INVERSION OF UNIFORM FIELD EDDY CURRENT DATA USING NEURAL NETWORKS

J. M. Mann, L. W. Schmerr,
J. C. Moulder, and M. W. Kubovich

Center for NDE, Iowa State University
Ames, IA 50011

INTRODUCTION

A resurgence of research in artificial neural networks has sparked interest in applying these networks to difficult computational tasks such as inversion. Artificial neural networks are composed of simple processing elements, richly interconnected. These networks can be trained to perform arbitrary mappings between sets of input-output pairs by adjusting the weights of interconnections. They require no a priori information or built-in rules; rather, they acquire knowledge through the presentation of examples. This characteristic allows neural networks to approximate mappings for functions which do not appear to have a clearly defined algorithm or theory. Neural network performance has proven robust when faced with incomplete, fuzzy, or novel data.

Inversion of eddy current flaw signals has typically been based upon models of the field-flaw interaction, a so-called model-based inversion procedure. Although the feasibility of inverting eddy current data in this fashion has been demonstrated before [1-3], the complexity of such procedures has hampered their widespread acceptance and use in industry. The goal of this study is to develop an inversion method that is easy to use and implement outside the research community. This paper presents preliminary results on the use of neural networks for the inversion of eddy current flaw signals to obtain flaw sizes.

UNIFORM FIELD PROBE

A uniform field eddy current probe was selected for use in this study because a substantial body of experimental and theoretical work exists. The theory for the interaction of a spatially uniform electromagnetic field with a three-dimensional flaw developed by B. A. Auld [4,5] is well-known and has been shown to agree with experiment [6,7].

The probe used here is based on a design proposed by E. Smith [6], but in a slightly different configuration[8]. This probe consists of a C-shaped ferrite core wound with 65 turns of wire, creating an active region between the two tips of the ferrite. To increase uniformity of the magnetic field in the active region, the tips of the ferrite were shaped and chamfered [8], and a copper foil surrounds the probe to diminish effects of field leakage.

NEURAL NETWORKS

Neural networks are intended to model the structure and operation of the brain. They are composed of computationally simple processing elements and connections. Succinctly, a neural network can be described as a directed graph, with the nodes of the graph represented by the processing elements and the edges of the graph represented by the connections between the processing elements [9]. Each processing element receives inputs from several places – other processing elements, the outside world, or both – and operates on its inputs with a transfer function to produce a single output. This output becomes the input either to other processing elements in the network or the final output of the network.

The networks studied here are multilayered feed-forward networks and use a backpropagation learning algorithm [10]. In this implementation, networks are composed of several separate layers (groups of processing elements), including an input layer, an output layer, and possibly one or more hidden layers (layers which do not receive inputs from or send outputs to the outside world), as shown in Fig. 1. The interconnections transmit information in only one direction during operation and each interconnection has an associated weighting factor.

The transfer function for a processing element comprises two steps. The first step is to compute the weighted sum of inputs, I

$$I = \sum_i w_{ji} x_i, \tag{1}$$

where w_{ji} is the weight on the connection from the ith processing element in the preceding layer to the jth processing element (the processing

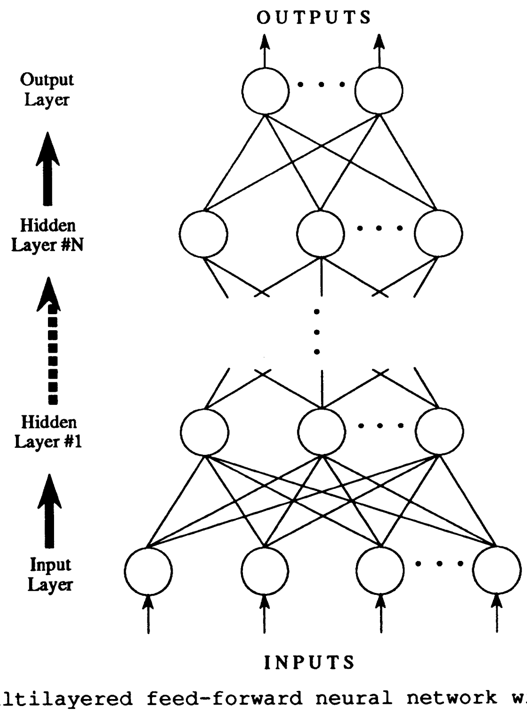

Figure 1. A multilayered feed-forward neural network with hidden layers.

