2002

Neural network based iterative algorithms for solving electromagnetic NDE inverse problems

Pradeep Ramuhalli
Iowa State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

Neural network based iterative algorithms for solving electromagnetic NDE inverse problems

by

Pradeep Ramuhalli

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering (Communications and Signal Processing)

Program of Study Committee:
Lalita Udpa, Major Professor
Satish Udpa
Shanker Balasubramaniam
James McCalley
Fritz Keinert

Iowa State University

Ames, Iowa

2002

UMI Number: 3051495

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Graduate College
Iowa State University

This is to certify that the doctoral dissertation of

Pradeep Ramuhalli

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This thesis would not have been possible without the help and support of several people. I would like to thank my major professor, Dr. Lalita Udpa, for her guidance and advice during the course of my graduate studies. I would also like to thank the members of my committee, Dr. Satish Udpa, Dr. Shanker Balasubramaniam, Dr. James McCalley and Dr. Fritz Keinert, for their guidance and support. In particular, I would like to thank Dr. Satish Udpa for the personal interest he has taken in my work, and for his advice and help.

Thanks are also due to the members of the Materials Assessment Research Group for making my stay at Iowa State University an enjoyable experience. The students and staff affiliated with this group have always been ready with their help and advice. In particular, I would like to thank the group administrator, Mrs. Linda Clifford, for her advice, support and help during my stay here. I would also like to thank Dr. Robi Polikar, Dr. Mohammed Afzal and Dr. Jaejoon Kim for their helpful suggestions whenever I needed them.

The graduate fellowship established by Takano Co. Ltd., Nagano, Japan at the Department of Electrical and Computer Engineering at Iowa State University was an important ingredient in making this thesis possible. I would like to take this opportunity to thank the Takano fellowship committee at Iowa State University and the president of Takano Co. Ltd., Mr. Horii, for the fellowship award.

Finally, and most importantly, I would like to thank my parents and sister for their unwavering support during my (long) career as a student. They have always pushed me to attain higher goals, and my pursuit of a PhD would not have been possible without them.

ABSTRACT

The solution of inverse problems is of interest in a variety of applications ranging from geophysical exploration to medical diagnosis and non-destructive evaluation (NDE). Electromagnetic methods are often used in the nondestructive inspection of conducting and ferromagnetic materials. A crucial problem in electromagnetic NDE is signal inversion wherein the defect parameters must be recovered from the measured signals. Iterative algorithms are commonly used to solve this inverse problem. Typical iterative inversion approaches use a numerical forward model to predict the measurement signal for a given defect profile. The desired defect profile can then be found by iteratively minimizing a cost function. The use of numerical models is computationally expensive, and therefore, alternative forward models need to be explored. This thesis proposes neural network based forward models in iterative inversion algorithms for solving inverse problems in NDE.

This study proposes two different neural network based iterative inverse problem solutions. In addition, specialized neural networks forward models that closely model the physical processes in electromagnetic NDE are proposed and used in place of numerical forward models. The first approach uses basis function networks (radial basis function (RBFNN) and wavelet basis function (WBFNN)) to approximate the mapping from the defect space to the signal space. The trained networks are then used in an iterative algorithm to estimate the profile given the measurement signal. The second approach proposes the use of two networks in a feedback configuration. This approach stabilizes the solution process and provides a confidence measure of the inversion result. Furthermore, specialized finite element model based neural networks (FENN) are proposed to model the forward problem.

These networks are derived from conventional finite element models and offer several advantages over conventional numerical models as well as neural network based forward models. These neural networks are then applied in an iterative algorithm to solve the inverse problem. Results of applying these algorithms to several examples including synthetic magnetic flux leakage (MFL) data are presented.

1. INTRODUCTION

Non-destructive evaluation (NDE) is the science of inspecting materials for flaws without compromising their usefulness. A generic NDE system is shown in Figure 1. An input transducer is used to couple energy into the test sample. The response of the material-energy interaction is captured by means of a receiving transducer and the resulting signal is analyzed to determine the existence of a flaw in the specimen. A wide range of energy sources including electromagnetic, ultrasonic and x-rays have been used in different applications. Commonly used electromagnetic techniques for NDE include magnetic flux leakage and eddy current methods.

The primary objective of NDE is to characterize the flaw based on the measurement signal. This can be accomplished using signal classification algorithms or alternately via algorithms for estimating flaw parameters given the measured signal. The problem of flaw characterization can be represented using a systems approach to NDE.

1.1. Inverse Problems in NDE

A typical NDE system can be represented by the linear model (Figure 2) where $x(t)$ is the excitation source, $y(t)$ is the probe measurement and $H(\omega)$ is the transfer function of the field/flaw interaction [1]. Three classes of problems may be defined using this approach:

(i) Given input $x(t)$ and system $H(\omega)$, determine the output $y(t)$.

(ii) Given input $x(t)$ and output $y(t)$, determine $H(\omega)$.

(iii) Given system $H(\omega)$ and the output $y(t)$, determine $x(t)$.

The first case presents the forward problem while the second and third cases are related to inverse problems. The second problem is one of system identification and the third

is commonly referred to as deconvolution. In NDE, the forward problem involves estimating the measurement signal due to a flaw and applied input energy whereas inverse problems involve the estimation of defect parameters using information contained in the measurement signal. Defect parameters can range from simple estimates of equivalent length, width and depth to a full three-dimensional profile.

An inverse problem is said to be well-posed in the sense of Hadamard if the solution satisfies three properties:

(i) Existence

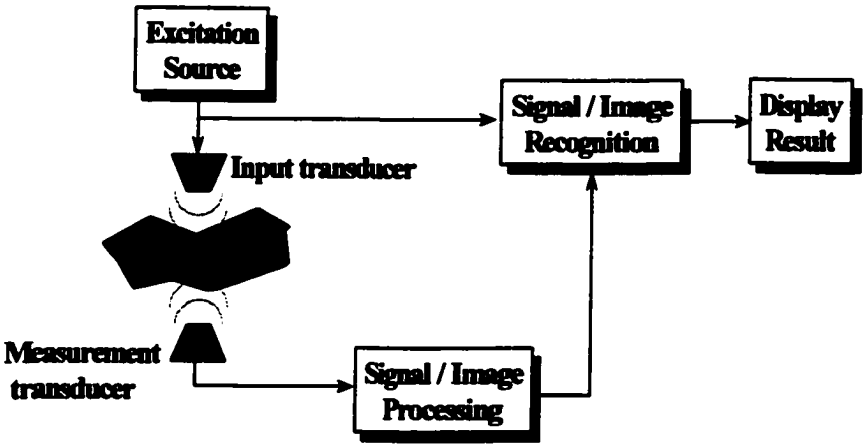


Figure 1. A generic NDE system.

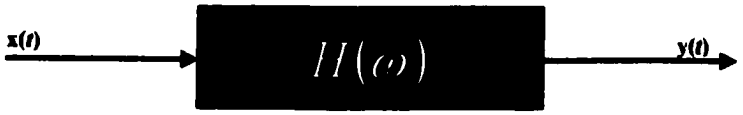


Figure 2. Systems approach to NDE.

(ii) Uniqueness and

(iii) Continuity: the solution depends continuously on the input.

The forward problem in general is well-posed and is solved analytically or by means of numerical modeling. In contrast, inverse problems in general are ill-posed, lacking both uniqueness and continuous dependence of the measured signals on defects. This has resulted in the development of a variety of solution techniques ranging from simple calibration procedures to other direct and iterative approaches [2]. These solution techniques can be divided into two broad categories: phenomenological and non-phenomenological approaches.

The first class of approaches – non-phenomenological approaches – attempts to solve the inverse problem by using signal processing techniques. A survey of signal processing methods as applied to inverse problems in NDE is available in [2]. These methods typically range from simple calibration methods to the more recent procedures based on neural networks. Calibration curves are obtained by first generating a set of signals either experimentally or numerically for a range of defect parameter values. A family of calibration curves is obtained by plotting signal features such as peak values against defect parameters such as defect depth while holding other parameters constant. These curves are then used to predict defect parameters for a given signal. Direct solutions involve mapping the measured signal directly to the flaw parameters. An example of this is approach is the use of neural networks to map the measurement to the required defect profile. In this case, the problem is formulated as a function approximation problem and the underlying function mapping the input signal to the output (profile) is "learnt" by a neural network.

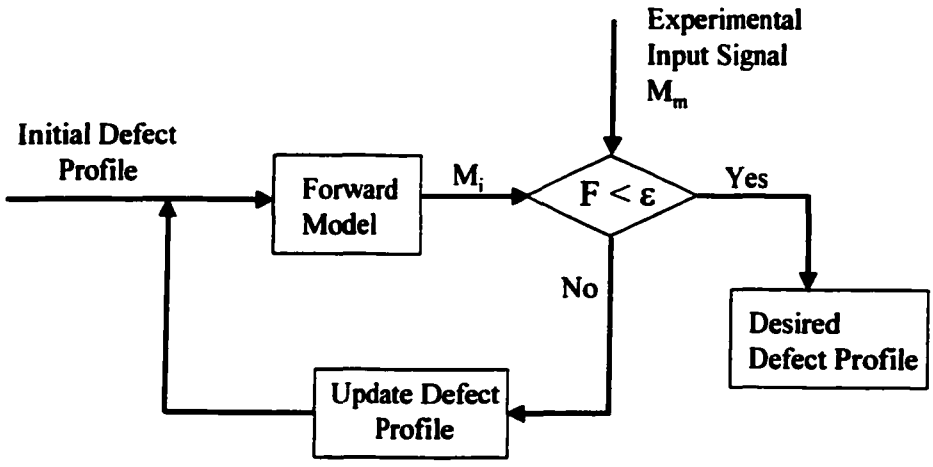


Figure 3. Iterative inversion method for solving inverse problems.

Phenomenological approaches can be direct or iterative. Iterative approaches typically employ a forward model that simulates the underlying physical process (Figure 3) [3]. The physical process in NDE is usually represented by means of differential or integral equations, and the corresponding forward model is typically a numerical model such as a finite element model. The algorithm starts with an initial estimate of the defect parameters and solves the corresponding forward problem to determine the signal. The error between the measured and predicted signals is minimized iteratively by updating the defect profile. When the error is below a pre-set threshold, the defect parameters represent the desired solution.

Methods utilizing both phenomenological and non-phenomenological approaches have been reported extensively in literature [4]. Iterative techniques for inverse problems in NDE have been developed using numerical models [3, 5] based on integral and differential formulations [6, 7, 8] to represent the forward process. In addition, various non-phenomenological approaches have also been reported. For instance, Hwang et al report the use of a wavelet basis function neural network to learn the profile for a given MFL signal [9].

However, all of these methods have certain drawbacks. Iterative methods using three-dimensional numerical models are, in general, computationally intensive, and therefore have limited practical application. In addition, updating the defect profile is also difficult, since gradient-based approaches cannot be easily applied to solutions of numerical models such as finite element models. On the other hand, neural network based non-phenomenological techniques are open loop in nature and are capable of providing a confidence measure of the accuracy only during the training phase. In addition, non-phenomenological techniques suffer from the drawback that they cannot be used in cases where the solution is non-unique.

The major objective of this study is the development of solutions to inverse problems in NDE that overcome the disadvantages of the conventional approaches. The proposed solutions are described below with a brief discussion of their advantages and disadvantages. In addition, we also compare the proposed solution methods to existing algorithms presented in the literature, and present the differences between the existing and proposed algorithms.

1.2. Neural Network Based Iterative Inversion Algorithms

In this study, we focus on developing solutions that try to incorporate the best components of both phenomenological and non-phenomenological approaches. Specifically, two different neural network based approaches to solving the inverse problem are proposed. Both approaches are iterative in nature and involve the use of neural networks as forward models. These approaches are

1. **Approach I: Neural network based iterative inversion**: A single neural network is used instead of a numerical model as the forward model in the inversion approach shown in Figure 3. The advantages of this approach include its speed and simplicity as the forward model inherits many of the advantages of neural networks. Similar

approaches have been used in the past in sonar performance analysis, power system security assessment and control (a survey of iterative inversion algorithms and applications is given in [10]). However, the proposed approach is different since it uses radial basis function and wavelet basis function neural networks to model the forward process, as opposed to multilayer perceptron neural networks used in the literature.

2. **Approach II: Feedback Neural Networks**: This approach is a modification of Approach I, and uses two neural networks in feedback configuration, with one neural network modeling the forward problem while the other models the inverse problem. This approach allows us to incorporate the underlying physics of the problem in the forward model, thus providing an accurate solution. The suggested technique is also capable of incremental learning, provides an online measure for accuracy of the defect estimate, and is computationally efficient.

The major drawback of both Approach I and Approach II is that the performance of the neural networks depends on the data used in training and testing. Mathematically, each of the neural networks approximates the function mapping the input to the output, and as long the test data is similar to the training data, the network can interpolate between the training data points to obtain a reasonable prediction. However, when the test signal is no longer similar to the training data, the network is forced to extrapolate and the performance is seen to degrade. For example, a network trained with rectangular defects will not predict with high accuracy when a signal from a circular defect is given as input. This may be a disadvantage in NDE signal inversion, where the shape of the flaw is not known *a priori*. Hence, there is a need for developing neural networks that can "extrapolate".

An easy way around this difficulty is to ensure that the training database has enough data to cover a diverse set of signals. However, this is difficult to ensure in practice, and even if we can generate data from all possible defect profiles, the database and associated storage costs would be enormous. Alternatively, we have to consider the design of neural networks that are capable of extrapolation. Extrapolation methods are discussed extensively in the literature ([11, 12, 13, 14]), but the design of an extrapolation neural network involves several issues. For instance, there are no methods for ensuring that the error in the network prediction is within reasonable bounds during the extrapolation procedure.

Model based methods for extrapolation use numerical models in an iterative approach. These models are capable of correctly predicting the signal given any reasonable defect profile, since they solve the underlying governing equations. However, numerical models are computationally intensive, which limits their application. An ideal solution therefore would be to combine the power of numerical models with the computational speed of neural networks, i.e., to create neural networks that are capable of solving the underlying partial differential equations in an electromagnetic NDE problem. Specifically, we are interested in designing neural networks that are closely related to a numerical model such as the finite element model. This finite element neural network (FENN) can then be used as the forward model in either Approach I or Approach II to solve the inverse problem. Using a numerical model in a neural network framework allows parallel implementation, thus resulting in potential savings in computational effort. Furthermore, the neural network would require a minimal amount of training and therefore, the iterative algorithm would be faster, and not be training database dependent like regular neural networks.

Finite element based neural networks have not been explored extensively in the literature. A large number of publications refer to the finite element neural network as a neural network that has been trained using FEM data (for instance, [15, 16]). One of the few finite element neural network formulations that have been reported has been by Takeuchi and Kosugi [17]. This approach is based on error minimization and the neural network is designed from the energy functional derived during the finite element method. Furthermore, the network is designed to solve the forward problem, and must be modified to solve the inverse problem. Other reports of finite element neural network combinations are either similar to the Takeuchi method [18, 19] or use Hopfield neural networks to solve the forward problem [20]. The use of Hopfield networks makes it difficult to solve the inverse problem, especially if derivatives need to be computed in the inversion procedure. Such networks are therefore not considered in this study. An alternative neural network approach to solving differential equations is proposed in [21]. Here, the solution to a differential equation is written as a sum of two terms, with one term having no adjustable parameters and a second term with adjustable parameters. These parameters are modeled using a neural network, and a training procedure is used to determine the optimal parameter set. Boundary conditions are taken into account when the two terms are formed. The drawback of this approach is that it is limited to rectangular domains. In addition, the neural network requires a training stage.

Our proposed approach is different in that we derive the neural network from the point of view of the inverse problem. The neural network architecture that is eventually developed also makes it easy to solve the forward problem. The structure of the neural network is also simpler than those reported in the literature, making it easier to implement in parallel in both hardware and software. Furthermore, the neural network is not limited to a

specific type of domain, and does not require any training. In fact, the FENN weights are determined solely by the differential equation and associated boundary conditions – an advantage in solving inverse problems.

1.3. Organization of this Dissertation

This thesis is organized as follows. Chapter 2 presents a brief introduction to neural networks. This section contains a description of the radial basis function neural network (RBFNN) and the wavelet basis function neural network (WBFNN), along with an introduction to function approximation. The relationship between these neural networks and function approximation theory is also shown. Chapters 3 and 4 present the proposed approaches to solving inverse problems, namely, a simple neural network based iterative inversion algorithm and a feedback neural network algorithm respectively. The necessary update equations along with a complete description of the algorithm are presented. Results of applying these algorithms are also presented in the corresponding sections. This is followed by a description of the proposed finite element neural network in Chapter 5, along with initial results. Chapter 5 contains an introduction to finite element models, the formulation of the FENN, and the necessary update equations for solving the forward and inverse problems. An analysis of the sensitivity of the FENN to measurement errors is also provided in Chapter 5. Finally, Chapter 6 summarizes the various approaches and presents ideas for future work. In addition, a brief introduction to magnetic flux leakage theory is provided in the Appendix.

2. NEURAL NETWORKS

Neural networks are connectionist models proposed in an attempt to mimic the function of the human brain. A neural network consists of a large number of simple processing elements called neurons (or nodes) [22, 23]. Neurons implement simple functions and are massively interconnected by means of weighted interconnections. These weights, determined by means of a training process, determine the functionality of the neural network. The training process uses a training database to determine the network parameters (weights).

The functionality of the neural network is also determined by its topology. Most networks have a large number of neurons, with the neurons arranged in layers. In addition to input and output layers, there are usually layers of neurons that are not directly connected to either the input or the output, called hidden layers. The corresponding nodes are referred to as hidden nodes. Hidden layers give the network the ability to approximate complex, nonlinear functions.

The advantages of using neural networks are numerous: neural networks are learning machines that can learn any arbitrary functional mapping between input and output, they are fast machines and can be implemented in parallel, either in software or in hardware. In fact, the computational complexity of neural networks is polynomial in the number of neurons used in the network. Parallelism also brings with it the advantages of robustness and fault tolerance. Efficient learning algorithms ensure that the network can learn mappings to any arbitrary precision in a short amount of time. Furthermore, the input-output mapping is explicitly known in a neural network and gradient descent procedures can be used advantageously to perform the inversion process.

Neural networks have been widely used for function approximation and multidimensional interpolation [23]. Given a set of p ordered pairs $(\mathbf{x}_i, d_i)$, $i = 1,2,..., p$ with $\mathbf{x}_i \in \mathbf{R}^N$ and $d_i \in \mathbf{R}$, the problem of interpolation is to find a function $F: \mathbf{R}^N \rightarrow \mathbf{R}^1$ that satisfies the interpolation condition

$$F(\mathbf{x}_i) = d_i, \quad i = 1,2,..., p \tag{2.1}$$

For strict interpolation, the function F is constrained to pass through all the p data points. The definition can be easily extended to the case where the output is M-dimensional. The desired function is then $F: \mathbf{R}^N \rightarrow \mathbf{R}^M$.

In practice, the function F is unknown and must be determined from the given data $(\mathbf{x}_i, d_i)$, $i = 1,2,..., p$. A typical neural network implementation of this problem is a two-step process: Training, where the neural network learns the function F given the *training data* $\{\mathbf{x}_i, \mathbf{d}_i\}$, and generalization, where the neural network predicts the output for a test input. Two different neural network architectures for interpolation are described in the sections below.

2.1. Regularization Theory

The problem of estimating the function F above can be thought of as an ill-posed problem, since the solution is in general not unique. Additional constraints are therefore necessary to convert the ill-posed problem to a well-posed one. Standard regularization procedures involve imposing additional constraints on the solution space by defining an error function. Consider the interpolation problem defined above. Let the desired function be represented by $F(\mathbf{x})$. Then, according to Tikhonov regularization theory [23, 24, 25], the function F can be obtained by minimizing an error functional given by

$$E(F) = E_s(F) + \lambda E_r(F) \tag{2.2}$$

where λ is the regularizing parameter, E_s is the standard error between the desired output and the actual response y

$$E_s = \frac{1}{2}\sum_{i=1}^{p}(d_i - y_i)^2 \tag{2.3}$$

and E_r is the regularizing term that depends on the properties of F. If $\mathbf{P}$ is a linear pseudo-differential operator embedding a smoothness constraint,

$$E_r = \frac{1}{2}\|\mathbf{P}F\|^2 \tag{2.4}$$

The resulting solution is smooth and therefore, continuous.

In order to find F that minimizes the total error, we differentiate E with respect to F using the Fréchet differential [23] and set it equal to zero.

$$dE(F,h) = 2\left[h, \mathbf{P}^*\mathbf{P}F - \frac{1}{\lambda}\sum_{i=1}^{p}(d_i - F)\delta_{\mathbf{x}_i}\right]_H \tag{2.5}$$

where $h(\mathbf{x})$ is a fixed function of the vector $\mathbf{x}$, $\delta_{\mathbf{x}_i} = \delta(\mathbf{x} - \mathbf{x}_i)$, $\mathbf{P}^*$ is the adjoint of $\mathbf{P}$, and the symbol $(.,.)_H$ denotes the inner product in H space. Since $\lambda \in (0,\infty)$, the Fréchet differential is zero for any $h(\mathbf{x})$ in H if and only if

$$\mathbf{P}^*\mathbf{P}F = \frac{1}{\lambda}\sum_{i=1}^{p}(d_i - F)\delta(\mathbf{x} - \mathbf{x}_i) \tag{2.6}$$

Equation (2.6) is referred to as the Euler-Lagrange equation for the cost functional $E(F)$ and its solution is given by

$$F(\mathbf{x}) = \int_{\mathbf{R}^N} G(\mathbf{x},\boldsymbol{\theta})\frac{1}{\lambda}\sum_{i=1}^{p}(d_i - F(\boldsymbol{\theta}_i))\delta(\boldsymbol{\theta} - \mathbf{x}_i)\,d\boldsymbol{\theta} \tag{2.7}$$

where θ is the variable of integration and $G(\mathbf{x}, \mathbf{x}_i)$ is the Green's function for the self-adjoint operator $\mathbf{P}^*\mathbf{P}$, i.e.,

$$\mathbf{P}^*\mathbf{P}G(\mathbf{x}, \mathbf{x}_i) = \delta(\mathbf{x} - \mathbf{x}_i) \tag{2.8}$$

Integrating, we get

$$F(\mathbf{x}) = \frac{1}{\lambda}\sum_{i=1}^{p}(d_i - F(\mathbf{x}_i))G(\mathbf{x}, \mathbf{x}_i) \tag{2.9}$$

which can be written in matrix-vector form as

$$\mathbf{F} = \mathbf{G}\mathbf{w} \tag{2.10}$$

with

$$\mathbf{w} = \frac{1}{\lambda}(\mathbf{d} - \mathbf{F}) \tag{2.11}$$

and

$$\mathbf{G} = \begin{bmatrix} G(x_1, x_1) & G(x_1, x_2) & \ldots & G(x_1, x_p) \\ G(x_2, x_1) & G(x_2, x_2) & \ldots & G(x_2, x_p) \\ \ldots & \ldots & & \ldots \\ \ldots & \ldots & & \ldots \\ G(x_p, x_1) & G(x_p, x_2) & \ldots & G(x_p, x_p) \end{bmatrix} \tag{2.12}$$

Since the operator $\mathbf{P}^*\mathbf{P}$ is self-adjoint, the associated Green's function and consequently the matrix **G** will be symmetric. Further, Light [26] has proved that the matrix **G** is positive definite provided that the data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$ are distinct. In practice, λ may be chosen to be sufficiently large so that the matrix $\mathbf{G} + \lambda\mathbf{I}$ is positive definite. This implies that the system of equations (2.10) has a unique solution given by

$$\mathbf{w} = (\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{d} \tag{2.13}$$

and the function F is given by

$$F(\mathbf{x}) = \sum_{i=1}^{p} w_i G(\mathbf{x}, \mathbf{x}_i) \tag{2.14}$$

The number of Green's functions used in this expansion is equal to the number of data points.

2.2. Radial Basis Function Neural Networks

The theory described above can be implemented as a radial basis function (RBF) neural network. Radial basis function (RBF) neural networks are a class of networks that are widely used for solving multivariate function approximation problems [23, 24]. An RBF neural network consists of an input and output layer of nodes and a single hidden layer (Figure 4). Each node in the hidden layer implements a basis function $G(\mathbf{x}, \mathbf{x}_i)$ and the number of hidden nodes is equal to the number of data points in the training database. The RBFNN approximates the unknown function that maps the input to the output in terms of a basis function expansion, with the functions $G(\mathbf{x}, \mathbf{x}_i)$ as the basis functions. The input-output relation for the RBFNN is given by

$$y_l = \sum_{j=1}^{N} w_{lj} G(\mathbf{x}, \mathbf{x}_j), \quad l = 1,2,..., M \tag{2.15}$$

where N is the number of basis functions used, $\mathbf{y} = (y_1, y_2, ... y_M)^T$ is the output of the RBFNN, $\mathbf{x}$ is the test input, $\mathbf{x}_j$ is the center of the basis function and w_{lj} are the expansion coefficients or weights associated with each basis function. Each training data sample is selected as the center of a basis function. Basis functions $G(\mathbf{x}, \mathbf{x}_i)$ that are radially symmetric are called radial basis functions. Commonly used radial basis functions include the Gaussian and inverse multiquadrics.

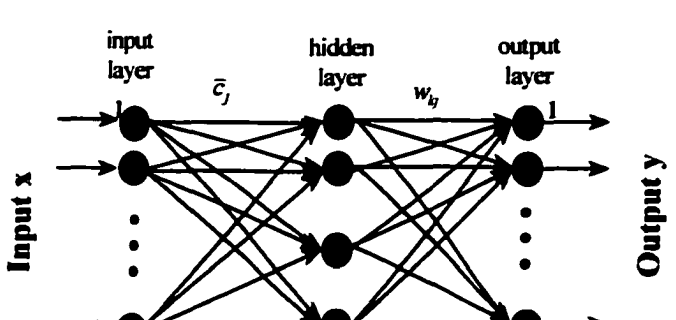


Figure 4. The radial basis function neural network.

The network described above is called an exact RBFNN, since each training data point is used as a basis center. The storage costs of an exact RBFNN can be enormous, especially when the training database is large. An alternative to an exact RBFNN is a generalized RBFNN where the number of basis functions is less than the number of training data points. The problem then changes from strict interpolation (in an exact RBFNN) to an approximation, where certain error constraints are to be satisfied. The operation of the generalized RBFNN is summarized in the following steps.

Step 1. Center selection: This is achieved by using either the K-means clustering algorithm [27, 28] or other optimization techniques that select the basis function locations by minimizing the error in the approximation. The input-output relation for a generalized RBFNN using Gaussian basis functions is given by

$$y_l = \sum_{j=1}^{H} w_{lj} \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2\sigma_j^2} \right) \tag{2.16}$$

where H is the total number of basis functions used, $\mathbf{c}_j$ is the center of the j^{th} Gaussian basis function and σ_j is the width of the Gaussian. The neural network architecture is then

selected by setting the number of input nodes equal to the input dimension, the number of hidden nodes to the number of centers obtained in Step 1, and the number of output nodes equal to the output dimension.

