physical dimensions. In this situation, a conflict is flagged and the information is passed to a decision module to determine if a change should be allowed. The conflict bit is obtained simply by ANDing the (Increase X) and the (Decrease X) commands. If both are true, then a conflict is detected.

After simplification, a set of 21 symbolic rules acting on 20 input symbols and giving 21 outputs is obtained. The input symbols come from the seven structural parameters plus the Force, Current and Resistance (1), (2), (3) and each parameter has two inputs - "too big" and a "too small". The 21 output symbols consist of three for each structural parameter - increase, decrease and conflict.

The complete set of trend and conflict rules could be implemented in a traditional rule based system. However, such a system would only operate within the parameters of the original simple algebraic model with all its simplifying assumptions. The intention here is to construct a Knowledge Based Artificial Neural Network (KBANN) [4], [5], [6], [7], [8], to implement the rules instead, and then, to retrain the network to remove the assumptions by using a two-dimensional, finite element based, numerical model that provides training data, which include the effects of non-linearity, fringing, etc [9].

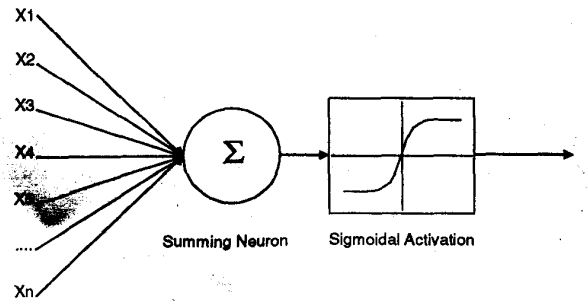

Fig. 2. A Single Neuron.

III. A Knowledge Based Artificial Neural Network

The symbolic rules derived from the algebraic model can be used to create a backpropagation neural network, whose topology and initial connection weights are based on the rules rather than being arbitrarily chosen. A single neuron of the network is shown in Fig. 2. A weighted sum of the inputs to the neuron is created and then passed through a sigmoidal activation function to generate an output between 0 and 1. The complete structure is created by mapping the dependencies present in the rules onto a network topology. The resulting structure implements the qualitative effects of the algebraic model. While the network is fully interconnected, in fact, only those connections relating to the symbolic rules are non-zero initially.

The network can now be refined, i.e. modified, through learning by using empirical information obtained from a numerical simulation of the device. This allows real world effects such as magnetic saturation and field fringing to be superimposed on the original algebraic system. Training data (Empirical Knowledge) is obtained from an electromagnetic device analysis package [9] by varying each of the structural parameters independently over a wide range of values. The training samples are then obtained by examining the changes in the design parameters for each change in the structural parameters. The sign of the change (positive or negative) can then be mapped onto the concepts of input state and output commands.

Using the KBANN methodology, the resulting network has 4 layers (1 input, 2 hidden and 1 output) containing 20, 28, 28 and 21 neurons, respectively. This results in a total of 1932 weights to be computed in the final training although less than 10% of these are non-zero in the initial creation of the network. The starting values of the non-zero weights are all set equal to a pre-determined level, *W*.

As a reference to compare performance, a conventional backpropagation neural network has also been implemented. Its topology was adjusted experimentally until a performance equivalent to the KBANN was achieved. It has 3 layers containing 20, 9 and 21 neurons respectively, giving a total of 369 weights. Note that the KBANN is considerably larger than the reference network. It possesses 47 more neurons and 5.2 times more connection weights. Note that both networks are using the same neuron structures, i.e. summing neurons thresholded on their outputs.

IV. Training the Network

After an analysis of the terminal parameters with respect to design parameters, the training and testing sets were carefully chosen to include regions in the design space where saturation was critical and others where it did not exist. Because KBANN is based on a model of the device that assumes no saturation, the proportion of samples taken from the saturation region was over-represented (40%) in order to facilitate learning by compensating for the bias towards non-saturated regions. The training set contains 2450 samples and the testing set, 600. Subsets containing samples from the saturation or non-saturation regions were also considered.

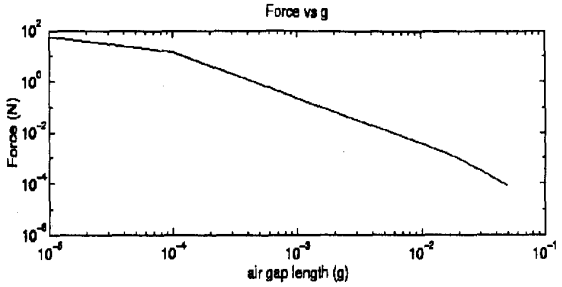

Fig. 3. Analysis of Saturation with respect to the Airgap (g).

Because of the difficulty of extracting conflict information from the empirical data for this model, only non-conflicting samples were used to train the network.

V. Experiments

The first experiment performed on the KBANN was to evaluate the effect of the parameter *W*, the initial connection weight. Larger values mean that the symbolic knowledge is more strongly encoded, resulting in a better performance on data samples corresponding to the device model, i.e. non-saturated regions. Since large weights tend to saturate the sigmoid activation function elements (i.e. where the slope is small), learning of saturation samples is more difficult and the generalization ability is also globally reduced. On the other hand, small values of *W* generate a network that does not exploit the full potential of symbolic knowledge (i.e. rules). Sample results are presented in Table II.