element whose transfer function is being computed) in the current layer and x_i is the output of the ith processing element. The second step is to apply an activation function, f, to I to produce OUT

$$OUT = f(I). \qquad (2)$$

The activation function generally used for backpropagation is a sigmoid [10] of the form

$$f(I) = \frac{1}{1 + e^{-I}}. \qquad (3)$$

A sigmoid satisfies the requirement, discussed later, that the activation function be everywhere differentiable. Any activation function which is everywhere differentiable could be used. It also acts as an automatic gain control [11], so that $OUT \rightarrow 0$ as $I \rightarrow -\infty$ and $OUT \rightarrow 1$ as $I \rightarrow +\infty$. This prevents the outputs of the processing elements from growing without bound, which could allow some processing elements to dominate the operation of the network and cause a kind of "paralysis."

BACKPROPAGATION LEARNING

By appropriately adjusting the weights of interconnections, a neural network can be configured to approximate a particular function; in our case, the flaw inversion transform. Weights are adaptively adjusted through the application of a learning algorithm during a process called training. Training a neural network using a backpropagation learning algorithm consists of: 1) presenting inputs to the network's input layer; 2) allowing the network to compute its outputs; 3) computing the errors at the output layer by presenting the network with the target outputs for the associated inputs; 4) propagating errors back through the network and adjusting weights so as to minimize the errors at the outputs.

For processing elements in the output layer, an error signal, δ_k, is calculated for the kth processing element as

$$\delta_k = f'(I_k)(y_k - y'_k), \qquad (4)$$

where f' is the derivative of the activation function, y_k is the target output, and y'_k is the network estimate of y_k. The calculation of the error signal shows why the activation function f must be everywhere differentiable. The change of weight u_{kj} connecting the kth processing element and the jth processing element in the final hidden layer is defined as

$$\Delta u_{kj} = \alpha \delta_k z_j, \qquad (5)$$

where α is the learning rate and z_j is the output of the jth processing element. For all other processing elements not in the output layer, an error signal δ_j for the jth processing element is calculated from

$$\delta_j = f'(I_j) \sum_k \delta_k u_{kj} \qquad (6)$$

and the change of weight w_{ji} connecting the jth processing element and the ith processing element in the previous layer is defined as

$$\Delta w_{ji} = \alpha \delta_j x_i, \qquad (7)$$

where x_i is the output of the ith processing element in the previous layer. This backward propagation of error signals continues to the first hidden layer. Therefore, two passes through the network are required for each iteration – one forward to compute outputs and one backward to compute errors and adjust weights. The weight adjustment process is repeated for many input-output pairs, known as a training set. Training is stopped when the errors at the output layer reach a sufficiently low level.

The goal of backpropagation learning is to perform a gradient descent search in weight space to find the minimum error for all pairs in the training set. The error surface is determined by the set of possible weights for the network for a given input-output pair. As with any gradient search method, backpropagation is susceptible to becoming trapped in local minima. Several methods for escaping local minima have been proposed, including momentum [11]. Using a momentum term essentially adds some part of past weight changes to the current weight change. This can often help move over small "bumps" in the error surface that otherwise would trap the algorithm.

APPROACH

Two approaches were taken to determine the feasibility of using a neural network for inversion of eddy current flaw signals. The first approach involved the development of a network for inverting synthetic (noise-free) data. Flaw dimensions were generated randomly such that $c \leq 2.00$ mm, $a/c \leq 1$, $\Delta u \leq 0.1c$, and $a/\delta \geq 2$, where c is the flaw half-length, a is the flaw depth, Δu is the flaw width, and δ is the skin depth. Flaw impedance magnitude, $|\Delta Z|$, and phase, θ, were then calculated at seven frequencies (2 MHz - 8 MHz at 1 MHz intervals) according to Auld's ΔZ theory [5] for each of the flaw geometries.

The second approach was the trial inversion of experimental data obtained with a uniform field probe. Data were taken with the uniform field probe described earlier using computer controlled x-y positioners to move a sample under the stationary probe. Real and imaginary values of probe impedance were acquired by a personal computer connected to a Hewlett-Packard 4194A Impedance Analyzer over an IEEE-488 bus. Each measurement consisted of scanning the probe over a flaw in one-dimension, giving impedance values at a number of points both near the flaw and away from the flaw. This allowed for preprocessing of the data to remove effects caused by tilt. Flaw impedance magnitude, $|\Delta Z|$, and phase, θ, at the center of the flaw was then calculated. The flaws consisted of five semi-elliptical EDM notches and one "no flaw" in Ti 6Al-4V. The dimensions of these flaws are shown in Table 1.