Step 2. Training: Training of the neural network involves determining the weights w_{lj}, in addition to the centers and widths of the basis functions. Writing (2.16) in matrix-vector form as

$$\mathbf{Y} = \mathbf{GW} \tag{2.17}$$

where

$$\mathbf{Y} = \begin{bmatrix} \mathbf{d}_1^T \\ \mathbf{d}_2^T \\ \cdots \\ \mathbf{d}_P^T \end{bmatrix} \tag{2.18}$$

is the desired M-dimensional output for all P input samples,

$$\mathbf{G} = \begin{bmatrix} G(\mathbf{x}_1,\mathbf{c}_1) & G(\mathbf{x}_1,\mathbf{c}_2) & \cdots & G(\mathbf{x}_1,\mathbf{c}_H) \\ G(\mathbf{x}_2,\mathbf{c}_1) & G(\mathbf{x}_2,\mathbf{c}_2) & \cdots & G(\mathbf{x}_2,\mathbf{c}_H) \\ \cdots\cdots & & & \\ G(\mathbf{x}_P,\mathbf{c}_1) & G(\mathbf{x}_P,\mathbf{c}_2) & \cdots & G(\mathbf{x}_P,\mathbf{c}_H) \end{bmatrix} \tag{2.19}$$

is the output of the basis functions,

$$\mathbf{W} = \{w_{jl}\}, \quad j = 1,2,...,H, \quad l = 1,2,...,M \tag{2.20}$$

is the weight matrix and M is the output dimension. Equation (2.17) can be solved for $\mathbf{W}$ as:

$$\mathbf{W} = \mathbf{G}^+\mathbf{Y} \tag{2.21}$$

where $\mathbf{G}^+$ is the pseudoinverse defined as

$$\mathbf{G}^+ = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T. \tag{2.22}$$

Step 3. Generalization: In the test phase, the unknown pattern $\mathbf{x}$ is mapped using the relation

$$F(\mathbf{x}) = \sum_{j=1}^{H} w_{lj} \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2\sigma_j^2} \right) \tag{2.23}$$

2.3. Wavelet Basis Function Neural Networks

The wavelet transform is a time-frequency transform that provides both the frequency as well as time localization in the form of a multiresolution decomposition of the signal [29].

Consider a square-integrable function $F(x)$ and let V_m be the vector space containing all possible projections of F at the resolution m where 2^m is the sampling interval at this resolution [30]. Obviously, as m increases, the number of samples at that resolution decreases and the approximation gets coarser. Now, consider all approximations of F at all resolutions. The associated vector spaces are nested as follows

$$\ldots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \ldots \tag{2.24}$$

due to the fact the finer resolutions contain all the required information to compute the coarser approximation of the function F. It is also obvious that as the resolution decreases, the approximation gets coarser and contains less and less information. In the limit, it converges to zero:

$$\lim_{m \to \infty} V_m = \bigcap_{m=-\infty}^{\infty} V_m = \{0\} \tag{2.25}$$

On the other hand, as the resolution increases, the approximation has more information and eventually converges to the original signal:

$$\lim_{m \to -\infty} V_m = \bigcup_{m=-\infty}^{\infty} V_m \text{ is dense in } L^2(\mathrm{R}) \tag{2.26}$$

Mallat [31] showed that a unique function, called the scaling function exists such that the family of functions resulting from the translation and dilation of the scaling function forms an orthonormal basis for V_m. In other words, if $\phi(x)$ denotes the scaling function, then

$$V_m = \text{linear span}\{\phi_{mk}, k \in Z\} \tag{2.27}$$

where

$$\phi_{mk} = \sqrt{2^{-m}}\,\phi\left(2^{-m}x - k\right), \quad (m,k) \in Z^2 \tag{2.28}$$

is the dilated and translated version of $\phi(x)$.

Since the family of functions $\left\{\phi_{mk}(x) \mid (m,k) \in Z^2\right\}$ forms an orthonormal basis for V_m, F can be written as

$$F_m(x) = \sum_{k=-\infty}^{\infty} s_{mk}\phi_{mk}(x) \tag{2.29}$$

where

$$s_{mk} = \int_{-\infty}^{\infty} F(x)\phi_{mk}(x)dx \tag{2.30}$$

is the projection of F onto the orthonormal basis functions $\phi_{mk}(x)$.

Further, suppose W_m is the orthogonal complement of V_m in V_{m-1}. Then

$$V_{m-1} = V_m \oplus W_m \text{ with } V_m \perp W_m. \tag{2.31}$$

The $(m\text{-}1)^{\text{th}}$ approximation can be written as the sum of the projections of F onto V_m and W_m. Equivalently, the difference in information (called the detail) between the m^{th} and $(m\text{-}1)^{\text{th}}$ approximations is given by the projection of F onto W_m. Mallat [31] shows that there exists a unique function, called the wavelet function, whose translates and dilates form an orthonormal basis for the space W_m. In other words, the detail of F at the m^{th} resolution is given by

$$D_m F(x) = \sum_{k=-\infty}^{\infty} d_{mk} \psi_{mk}(x) \tag{2.32}$$

where $\psi(x)$ is the wavelet,

$$\psi_{mk}(x) = \sqrt{2^{-m}}\,\psi\left(2^{-m}x - k\right), \quad (m,k) \in Z^2 \tag{2.33}$$

are the translates and dilates of $\psi(x)$ and

$$d_{mk} = \int_{-\infty}^{\infty} F(x)\psi_{mk}(x)dx \tag{2.34}$$

are the projections of F onto W_m. Further, from (2.31), we get

$$F_{m-1}(x) = F_m(x) + \sum_{-\infty}^{\infty} d_{mk}\psi_{mk}(x) \tag{2.35}$$

Since the V-spaces form a nested set of subspaces, F can be written as

$$F(x) = \sum_{k=-\infty}^{\infty} s_{k,-\infty}\phi_{k,-\infty}(x) + \sum_{l=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} d_{lk}\psi_{lk}(x) \tag{2.36}$$

where l indexes over the different resolutions. In practice, the limits of summation are chosen to be finite.

Figure 5 shows an example of the multiresolution decomposition of a signal into six levels. Figure 5(a) is the original signal while 5(b) shows the approximation at the coarsest level. Figures 5(c)-(h) show the details at different levels of resolution.

A neural network architecture that implements a multiresolution approximation is shown in Figure 6. The network consists of an input and an output layer with a single hidden layer of nodes [30]. The hidden layer nodes are grouped by resolution level. We have as many groups as resolution levels, with the number of basis functions at each resolution decided by a dyadic center selection method to be described later. The input-output relation is given by

$$y_l = \sum_{j=1}^{H_1} w_{lj}\phi_j(\mathbf{x}, \mathbf{c}_j) + \sum_{n=1}^{L}\sum_{k=1}^{K_n} w_{lkn}\psi_{nk}(\mathbf{x}, \mathbf{c}_{nk}) \tag{2.37}$$

where L is the total number of resolutions, H_1 is the number of scaling functions used at the coarsest resolution, K_n is the number of wavelet functions used at resolution n, $\mathbf{c}_i$ is the center of the corresponding basis function and w_{lj} is the weight of the interconnection connecting the j^{th} hidden node to the l^{th} output node. The weights are determined in a similar manner to the weights in the RBFNN described earlier.

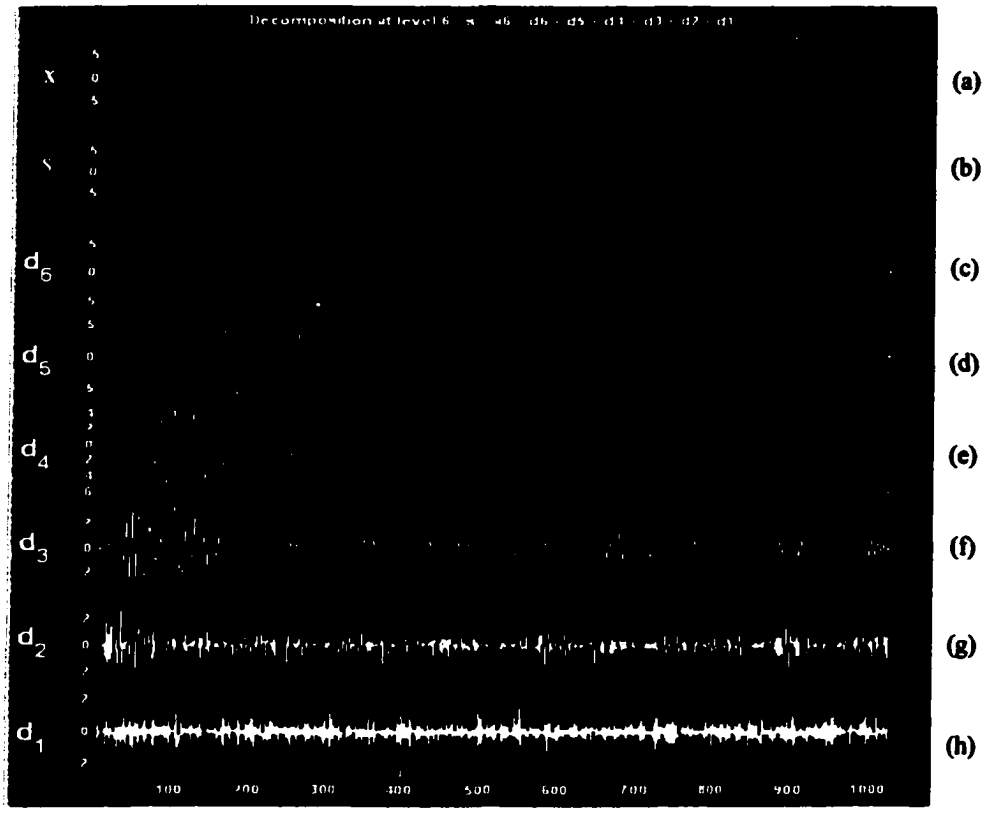


Figure 5. Multiresolution analysis.

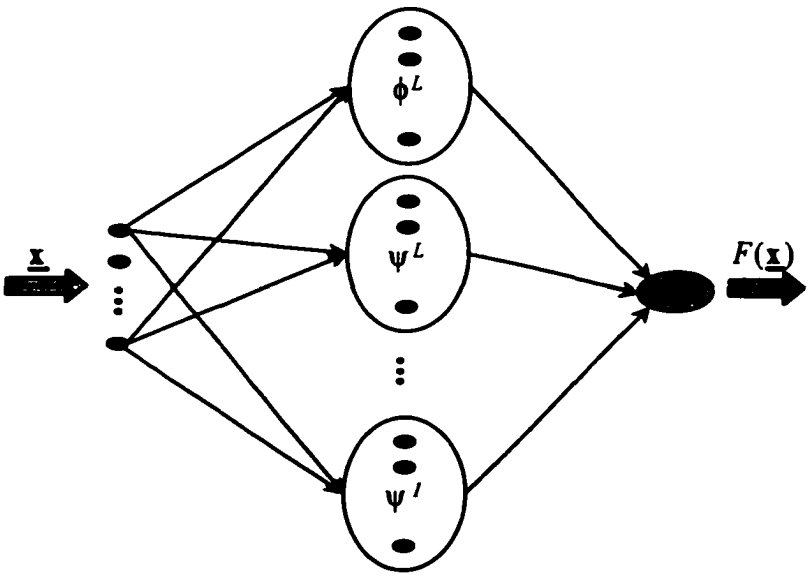


Figure 6. The wavelet basis function neural network.

The primary advantage of using wavelet basis functions is orthonormality [30]. Orthonormality of wavelets ensures that the number of basis functions required to approximate the function F is minimum. The second advantage is that wavelets are local basis functions (localization property of wavelets [30]). The multiresolution approximation (MRA) using wavelets allows distribution of basis functions based on the resolution required in different parts of the input space. In addition, the ability to add details at higher resolutions as more data become available allows the network to learn in an incremental fashion and allows the user to control the degree of accuracy of the approximation.

Equation (2.36) formulated for scalar inputs can be extended for multidimensional inputs. The corresponding multidimensional scaling functions and wavelets are formed by tensor products of the one-dimensional scaling functions and wavelets. Consider the 2-

dimensional case with $\mathbf{x} = (x_1, x_2)^T$. Denoting the 1-D scaling function by $\phi(x)$ and the 1-D wavelet by $\psi(x)$, one can show that the two-dimensional scaling function is given by [30]

$$\Phi(x_1, x_2) = \phi(x_1)\phi(x_2) \tag{2.38}$$

Similarly, the corresponding wavelet functions are given by

$$\begin{aligned} \Psi^1(x_1, x_2) &= \phi(x_1)\psi(x_2) \\ \Psi^2(x_1, x_2) &= \psi(x_1)\phi(x_2) \\ \Psi^3(x_1, x_2) &= \psi(x_1)\psi(x_2) \end{aligned} \tag{2.39}$$

For an accurate approximation, all the four basis functions must be used at each hidden node. Kugarajah and Zhang [32] have shown that, under certain conditions, a radial basis scaling function $\phi(\|\mathbf{x} - \mathbf{x}_i\|)$ and wavelet $\psi(\|\mathbf{x} - \mathbf{x}_i\|)$ constitute a frame, and that these functions can be used in place of the entire N-dimensional basis, resulting in a savings in storage and execution time while minimally affecting the accuracy of the approximation.

The operation of wavelet basis function neural networks is summarized in the following steps.

Step 1. Basis Function Selection: A significant issue in wavelet basis function neural networks is the selection of the basis functions. The wavelet family used in the WBFNN depends on the form of the function F that must be reconstructed. Even though this function is usually unknown, some important details may be obtained by inspecting the problem at hand. For instance, classification usually calls for a discontinuous or quantized function F where all the input data is to be mapped onto one of a few classes. In such cases, discontinuous wavelets, such as the Haar wavelet, may be used. Continuous wavelets may be used to approximate smoother functions.

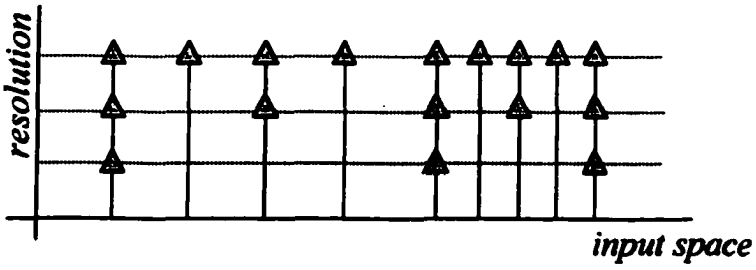


Figure 7. Dyadic center selection scheme.

Step 2. Center Selection: The location and number of basis functions are important since they determine the architecture of the neural network. Centers at the first (or coarsest) resolution are selected by using the K-means algorithm. Centers at finer resolution levels are selected using a dyadic scheme (Figure 7) [33]. Each center at successive resolutions is computed as the mean of two centers at a lower resolution.

Step 3. Training: Training the network involves determining the expansion coefficients associated with each resolution level. These coefficients are determined by using a matrix inversion operation, similar to the operation performed in RBF neural networks. The centers can also be dynamically varied during the training process till the error in the network prediction falls below a predetermined level. Over-fitting by the network can be avoided by pruning the centers one by one until the network performs at an acceptable level on a blind test database. In this study however, no optimization is performed after center selection.

Step 4. Generalization: In this step, the trained WBFNN is used to predict the output for a new test signal using (2.37).

3. ELECTROMAGNETIC NDE SIGNAL INVERSION USING NEURAL NETWORK FORWARD MODELS

The iterative inversion algorithm for electromagnetic signal inversion shown in Figure 3 was implemented using the neural networks as the forward model to predict the measurement signal given the defect profile. The algorithm starts with an initial estimate of the defect profile and computes the signal for this profile. This signal is compared to the measured signal. The basic principle underlying this algorithm is that, if the predicted signal is similar to the measured signal, then the corresponding defect profile is close to the desired defect profile. If the signals are not similar, the defect profile is updated iteratively to minimize the error.

The key element in the proposed approach is the forward model. The use of numerical models such as finite element models (FEM) in an iterative procedure is computationally expensive. In this section, the function mapping the input (defect profile) to the signal at the output is approximated using neural networks. Two different neural networks, namely the RBFNN and the WBFNN discussed earlier, are used to model the forward process.

The defect profile $\mathbf{x}_i$ and the corresponding signal $\mathbf{y}_i$ are presented to the input and output layer nodes respectively. The sum-squared error between the desired output $\mathbf{d} = (d_1, d_2, ...d_M)$ and the actual output $\mathbf{y} = (y_1, y_2, ...y_M)$ of the neural network is computed as

$$E = \frac{1}{2}\sum_{i=1}^{M}(d_i - y_i)^2 \tag{3.1}$$

where M is the number of output nodes for the neural network (equal to the dimensionality of the output). The defect profile is updated using a combination of gradient descent and

simulated annealing to minimize the error. In the gradient descent algorithm [34], at each iteration, the input is changed according to the relation

$$\mathbf{x}^{new} = \mathbf{x}^{old} + \eta\left(-\frac{\partial E}{\partial \mathbf{x}}\right) \tag{3.2}$$

where $\mathbf{x}$ is the input and η is called the learning rate. η controls the rate of convergence of the algorithm. Convergence is achieved when E falls below a given threshold.

One of the drawbacks of using gradient descent for minimization is that the algorithm may converge to a local minimum in the error surface. The use of simulated annealing in conjunction with gradient descent allows the algorithm to explore the error surface more thoroughly, resulting in a globally optimal solution. Simulated annealing is also an optimization algorithm [34] that simulates the annealing process in material science. Annealing is the process of gradually cooling a liquid till it freezes. By doing so, the material reaches a state of very low energy. It can be proved that if the temperature is lowered sufficiently slowly, the material will attain the lowest-energy configuration (optimal state).

In order to apply this technique to combinatorial optimization, we define a temperature variable T along with the cooling schedule. The cooling schedule determines how rapidly the temperature is lowered. An exponentially slow cooling schedule is defined as one where

$$1 + k \geq \exp\left(\frac{T_0}{T_k}\right) \tag{3.3}$$

with T_k being the temperature at iteration k and T_0 is a sufficiently large initial temperature. An exponential cooling schedule finds the optimal solution, if it exists. However, convergence time can be significantly large and in practice,

$$T_k = \alpha T_{k-1}, \qquad \alpha < 1 \tag{3.4}$$

is used.

Simulated annealing may be used in combination with gradient descent to adjust the learning rate. This is used to speed up the convergence of the algorithm. Once the network is trained, the complete algorithm for inverting a new test signal is summarized below:

Given the signal from an unknown defect profile,

(i) Initialize the defect profile randomly. Call this the estimated profile $\mathbf{x}(0)$.

(ii) At iteration k, present the defect profile $\mathbf{x}(k)$ to the neural network. Compute the predicted MFL signal

$$\mathbf{y}(k) = F(\mathbf{x}(k)) \tag{3.5}$$

(iii) Compute the sum-squared error

$$E(k) = \frac{1}{2}\|\mathbf{d} - \mathbf{y}(k)\|^2 \tag{3.6}$$

(iv) Compute $\partial E(k)/\partial \mathbf{x}(k)$, the gradient of E with respect to $\mathbf{x}(k)$.

(v) Update $\mathbf{x}$:

$$\hat{\mathbf{x}}(k+1) = \mathbf{x}(k) + \eta\left(-\frac{\partial E(k)}{\partial \mathbf{x}(k)}\right) \tag{3.7}$$

(vi) Compute the output of the neural network:

$$\hat{\mathbf{y}}(k+1) = F(\hat{\mathbf{x}}(k+1)) \tag{3.8}$$

(vii) Compute the new error

$$E(k+1) = \frac{1}{2}\|\mathbf{d} - \hat{\mathbf{y}}(k+1)\|^2 \tag{3.9}$$

(viii) If $E(k+1) < E(k)$, set

$$\mathbf{x}(k+1) \leftarrow \hat{\mathbf{x}}(k+1), \tag{3.10}$$

$$\eta \leftarrow \eta \times \eta_{inc} \tag{3.11}$$

where η_{inc} is an increment factor.

(ix) If $E(k+1) > E(k)$, set

$$\eta \leftarrow \eta \times \eta_{dec}. \tag{3.12}$$

(x) Go to (ii). Repeat till $E(k) < ERROR\_THRESHOLD$.

In order to implement the algorithm above, it is necessary to compute the gradient of the error between the predicted and measured signals with respect to the input. This requires the input-output relationship for the different neural networks. In the RBFNN with Gaussian basis functions, the output and the input are related by a basis function expansion as given in (2.16) and repeated here:

$$y_l = F(\mathbf{x}) = \sum_{j=1}^{H} w_{lj} \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2\sigma_j^2} \right). \tag{3.13}$$

Substituting in (3.1) and computing the derivative of E with respect to $\mathbf{x} = (x_1, x_2, ...x_N)$, we get

$$\frac{\partial E(\mathbf{x})}{\partial x_i} = -\sum_{l=1}^{M} \left[(d_l - y_l) \left(-\frac{1}{\sigma_j^2} \sum_{j=1}^{H} w_{lj} (x_i - c_{ji}) \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2\sigma_j^2} \right) \right) \right], \quad i = 1,2,..,N \tag{3.14}$$

Similarly, the input-output relation for a WBFNN using Gaussian scaling functions and Mexican hat wavelets is given by

$$y_l = \sum_{j=1}^{H_1} w_{lj} \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_j\|^2}{2\sigma_j^2}\right) + \sum_{n=1}^{L}\sum_{k=1}^{K_n} w_{lkn}\left(\frac{1-\|\mathbf{x}-\mathbf{c}_{kn}\|^2 2^{m_n^2}}{2\sigma_{kn}^2}\right)\exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_{kn}\|^2 2^{m_n^2}}{2\sigma_{kn}^2}\right)$$
$$= \sum_{j=1}^{H_1} w_{lj} \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_j\|^2}{2\sigma_j^2}\right) + \sum_{k=1}^{H_2} w_{lk}\left(\frac{1-\|\mathbf{x}-\mathbf{c}_k\|^2 2^{m_k^2}}{2\sigma_k^2}\right)\exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_k\|^2 2^{m_k^2}}{2\sigma_k^2}\right) \tag{3.15}$$

with $H_2 = K_1 + K_2 + ... + K_L$ is the total number of wavelet basis functions used over all L resolutions and $l = 1,2,...,M$. Smooth basis functions are desirable in this application since derivatives are to be computed. This precludes the use of discontinuous Haar basis functions. Again, taking the derivative of E with respect to the input yields

$$\frac{\partial E(\mathbf{x})}{\partial x_i} = \sum_{l=1}^{M}\left[(d_l - y_l)\left(\frac{1}{\sigma_j^2}\sum_{j=1}^{H_1} w_{lj}(x_i - c_{ji})\exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_j\|^2}{2\sigma_j^2}\right)\right.\right.$$
$$\left.\left. + \frac{1}{\sigma_k^2}\sum_{k=1}^{H_2} w_{lk}(x_i - c_{ki})\left\{1+\frac{\left(1-\|\mathbf{x}-\mathbf{c}_k\|^2 2^{m_k^2}\right)}{2\sigma_k^2}\right\}\exp\left(-2^{m_k^2}\frac{\|\mathbf{x}-\mathbf{c}_k\|^2}{2\sigma_k^2}\right)\right)\right] \tag{3.16}$$

for $i = 1,2,...,N$. The first term in the derivative in (3.16) is due to the scaling function while the second term is due to the wavelets.

These derivatives are then substituted into the gradient descent algorithm for signal inversion in electromagnetic NDE.

3.1. Results

The iterative inversion algorithm was tested on magnetic flux leakage (MFL) data (Appendix) generated by means of a two-dimensional finite element model (FEM) with a 100x100 node mesh [6] so that only the cross-section of the flaw with varying widths and depths is modeled. MFL techniques are used extensively for the inspection of ferromagnetic materials where the measured signal consists of changes in leakage magnetic flux density as

the probe scans the sample surface. A set of 240 defects was used to generate the corresponding MFL signals. The defects varied in width from 1" to 7" and depth from 0.15" to 0.85" in a sample of unit thickness. Figure 8 shows two examples of defect profiles and their corresponding MFL signals. Of these 240 defect profile-MFL signal pairs, 210 were used to train the neural networks while 30 were used as part of the test database with no overlap between the training and test sets.

3.1.1. Inversion Results Using RBFNN

The first set of results was obtained by using the RBFNN as the forward model. An RBFNN with 100 input nodes, 100 output nodes and 140 centers and Gaussian basis functions was used. The centers were determined using a K-Means clustering algorithm. The spread (or width) of each basis function was determined from the corresponding cluster spread. Figure 9 shows the performance of the RBFNN as a forward model. The solid line is the true MFL signal while the dotted line shows the prediction of the neural network. These results indicate the feasibility of using an RBFNN to accurately model the forward process.

Figure 10 shows the results of iterative inversion of a signal from a 2.6" long, 0.75"deep defect. Figure 10(a) shows the true signal as a solid line. The corresponding true defect profile is shown in Figure 10(b) as a solid curve. The predicted defect profile (reached after convergence of the algorithm) is shown in Figure 10 (b) as a dotted line while the corresponding MFL signal is shown in Figure 10 (a), again as a dotted curve. These results show a perfect match in the MFL signals although the defect profile does not match the true profile exactly. This could be due to either an imperfect forward model or the algorithm converging to a local minimum during the inversion process. Figure 11 (a) and (b) show similar results for a different defect (6.2" wide and 0.40" deep). The same trend is observed

in that the true and predicted signals match well although the true and predicted defect profiles do not match exactly. These results can be compared to results presented in Figure 12 and Figure 13, where the signals to be inverted have been corrupted with additive noise. Figure 12 (a) shows the results for the 2.6", 0.75" deep flaw when the noise level is 5%, while Figure 12 (b) shows the corresponding results for a 15% noise level. Similar results for the 6.2", 0.40" deep flaw are shown in Figure 13 (a) and (b). These results indicate that the algorithm is robust under reasonable levels of noise in the measurements. A slight change in the predicted depth can be observed when the data is noisy. This is particularly so when the actual depth is small. This result can be attributed to the fact that additive noise changes the amplitude of the signal. Since information about the depth of the flaw is present in the amplitude, a slight change in signal amplitude results in a corresponding change in the predicted depth. However, the error in predicted depth is only 5% for the 0.75" deep flaw.

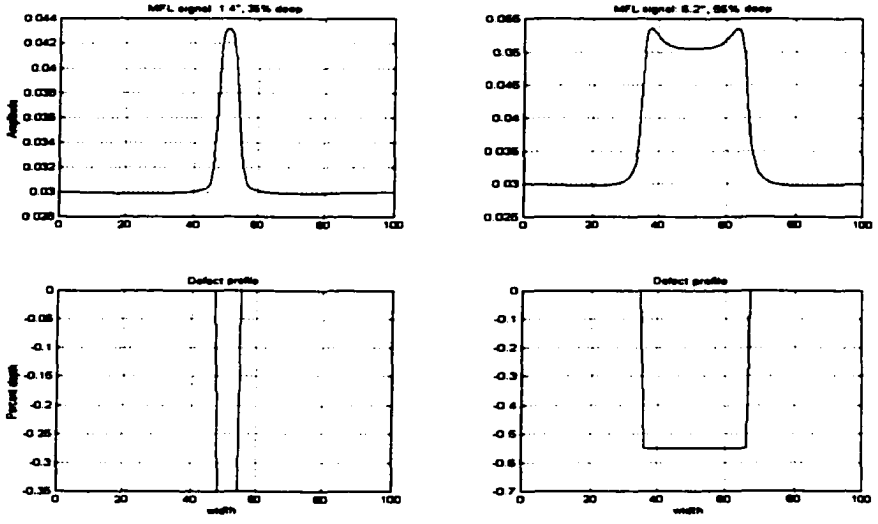


Figure 8. Examples of defect profiles and MFL signals.