As can be seen, the optimum trade-off was *W*=4.0 for this problem, and was used for all subsequent experiments. This corresponds to the value suggested by Shavlik [5].

TABLE II
KBANN Training Performance with *W*

W	Training	Error on testing set
3.0	86 epochs	0.044
4.0	73 epochs	0.042
5.0	76 epochs	0.044

TABLE III
A COMPARISON BETWEEN KBANN AND THE REFERENCE NETWORK

	KBANN	Reference Network
Training	73 epochs	219 epochs
Residual Error (on Training Set)	0.40%	0.40%
Error on testing sets		
Full	0.042	0.049
Saturation only	0.014	0.0056
Non-saturation only	0.058	0.072

VI. Performance Comparisons

Table III presents a summary of the results obtained from KBANN and the reference network. Despite the fact that the training is 3 times shorter, KBANN is overall more computationally expensive to train, because of the larger number of connection weights. For the same reason, recall is more expensive on KBANN.

However, despite its additional complexity, the KBANN did not suffer from local minimum problems during training. An analysis of the average variation of weights shows that the initial point in the search space determined by the KBANN methodology is a good estimate of the final solution, and therefore the training phase involves less weight variations.

The last experiment allows the determination of what the trained neural networks have actually learned. For this purpose, a rule extraction process is performed. Each possible input state (e.g. *gap is too large*) is individually applied to the network (20 cases) then the output is computed and passed through a threshold operator. Only significant values are kept. Table IV presents the results for a subset of 5 parameters.

As can be seen, KBANN has acted as a "Rule Optimizer" by pruning 45% of the initial rules. This is explained by the fact that the training set contained only non-conflicting data.

On the other hand, the reference neural network did significantly worse. It was only able to learn 15% of the relationships described in the ideal C-core model, while also learning spurious relationships (15%).

This experiment shows that KBANN performs much better in the presence of an incomplete training set because the symbolic knowledge used to build it sets a better initial point to start exploring the search space.

Thus, in situations where a complete set of training data is either difficult to obtain or costly, or both, but some amount of formal design knowledge exists, the KBANN approach may prove to be both efficient and effective.

TABLE IV
RULE OPTIMIZATION BY KBANN

Parameter	Retained Rule Occurrences	Deleted Rule Occurrences	Ratio
g	2	0	100%
V	3	1	75%
a_{wire}	3	3	50%
N	1	3	25%
L_{dev}	0	4	0%
Total	9	11	45%

In summary, the strength of KBANN resides in a *Symbolic Conceptualization* of the problem, associated with a neural network implementation that adds learning capacities.

VII. Conclusions

The major advantage of the KBANN was that, even after training, it maintained a large proportion of the knowledge used to construct it thus, it performed better than the conventional network in the non-saturated region, i.e. where the problem matched the algebraic assumptions.

However, the main limitations of the method are:

Generated networks are not minimal in size (number of layers and neurons) and also have a fixed size for a particular problem. However, KBANN partially removes the need to determine an optimal neural network structure for a particular problem.

The computational cost to obtain the output in the recall phase) is directly proportional to the number of connection weights. This makes KBANN-generated networks less computationally efficient to use.

It learns new or implicit knowledge representations or associations contained in exceptions less easily. The internal representation is biased towards the symbolic representation of the ideal model.

In terms of performance, KBANN makes a trade-off between retention of non-saturated regions and learning saturation data. Generally, a better error rate on one results in a worse one on the other.

References

[1] T.E.McDermott, P.Zhou, J.Gilmore, Z.Cendes, "Electromechanical System Simulation with models Generated from Finite Element Solutions," *IEEE Transactions on Magnetics*, Vol. 33, 2, 1997, pp. 1682-1685.

[2] R.Rong,, D.A.Lowther., Z.Malik, H.Su, J.Nelder, R.Spence, "Applying Response Surface Methodology in the Design and Optimization of Electromagnetic Devices," *IEEE Transactions on Magnetics*, Vol. 33, 2, 1997, pp. 1916-1919.

[3] C.M.Saldanha, D.A.Lowther, "Knowledge-Based Computation of Electromagnetic Device Parameters," *IEEE Transactions on Magnetics*, Vol. 24, 1, 1988, pp. 334-337.

[4] J.W.Shavlik, G.G.Towell, "Knowledge-Based Artificial Neural Networks," *Artificial Intelligence*, 70, 1994, pp. 119-165.

[5] J.W.Shavlik, "A Framework for Combining Symbolic and Neural Learning," Technical Report 1123, Computer Sciences Department, University of Wisconsin - Madison, Nov. 1992

[6] A.Tan, "Integrating Rules and Neural Computation," *1995 IEEE International Conference on Neural Networks*, pp. 1794-1799

[7] K.Tsujino, "Hybrid Knowledge Acquisition by Integrating Decision Trees and Neural Networks," *1995 IEEE International Conference on Neural Networks*, pp. 1379-1383

[8] Q.Yang, V.K.Bhargava, "Building Expert Systems by a Modified Perceptron Network with Rule-Transfer Algorithms," *International Joint Conference on Neural Networks*, 1991, pp. II-77 - II-82

[9] Infolytica Corporation, *MagNet 5 Reference Manual*, Montreal, Canada, 1995.