In both approaches, the flaw dimensions are the outputs of the neural network and the ΔZ magnitude-phase information is the input. This requires a network with an input layer of 14 processing elements (magnitude-phase values at seven frequencies) and an output layer of three processing elements (flaw half-length, depth, and width). The number of hidden layers and hidden processing elements for each case are discussed below. After training, the networks were tested with data not used during training in order to evaluate the network's ability to generalize.

RESULTS USING SYNTHETIC DATA

Two sets of flaw dimensions and associated ΔZ magnitude-phase information were generated for training a network. One set consisted of 100 pairs, the other consisted of 1000 pairs. The second set included the same 100 pairs as the first set, but with an additional 900 pairs.

Table 1. Dimensions of flaws used in the experimental portion of this study. A * denotes a flaw used in training the neural network. The remaining two flaws were used for testing the network after training.

Flaw #	Length, 2c (mm)	Depth, a (mm)	Width, Δu (mm)
* 1	2.48	1.05	0.16
2	2.01	0.85	0.20
* 3	1.60	0.63	0.12
4	1.18	0.40	0.12
* 5	0.61	0.33	0.10
* 6	0.00	0.00	0.00

After training separate networks using the two training sets, the performance of each network was tested with a 100-pair testing set different from either of the training sets. The estimates computed by the network for the testing set were then compared with the actual flaw dimensions to evaluate the network's performance. The network trained on 100 pairs had one hidden layer of 14 processing elements while the network trained on 1000 pairs had two hidden layers of 14 processing elements each.

Figure 2 shows the results of both networks' estimates of flaw depth vs. the actual depth for each of the 100 flaws in the testing set. The figure shows that the network trained on 100 pairs (Fig. 2a) has fair performance, but that the network has not been able to closely approximate the function involved. The mean error for the testing set is -8.941 and the standard deviation is 18.856. The network trained on 1000 pairs (Fig. 2b), however, produced much better results, having a mean error of -0.770 and a standard deviation of 4.136. This demonstrates that the second network has already begun to closely approximate the mapping for flaw depth inversion.

Both training sets represent a small percentage of the total population of flaws. This suggests that the performance of the first network is good considering the training set size, but that performance was dramatically improved by training on a larger set of flaws as demonstrated by the second network. This suggests that the performance of the network might be further improved by using a still larger training set.

RESULTS USING EXPERIMENTAL DATA

Due to the limited number of flaws available for measurement, only one set of data was obtained for training a network to invert experimental flaw data. In order to generate training and testing sets of useful size, 10 independent measurements at seven frequencies (2 MHz - 8 MHz at 1 MHz intervals) were made on each of the flaws in Table 1, giving a total of 60 measurements. Training and testing sets were created by dividing the six flaws into two disjoint subsets. Because the training set needed to span the range of flaw dimensions that would be seen during testing, we chose to use four flaws, #1, #3, #5, and #6, for training and the remaining two flaws, #2 and #4, for testing. Although the

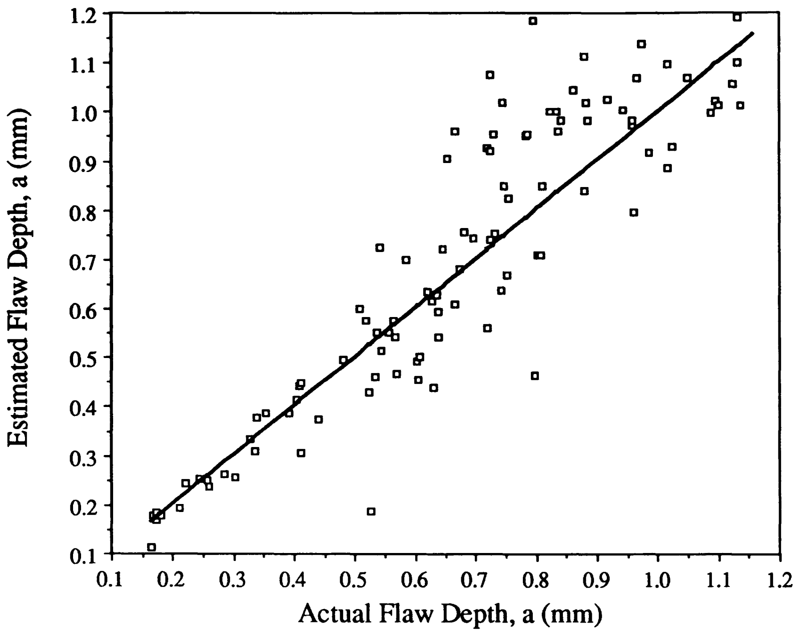

Figure 2a. Results of testing after having trained a neural network with synthetic data generated according to Auld's ΔZ theory for 100 randomly chosen flaw geometries. The graph shows estimated flaw depth vs. actual flaw depth.