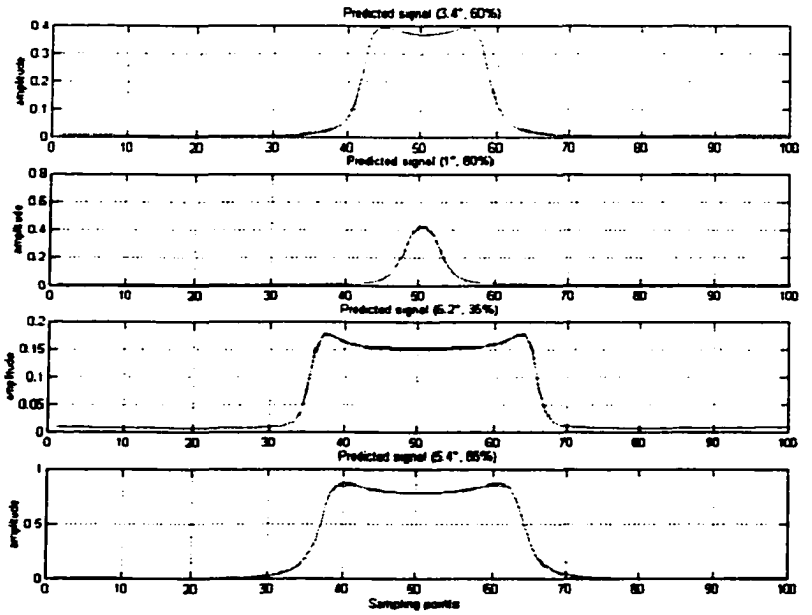


Figure 9. Performance of the RBFNN as a forward model.

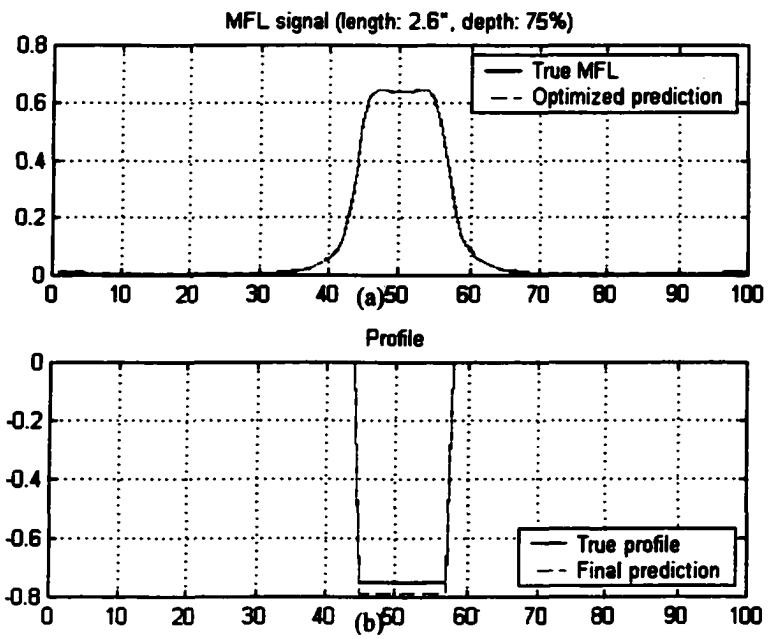


Figure 10. Results of iterative inversion, RBFNN as forward model (2.6", 0.75" deep).

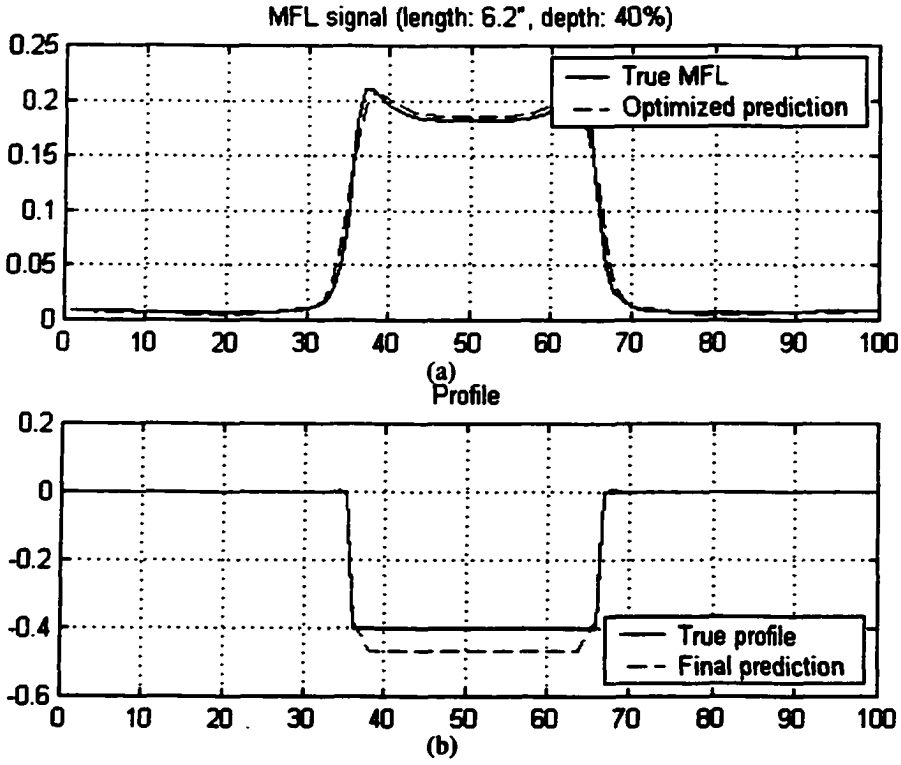
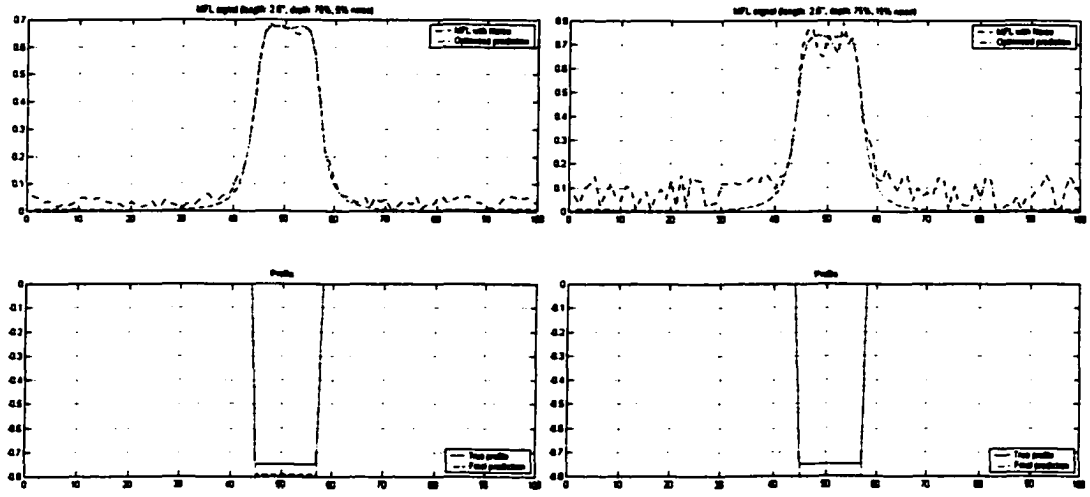
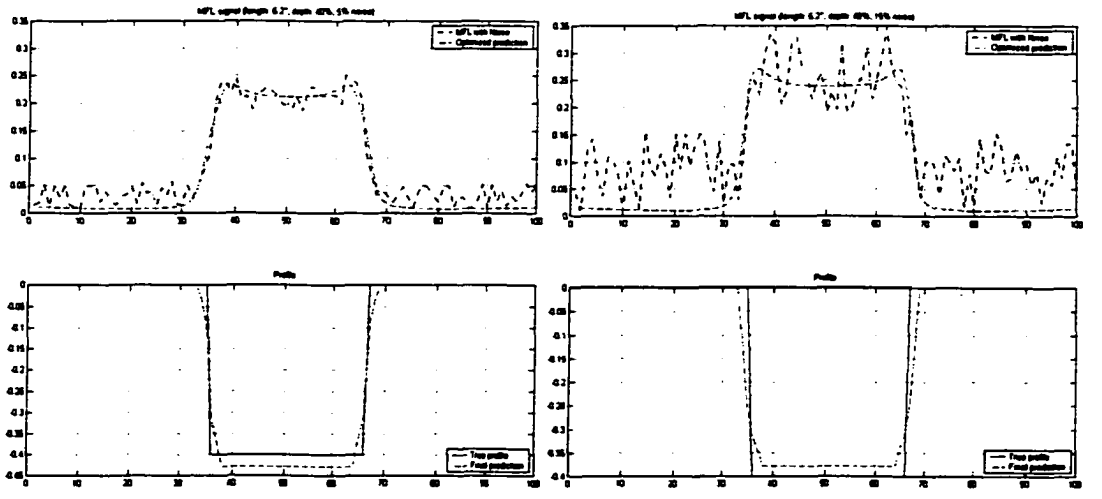


Figure 11. Results of iterative inversion, RBFNN as forward model (6.2", 0.40" deep).



(a) (b)

Figure 12. Performance of RBFNN with noise for 2.6", 0.75" deep flaw (a) 5% noise, (b) 15% noise.



(a) (b)

Figure 13. Performance of RBFNN with noise for 6.2", 0.40" deep flaw (a) 5% noise (b) 15% noise.

3.1.2. Inversion Results Using WBFNN

Results obtained using the WBFNN as the forward model are presented in Figures 12 through 16. Two resolution levels, with 10 centers at the coarsest resolution selected using the K-Means clustering algorithm, were used. Centers at higher resolution were selected using the dyadic center selection method to give a total of 29 centers. No optimization was performed after center selection to reduce the number of basis functions used. The scaling function used was a Gaussian function

$$\phi(\mathbf{x},\mathbf{c})=\exp\left(-\frac{\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2}\right) \tag{3.17}$$

where $\mathbf{c}$ and σ are the center and spread of the scaling function, respectively. The wavelet functions were Mexican hat wavelets as shown below

$$\psi(\mathbf{x},\mathbf{c})=\left(\frac{1-\|\mathbf{x}-\mathbf{c}\|^2 2^{m^2}}{2\sigma^2}\right)\exp\left(-\frac{\|\mathbf{x}-\mathbf{c}\|^2 2^{m^2}}{2\sigma^2}\right) \tag{3.18}$$

where c and σ are the center and spread of the wavelet function respectively. m is a parameter controlling the dilation of the wavelet, whose value depends on the resolution level. Figure 14 shows the performance of the WBFNN as a forward model. This result indicates that a WBFNN is capable of accurately modeling the magnetic flux leakage phenomenon. A slight error in the estimated signals can be attributed to the use of fewer basis functions in the expansion. Figure 15 and Figure 16 show the performance of the iterative inversion process while Figure 17 and Figure 18 show the corresponding results when the measurements have been corrupted with 5% and 15% noise.

Comparing the results in Figures 7 and 12, we see that the WBFNN is a better forward model than the RBFNN. However, the iterative inversion results of the WBFNN are less accurate than those obtained by using the RBFNN. One possible reason for this could be that the WBFNN is over-fitting the training data due to a high number of resolutions. This can be resolved by using a WBFNN with fewer basis functions and/or resolutions. Simulations with a WBFNN using three resolution levels resulted in a higher error in the predicted defect signal. The results also show the robustness of the algorithm with respect to additive noise demonstrating the feasibility of the inversion technique.

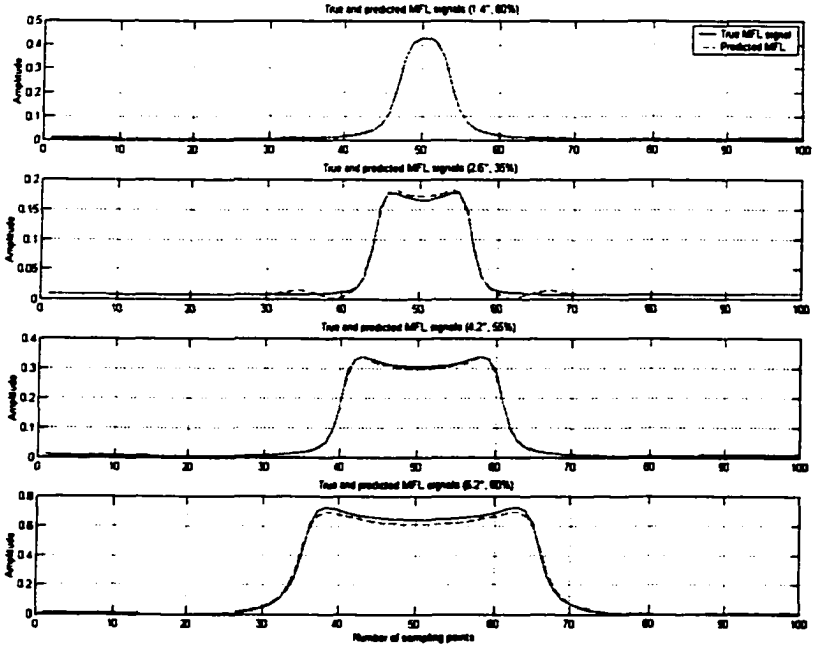


Figure 14. Performance of WBFNN as a forward model.

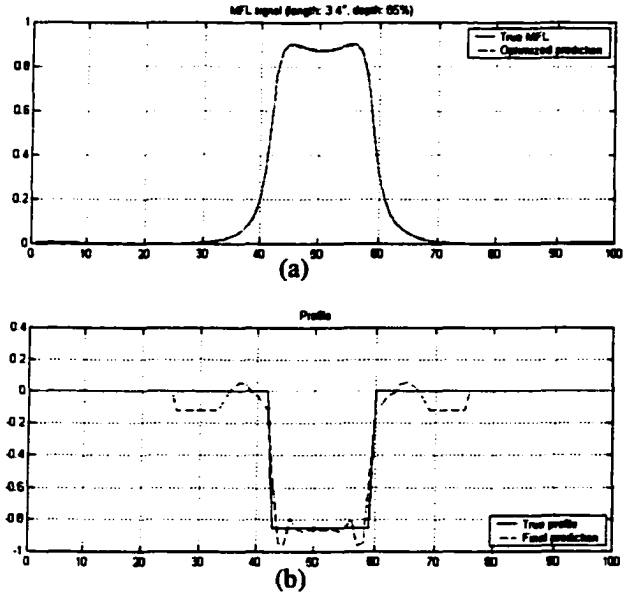


Figure 15. Results of iterative inversion, WBFNN as forward model (3.4", 0.85" deep).

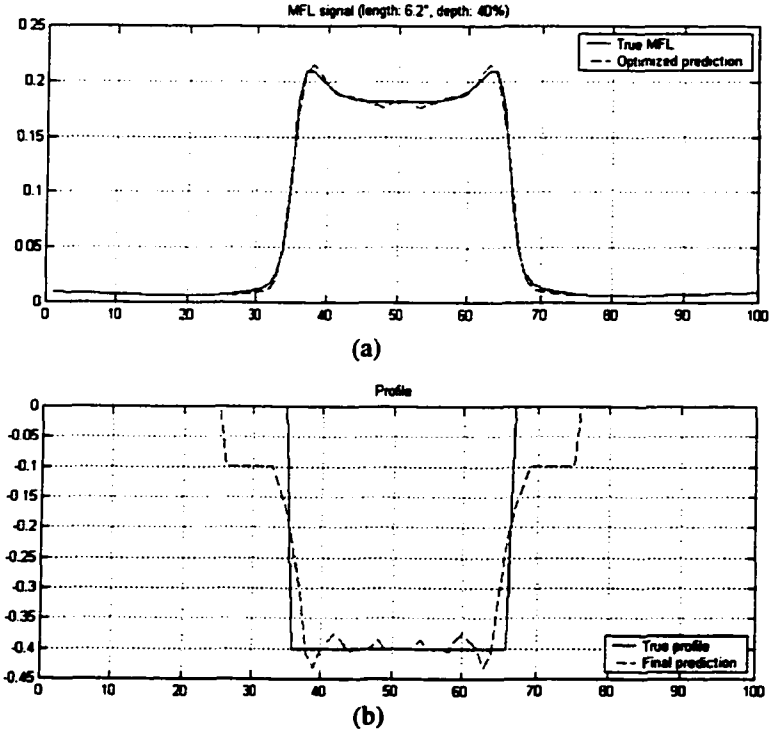


Figure 16. Results of iterative inversion, WBFNN as forward model (6.2", 0.40" deep).

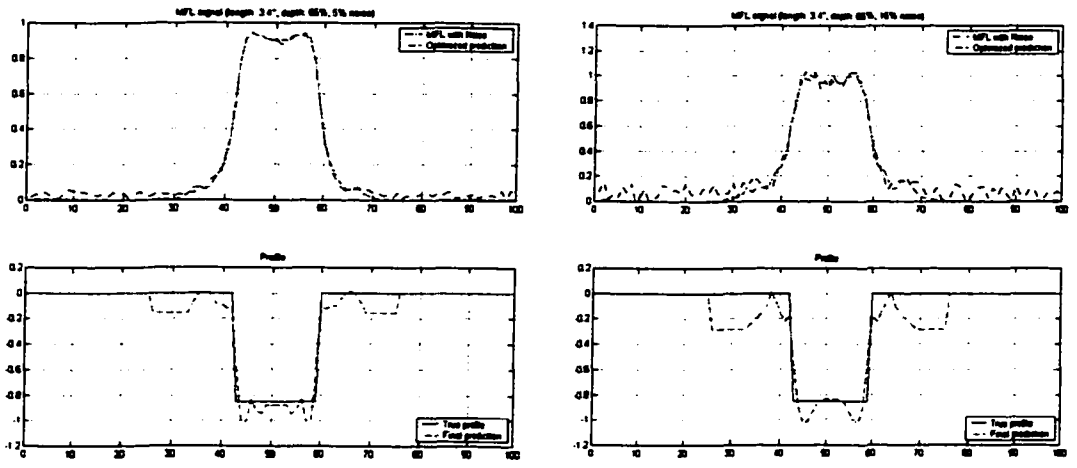
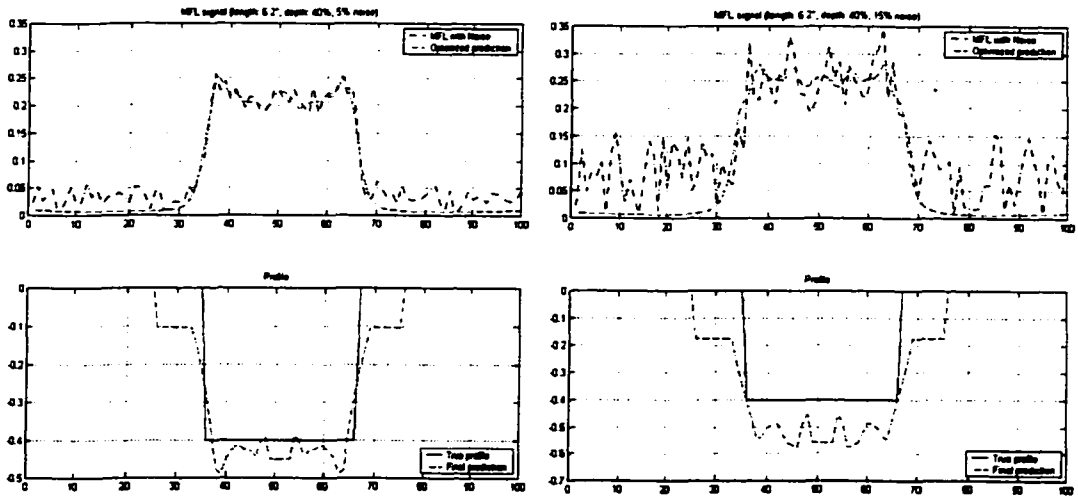


Figure 17. Performance of WBFNN with noise for 3.4", 0.85" deep flaw (a) 5% noise (b) 15% noise.



(a) (b)

Figure 18. Performance of WBFNN with noise for 6.2", 0.40" deep flaw (a) 5% noise (b) 15% noise.

4. ELECTROMAGNETIC NDE SIGNAL INVERSION USING FEEDBACK NEURAL NETWORKS

The second approach to solving the inverse problem is a feedback neural network scheme. The feedback neural network (FBNN) approach is depicted in Figure 19. Two neural networks are used in a feedback configuration. The forward network predicts the signal corresponding to a defect profile while the inverse (characterization) network predicts a profile given an NDE signal. The forward network replaces the finite element model employed in a typical phenomenological approach and provides a reference for comparing the defect profile predicted by the inverse neural network.

The overall approach to solving the inverse problem is as follows. The signal from a defect with unknown shape is input to the characterization neural network to obtain an estimate of the profile. This estimate is then input into the forward network to obtain the corresponding prediction of the MFL signal for that estimate of the profile. If the estimated defect profile is close to the true profile, the measured MFL signal and the predicted signal from the forward network will be similar to each other. On the other hand, if the error exceeds a threshold, the training mode is invoked and the networks are retrained with the correct defect profile-MFL signal dataset.

Since the forward neural network serves as a "standard' for measuring the performance of the FBNN scheme, it must be capable of accurately estimating the signal obtained from a variety of defect profiles. The wavelet basis function neural network described in Chapter 2 is used for implementing the forward network. A radial basis function (RBFNN) neural network is used as an inverse network for characterizing the defect profiles.

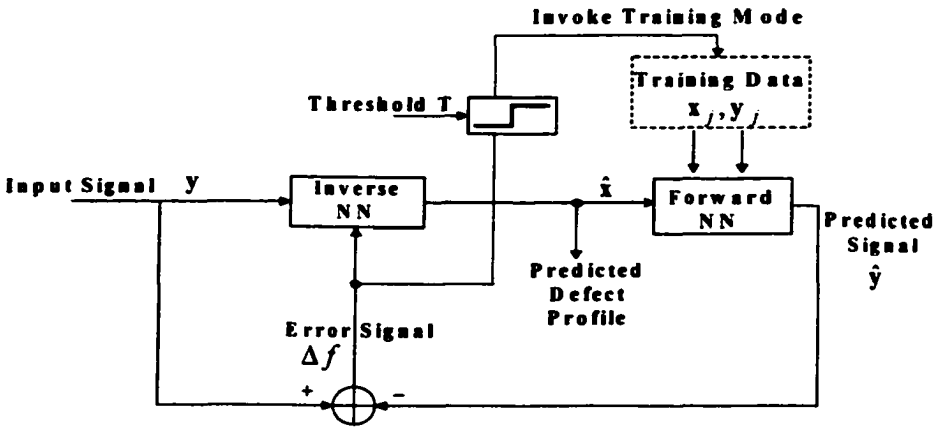


Figure 19. Schematic of the feedback neural network approach (Prediction mode).

4.1. FBNN Training and Optimization

The forward and inverse networks are first trained using the training database. The forward network is tested to ensure that it is capable of accurately modeling the forward process for the training database, and if necessary, the network parameters are optimized. In addition, the inverse network is also trained and its parameters optimized. This process is referred to as the training mode. The goal of the optimization step is to minimize the error due to the inverse RBFNN. Let Δf be the error between the actual MFL signal and the prediction of the forward network in the feedback configuration. In order for Δf to be zero, the characterization network must be an exact inverse of the forward network. While the functional form of the forward network can be derived easily, obtaining its inverse analytically is difficult. This is due to the fact that the output of the forward network is a function of the number and location of the respective basis function centers in each network. The inverse is, therefore, estimated numerically.

An adaptive scheme is used to estimate the inverse of the forward network as shown in Figure 20. The adaptive scheme uses the same gradient descent – simulated annealing combination described in the previous chapter. This "inverse network" is used as the characterization network.

Let E = the error at the output of the inverse network in Figure 20,

w_{kj} = interconnection weight from node j in the hidden layer to node k in the output layer

c_j = center of the j^{th} basis function (at node j in the hidden layer)

σ_j = spread of the j^{th} basis function

y = the measured signal

$\mathbf{x} = (x_1, x_2, ..., x_k, ...x_n)$ be the desired output of the RBF network

$\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, ..., \hat{x}_k, ...\hat{x}_n)$ be the actual output of the RBF network

Then, the error E can be defined as

$$E = \frac{1}{2}\|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \sum_{k=1}^{N}(x_k - \hat{x}_k)^2 \tag{4.1}$$

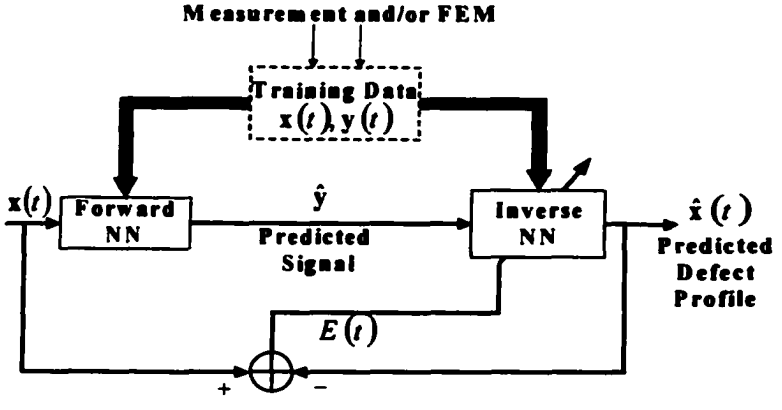


Figure 20. Feedback neural network: Training mode.

where $\hat{x}_k$ is given by

$$\hat{x}_k = \sum_{j=1}^{l} w_{kj} \phi\left(\frac{\|\mathbf{y} - \mathbf{c}_j\|}{2\sigma_j^2} \right) \tag{4.2}$$

and the basis function is chosen to be a Gaussian function:

$$\phi\left(\frac{\|\mathbf{y} - \mathbf{c}_j\|}{2\sigma_j^2} \right) = \exp\left(-\frac{\|\mathbf{y} - \mathbf{c}_j\|^2}{2\sigma_j^2} \right) \tag{4.3}$$

Substituting (4.2) and (4.3) into (4.1) and taking the derivative with respect to the weights w_{kj}, we have

$$\frac{\partial E}{\partial w_{kj}} = -(x_k - \hat{x}_k)\phi\left(\frac{\|\mathbf{y} - \mathbf{c}_j\|}{2\sigma_j^2} \right) \tag{4.4}$$

Similarly, the derivative of the error with respect to the other two parameters ($\mathbf{c}_j$ and σ_j) can be computed as follows:

$$\frac{\partial E}{\partial c_{ji}} = \sum_{k=1}^{n} -(x_k - \hat{x}_k)\left[w_{kj}\phi\left(\frac{\|\mathbf{y} - \mathbf{c}_j\|}{2\sigma_j^2} \right)\left(\frac{y_i - c_{ji}}{\sigma_j^2} \right) \right] \tag{4.5}$$

$$\frac{\partial E}{\partial \sigma_j} = \sum_{k=1}^{n} -(x_k - \hat{x}_k)\left[w_{kj}\phi\left(\frac{\|\mathbf{y} - \mathbf{c}_j\|}{2\sigma_j^2} \right)\left(\frac{\|\mathbf{y} - \mathbf{c}_j\|^2}{\sigma_j^3} \right) \right] \tag{4.6}$$

The derivatives are then substituted into the gradient descent equation to derive the update equations for the three parameters. The gradient descent equation is given by

$$d^{new} = d^{old} + \eta\left(-\frac{\partial E}{\partial d} \right) \tag{4.7}$$

where d is the parameter of interest w_{kj}, c_{ji} or σ_j. The algorithm is summarized below.

Train the wavelet basis function neural network to predict the measured signal given the defect profile. Using the same data, train the inverse network to predict the profile given the measured signal. For each profile-signal pair $\{\mathbf{x}(t), \mathbf{y}(t)\}$ in the training database,

(i) Apply the defect profile to the WBFNN and obtain the predicted signal. Call this $\hat{\mathbf{y}}$.

(ii) Present the predicted signal $\hat{\mathbf{y}}$ to the inverse (RBF) neural network. Compute the predicted defect profile $\hat{\mathbf{x}}(t)=(\hat{x}_1, \hat{x}_2, ..., \hat{x}_k, ...\hat{x}_n)$ where

$$\hat{x}_k = \sum_{j=1}^{l} w_{kj} \phi\left(\frac{\|\mathbf{y}-\mathbf{c}_j\|}{2\sigma_j^2}\right) \tag{4.8}$$

(iii) Compute the sum-squared error

$$E(t)=\frac{1}{2}\|\mathbf{x}(t)-\hat{\mathbf{x}}(t)\|^2 \tag{4.9}$$

(iv) Compute the gradient of E with respect to the RBFNN parameters, $\frac{\partial E(t)}{\partial w_{kj}}$, $\frac{\partial E(t)}{\partial c_{ji}}$ and $\frac{\partial E(t)}{\partial \sigma_j}$.

(v) Update the parameters w_{kj}, c_{ji} or σ_j using the gradient descent equation

$$\hat{d}(t+1)=d(t)+\eta\left(-\frac{\partial E(t)}{\partial d(t)}\right) \tag{4.10}$$

(vi) Compute the new output of the RBF neural network $\hat{\mathbf{x}}_{new}(t)$

(vii) Compute the new error

$$E_{new}(t)=\frac{1}{2}\|\mathbf{x}-\hat{\mathbf{x}}_{new}(t)\|^2 \tag{4.11}$$

(viii) If $E_{new}(t) < E(t)$, set

$$d(t+1) \leftarrow \hat{d}(t+1), \quad \text{where } d = w_{kj}, d = c_{ji} \text{ or } d = \sigma_j \,, \tag{4.12}$$

$$\eta \leftarrow \eta \times \eta_{inc} \tag{4.13}$$

where η_{inc} is an increment factor.

(ix) If $E_{new}(t) > E(t)$, set

$$\eta \leftarrow \eta \times \eta_{dec} \,. \tag{4.14}$$

Go to (ii). Repeat till $E(t) < ERROR\_THRESHOLD$.

Once the characterization network is trained and optimized, the two networks are connected in the feedback configuration shown in Figure 19. The characterization network can then be used for predicting flaw profiles using signals obtained from defects of unknown shape and size.

4.2. Results and Discussions

The algorithm was tested with the tangential component of magnetic flux leakage (MFL) data generated by a 2D FEM employing a 100x100-element mesh. The database is the same database used to test the algorithm described in Chapter 3. A wavelet basis function neural network (WBFNN) is used as the forward network while the radial basis function (RBFNN) network is used as the inverse network for characterization. The WBFNN uses 3 resolution levels with 10 centers at the coarsest resolution. The centers at other resolutions are computed using a dyadic grid (a total of 66 hidden nodes). The number of input nodes is equal to the number of points (100) used to approximate the defect profile. The number of output nodes is equal to the length of each signal (also 100 points). The RBFNN uses 140 basis functions in the hidden layer. 210 defect profile-MFL signal pairs were used in the training set and 30 signals were used for testing with no overlap between the two data sets.

Figure 21 shows the results of training the forward network. The solid line shows the true signal while the dotted line shows the neural network prediction. These plots indicate that the forward network is capable of predicting the signal with little error. A typical prediction result is shown in Figure 22. The solid line in Figure 22 (a) shows the true signal. This is applied to the RBFNN network, which has not been optimized. The prediction result of the RBFNN network is shown in Figure 22 (b). The resulting signal is then applied to the forward network. The corresponding output is shown in Figure 22 (a). The results after optimizing the inverse network are also shown in Figure 22 (a) and (b). Similar results obtained using signals from defects with other geometries are shown in Figure 23 and Figure 24.

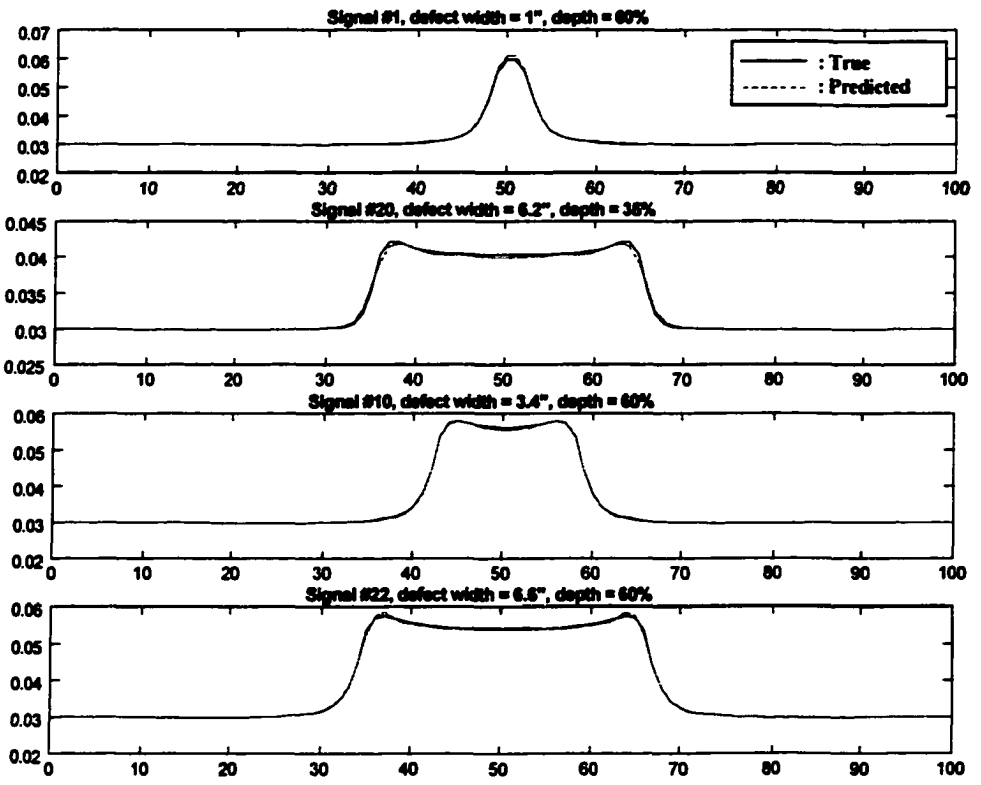


Figure 21. Training results for the forward network.

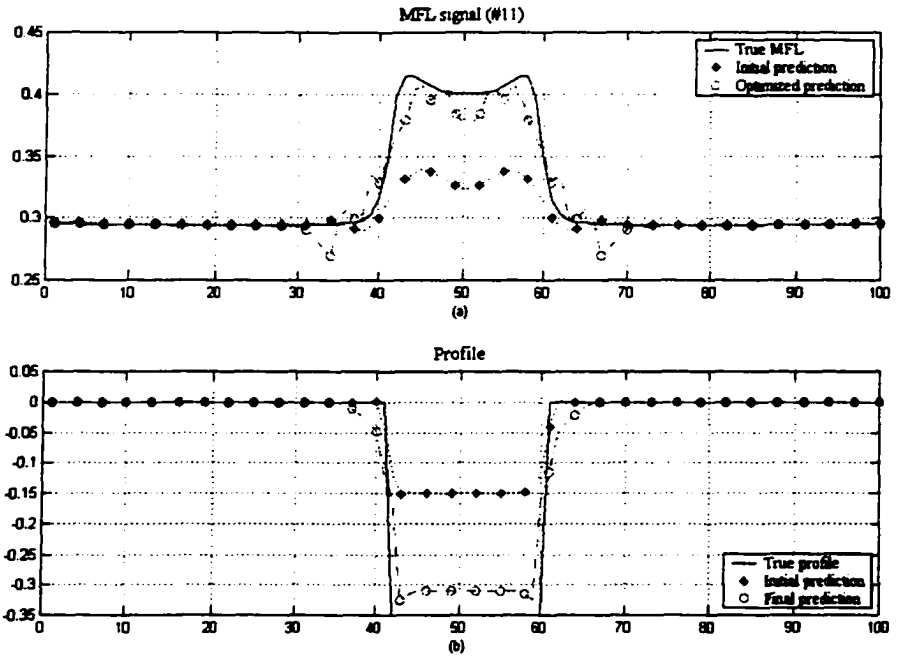


Figure 22. Feedback neural network results (3.8", 0.35" deep).

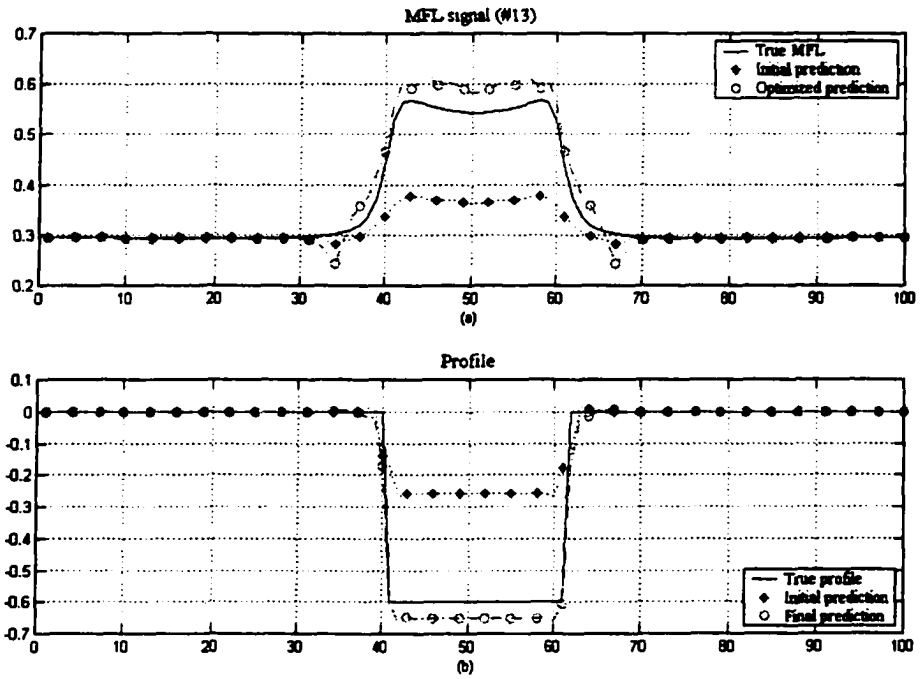


Figure 23. Feedback neural network result (4.2", 0.60" deep).

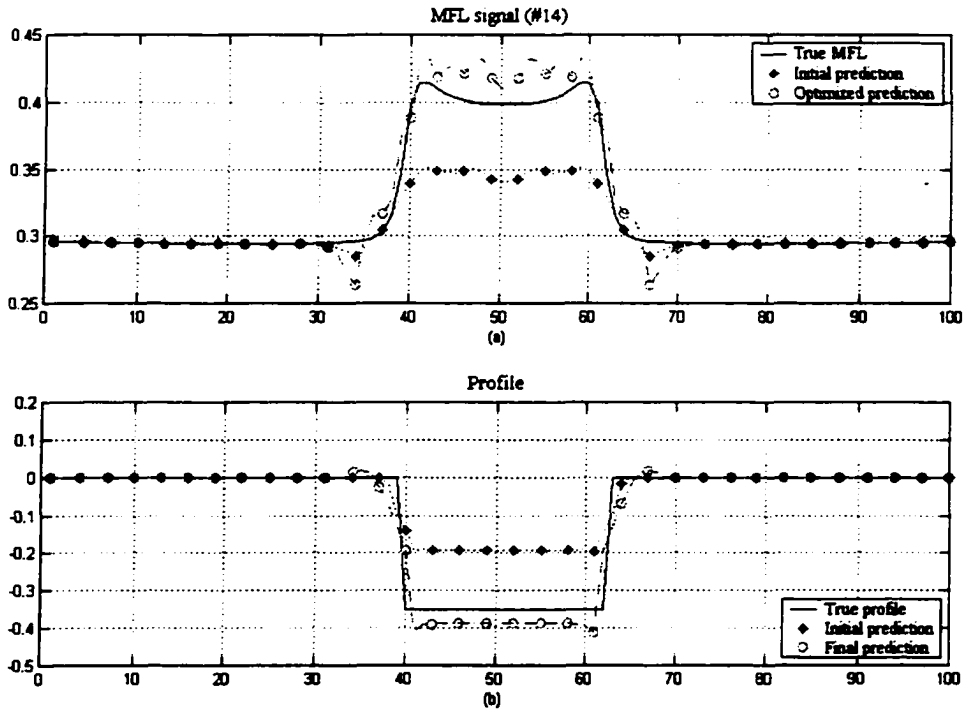


Figure 24. Feedback neural network result (4.6", 0.35" deep).

These results indicate that the optimization process improves the prediction results. In addition, the use of a forward network in a feedback configuration provides a measure of the error in the characterization with the error in the defect profile prediction being proportional to the error in the signal prediction. This fact is also illustrated in the results presented in Figure 25 - Figure 28, where the inversion results on signals with and without noise is compared. Figure 25 shows the inversion result for a 4.2" wide, 0.55" deep rectangular flaw, while Figure 26 (a) and (b) show the corresponding results with 5% and 15% additive noise. Similar results are shown in Figure 27 and Figure 28 for a 1.4" wide, 0.20" deep flaw. In general, we expect the feedback network combination to perform poorly when the test data is not very similar to the training data. These figures show that the noisy measurements are not very similar to the original training data, and as such, the performance of the feedback

network algorithm will degrade. However, as mentioned earlier, the error in the predicted profile increases with an increase in the amount of noise, and this fact can be used to provide a measure of confidence in the network prediction.

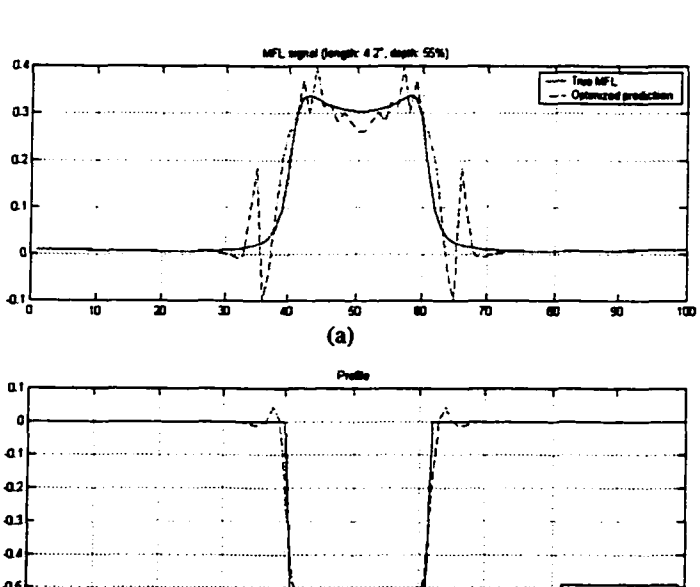


Figure 25. Inversion results for a 4.2" wide, 0.55" deep flaw (no noise).

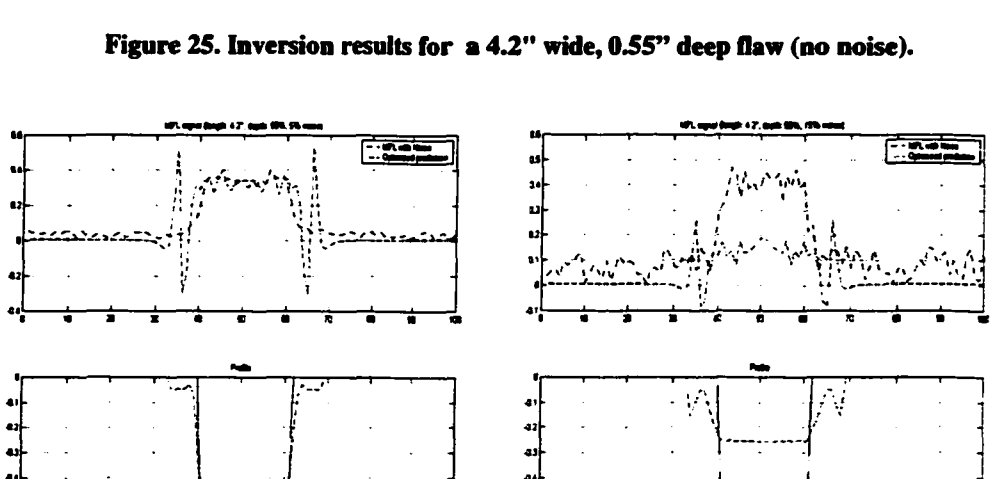


Figure 26. Inversion results for a 4.2" wide, 0.55" deep flaw (a) 5% noise, (b) 10% noise.

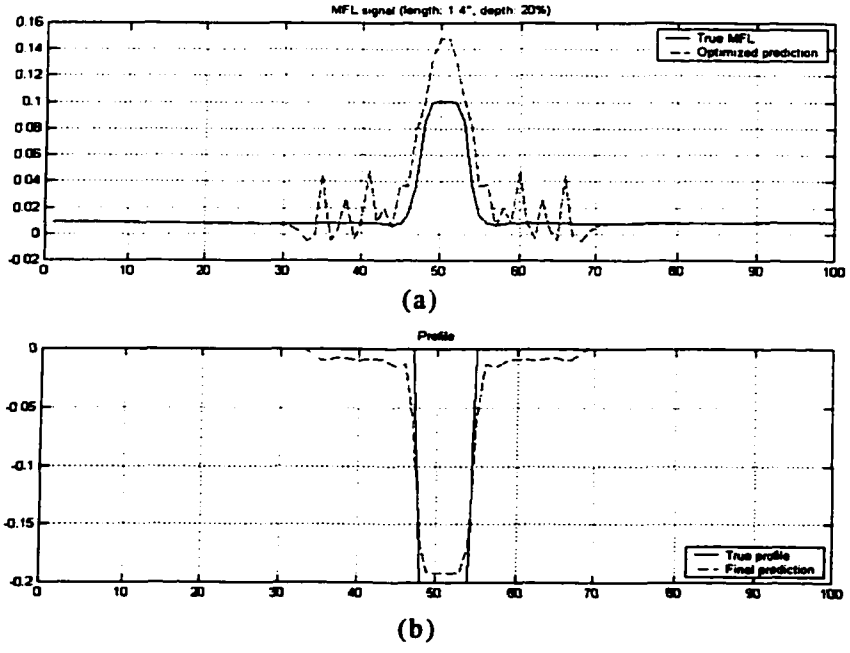


Figure 27. Inversion results for a 1.4" wide, 0.20" deep flaw (no noise).

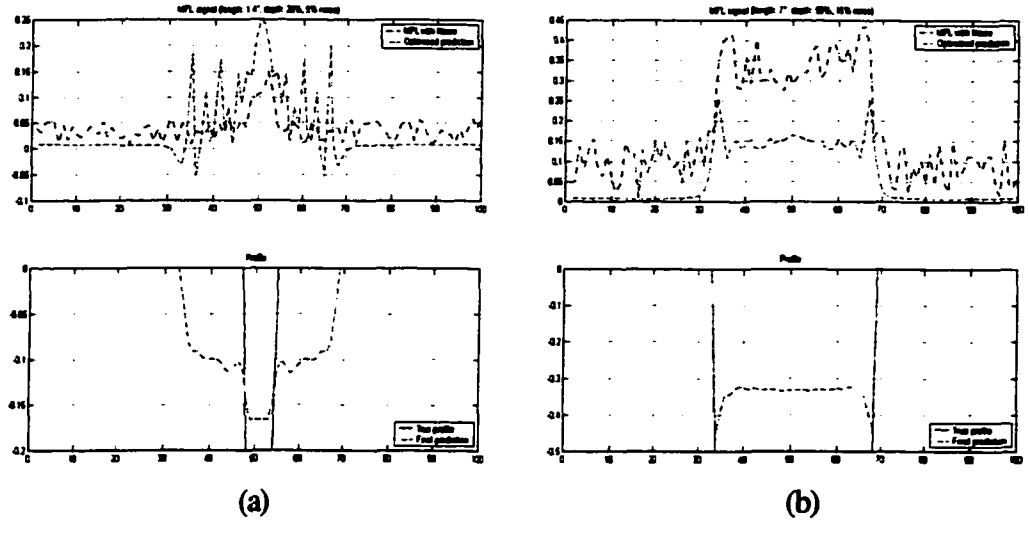


Figure 28. Inversion results for a 1.4" wide, 0.20" deep flaw (a) 5% noise (b) 15% noise.

5. THE FINITE ELEMENT NEURAL NETWORK AND ITS APPLICATION TO SIGNAL INVERSION

This section first describes the finite element model (FEM) and describes the reformulation of the FEM into a neural network structure using a simple two-dimensional problem. The structure of this neural network is described, followed by its application to solving the forward and inverse problems. This model is then extended to the general case and the advantages and disadvantages of this approach are described along with an analysis of the sensitivity of the inversion algorithm to errors in the measurements.

5.1. The Finite Element Method

Consider a typical boundary value problem with the governing differential equation

$$\mathrm{L}\phi = f \tag{5.1}$$

where L is a differential operator, f is the applied source or forcing function and ϕ is the unknown quantity. This differential equation can be solved in conjunction with boundary conditions on the boundary Γ enclosing the domain. A commonly used approach to solve this problem is to use the finite element approach. The variational formulation of this approach determines the unknown ϕ by minimizing the functional [7, 8]

$$F(\tilde{\phi}) = \frac{1}{2}\langle \mathrm{L}\tilde{\phi}, \tilde{\phi} \rangle - \frac{1}{2}\langle \tilde{\phi}, f \rangle - \frac{1}{2}\langle f, \tilde{\phi} \rangle \tag{5.2}$$

with respect to the trial function $\tilde{\phi}$. The minimization procedure starts by dividing the domain of interest into small subdomains called elements and representing $\tilde{\phi}$ in each element by means of basis functions defined over the element:

$$\tilde{\phi}^e = \sum_{j=1}^{n} N_j^e \phi_j^e \tag{5.3}$$

where $\tilde{\phi}^e$ is the unknown solution in element e, N_j^e is the basis function associated with node j in element e, ϕ_j^e is the value of the unknown quantity at node j and n is the total number of nodes associated with element e. In general, the basis functions (also referred to as interpolation functions or shape functions) can be linear, quadratic or higher order basis functions. Higher order polynomials, though more accurate, generally result in higher computational complexity, and hence, linear basis functions are commonly used.

Once the domain is divided into smaller elements, the functional can be expressed as

$$F(\tilde{\phi}) = \sum_{e=1}^{M} F(\tilde{\phi}^e) \tag{5.4}$$

where M is the total number of elements and $F(\tilde{\phi}^e)$ represents the value of the functional within element e:

$$F(\phi^e) = \frac{1}{2}\int_{\Omega^e} \tilde{\phi}^e \mathcal{L}\tilde{\phi}^e d\Omega - \int_{\Omega^e} f\tilde{\phi}^e d\Omega \tag{5.5}$$

By substituting (5.3) in (5.5), we obtain the discrete version of the functional within each element:

$$F(\tilde{\phi}^e) = \frac{1}{2}\boldsymbol{\varphi}^{eT}\int_{\Omega^e} \mathbf{N}^e \mathcal{L}\mathbf{N}^{eT} d\Omega \boldsymbol{\varphi}^e - \boldsymbol{\varphi}^{eT}\int_{\Omega^e} f\mathbf{N}^e d\Omega \tag{5.6}$$

where $(...)^T$ is the transpose of a matrix or a vector. This can be written in matrix-vector form as

$$F(\tilde{\phi}^e) = \frac{1}{2}\boldsymbol{\varphi}^{eT}\mathbf{K}^e\boldsymbol{\varphi}^e - \boldsymbol{\varphi}^{eT}\mathbf{b}^e \tag{5.7}$$

where $\mathbf{K}^e$ is the $n \times n$ elemental matrix with elements

$$K_{ij}^e = \int_{\Omega^e} N_i^e \mathcal{L} N_j^e d\Omega \tag{5.8}$$

and $\mathbf{b}^e$ is an $n \times 1$ vector with elements

$$b_i^e = \int_{\Omega^e} f N_i^e \, d\Omega . \tag{5.9}$$

From (5.4) and (5.7), we obtain

$$F = \frac{1}{2}\boldsymbol{\varphi}^T \mathbf{K}\boldsymbol{\varphi} - \boldsymbol{\varphi}^T \mathbf{b} \tag{5.10}$$

where $\mathbf{K}$ is the $N \times N$ global matrix derived from the terms of the elemental matrices for different elements, and N is the total number of nodes. Equation (5.10) is the discrete version of the functional and can be minimized with respect to the nodal parameters $\boldsymbol{\varphi}$ by taking the derivative of F with respect to $\boldsymbol{\varphi}$ and setting it equal to zero. The resulting equation

$$\frac{\partial F}{\partial \boldsymbol{\varphi}} = \mathbf{K}\boldsymbol{\varphi} - \mathbf{b} = 0 \tag{5.11}$$

can be solved for the nodal parameters $\boldsymbol{\varphi}$:

$$\boldsymbol{\varphi} = \mathbf{K}^{-1}\mathbf{b} . \tag{5.12}$$

Boundary conditions for these problems are usually of two types: natural boundary conditions and essential boundary conditions. Essential boundary conditions (also referred to as Dirichlet boundary conditions) impose constraints on the value of the unknown ϕ at several nodes. Natural boundary conditions (of which Neumann boundary conditions are a special case) impose constraints on the change in ϕ across a boundary. Imposition of these conditions into the finite element formulation is straightforward. Natural boundary conditions are incorporated into the functional and are satisfied automatically during the solution procedure. Dirichlet boundary conditions, on the other hand, need to be handled separately. These conditions are imposed on the functional minimization equation (5.11), by deleting the

rows and columns of the **K** matrix corresponding to the nodes that are part of the boundary. Suppose the given boundary condition is

$$\phi = p \text{ on the boundary } \Gamma \tag{5.13}$$

and let the i^{th} node represent the boundary Γ. Then, equations (5.11) are modified as follows:

$$b_i \leftarrow p,\ b_j \leftarrow b_j - K_{ji}p,\ j \neq i \tag{5.14}$$

$$K_{ii} \leftarrow 1,\ K_{ij} \leftarrow 0,\ K_{ji} \leftarrow 0,\ j \neq i \tag{5.15}$$

This process can be repeated for each node on the boundary Γ and the resulting matrix can be solved as indicated in (5.12) to obtain the solution subject to the Dirichlet conditions.

5.2. The Finite Element Neural Network

The finite element model can be easily converted into a neural network form. To see this, consider the simple two-dimensional example:

$$-\nabla \bullet (\alpha \nabla \phi) + \beta\phi = f \tag{5.16}$$

with boundary conditions

$$\phi = p \text{ on } \Gamma_1 \tag{5.17}$$

and

$$\alpha \nabla \phi \bullet \hat{n} + \gamma\phi = q \text{ on } \Gamma_2 \tag{5.18}$$

where α and β are constants depending on the material, f is the applied source, $\Gamma = \Gamma_1 + \Gamma_2$ is the boundary enclosing the domain, $\hat{n}$ is its outward normal unit vector, and γ, p and q are known parameters associated with the boundary. Assume that the domain is divided into two elements (Figure 29), with four nodes. The elemental matrices $\mathbf{K}^e$ and $\mathbf{b}^e$ can be derived as [8]

$$K_{ij}^e = \int_{\Omega^e} \left(\alpha \nabla N_i^e \bullet \nabla N_j^e + \beta N_i^e N_j^e\right) d\Omega \tag{5.19}$$

and

$$b_i^e = \int_{\Omega^e} f N_i^e \, d\Omega \,. \tag{5.20}$$

The global matrix equation can be assembled by combining the two elemental matrices. To do this, we need the node-element connectivity information given in Table 1. This table contains information about the various nodes that make up each element, as well as their position in the element (often called the local node number). Each node also has a global node number, indicating its position in the entire finite element model system. The columns in the table marked $n(i,e)$ refer to the i^{th} node in element e and the value of $n(i,e)$ is the global node number. For instance, node 2 in Figure 29 appears as the second node in element 1 and the third node in element 2.