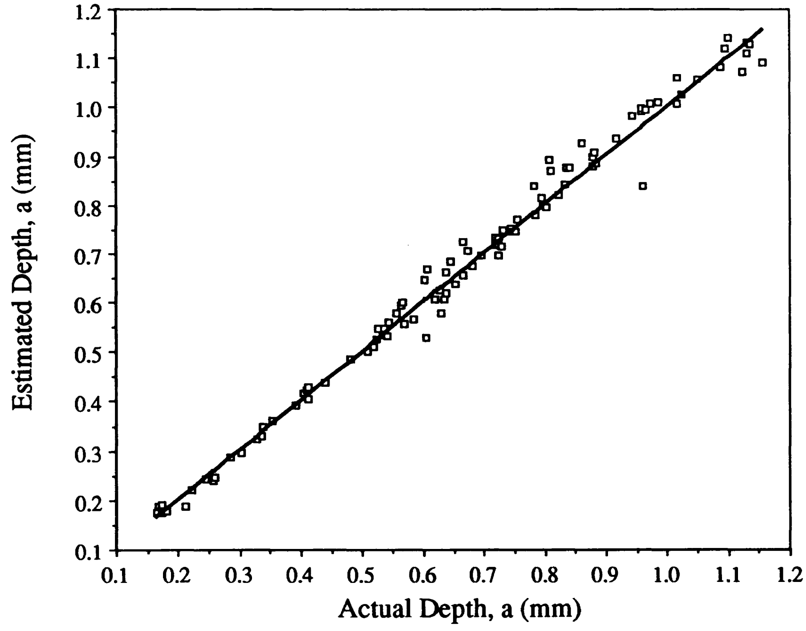

Figure 2b. Results of testing after having trained a neural network with synthetic data generated according to Auld's ΔZ theory for 1000 randomly chosen flaw geometries. The graph shows estimated flaw depth vs. actual flaw depth.

training flaws cover the range of dimensions of the test flaws for both the flaw half-length and depth, the width for flaw #2 falls outside those in the training set.

Figure 3 shows a comparison of the network's estimates for flaw depth with the actual depth for each of the ten measurements on flaws #2 and #4. The network's estimates show excellent agreement with the actual flaw depth, and all estimates are within about ±5% of the actual size.

DISCUSSION

Because space is limited, we are not able to present results estimating flaw half-length and flaw width for either approach. However, the results of depth estimation for both cases are representative of estimations for all three dimensions. Some notable exceptions occurred in the experimental case. For half-length, the network estimates fell within ±3% for flaw #2, but overestimated c by as much as 25% for flaw #4. Width estimations for flaw #2 were deemed invalid because its actual width fell outside that of the training set, but estimations of Δu for flaw #4 were within +20%.

Several steps can be taken to improve performance of the networks, particularly for experimental data. One such step is to measure more flaws, thereby creating a larger training set. We believe that this will improve performance for experimental data as it did with synthetic data. This means, however, measuring a large number of flaws covering a wide range of shapes and sizes. One way to overcome this problem is to train a network with synthetic data, from which a large training set can easily and readily be obtained, and then test the network with experimental data. This would provide a virtually unlimited training set size and could be configured to meet any criteria.