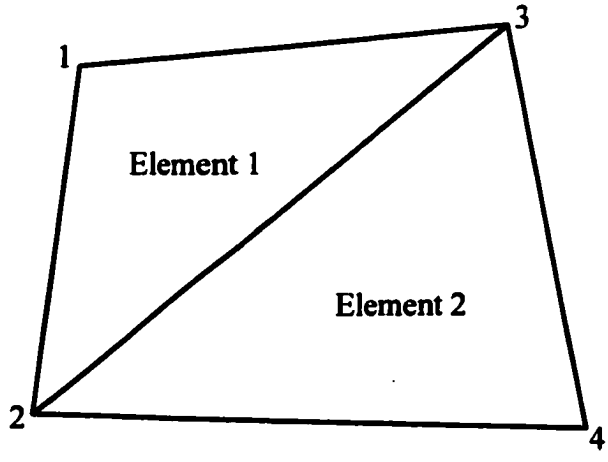


Figure 29. FEM domain discretization using two elements and four nodes.

Table 1. Node-element connectivity array for the two-element mesh given in Figure 29.

e	$n(1,e)$	$n(2,e)$	$n(3,e)$
1	1	2	3
2	4	3	2

The connectivity array shown in Table 1 can be used to obtain the global matrix **K**, by combining the appropriate members of the elemental matrices. Consider node 2 as an example. Since node 2 appears as the second node in element 1 and the third node in element 2, we combine the corresponding members K_{22}^1 and K_{33}^2 of the elemental matrices to obtain

$$K_{22} = K_{22}^1 + K_{33}^2. \tag{5.21}$$

This process is repeated for each of the four nodes, giving

$$\mathbf{K} = \begin{bmatrix} K_{11}^1 & K_{12}^1 & K_{13}^1 & 0 \\ K_{21}^1 & K_{22}^1 + K_{33}^2 & K_{23}^1 + K_{32}^2 & K_{31}^2 \\ K_{31}^1 & K_{32}^1 + K_{23}^2 & K_{33}^1 + K_{22}^2 & K_{21}^2 \\ 0 & K_{13}^2 & K_{12}^2 & K_{11}^2 \end{bmatrix} \tag{5.22}$$

Similarly, the vector **b** is given by

$$\mathbf{b} = \begin{bmatrix} b_1^1 \\ b_2^1 + b_3^2 \\ b_3^1 + b_2^2 \\ b_1^2 \end{bmatrix} \tag{5.23}$$

In order to convert the FEM into a neural network, we start by first separating K_{ij}^e into two components: one component dependent on the material properties α and β, and the second

component independent of these material properties. This can be achieved by rewriting (5.19) as

$$K_{ij}^e = \alpha \int_{\Omega^e} \nabla N_i^e \bullet \nabla N_j^e d\Omega + \beta \int_{\Omega^e} N_i^e N_j^e d\Omega$$
$$= \alpha S_{ij}^e + \beta T_{ij}^e \tag{5.24}$$

where

$$S_{ij}^e = \int_{\Omega^e} \nabla N_i^e \bullet \nabla N_j^e d\Omega \tag{5.25}$$

and

$$T_{ij}^e = \int_{\Omega^e} N_i^e N_j^e d\Omega \tag{5.26}$$

Rewriting the original equation as shown in (5.24) assumes that the material properties are constant in an element. In general, this is not an unreasonable assumption, since the elements are usually small enough for this assumption to be true. Equations (5.24) and (5.11) can be converted into a neural network. The structure of this network is shown in Figure 30. The neural network is a three layer neural network, with input, output and hidden layers. The input layer has two groups of 2 neurons, with one group taking the α values in each element as input and the second group taking in the values of β in each element as input. The hidden layer has 16 neurons that are arranged in groups of 4 neurons for convenience. The output of each group of hidden layer neurons is the corresponding row vector of **K**. In the general case with M elements and N neurons in the FEM mesh, the input layer has $2M$ neurons, with the inputs being the material properties α and β in each of the M elements. The hidden layer has N^2 neurons arranged in N groups of N neurons, corresponding to the N^2 elements in the global matrix **K**. The weights from the input to the

hidden layer are set to the appropriate values of S_{ij}^e and T_{ij}^e. Examples of these weights are shown in Figure 30. Each neuron in the hidden layer acts as a summation unit, and the outputs of the hidden layer neurons are the elements K_{ij} of the global matrix **K**:

$$K_{ij} = \sum_{e=1}^{2} \alpha_e w_{ij}^e + \beta_e g_{ij}^e \tag{5.27}$$

where $w_{ij}^e = S_{ij}^e$ if nodes i and j are part of element e, and $w_{ij}^e = 0$ otherwise. Similarly, $g_{ij}^e = T_{ij}^e$ if nodes i and j are part of element e, and $g_{ij}^e = 0$ else.

Each group of hidden neurons is connected to one output neuron (giving a total of four output neurons) by a set of weights φ, with each element of φ representing the nodal values ϕ_j. Each output neuron is also a summation unit, and the output of each neuron is equal to b_i:

$$b_i = \sum_{j=1}^{4} K_{ij}\phi_j = \sum_{j=1}^{4} \phi_j \left(\sum_{e=1}^{2} \alpha_e w_{ij}^e + \beta_e g_{ij}^e \right) \tag{5.28}$$

where the second part of (5.28) is obtained by using (5.27). The number of output neurons in the general case increases to N.

5.2.1. Incorporation of Boundary Conditions

Natural boundary conditions are applied in the finite element method by adding an additional term to the functional. For the example under consideration, this functional takes the form [8]

$$F_b(\phi) = \int_{\Gamma_2} \left(\frac{\gamma}{2}\phi^2 - q\phi \right) d\Gamma \tag{5.29}$$

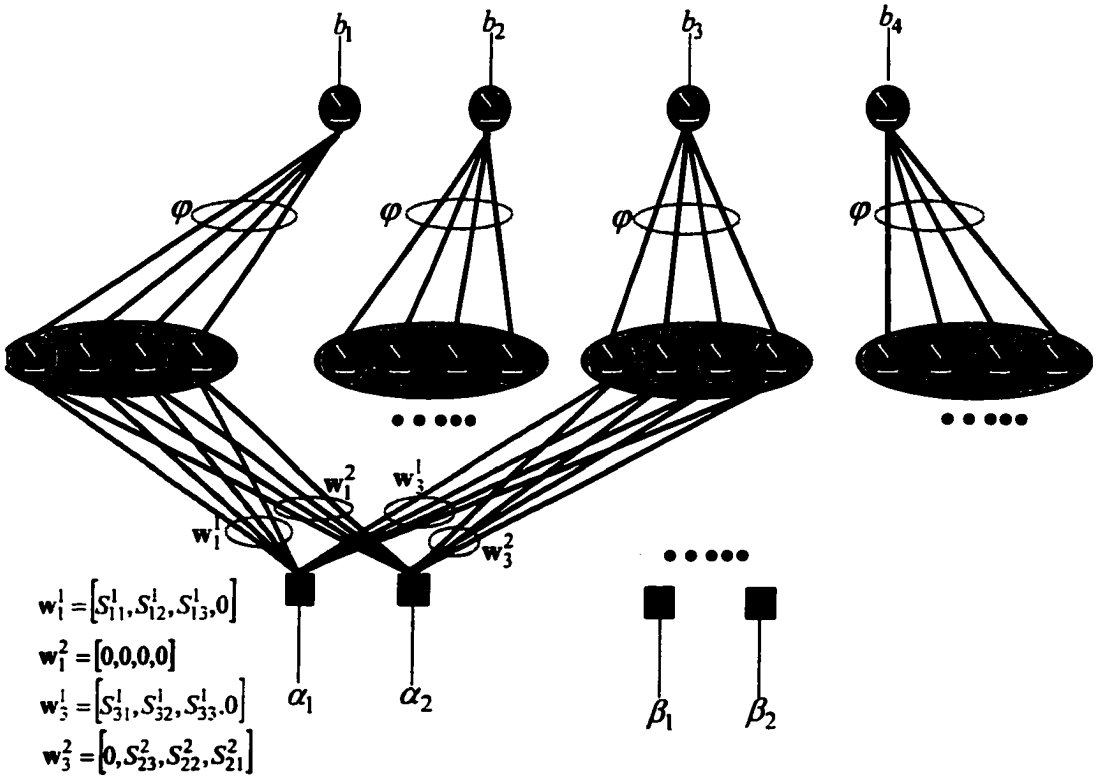


Figure 30. The finite element neural network.

Assuming that the boundary Γ_2 is made up of M_s segments, (5.29) can be written as

$$F_b(\phi)=\sum_{s=1}^{M_s}F_b^s(\phi^s) \tag{5.30}$$

Further, suppose that the unknown function ϕ^s in each segment s can be approximated as

$$\phi^s=\sum_{j=1}^{2}N_j^s\phi_j^s \tag{5.31}$$

where N_j^s are the basis functions defined over the segment s:

$$N_1^s=1-\xi,\ \ N_2^s=\xi \tag{5.32}$$

Here, ξ is the normalized distance between node 1 and node 2 on the segment. From (5.29) and (5.31), if l^s is the length of the segment, we get

$$\frac{\partial F_b^s}{\partial \phi_j^s} = \mathbf{K}^s \boldsymbol{\varphi}^s - \mathbf{b}^s \tag{5.33}$$

where

$$K_{ij}^s = \int_0^1 \gamma N_i^s N_j^s l^s d\xi$$

$$b_i^s = \int_0^1 q N_i^s l^s d\xi$$

$$\boldsymbol{\varphi}^s = \left[\phi_1^s, \phi_2^s\right]^T \tag{5.34}$$

Finally, the global matrix equation is modified as follows:

$$\frac{\partial F}{\partial \boldsymbol{\phi}} = \left(\mathbf{K}\boldsymbol{\varphi} - \mathbf{b}\right) + \left(\mathbf{K}^s \boldsymbol{\varphi}^s - \mathbf{b}^s\right) = 0 \tag{5.35}$$

Equation (5.35) indicates that the elements of $\mathbf{K}^s$ are added to the elements of $\mathbf{K}$ that correspond to the nodes on the boundary Γ_2. Similarly, the elements of $\mathbf{b}^s$ are added to the corresponding elements of $\mathbf{b}$. Note that the elements of $\mathbf{K}^s$ and $\mathbf{b}^s$ do not depend on the material properties α and β. Equation (5.35) thus implies that natural boundary conditions can be applied in the finite element neural network as bias inputs to the hidden layer nodes that are a part of the boundary, and the corresponding output nodes.

Dirichlet boundary conditions, as explained in the previous section, are applied by clamping the potential values at the nodes on the boundary Γ_1, and updating the matrix $\mathbf{K}$ according to equations (5.14) and (5.15). In the FENN, these boundary conditions can be applied by clamping the corresponding weights between the hidden layer and output layer

neurons. These weights will be referred to as the clamped weights, while the remaining weights will be referred to as the free weights. In addition, we modify the weights between the input and the hidden layers, and the source terms according to equations (5.14) and (5.15).

As mentioned earlier, the FENN can be easily extended to the case where the FEM mesh has N nodes and M elements. Similarly, extension to higher dimensional problems is straightforward. The number of nodes and elements dictates the number of neurons in the three layers. The weights between the input and hidden layer change depending on node-element connectivity information.

5.3. Forward and Inverse Problem Formulation Using FENN

Applying the FENN to solve the forward problem is straightforward. The forward problem involves determining the weights φ given the material parameters and the applied source. Given these values, the procedure for solving the forward problem is as follows.

1. Apply the material parameters α and β to the input neurons and compute the output of each of the hidden neurons. Any natural boundary conditions are applied as bias inputs to the hidden layer neurons. Initialize the value of φ randomly for all the free weights. Fix the values of the clamped weights according to the Dirichlet boundary conditions. Let the value of φ at iteration t be denoted by $\varphi(t)$.
2. At iteration t, compute the output of the neural network:

$$\hat{b}_i(t) = \sum_{j=1}^{N} K_{ij}\phi_j(t), \quad i = 1,2,...,N \tag{5.36}$$

3. Compute the error at the output of the neural network:

$$E(t)=\frac{1}{2}\left\|\mathbf{b}-\hat{\mathbf{b}}(t)\right\|=\frac{1}{2}\sum_{i=1}^{N}\left(b_i-\hat{b}_i(t)\right)^2=\frac{1}{2}\sum_{i=1}^{N}\left(E(t)\right)^2 \tag{5.37}$$

4. Compute the gradient of the error with respect to the free hidden layer weights

$$\frac{\partial E(t)}{\partial \phi_j}=-\sum_{i=1}^{N}E_i(t)K_{ij} \tag{5.38}$$

5. Update the free weights using the gradient descent equation

$$\phi_j(t+1)=\phi_j(t)+\eta\left(-\frac{\partial E(t)}{\partial \phi_j}\right) \tag{5.39}$$

6. Repeat steps 2-5 till convergence is achieved. The convergence criterion considered here is that the output error must fall below a threshold.

This approach is equivalent to the iterative approaches used to solve for the unknown nodal values [7].

The same neural network can be applied easily to solve the inverse problem. The inverse problem involves determining the material properties α and β, given the measurement $\boldsymbol{\varphi}$ and the applied source $\mathbf{b}$. We apply the iterative approach described in Approach I to solve the inverse problem. The overall algorithm is as follows.

1. Initialize the values of α and β. Let $\alpha(t)$ and $\beta(t)$ denote the values of the material properties at iteration t. Fix the weights between the hidden layer and the output layer neurons to the measurement $\boldsymbol{\varphi}$.
2. At iteration t, apply $\alpha(t)$ and $\beta(t)$ at the input layer of the FENN. Compute the output of the network using equations (5.27) and (5.28). Call this output $\hat{b}(t)$.
3. Compute the error at the output

$$E(t)=\frac{1}{2}\left\|\mathbf{b}-\hat{\mathbf{b}}(t)\right\|=\frac{1}{2}\sum_{i=1}^{N}\left(b_i-\hat{b}_i(t)\right)^2=\frac{1}{2}\sum_{i=1}^{N}E_i{}^2 \tag{5.40}$$

4. Compute the gradient of the error due to $\alpha(t)$and $\beta(t)$

$$\frac{\partial E(t)}{\partial \alpha_e}=-\sum_{i=1}^{N}E_i\left(\sum_{j=1}^{N}\phi_j w_{ij}^{e}\right) \tag{5.41}$$

$$\frac{\partial E(t)}{\partial \beta_e}=-\sum_{i=1}^{N}E_i\left(\sum_{j=1}^{N}\phi_j g_{ij}^{e}\right) \tag{5.42}$$

5. Update the values of $\alpha(t)$and $\beta(t)$ using the gradient descent equation

$$\alpha_e(t+1)=\alpha_e(t)+\eta\left(-\frac{\partial E(t)}{\partial \alpha_e}\right) \tag{5.43}$$

$$\beta_e(t+1)=\beta_e(t)+\eta\left(-\frac{\partial E(t)}{\partial \beta_e}\right) \tag{5.44}$$

6. Repeat steps 2-5 till convergence. Again, the algorithm converges when the error falls below a threshold.

5.4. Advantages and Modifications

The major advantage of this formulation of the FENN is that it represents the finite element model in a parallel form, enabling parallel implementation in either hardware or software. Further, computing gradients in the FENN is very simple. This is an advantage in solving both forward and inverse problems using gradient-based methods, as described in Approaches I and II. The expressions for the gradients (shown in the previous subsection) also indicate that the gradients can be computed in parallel, enabling even faster solution of both the forward and inverse problems. Secondly, the network has been derived to make the solution of inverse problems tractable. A major advantage of this approach for solving inverse problems is that it avoids inverting the global matrix in each iteration. This results in

considerable computational savings. In addition, the approach lends itself easily to solution of the forward problem. This is in contrast to other approaches described in the literature, where the networks are derived to simplify the solution procedure for the forward problem, and need considerable modification to solve the inverse problem.

The FENN also does not require any training, since most of its weights can be computed in advance and stored. The weights depend on the governing differential equation and its associated boundary conditions, and as long as these two factors do not change, the weights do not change. This is especially an advantage in solving inverse problems in electromagnetic NDE. This approach also reduces the computational effort associated with the network.

The major drawback of the FENN is the number of neurons and weights necessary. However, the memory requirements can be reduced considerably, since most of the weights between the input and hidden layer are zero. These weights, and the corresponding connections, can be discarded. Similarly, most of the elements of the **K** matrix are also zero (**K** is a banded matrix). The corresponding neurons in the hidden layer and the associated connections can also be discarded, reducing memory and computation requirements considerably. Furthermore, the weights between each group of hidden layer neurons and the output layer are the same (φ). Weight-sharing approaches can be used here to further reduce the storage requirements.

5.5. Sensitivity Analysis of the Inverse Problem

Intuitively, an error in the measurement φ will result in an error (which can be large for ill-posed problems) in the estimate of the material parameters α and β. In order to

quantify the error in the material parameters, we assume that $\tilde{\phi}_j = \phi_j + \Delta\phi_j$ is the measured value of the potentials where ϕ_j is the true value of the potential and $\Delta\phi_j$ is the error in the measurement at node j. Let $\tilde{\alpha}_e(n)$ and $\tilde{\beta}_e(n)$ be the corresponding estimated values of the material parameters in element e at iteration n. We assume that $\tilde{\alpha}_e(n) = \alpha_e(n) + \delta_e(n)$ where $\alpha_e(n)$ is the corresponding estimate of α when the measurement error in ϕ_j is zero. Similarly, let $\tilde{\beta}_e(n) = \beta_e(n) + \varepsilon_e(n)$. Then, define the output of the FENN as

$$\begin{aligned}
\tilde{b}_i(n) &= \sum_{j=1}^{N}\left(\sum_{e=1}^{M}\tilde{\alpha}_e(n)w_{ij}^e + \tilde{\beta}_e(n)g_{ij}^e\right)\tilde{\phi}_j \\
&= \sum_{j=1}^{N}\left(\sum_{e=1}^{M}\alpha_e(n)w_{ij}^e + \beta_e(n)g_{ij}^e + \delta_e(n)w_{ij}^e + \varepsilon_e(n)g_{ij}^e\right)\left(\phi_j + \Delta\phi_j\right) \\
&= \sum_{j=1}^{N}\left(\sum_{e=1}^{M}\alpha_e(n)w_{ij}^e + \beta_e(n)g_{ij}^e\right)\phi_j + \sum_{j=1}^{N}\left(\sum_{e=1}^{M}\alpha_e(n)w_{ij}^e + \beta_e(n)g_{ij}^e\right)\Delta\phi_j \\
&\quad + \sum_{j=1}^{N}\left(\sum_{e=1}^{M}\delta_e(n)w_{ij}^e + \varepsilon_e(n)g_{ij}^e\right)\left(\phi_j + \Delta\phi_j\right) \\
&= \hat{b}_i(n) + \sum_{j=1}^{N}\left(\sum_{e=1}^{M}\alpha_e(n)w_{ij}^e + \beta_e(n)g_{ij}^e\right)\Delta\phi_j \\
&\quad + \sum_{j=1}^{N}\left(\sum_{e=1}^{M}\delta_e(n)w_{ij}^e + \varepsilon_e(n)g_{ij}^e\right)\left(\phi_j + \Delta\phi_j\right)
\end{aligned} \tag{5.45}$$

where

$$\hat{b}_i(n) = \sum_{j=1}^{N}\left(\sum_{e=1}^{M}\alpha_e(n)w_{ij}^e + \beta_e(n)g_{ij}^e\right)\phi_j \tag{5.46}$$

is the output of the FENN when the measurement noise is zero. The corresponding sum-squared error at the output of the FENN is

$$\tilde{E}(n)=\frac{1}{2}\sum_{i=1}^{N}\tilde{E}_i^2(n) \tag{5.47}$$

where

$$\tilde{E}_i(n)=b_i-\tilde{b}_i(n)$$

$$=b_i-\hat{b}_i(n)-\sum_{j=1}^{N}\left(\sum_{e=1}^{M}\alpha_e(n)w_{ij}^e+\beta_e(n)g_{ij}^e\right)\Delta\phi_j$$

$$-\sum_{j=1}^{N}\left(\sum_{e=1}^{M}\delta_e(n)w_{ij}^e+\varepsilon_e(n)g_{ij}^e\right)\left(\phi_j+\Delta\phi_j\right)$$

$$=E_i(n)-\sum_{j=1}^{N}\left(\sum_{e=1}^{M}\alpha_e(n)w_{ij}^e+\beta_e(n)g_{ij}^e\right)\Delta\phi_j$$

$$-\sum_{j=1}^{N}\left(\sum_{e=1}^{M}\delta_e(n)w_{ij}^e+\varepsilon_e(n)g_{ij}^e\right)\left(\phi_j+\Delta\phi_j\right) \tag{5.48}$$

and $E_i(n)$ is the error at the FENN output when the measurement noise is zero. Then, the gradient-descent update equations for $\tilde{\alpha}$ and $\tilde{\beta}$ are given by

$$\tilde{\alpha}_e(n)=\tilde{\alpha}_e(n-1)+\eta\left(-\frac{\partial\tilde{E}(n-1)}{\partial\tilde{\alpha}_e}\right) \tag{5.49a}$$

$$\tilde{\beta}_e(n)=\tilde{\beta}_e(n-1)+\eta\left(-\frac{\partial\tilde{E}(n-1)}{\partial\tilde{\beta}_e}\right). \tag{5.49b}$$

The derivatives in (5.49) can be obtained from (5.41) and (5.42):

$$\frac{\partial\tilde{E}(n)}{\partial\tilde{\alpha}_e}=-\sum_{i=1}^{N}\tilde{E}_i(n)\left(\sum_{j=1}^{N}\tilde{\phi}_j w_{ij}^e\right)=-\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{E}_i(n)\tilde{\phi}_j w_{ij}^e \tag{5.50}$$

$$\frac{\partial\tilde{E}(n)}{\partial\tilde{\beta}_e}=-\sum_{i=1}^{N}\tilde{E}_i(n)\left(\sum_{j=1}^{N}\tilde{\phi}_j g_{ij}^e\right)=-\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{E}_i(n)\tilde{\phi}_j g_{ij}^e \tag{5.51}$$

Then, the error in the estimates $\tilde{\alpha}$ and $\tilde{\beta}$ can be computed according to Theorem I below.

Theorem I: The following results hold for the error in the estimates in $\tilde{\alpha}$ and $\tilde{\beta}$:

1. $\delta_e(n) = \delta_e(n-1) + f\left(E_i(n-1), \alpha_e(n-1), \beta_e(n-1), \delta_e(n-1), \varepsilon_e(n-1), \tilde{\phi}_j\right)$

2. $\varepsilon_e(n) = \varepsilon_e(n-1) + h\left(E_i(n-1), \alpha_e(n-1), \beta_e(n-1), \delta_e(n-1), \varepsilon_e(n-1), \tilde{\phi}_j\right)$

3. If the measurement error is bounded, i.e., $A \le \Delta\phi_j/\phi_j \le B$, then $\delta_e(n)$ and $\varepsilon_e(n)$ are also bounded.

Proof: In order to prove Theorem I (1), we start with (5.49):

$$\tilde{\alpha}_e(n) = \tilde{\alpha}_e(n-1) + \eta\left(-\frac{\partial \tilde{E}(n-1)}{\partial \tilde{\alpha}_e}\right) \tag{5.52}$$

Substituting (5.50) in (5.49),

$$\tilde{\alpha}_e(n) = \tilde{\alpha}_e(n-1) + \eta\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{E}_i(n-1)\tilde{\phi}_j w_{ij}^e$$

$$= \tilde{\alpha}_e(n-2) + \eta\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{E}_i(n-2)\tilde{\phi}_j w_{ij}^e + \eta\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{E}_i(n-1)\tilde{\phi}_j w_{ij}^e \tag{5.53}$$

Continuing with the recursion, we get

$$\tilde{\alpha}_e(n) = \alpha_e(0) + \eta\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{E}_i(0)\tilde{\phi}_j w_{ij}^e + \eta\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{E}_i(1)\tilde{\phi}_j w_{ij}^e + \ldots + \eta\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{E}_i(n-1)\tilde{\phi}_j w_{ij}^e \tag{5.54}$$

or

$$\tilde{\alpha}_e(n) = \alpha_e(0) + \eta\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{\phi}_j w_{ij}^e\sum_{t=0}^{n-1}\tilde{E}_i(t) \tag{5.55}$$

From (5.48) and (5.55), we have

$$\tilde{\alpha}_e(n) = \alpha_e(0)$$

$$+\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{n-1}\left[E_i(t)-\sum_{k=1}^{N}\Delta\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(t)w_{ij}^{e'}+\beta_{e'}(t)g_{ij}^{e'}\right)-\sum_{k=1}^{N}\tilde{\phi}_k\sum_{e'=1}^{M}\left(\delta_{e'}(t)w_{ij}^{e'}+\varepsilon_{e'}(t)g_{ij}^{e'}\right)\right] \tag{5.56}$$

This can be simplified to give

$$\tilde{\alpha}_e(n)=\alpha_e(0)+\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{n-1}E_i(t)-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{n-1}\sum_{k=1}^{N}\Delta\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(t)w_{ij}^{e'}+\beta_{e'}(t)g_{ij}^{e'}\right)$$

$$-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{n-1}\sum_{k=1}^{N}\tilde{\phi}_k\sum_{e'=1}^{M}\left(\delta_{e'}(t)w_{ij}^{e'}+\varepsilon_{e'}(t)g_{ij}^{e'}\right) \tag{5.57}$$

From the definition of $\tilde{\phi}_i$, we get

$$\tilde{\alpha}_e(n)=\alpha_e(0)+\eta\sum_{i,j=1}^{N}\phi_j w_{ij}^{e}\sum_{t=0}^{n-1}E_i(t)+\eta\sum_{i,j=1}^{N}\Delta\phi_j w_{ij}^{e}\sum_{t=0}^{n-1}E_i(t)$$
$$-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{n-1}\sum_{k=1}^{N}\Delta\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(t)w_{ij}^{e'}+\beta_{e'}(t)g_{ij}^{e'}\right)-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{n-1}\sum_{k=1}^{N}\tilde{\phi}_k\sum_{e'=1}^{M}\left(\delta_{e'}(t)w_{ij}^{e'}+\varepsilon_{e'}(t)g_{ij}^{e'}\right) \tag{5.58}$$

But

$$\alpha_e(n)=\alpha_e(0)+\eta\sum_{i,j}\phi_j w_{ij}^{e}\sum_{t=0}^{n-1}E_i(t) \tag{5.59}$$

So,

$$\tilde{\alpha}_e(n)=\alpha_e(n)+\eta\sum_{i,j=1}^{N}\Delta\phi_j w_{ij}^{e}\sum_{t=0}^{n-1}E_i(t)-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{n-1}\sum_{k=1}^{N}\Delta\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(t)w_{ij}^{e'}+\beta_{e'}(t)g_{ij}^{e'}\right)$$

$$-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{n-1}\sum_{k=1}^{N}\tilde{\phi}_k\sum_{e'=1}^{M}\left(\delta_{e'}(t)w_{ij}^{e'}+\varepsilon_{e'}(t)g_{ij}^{e'}\right) \tag{5.60}$$

or

$$\tilde{\alpha}_e(n)=\alpha_e(n)+\delta_e(n) \tag{5.61}$$

where

$$\delta_e(n)=\eta\sum_{i,j=1}^{N}\Delta\phi_j w_{ij}^{e}\sum_{t=0}^{n-1}E_i(t)-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{n-1}\sum_{k=1}^{N}\Delta\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(t)w_{ij}^{e'}+\beta_{e'}(t)g_{ij}^{e'}\right)$$

$$-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{n-1}\sum_{k=1}^{N}\tilde{\phi}_k\sum_{e'=1}^{M}\left(\delta_{e'}(t)w_{ij}^{e'}+\varepsilon_{e'}(t)g_{ij}^{e'}\right) \tag{5.62}$$

with $\delta_e(0)=\varepsilon_e(0)=0$. Using this fact and (5.62), we have

$$\delta_e(1)=\eta\sum_{i,j=1}^{N}\Delta\phi_j w_{ij}^{e}E_i(0)-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{k=1}^{N}\Delta\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(0)w_{ij}^{e'}+\beta_{e'}(0)g_{ij}^{e'}\right) \tag{5.63}$$

$$\delta_e(2)=\eta\sum_{i,j=1}^{N}\Delta\phi_j w_{ij}^{e}\sum_{t=0}^{1}E_i(t)-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{1}\sum_{k=1}^{N}\Delta\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(t)w_{ij}^{e'}+\beta_{e'}(t)g_{ij}^{e'}\right)$$

$$-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{t=0}^{1}\sum_{k=1}^{N}\tilde{\phi}_k\sum_{e'=1}^{M}\left(\delta_{e'}(t)w_{ij}^{e'}+\varepsilon_{e'}(t)g_{ij}^{e'}\right)$$

or

$$\delta_e(2)=\delta_e(1)+\eta\sum_{i,j=1}^{N}\Delta\phi_j w_{ij}^{e}E_i(1)-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{k=1}^{N}\Delta\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(1)w_{ij}^{e'}+\beta_{e'}(1)g_{ij}^{e'}\right)$$

$$-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{k=1}^{N}\tilde{\phi}_k\sum_{e'=1}^{M}\left(\delta_{e'}(1)w_{ij}^{e'}+\varepsilon_{e'}(1)g_{ij}^{e'}\right) \tag{5.64}$$

This process can be carried out for all n, and the recursive relation is given by

$$\delta_e(n)=\delta_e(n-1)+\eta\sum_{i,j=1}^{N}E_i(n-1)\Delta\phi_j w_{ij}^{e}-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{k=1}^{N}\Delta\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(n-1)w_{ik}^{e'}+\beta_{e'}(n-1)g_{ik}^{e'}\right)$$

$$-\eta\sum_{i,j=1}^{N}\tilde{\phi}_j w_{ij}^{e}\sum_{k=1}^{N}\tilde{\phi}_k\sum_{e'=1}^{M}\left(\delta_{e'}(n-1)w_{ik}^{e'}+\varepsilon_{e'}(n-1)g_{ik}^{e'}\right) \tag{5.65}$$

or

$$\delta_e(n)=\delta_e(n-1)+f\left(E_i(n-1),\alpha_e(n-1),\beta_e(n-1),\delta_e(n-1),\varepsilon_e(n-1),\tilde{\phi}_j\right) \tag{5.66}$$

By a similar process, we can prove Theorem I (2):

$$\varepsilon_e(n)=\varepsilon_e(n-1)+h\left(E_i(n-1),\alpha_e(n-1),\beta_e(n-1),\delta_e(n-1),\varepsilon_e(n-1),\tilde{\phi}_j\right) \tag{5.67}$$

To prove Theorem I (3), we start with the fact that the measurement error is bounded:

$$A\le\Delta\phi_j/\phi_j\le B \tag{5.68}$$

Then, from (5.65),

$$\begin{aligned}\delta_e(n)\le\delta_e(n-1)&+\eta\max\left(A\sum_{i,j=1}^{N}E_i(n-1)\phi_j w_{ij}^e,B\sum_{i,j=1}^{N}E_i(n-1)\phi_j w_{ij}^e\right)\\&-\eta\min\Bigg[A(1+A)\sum_{i,j=1}^{N}\phi_j w_{ij}^e\sum_{k=1}^{N}\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(n-1)w_{ik}^{e'}+\beta_{e'}(n-1)g_{ik}^{e'}\right),\\&\qquad B(1+B)\sum_{i,j=1}^{N}\phi_j w_{ij}^e\sum_{k=1}^{N}\phi_k\sum_{e'=1}^{M}\left(\alpha_{e'}(n-1)w_{ik}^{e'}+\beta_{e'}(n-1)g_{ik}^{e'}\right)\Bigg]\\&-\eta\min\Bigg[(1+A)^2\sum_{i,j=1}^{N}\phi_j w_{ij}^e\sum_{k=1}^{N}\phi_k\sum_{e'=1}^{M}\left(\delta_{e'}(n-1)w_{ik}^{e'}+\varepsilon_{e'}(n-1)g_{ik}^{e'}\right),\\&\qquad(1+B)^2\sum_{i,j=1}^{N}\phi_j w_{ij}^e\sum_{k=1}^{N}\phi_k\sum_{e'=1}^{M}\left(\delta_{e'}(n-1)w_{ik}^{e'}+\varepsilon_{e'}(n-1)g_{ik}^{e'}\right)\Bigg]\end{aligned} \tag{5.69}$$

Recognizing that

$$\hat{b}_i(n)=\sum_{j=1}^{N}\left(\sum_{e=1}^{M}\alpha_e(n)w_{ij}^e+\beta_e(n)g_{ij}^e\right)\phi_j\,, \tag{5.70}$$

$$\begin{aligned}\delta_e(n)\le\delta_e(n-1)&+\eta\max\left(A\sum_{i,j=1}^{N}E_i(n-1)\phi_j w_{ij}^e,B\sum_{i,j=1}^{N}E_i(n-1)\phi_j w_{ij}^e\right)\\&-\eta\min\left(A(1+A)\sum_{i,j=1}^{N}\phi_j w_{ij}^e\hat{b}_i(n-1),B(1+B)\sum_{i,j=1}^{N}\phi_j w_{ij}^e\hat{b}_i(n-1)\right)\\&-\eta\min\Bigg[(1+A)^2\sum_{i,j=1}^{N}\phi_j w_{ij}^e\sum_{k=1}^{N}\phi_k\sum_{e'=1}^{M}\left(\delta_{e'}(n-1)w_{ik}^{e'}+\varepsilon_{e'}(n-1)g_{ik}^{e'}\right),\end{aligned}$$

$$(1+B)^2\sum_{i,j=1}^{N}\phi_j w_{ij}^{\epsilon}\sum_{k=1}^{N}\phi_k\sum_{\epsilon'=1}^{M}\left(\delta_{\epsilon'}(n-1)w_{ik}^{\epsilon'}+\varepsilon_{\epsilon'}(n-1)g_{ik}^{\epsilon'}\right)\Bigg] \tag{5.71}$$

Also,

$$\delta_{\epsilon}(n)\geq\delta_{\epsilon}(n-1)+\eta\min\left(A\sum_{i,j=1}^{N}E_i(n-1)\phi_j w_{ij}^{\epsilon},B\sum_{i,j=1}^{N}E_i(n-1)\phi_j w_{ij}^{\epsilon}\right)$$

$$-\eta\max\left(A(1+A)\sum_{i,j=1}^{N}\phi_j w_{ij}^{\epsilon}\hat{b}_i(n-1),B(1+B)\sum_{i,j=1}^{N}\phi_j w_{ij}^{\epsilon}\hat{b}_i(n-1)\right)$$

$$-\eta\max\Bigg[(1+A)^2\sum_{i,j=1}^{N}\phi_j w_{ij}^{\epsilon}\sum_{k=1}^{N}\phi_k\sum_{\epsilon'=1}^{M}\left(\delta_{\epsilon'}(n-1)w_{ik}^{\epsilon'}+\varepsilon_{\epsilon'}(n-1)g_{ik}^{\epsilon'}\right),$$

$$(1+B)^2\sum_{i,j=1}^{N}\phi_j w_{ij}^{\epsilon}\sum_{k=1}^{N}\phi_k\sum_{\epsilon'=1}^{M}\left(\delta_{\epsilon'}(n-1)w_{ik}^{\epsilon'}+\varepsilon_{\epsilon'}(n-1)g_{ik}^{\epsilon'}\right)\Bigg] \tag{5.72}$$

Similarly,

$$\varepsilon_{\epsilon}(n)\leq\varepsilon_{\epsilon}(n-1)+\eta\max\left(A\sum_{i,j=1}^{N}E_i(n-1)\phi_j g_{ij}^{\epsilon},B\sum_{i,j=1}^{N}E_i(n-1)\phi_j g_{ij}^{\epsilon}\right)$$

$$-\eta\min\left(A(1+A)\sum_{i,j=1}^{N}\phi_j g_{ij}^{\epsilon}\hat{b}_i(n-1),B(1+B)\sum_{i,j=1}^{N}\phi_j g_{ij}^{\epsilon}\hat{b}_i(n-1)\right)$$

$$-\eta\min\Bigg[(1+A)^2\sum_{i,j=1}^{N}\phi_j g_{ij}^{\epsilon}\sum_{k=1}^{N}\phi_k\sum_{\epsilon'=1}^{M}\left(\delta_{\epsilon'}(n-1)w_{ik}^{\epsilon'}+\varepsilon_{\epsilon'}(n-1)g_{ik}^{\epsilon'}\right),$$

$$(1+B)^2\sum_{i,j=1}^{N}\phi_j g_{ij}^{\epsilon}\sum_{k=1}^{N}\phi_k\sum_{\epsilon'=1}^{M}\left(\delta_{\epsilon'}(n-1)w_{ik}^{\epsilon'}+\varepsilon_{\epsilon'}(n-1)g_{ik}^{\epsilon'}\right)\Bigg] \tag{5.73}$$

$$\varepsilon_{\epsilon}(n)\geq\varepsilon_{\epsilon}(n-1)+\eta\min\left(A\sum_{i,j=1}^{N}E_i(n-1)\phi_j g_{ij}^{\epsilon},B\sum_{i,j=1}^{N}E_i(n-1)\phi_j g_{ij}^{\epsilon}\right)$$

$$-\eta\max\left(A(1+A)\sum_{i,j=1}^{N}\phi_j g_{ij}^{\epsilon}\hat{b}_i(n-1),B(1+B)\sum_{i,j=1}^{N}\phi_j g_{ij}^{\epsilon}\hat{b}_i(n-1)\right)$$

$$-\eta \max\left[(1+A)^2 \sum_{i,j=1}^{N} \phi_j g_{ij}^{e} \sum_{k=1}^{N} \phi_k \sum_{e'=1}^{M} \left(\delta_{e'}(n-1) w_{ik}^{e'} + \varepsilon_{e'}(n-1) g_{ik}^{e'}\right),\right.$$

$$\left.(1+B)^2 \sum_{i,j=1}^{N} \phi_j g_{ij}^{e} \sum_{k=1}^{N} \phi_k \sum_{e'=1}^{M} \left(\delta_{e'}(n-1) w_{ik}^{e'} + \varepsilon_{e'}(n-1) g_{ik}^{e'}\right)\right] \tag{5.74}$$

Equations (5.71)-(5.74) indicate that the error in the estimates $\tilde{\alpha}$ and $\tilde{\beta}$ are bounded if the measurement error are.

□

Corollary I: If $A = 0$ and $B \ll 1$, then

$$\delta_e(n) \le \delta_e(n-1) + \eta \max\left(0, B \sum_{i,j=1}^{N} E_i(n-1) \phi_j w_{ij}^{e}\right) - \eta \min\left(0, B \sum_{i,j=1}^{N} \phi_j w_{ij}^{e} \hat{b}_i(n-1)\right)$$

$$-\eta \sum_{i,j=1}^{N} \phi_j w_{ij}^{e} \sum_{k=1}^{N} \phi_k \sum_{e'=1}^{M} \left(\delta_{e'}(n-1) w_{ik}^{e'} + \varepsilon_{e'}(n-1) g_{ik}^{e'}\right) \tag{5.75}$$

$$\delta_e(n) \ge \delta_e(n-1) + \eta \min\left(0, B \sum_{i,j=1}^{N} E_i(n-1) \phi_j w_{ij}^{e}\right) - \eta \max\left(0, B \sum_{i,j=1}^{N} \phi_j w_{ij}^{e} \hat{b}_i(n-1)\right)$$

$$-\eta \sum_{i,j=1}^{N} \phi_j w_{ij}^{e} \sum_{k=1}^{N} \phi_k \sum_{e'=1}^{M} \left(\delta_{e'}(n-1) w_{ik}^{e'} + \varepsilon_{e'}(n-1) g_{ik}^{e'}\right) \tag{5.76}$$

$$\varepsilon_e(n) \le \varepsilon_e(n-1) + \eta \max\left(0, B \sum_{i,j=1}^{N} E_i(n-1) \phi_j g_{ij}^{e}\right) - \eta \min\left(0, B \sum_{i,j=1}^{N} \phi_j g_{ij}^{e} \hat{b}_i(n-1)\right)$$

$$-\eta \sum_{i,j=1}^{N} \phi_j g_{ij}^{e} \sum_{k=1}^{N} \phi_k \sum_{e'=1}^{M} \left(\delta_{e'}(n-1) w_{ik}^{e'} + \varepsilon_{e'}(n-1) g_{ik}^{e'}\right) \tag{5.77}$$

$$\varepsilon_e(n) \ge \varepsilon_e(n-1) + \eta \min\left(0, B \sum_{i,j=1}^{N} E_i(n-1) \phi_j g_{ij}^{e}\right) - \eta \max\left(0, B \sum_{i,j=1}^{N} \phi_j g_{ij}^{e} \hat{b}_i(n-1)\right)$$

$$-\eta \sum_{i,j=1}^{N} \phi_j g_{ij}^{e} \sum_{k=1}^{N} \phi_k \sum_{e'=1}^{M} \left(\delta_{e'}(n-1) w_{ik}^{e'} + \varepsilon_{e'}(n-1) g_{ik}^{e'}\right) \tag{5.78}$$

Proof: Substitute $A=0$ and $B \ll 1$ in (5.71)-(5.74), and eliminate all terms which are multiplied by B^2.

□

It is important to note that, if $A=B=0$ (no noise), then $\delta_e(n)=\varepsilon_e(n)=0\ \forall n$ and the estimates $\tilde{\alpha}$ and $\tilde{\beta}$ indeed equal α and β (the estimates for the zero noise case).

Theorem I confirms an important and intuitive fact: that the error in the estimates of the material properties at any iteration depends on the error in the material property estimates in all previous iterations. This dependence is shown explicitly in equations (5.64)-(5.67). This fact is not a drawback of the FENN-based inversion algorithm. Rather, it shows the ill-posed nature of the inverse problem, and illustrates the point that the inversion results are only as good as the measured data. We addressed a similar issue in Chapters 3 and 4, where the error in the defect profile estimates from MFL signals was higher for shallow defects, when the percentage of additive noise was constant. In this case the depth information was carried in the amplitude of the MFL signal, and this information was corrupted by small amounts of noise.

Equations (5.75)-(5.78) can be modified to give certain requirements for convergence. To see this, we first, rewrite (5.75) and (5.76) as

$$\delta_e(n)+\eta\sum_{i,j=1}^{N}\phi_j w_{ij}^{e}\sum_{k=1}^{N}\phi_k\sum_{e'=1}^{M}\left(\delta_{e'}(n-1)w_{ik}^{e'}+\varepsilon_{e'}(n-1)g_{ik}^{e'}\right)$$

$$\le \delta_e(n-1)+\eta\max\left(0,B\sum_{i,j=1}^{N}E_i(n-1)\phi_j w_{ij}^{e}\right)-\eta\min\left(0,B\sum_{i,j=1}^{N}\phi_j w_{ij}^{e}\hat{b}_i(n-1)\right) \qquad (5.79)$$

$$\delta_e(n)+\eta\sum_{i,j=1}^{N}\phi_j w_{ij}^{e}\sum_{k=1}^{N}\phi_k\sum_{e'=1}^{M}\left(\delta_{e'}(n-1)w_{ik}^{e'}+\varepsilon_{e'}(n-1)g_{ik}^{e'}\right)$$

$$\geq \delta_e(n-1)+\eta\min\left(0,B\sum_{i,j=1}^{N}E_i(n-1)\phi_j w_{ij}^{e}\right)-\eta\max\left(0,B\sum_{i,j=1}^{N}\phi_j w_{ij}^{e}\hat{b}_i(n-1)\right) \tag{5.80}$$

Similarly,

$$\varepsilon_e(n)+\eta\sum_{i,j=1}^{N}\phi_j g_{ij}^{e}\sum_{k=1}^{N}\phi_k\sum_{e'=1}^{M}\left(\delta_{e'}(n-1)w_{ik}^{e'}+\varepsilon_{e'}(n-1)g_{ik}^{e'}\right)$$

$$\leq \varepsilon_e(n-1)+\eta\max\left(0,B\sum_{i,j=1}^{N}E_i(n-1)\phi_j g_{ij}^{e}\right)-\eta\min\left(0,B\sum_{i,j=1}^{N}\phi_j g_{ij}^{e}\hat{b}_i(n-1)\right) \tag{5.81}$$

$$\varepsilon_e(n)+\eta\sum_{i,j=1}^{N}\phi_j g_{ij}^{e}\sum_{k=1}^{N}\phi_k\sum_{e'=1}^{M}\left(\delta_{e'}(n-1)w_{ik}^{e'}+\varepsilon_{e'}(n-1)g_{ik}^{e'}\right)$$

$$\geq \varepsilon_e(n-1)+\eta\min\left(0,B\sum_{i,j=1}^{N}E_i(n-1)\phi_j g_{ij}^{e}\right)-\eta\max\left(0,B\sum_{i,j=1}^{N}\phi_j g_{ij}^{e}\hat{b}_i(n-1)\right) \tag{5.82}$$

Now, define the following terms:

$$K_1(n)=\eta B\sum_{i,j=1}^{N}E_i(n)\phi_j w_{ij}^{e} \tag{5.83}$$

$$K_2(n)=\eta B\sum_{i,j=1}^{N}\hat{b}_i(n)\phi_j w_{ij}^{e} \tag{5.84}$$

$$K_3=\eta\sum_{i,j=1}^{N}\phi_j w_{ij}^{e}\sum_{k=1}^{N}\phi_k w_{ik}^{e} \tag{5.85}$$

$$K_4(n)=\eta\sum_{i,j=1}^{N}\phi_j w_{ij}^{e}\sum_{k=1}^{N}\phi_k g_{ik}^{e}\varepsilon_e(n) \tag{5.86}$$

$$K_5(n)=\eta\sum_{i,j=1}^{N}\phi_j w_{ij}^{e}\sum_{k=1}^{N}\phi_k\sum_{1\leq e'\leq M,e'\neq e}\delta_{e'}(n)w_{ik}^{e'}+\varepsilon_{e'}(n)g_{ik}^{e'} \tag{5.87}$$

Also, let

$$f_n(K_2) = \min(0, K_2(n)) \tag{5.88}$$

$$h_n(K_2) = \max(0, K_2(n)) \tag{5.89}$$

$$s_n(K_1) = \max(0, K_1(n)) \tag{5.90}$$

$$t_n(K_1) = \min(0, K_1(n)) \tag{5.91}$$

Then, from (5.79), we have

$$\delta_e(n) + K_4(n-1) + K_5(n-1) \le \delta_e(n-1)(1-K_3) + s_{n-1}(K_1) - f_{n-1}(K_2) \tag{5.92}$$

Similarly, from (5.80), we get

$$\delta_e(n) + K_4(n-1) + K_5(n-1) \ge \delta_e(n-1)(1-K_3) + t_{n-1}(K_1) - h_{n-1}(K_2) \tag{5.93}$$

Using the fact that $\delta_e(0) = \varepsilon_e(0) = 0$, we get

$$\delta_e(1) \le s_0(K_1) - f_0(K_2) \tag{5.94}$$

Substituting the upper limit for $\delta_e(1)$ from (5.94), the upper limit for $\delta_e(2)$ can be computed as

$$\delta_e(2) + K_4(1) + K_5(1) \le [s_0(K_1) - f_0(K_2)](1-K_3) + s_1(K_1) - f_1(K_2) \tag{5.95}$$

Similarly,

$$\delta_e(3) + K_4(2) + K_5(2) \le [s_0(K_1) - f_0(K_2)](1-K_3)^2 + [s_1(K_1) - f_1(K_2)](1-K_3) + s_2(K_1) - f_2(K_2) \tag{5.96}$$

The general term in the recursion for iteration n+1 can be shown to be

$$\delta_e(n+1) + K_4(n) + K_5(n) \le \sum_{r=0}^{n} [s_r(K_1) - f_r(K_2)](1-K_3)^{n-r} \tag{5.97}$$

Similarly, we can show that

$$\delta_e(n+1)+K_4(n)+K_5(n)\geq\sum_{r=0}^{n}\left[t_r(K_1)-h_r(K_2)\right](1-K_3)^{n-r} \tag{5.98}$$

Equations (5.97) and (5.98) give the upper and lower bounds on the error in α in element e. In order for the summation to converge, we require that

$$|1-K_3|<1 \tag{5.99}$$

If this condition is met, as the number of iterations increases (i.e., $n\rightarrow\infty$), the terms in the summation corresponding to small values of r are weighted less and less. However, the bounds are never zero. The only manner in which the bounds can be zero is if $B=0$. Note that, if $K_3=1$, there is still one non-zero term (corresponding to $r=n$). Now, K_1 and K_2 depend on the weights (which in turn depend on the differential equation), source term, and the noise-free measurements and corresponding values of α and β. Thus, if $K_3=1$, then the bounds on δ_e at iteration n will depend on only the noise free parameters determined at iteration n-1, i.e.,

$$\delta_e(n+1)+K_4(n)+K_5(n)\leq s_n(K_1)-f_n(K_2) \tag{5.100}$$

$$\delta_e(n+1)+K_4(n)+K_5(n)\geq t_n(K_1)-h_n(K_2) \tag{5.101}$$

Since

$$K_3=\eta\sum_{i,j=1}^{N}\phi_j w_{ij}^e\sum_{k=1}^{N}\phi_k w_{ik}^e\,, \tag{5.102}$$

$K_3=1$ if

$$\eta=\frac{1}{\sum_{i,j=1}^{N}\phi_j w_{ij}^e\sum_{k=1}^{N}\phi_k w_{ik}^e} \tag{5.103}$$

This value of η is specific to element e. This result indicates that using different learning rates for different elements will result in a tighter error bound for the error in α.

In a similar manner, we can estimate the error bounds for ε. Define:

$$C_1(n) = \eta B \sum_{i,j=1}^{N} E_i(n) \phi_j g_{ij}^e \tag{5.104}$$

$$C_2(n) = \eta B \sum_{i,j=1}^{N} \hat{b}_i(n) \phi_j g_{ij}^e \tag{5.105}$$

$$C_3 = \eta \sum_{i,j=1}^{N} \phi_j g_{ij}^e \sum_{k=1}^{N} \phi_k g_{ik}^e \tag{5.106}$$

$$C_4(n) = \eta \sum_{i,j=1}^{N} \phi_j g_{ij}^e \sum_{k=1}^{N} \phi_k w_{ik}^e \delta_e(n) \tag{5.107}$$

$$C_5(n) = \eta \sum_{i,j=1}^{N} \phi_j g_{ij}^e \sum_{k=1}^{N} \phi_k \sum_{1 \le e' \le M, e' \ne e} \delta_{e'}(n) w_{ik}^{e'} + \varepsilon_{e'}(n) g_{ik}^{e'} \tag{5.108}$$

Then, using (5.88)-(5.91), and the recursions of (5.77) and (5.78), the limits on $\varepsilon_e(n+1)$ are given by

$$\varepsilon_e(n+1) + C_4(n) + C_5(n) \le \sum_{r=0}^{n} \left[s_r(C_1) - f_r(C_2) \right] (1 - C_3)^{n-r} \tag{5.109}$$

$$\varepsilon_e(n+1) + C_4(n) + C_5(n) \ge \sum_{r=0}^{n} \left[t_r(C_1) - h_r(C_2) \right] (1 - C_3)^{n-r} \tag{5.110}$$

The learning rate for ε_e is then

$$\eta = \frac{1}{\sum_{i,j=1}^{N} \phi_j g_{ij}^e \sum_{k=1}^{N} \phi_k g_{ik}^e} \tag{5.111}$$

Equations (5.103) and (5.111) indicate that the learning rate depends on the weights between the input and hidden layer, which in turn depend on the differential equation. These

weights can be pre-computed. However, the learning rate also depends on the error-free measurement ϕ_j. Since this quantity is unknown, the learning rate cannot be determined using (5.103) or (5.111), and a trial-and-error method must be used to determine the optimal learning rate. Alternatively, if several measurements are available, an average value of ϕ_j can be used to estimate η.

5.6. Results

5.6.1. One Dimensional Problems - Forward Model Results

The finite element model neural network (FENN) was tested using a one-dimensional version of Poisson's equation, which is a special case of (5.16):

$$-\nabla \bullet (\varepsilon \nabla \phi) = \rho \tag{5.112}$$

For a one-dimensional problem, this reduces to

$$-\frac{\partial}{\partial x}\left(\varepsilon \frac{\partial \phi}{\partial x}\right) = \rho . \tag{5.113}$$

Several different examples based on (5.113) were used to test the performance of the FENN on the forward problem. The first problem that was tested was

$$\frac{\partial^2 \phi}{\partial x^2} = 0, \quad x \in [0,1] \tag{5.114}$$

with the boundary conditions

$$\phi(0) = 0 \text{ and } \phi(1) = K, \quad K \in \{...,-3,-2,-1,1,2,3,...\} \tag{5.115}$$

The analytical solution to this problem is

$$\phi = \begin{cases} Kx, & x \in [0,1] \\ 0, & elsewhere \end{cases} \tag{5.116}$$

The FENN was tested by setting $\varepsilon = 1, \rho = 0$ and K=1. The domain of interest $[0,1]$ was divided into ten elements (with eleven nodes) and the weights for the first layer of neurons were pre-computed. These weight values are well documented in the literature [8]. The results for this problem are shown in Figure 31. The solid line shows the analytical solution while the stars show the result determined by the FENN. Similar results for K=5 are shown in Figure 32. These results were also obtained using ten elements and eleven nodes.

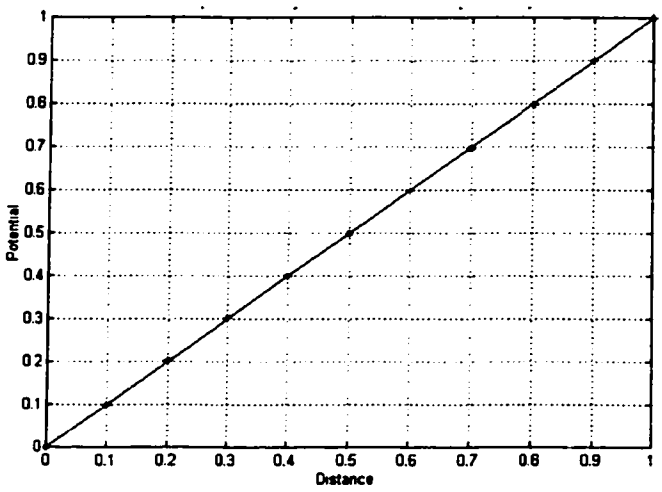


Figure 31. Comparison of analytical solution and FENN solution for Laplace's equation with K=1.