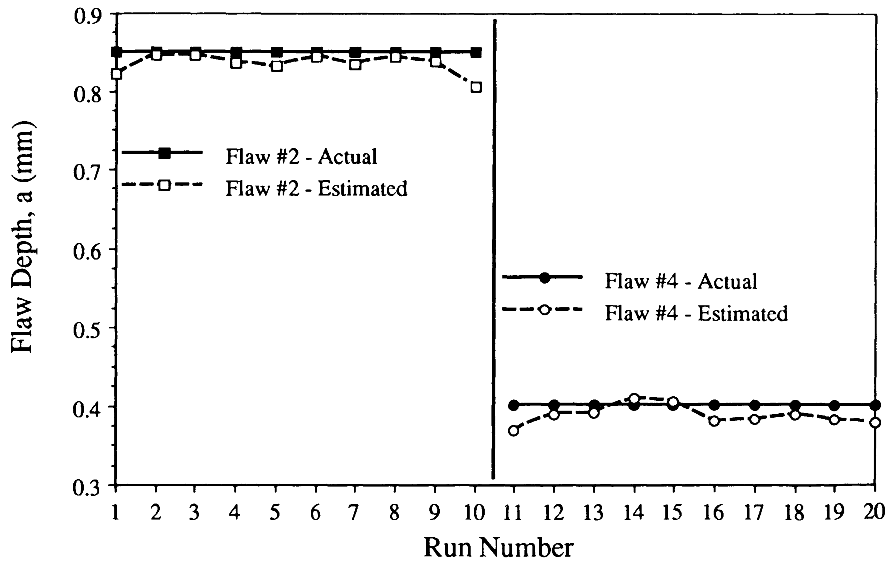

Figure 3. Results of testing after having trained a neural network with experimental data. The graph shows estimated and actual flaw depths for each of 10 independent measurements on flaws #2 and #4 (see Table 1).

No study was conducted to determine the optimal size, structure, or learning parameters of the networks. Almost certainly, other networks exist which will give better performance and an effort will be made in future work to find such networks. Possibilities for future study in this area include examining other network sizes, structures, and paradigms to find potentially better solutions. In fact, recent study indicates that possibly too many processing elements were used in the hidden layers for both approaches. This can hinder a network's ability to generalize as well as increase learning time.

CONCLUSIONS

Results from both the theoretical and experimental approaches to training neural networks for the inversion of uniform field eddy current data are very encouraging, but it should be emphasized that the results presented here are preliminary. There is certainly much more work to be done, especially with experimental data. Results from both parts of this study have shown the need for proper training sets: a large number of examples as demonstrated by the theoretical results, and a thorough and complete coverage of ranges as demonstrated by the experimental results. As our results demonstrate, neural networks show great promise in being able to solve the inverse problem for eddy current data.

ACKNOWLEDGMENTS

This work was supported by the Center for NDE at Iowa State University and was performed at the Ames Laboratory. Ames Laboratory is operated for the U. S. Department of Energy by Iowa State University under Contract No. W-7405-ENG-82. We are grateful to P. J. Shull of NIST-Boulder for the loan of flaw specimens.

REFERENCES

1. B. A. Auld, S. R. Jefferies, and J. C. Moulder, J. Nondestruct. Eval. 7, 79-94 (1988).
2. L. David Sabbaugh and Harold A. Sabbaugh, J. Nondestruct. Eval 7, 95-110 (1988).
3. L. Udpa and S. S. Udpa, J. Nondestruct. Eval. 7, 111-120 (1988).
4. B. A. Auld, F. G. Muennemann, and M. Riaziat, in Research Techniques in Nondestructive Evaluation, edited by R. S. Sharpe (Academic Press, London, 1984), Vol. VII, Chap. 2.
5. B. A. Auld, S. R. Jefferies, J. C. Moulder, and J. C. Gerlifz, in Review of Progress in Quantitative NDE, edited by D. O. Thompson and D. E. Chimenti (Plenum Press, New York, 1985), Vol. 5, pp. 383-393.
6. E. Smith, in Review of Progress in Quantitative NDE, edited by D. O. Thompson and D. E. Chimenti (Plenum Press, New York, 1985), Vol. 5, pp. 177-187.
7. J. C. Moulder, P. J. Shull, and T. E. Capobianco, in Review of Progress in Quantitative NDE, edited by D. O. Thompson and D. E. Chimenti (Plenum Press, New York, 1986), Vol. 6, pp. 601-610.
8. P. J. Shull, T. E. Capobianco, and J. C. Moulder, in Review of Progress in Quantitative NDE, edited by D. O. Thompson and D. E. Chimenti (Plenum Press, New York, 1986), Vol. 6, pp. 695-703.
9. Hecht-Nielsen, R., in Neurocomputing, in press, (Addison-Wesley Publishing Company, spring, 1990), Chap. 2.
10. Rumelhart, D. E., Hinton, G. E. and Williams, R. J., in Parallel Distributed Processing: Exploration in the Microstructure of Cognition, edited by D. E. Rumelhart and J. L. McClelland, (M.I.T. Press, Cambridge, MA, 1986), pp. 318-363.
11. Wasserman, P. D., in Neural Computing: Theory and Practice, (Van Nostrand Reinhold, New York, 1989), Chap. 3.