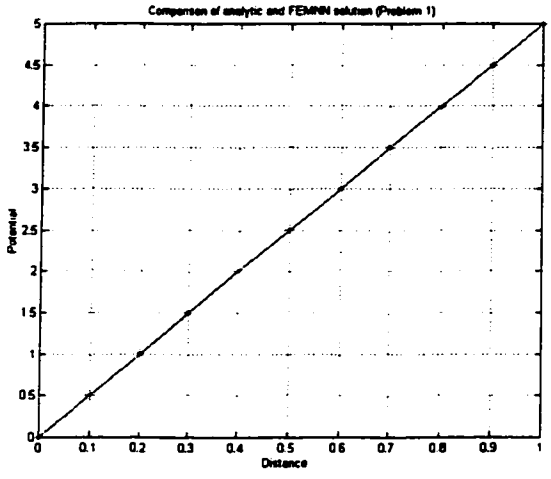


Figure 32. Comparison of analytical solution and FENN solution for Laplace's equation (K=5).

The second example that was tested was the one-dimensional Poisson's equation with $\varepsilon = 1$ and $\rho = -10$ with the same boundary conditions as above:

$$\phi(0) = 0 \text{ and } \phi(1) = K, \quad K \in \{..., -3, -2, -1, 1, 2, 3, ...\} \tag{5.117}$$

Again, the domain of interest was divided up into ten elements. The results for K=5 are shown in Figure 33. The analytical solution is shown using squares, while the FEM solution is shown using diamonds. The FENN result is shown using stars. The initial solution for the potential is indicated by the dashed line with triangles in the figure. The analytical solution for this problem is given by

$$\phi = \begin{cases} 5x^2, & x \in [0,1] \\ 0, & elsewhere \end{cases} \tag{5.118}$$

The results predicted by the FENN show excellent agreement with the analytical solution.

Similarly, for $\rho = 10$ and $K = 5$, the analytical solution is

$$\phi = \begin{cases} -5x^2 + 10x, & x \in [0,1] \\ 0, & elsewhere \end{cases} \tag{5.119}$$

and a comparison of the analytical solution (squares), the FEM solution (diamonds) and FENN solution (stars) for this problem is shown in Figure 34. Again, the initial solution is indicated by the dashed line with the triangles.

These results indicate that the FENN is capable of accurately solving for the potential ϕ. The algorithm also converged in relatively few iterations (approximately 500 iterations on average for all four problems) and the sum-squared error over all the output nodes was less than 0.0001 for all four sets of results.

As mentioned above, one advantage of the FENN approach is that the input-first hidden layer nodes can be computed once. These weights were used for all the four problems

described here. The only changes necessary to solve the different problems are changes in the input (ε) and the desired output (ρ).

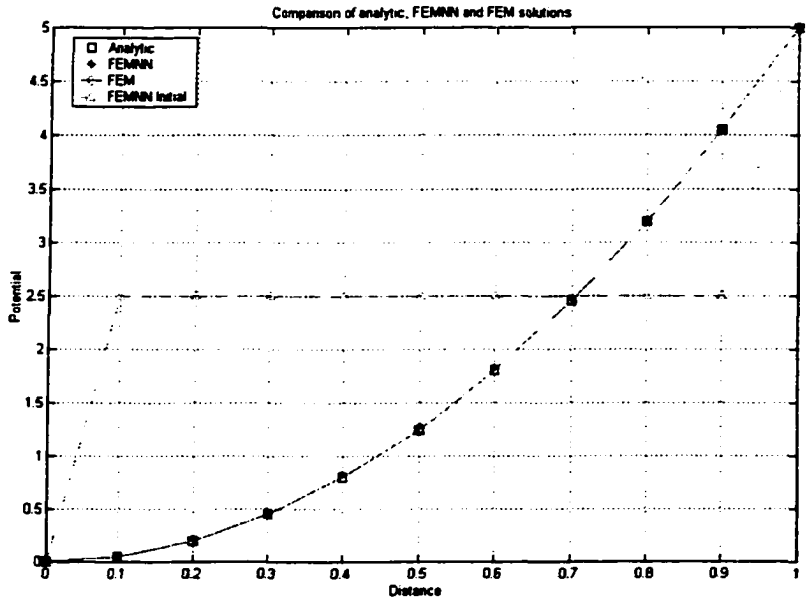


Figure 33. Comparison of analytical, FEM and FENN solutions for Poisson's equation (ρ=-10).

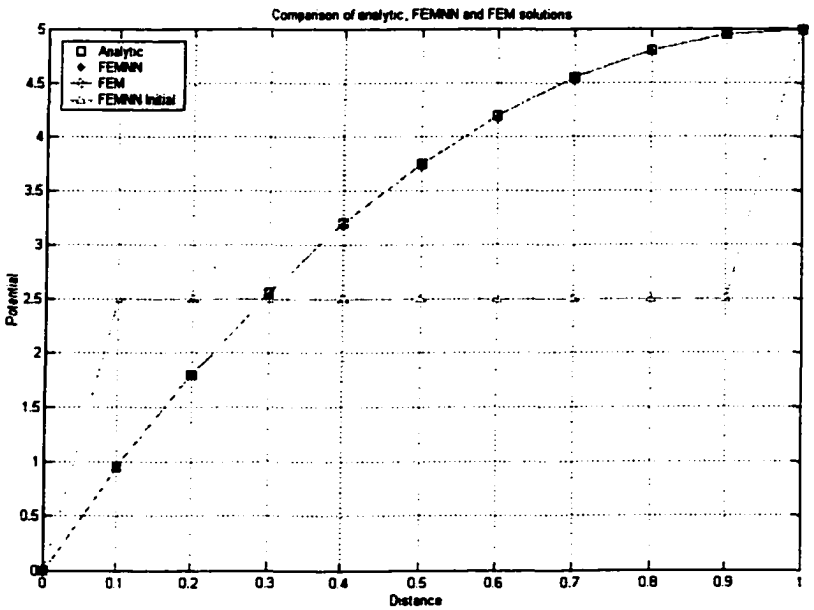


Figure 34. Comparison of analytical, FEM and FENN solutions for Poisson's equation (ρ=10).

5.6.2. One Dimensional Problems - Inverse Model Results

The FENN was also used to solve several simple inverse problems based on Poisson's equation and Laplace's equation. In all cases, the objective was to determine the value of ε for given values of ρ and ϕ. The results of this exercise are summarized below.

The first problem involves determining ε given $\rho = 1$ and $\phi = x, \quad x \in [0,1]$. The analytical solution to this inverse problem is

$$\varepsilon = K - x, \quad x \in [0,1] \text{and } K \in \mathbb{R} \tag{5.120}$$

As seen from (5.120), this inverse problem, and all the others that follow, have an infinite number of solutions and we expect the solution procedure to converge to one of these solutions depending on the initialization.

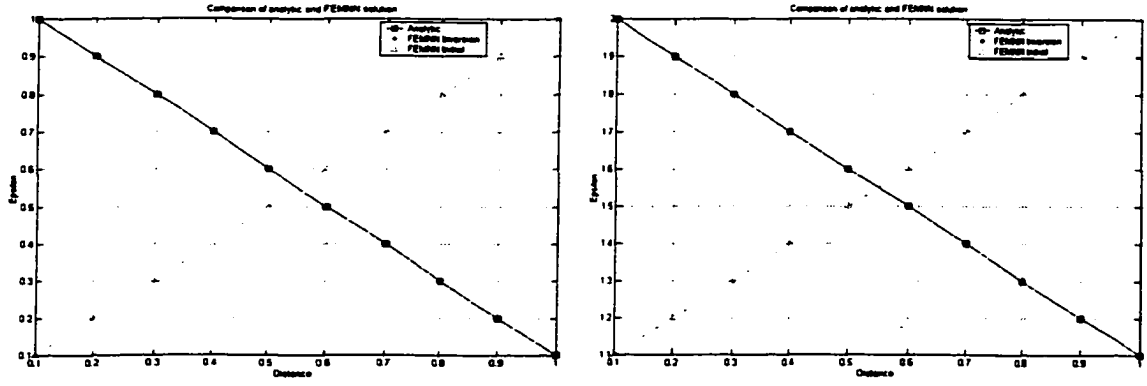
Figure 35 (a) and (b) show the two solutions to this inverse problem for two different initializations (shown using triangles): $\varepsilon = x$ and $\varepsilon = 1 + x$ respectively. The solution converges to $\varepsilon = 1 - x$ and $\varepsilon = 2 - x$ respectively and the FENN solution (in stars) is seen to match the analytical solution (squares) exactly.

In order to further test the algorithm, the same problem was solved using four more initializations. The first two initialized ε to a constant value and the results are shown in Figure 36. Similarly, Figure 37 shows the results for a random initialization. In this case, the analytical result was obtained by drawing a straight line between the first and last values of ε.

Similar results are presented in Figure 38 for the inverse problem when $\rho = -1$ and $\phi = x, \quad x \in [0,1]$. The analytical solution is

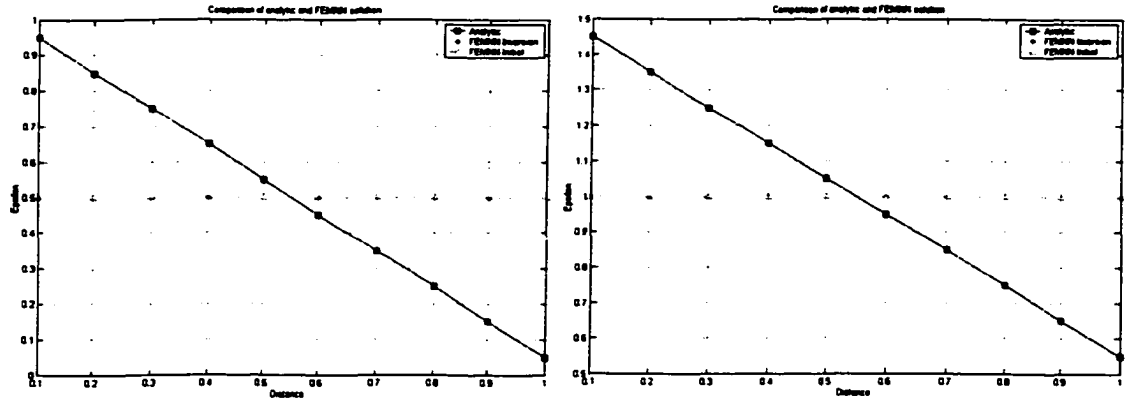
$$\varepsilon = K + x, \quad x \in [0,1] \text{and } K \in \mathbb{R} \tag{5.121}$$

The FENN results indicate that the algorithm converges to $\varepsilon = x$ and $\varepsilon = 1 + x$ for the initialization solutions $\varepsilon = 1 - x$ and $\varepsilon = 2 - x$ respectively.



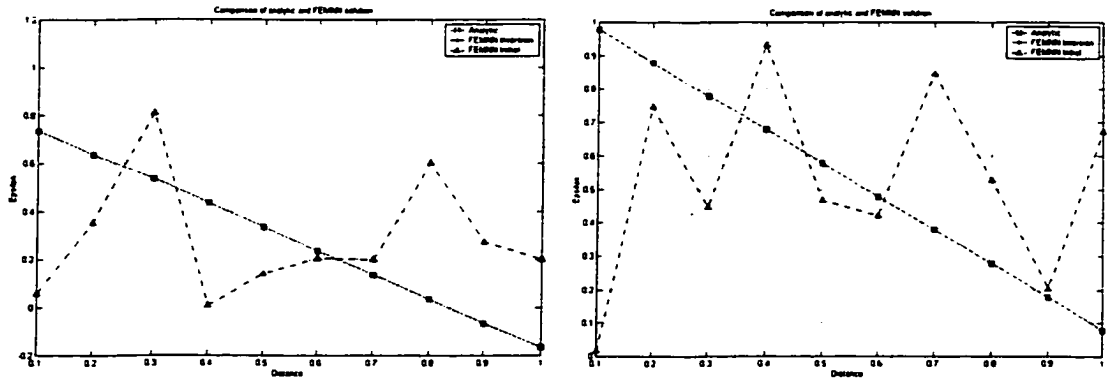
(a) (b)

Figure 35. FENN inversion results for Poisson's equation with (a) initial solution ε=x and (b) initial solution ε=1+x.



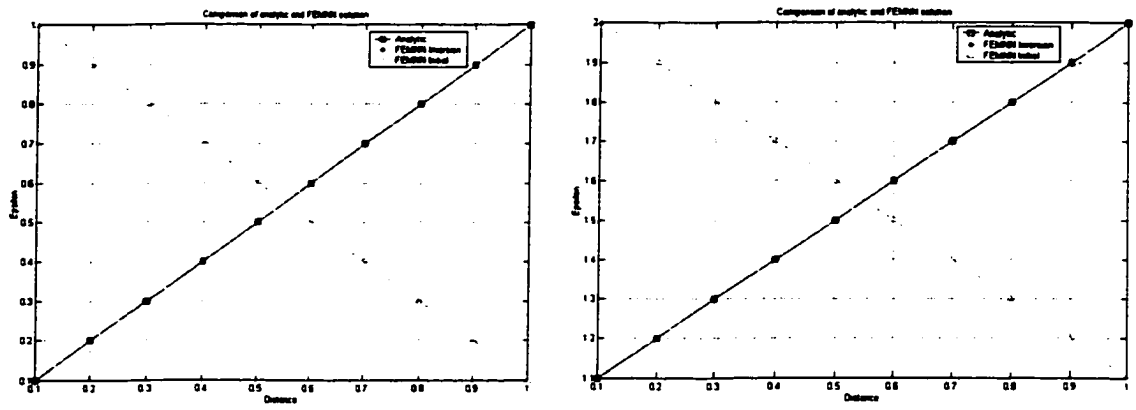
(a) (b)

Figure 36. Inversion result for Poisson's equation with initial solution (a) ε=0.5 (b) ε=1.



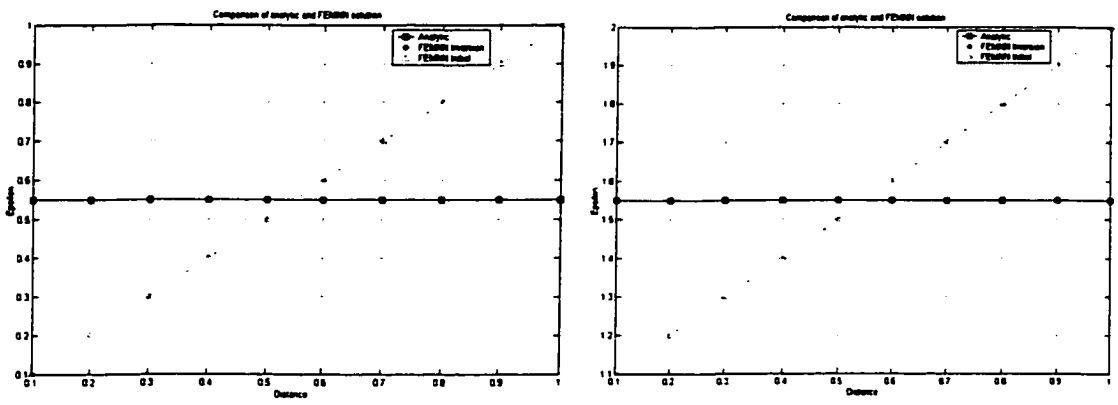
(a) (b)

Figure 37. Inversion result for Poisson's equation with (a) random initialization 1 (b) random initialization 2.



(a) (b)

Figure 38. FENN inversion results for Poisson's equation with initial solution (a) ε=1-x (b) ε=2-x.



(a) (b)

Figure 39. FENN inversion results for Laplace's equation with initialization (a) ε=x (b) ε=1+x.

The third example of the inverse problem that was used to test the algorithm was Laplace's equation with $\rho = 0$ and $\phi = x, \quad x \in [0,1]$. The analytical solution is $\varepsilon = K, \quad x \in [0,1]$ and $K \in \mathbb{R}$. Again, the results (Figure 39) show that the FENN solution matches the analytical solution and the exact solution to which the algorithm converges depends on the initialization.

The results presented for the inverse problem indicate that the solution is not unique and depends on the initialization. In order to obtain a unique solution, we need to impose constraints. For the second order differential equation (5.113), we need to constrain the value of ε at a known node on the sample in order to obtain a unique solution. This is usually possible to assign in practice. For instance, in electromagnetic NDE, we know the material properties at the boundary and this information can be used as the "ground truth" to constrain the solution. This constraint can be easily applied to the FENN by clamping the corresponding input nodes and not changing their values during the iterative inversion process.

This approach was applied to determine ε everywhere given that ϕ and f are specified as follows in (5.113):

$$\phi = x^2, \quad x \in [0,1] \tag{5.122}$$

and

$$f = -2K - 2\sin(x) - 2x\cos(x), \quad K \in \mathbb{R} \tag{5.123}$$

The analytical solution for this equation is

$$\varepsilon = \sin(x) + K \tag{5.124}$$

To solve this problem, we set $K = 1$ and clamp the value of ε at $x = 0$:

$$\varepsilon(x=0)=K \tag{5.125}$$

The results of the inversion are shown in Figure 40-Figure 43. Figure 40 shows the comparison between the analytical solution (solid line with squares) and the FENN result (solid line with stars). The initial value of ε was selected randomly and is shown in the figure as a dashed line. This result was obtained using 11 nodes and 10 elements in the corresponding finite element mesh. Figure 41 shows the error in the forcing function f at the FENN output. The squares indicate the desired value of f while the circles show the actual network output. This result indicates that, though the error in the forcing function is small, the error in the inversion result is fairly large. Similar results for 21 nodes (20 elements) in the mesh are shown in Figure 42 and Figure 43. It is seen that increasing the discretization significantly improves the solution. It should also be noted that the FENN inversion algorithm for 21 nodes has not converged to the desired error goal (as seen from Figure 43), and a larger number of iterations are necessary to further improve the solution.

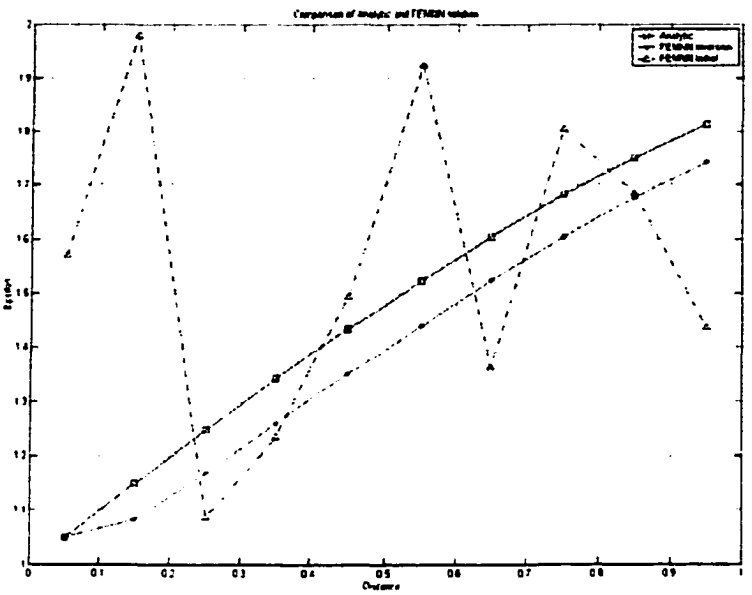


Figure 40. Constrained inversion result with eleven nodes.

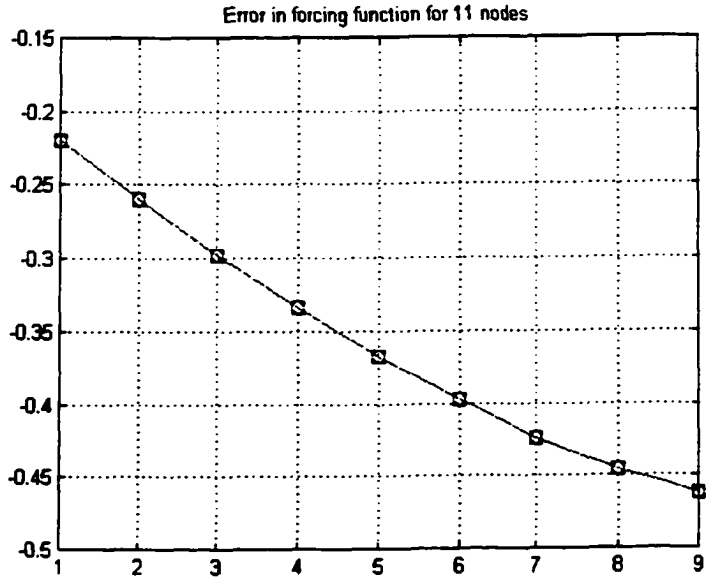


Figure 41. Error in the forcing function for an eleven node discretization.

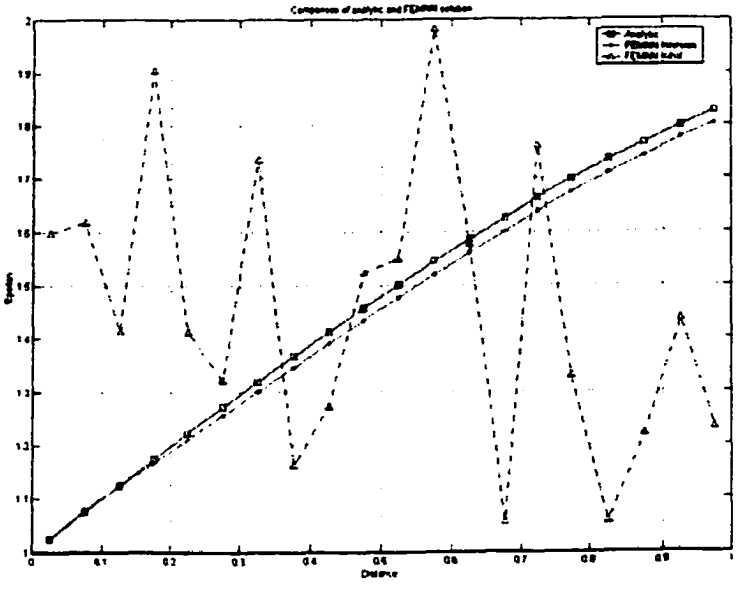


Figure 42. Constrained inversion results for a 21 node discretization.

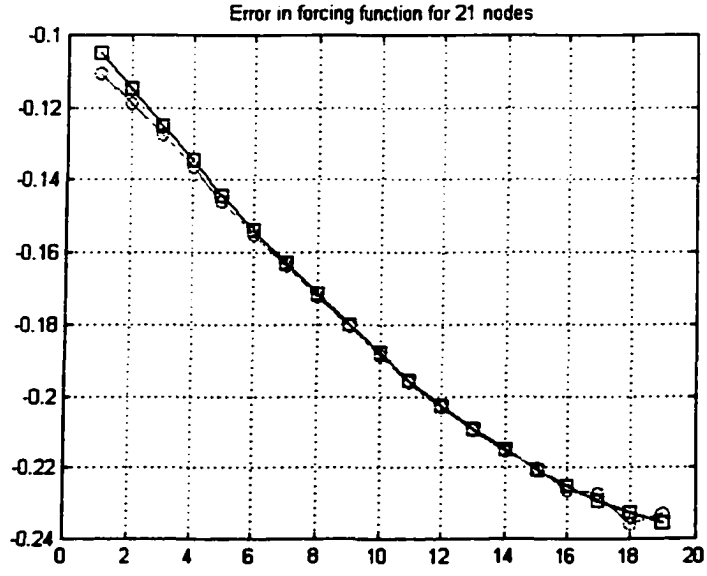


Figure 43. Error in the forcing function for a 21 node discretization.

5.6.3. Forward And Inverse Problems In Two Dimensions

The general form of Poisson's equation in two dimensions is

$$-\frac{\partial}{\partial x}\left(\alpha_x \frac{\partial \phi}{\partial x}\right)-\frac{\partial}{\partial y}\left(\alpha_y \frac{\partial \phi}{\partial y}\right)+\beta\phi = f \tag{5.126}$$

with boundary conditions

$$\phi = p \text{ on } \Gamma_1 \tag{5.127}$$

and

$$\left(\alpha_x \frac{\partial \phi}{\partial x}\hat{x}+\alpha_y \frac{\partial \phi}{\partial y}\hat{y}\right)\bullet \hat{n}+\gamma\phi = q \text{ on } \Gamma_2 \tag{5.128}$$

Several forward and inverse problem examples based on (5.126) were solved using the FENN algorithm. These are:

1. Problem I used $\alpha_x = \alpha_y = \alpha = x + y$, $(x, y) \in [0,1] \times [0,1]$, $\beta = \gamma = q = 0$ and $f = -2$. The analytical solution to the forward problem is $\phi = x + y$ when the Dirichlet boundary conditions are

$$\phi = y, \; x = 0$$

$$\phi = 1 + y, \; x = 1$$

$$\phi = x, \; y = 0$$

$$\phi = 1 + x, \; y = 1 \qquad (5.129)$$

Conversely, the inverse problem in this case is to estimate α in each element $(\beta = 0)$ given the potentials $\phi = x + y$ at each of the nodes.

Figure 44 shows the solution to the forward problem as a surface plot of ϕ, with Figure 44 (a) showing the analytical solution, Figure 44 (b) showing the FEM solution and the FENN solution in Figure 44 (c). The error between the FENN solution and the analytical solution is presented in Figure 44 (d). These results were obtained using a discretization of 11 nodes in each direction with triangular elements (a total of 121 nodes and 200 elements). The values of α and β were constant in each element and were set to their average values within each element. These results indicate that the FENN forward problem solution for a two-dimensional problem is comparable to the FEM solution. The error between the FENN solution and the analytical solution is also seen to be on the order of 10^{-7}.

The inverse problem solution is presented in Figure 45, with the analytical solution for α, the FENN inversion and the error between the analytical and FENN results in Figure 45 (a), (b) and (c) respectively. Several different discretizations were tested for solving the inverse problem, and the results presented here were obtained using 11 nodes in each

direction. Results obtained from a different discretization (5 nodes in each direction) are presented in Figure 46. The discretization was observed to affect the number of iterations needed for convergence, with the smaller mesh requiring a smaller number of iterations. Also, all the inverse problem solutions presented for this and other two-dimensional problems were obtained by constraining the material properties at the boundaries as mentioned in the section on one-dimensional inverse problems. The constraints were obtained from the definition of α_x, α_y and β for each of the problems.

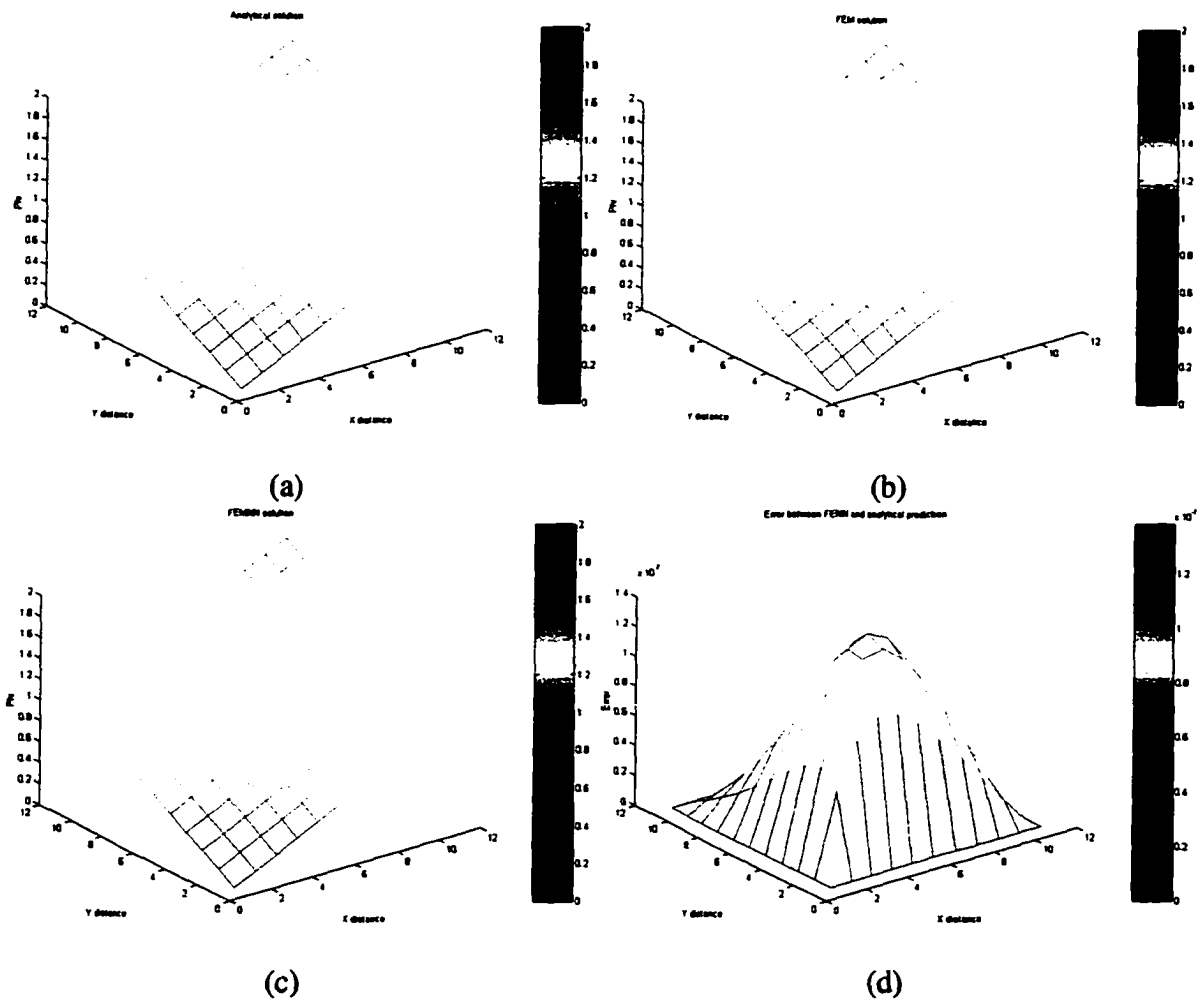
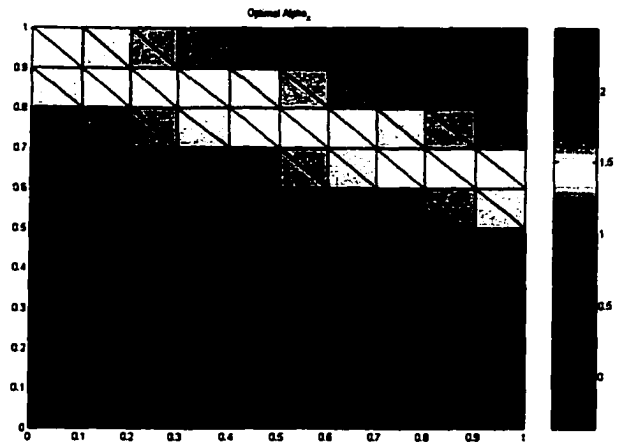
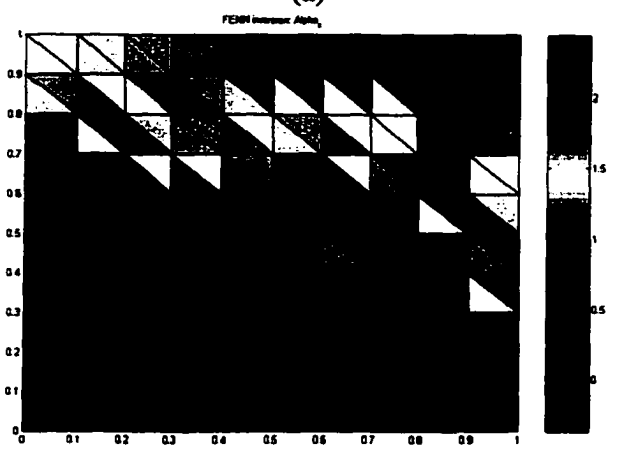


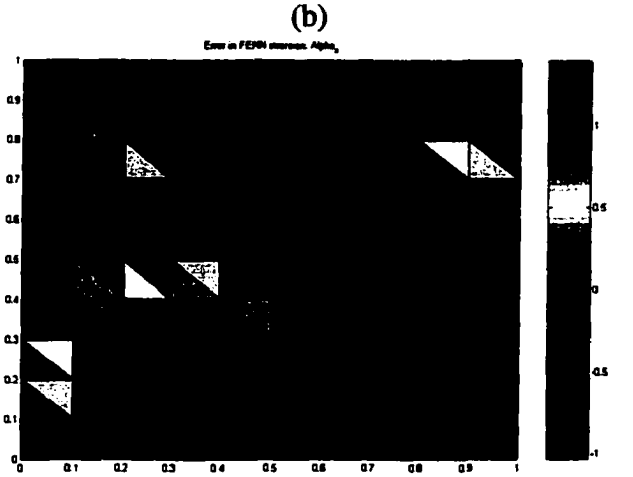
Figure 44. Solution of forward problem for Problem I (a) Analytical (b) FEM (c) FENN (d) error between (a) and (c).



(a)

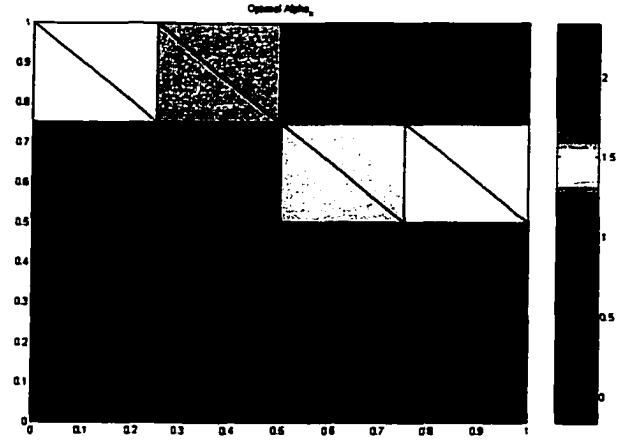


(b)

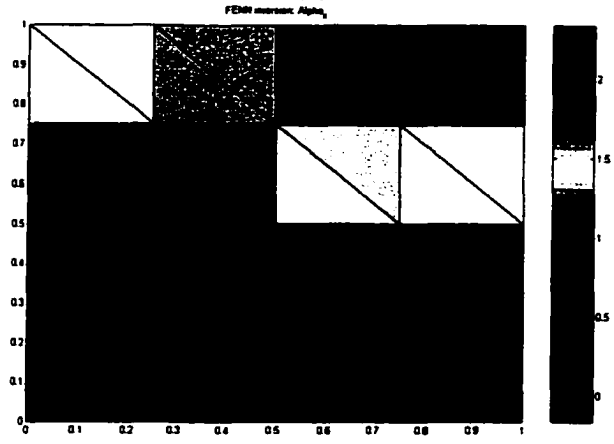


(c)

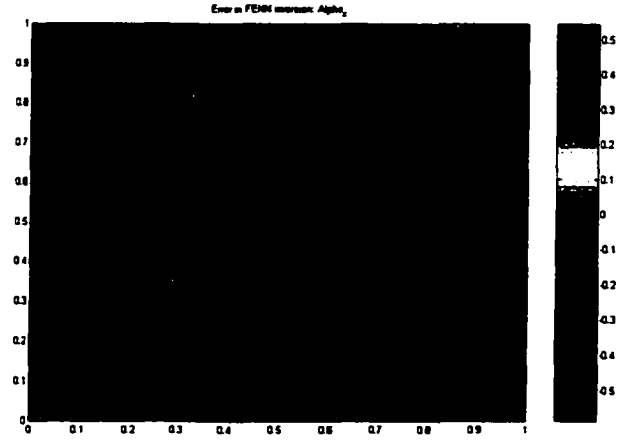
Figure 45. Inverse problem solution for Problem I with an 11x11 discretization (a) Analytical value of α (b) FENN inversion (c) Error between (a) and (b).



(a)



(b)



(c)

Figure 46. Inversion results for Problem I with a 5x5 mesh (a) Analytical value of α (b) FENN inversion (c) Error between (a) and (b).

2. Problem II used $\alpha_x = \alpha_y = \alpha = x + y$, $(x, y) \in [0,1] \times [0,1]$, $\beta = \gamma = q = 0$ and $f = -6(x + y)$. The analytical solution to the forward problem is $\phi = x^2 + y^2$ when the Dirichlet boundary conditions are

$$\phi = y^2, \; x = 0$$

$$\phi = 1 + y^2, \; x = 1$$

$$\phi = x^2, \; y = 0$$

$$\phi = 1 + x^2, \; y = 1 \qquad (5.130)$$

Conversely, the inverse problem in this case is to estimate α in each element given the potentials $\phi = x^2 + y^2$ at each of the nodes.

Figure 47 presents the solution to the forward problem as surface plots of ϕ, with Figure 47 (a) showing the analytical solution, Figure 47 (b) showing the FEM solution and Figure 47 (c) showing the FENN solution. The error between the analytical solution and the FENN solution is presented in Figure 47 (d). Again, the results for the forward problem were obtained using a discretization of 11 nodes in each direction with triangular elements. The results again indicate that the FENN and FEM solutions are similar, even though the error between the FENN and analytical solutions is high.

The inverse problem solution is presented in Figure 48, with Figure 48 (a), (b) and (c) showing analytical solution for α, the FENN inversion result and the error in the FENN inversion respectively. As in Problem I, several discretizations were used for solving the inverse problem, and the results presented in Figure 48 were obtained using 11 nodes in each direction.

3. Problem III used $\alpha_x = y$, $\alpha_y = x$, $(x,y) \in [0,1] \times [0,1]$, $\beta = \gamma = q = 0$ and $f = -2(x+y)$. The analytical solution to the forward problem is $\phi = x^2 + y^2$ when the Dirichlet boundary conditions are

$$\phi = y^2,\ x = 0$$

$$\phi = 1 + y^2,\ x = 1$$

$$\phi = x^2,\ y = 0$$

$$\phi = 1 + x^2,\ y = 1 \qquad (5.131)$$

Conversely, the inverse problem in this case is to estimate α_x and α_y in each element given the potentials $\phi = x^2 + y^2$ at each of the nodes.

Figure 49 presents the solution ϕ for Problem III, with the analytical solution in Figure 49 (a), the FEM solution in Figure 49 (b), the FENN solution in Figure 49 (c) and the error between the analytical and FENN solutions in Figure 49 (d). As in the previous examples, the results for the forward problem were obtained using a discretization of 11 nodes in each direction with triangular elements The inverse problem solution is presented in Figure 50, with Figure 50 (a), (b) and (c) showing the analytical solution, the FENN inversion and the error in the FENN inversion for α_x. Similar results for α_y are shown in Figure 51 (a), (b) and (c) respectively. As in the previous two examples, several discretizations were tried out for solving the inverse problem, and the results presented in Figure 50- Figure 51 were obtained using 11 nodes in each direction. Again, it was observed that the greater the discretization, the better the inverse problem solution.

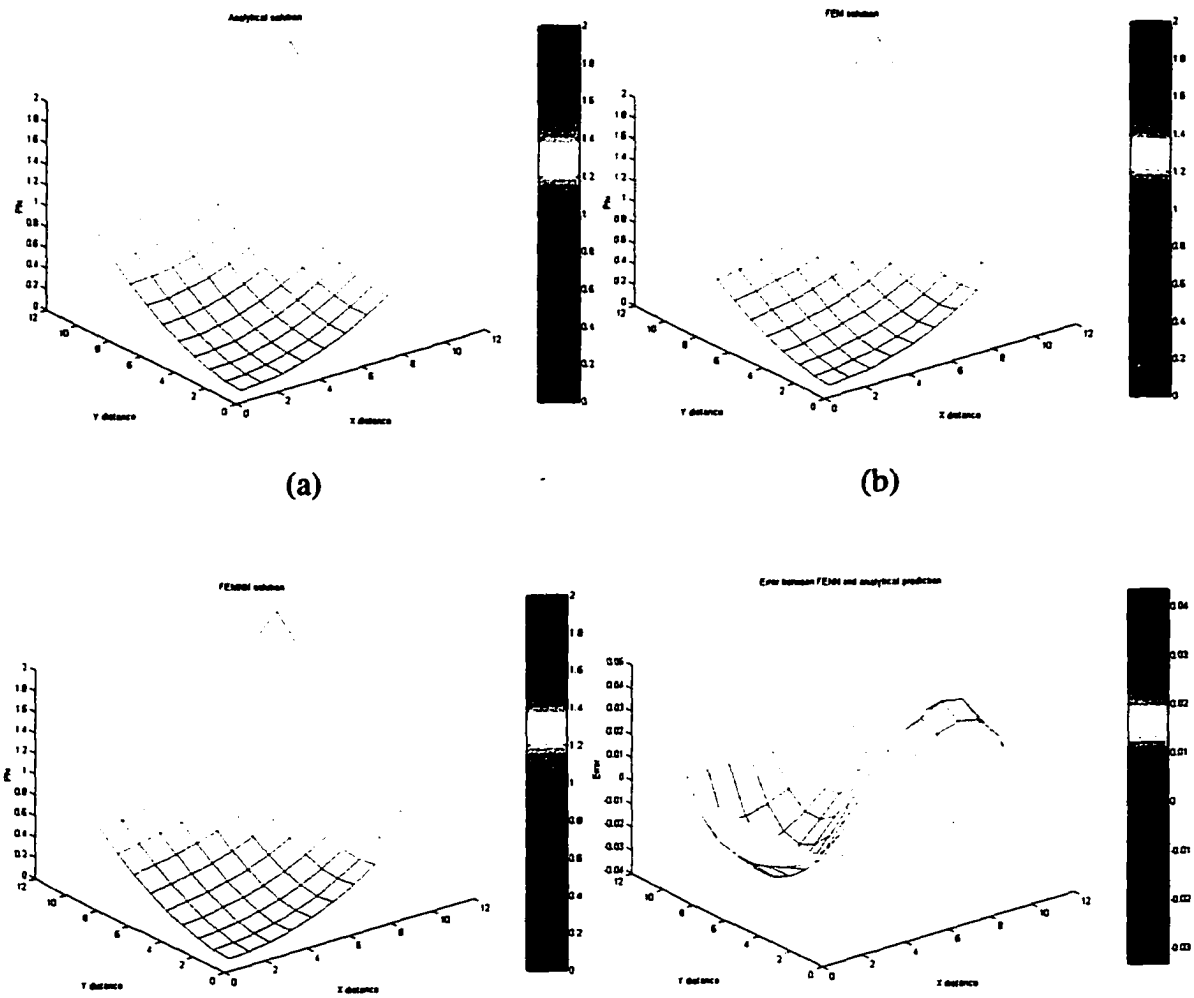
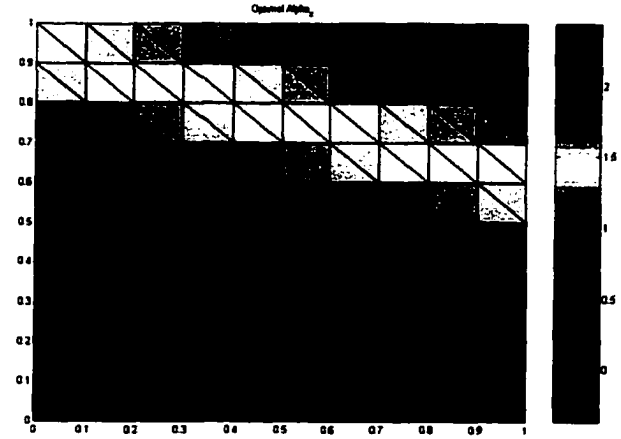
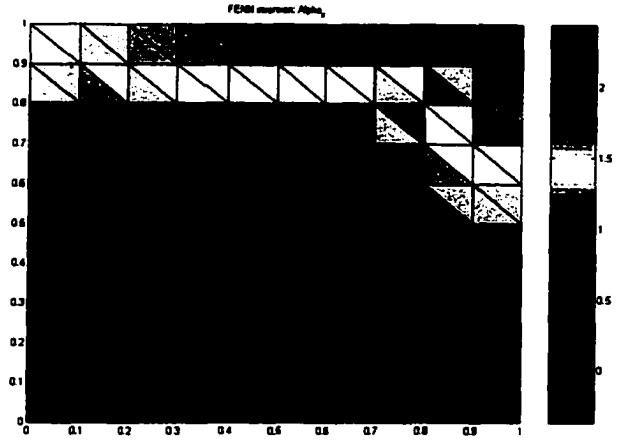


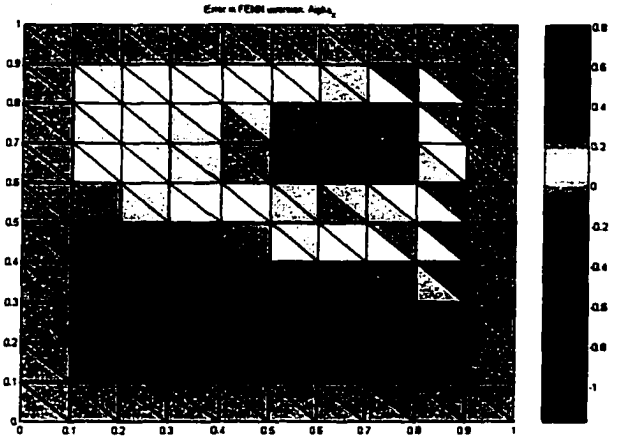
Figure 47. Forward problem solutions for Problem II (a) Analytical) (b) FEM (c) FENN.



(a)



(b)



(c)

Figure 48. Inversion results for Problem II with an 11x11 mesh (a) Analytical value of α (b) FENN inversion (c) Error between (a) and (b).

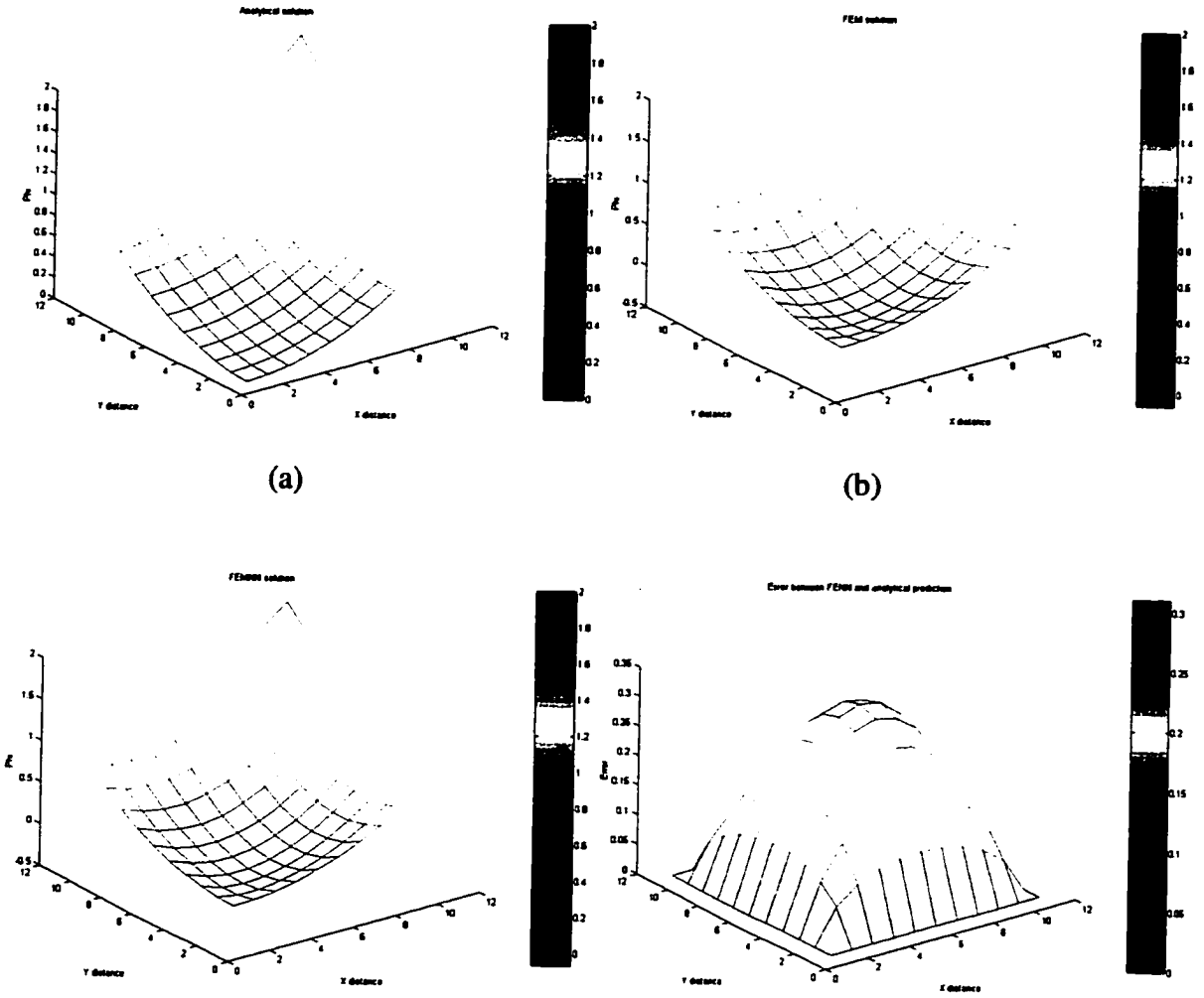
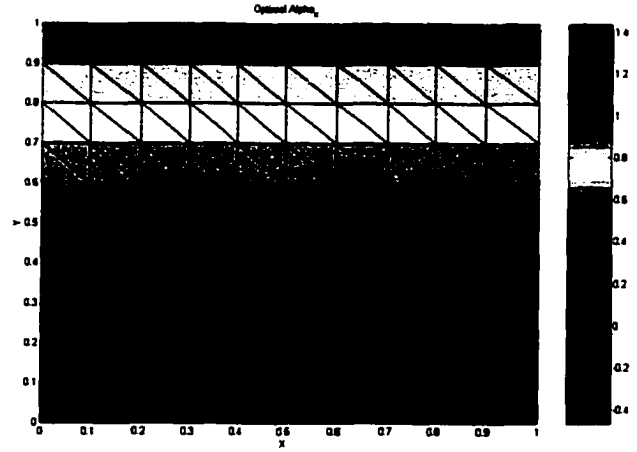
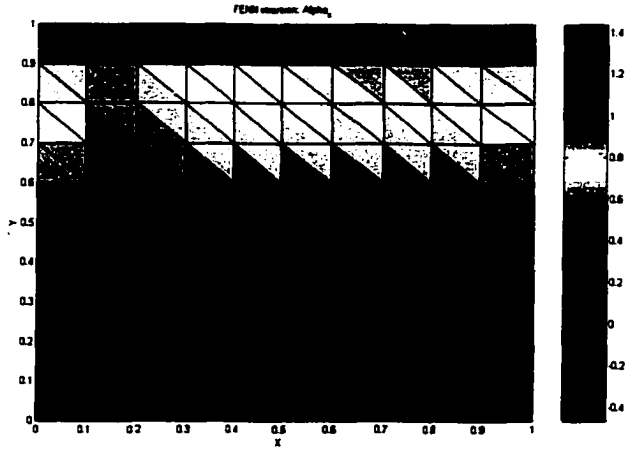


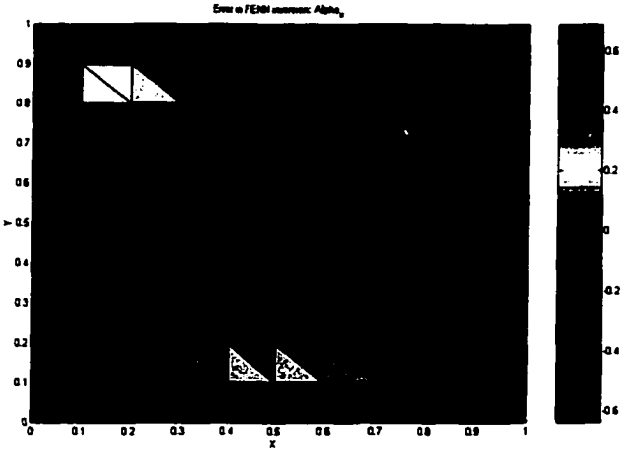
Figure 49. Solution for ϕ (Problem III) (a) Analytical (b) FEM (c) FENN (d) error between (a) and (c).



(a)

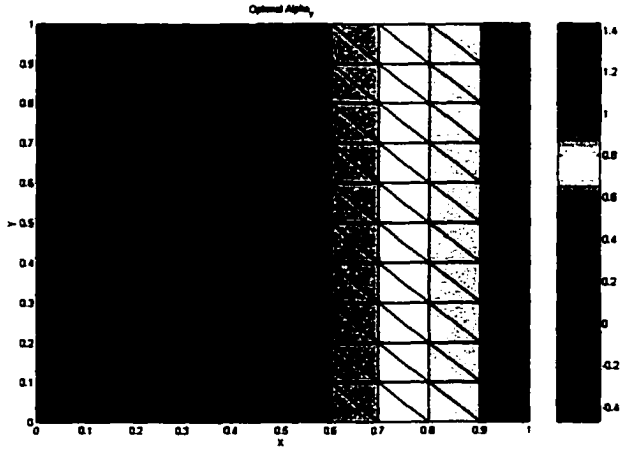


(b)

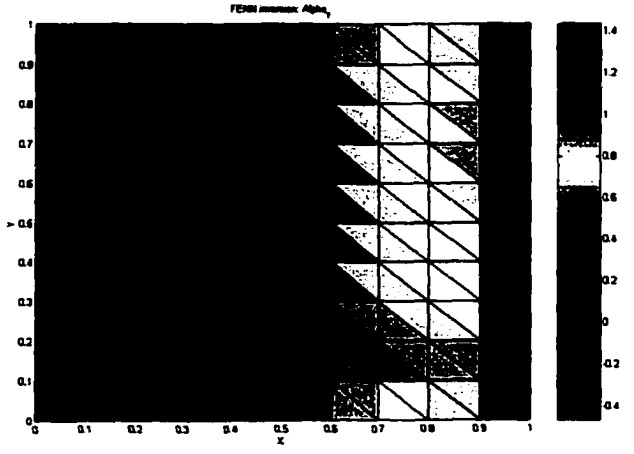


(c)

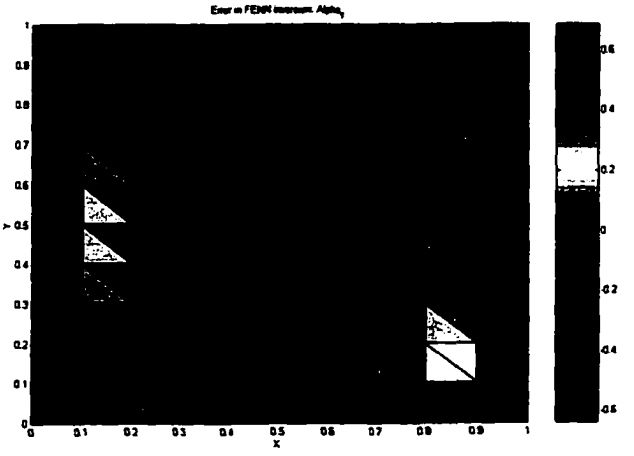
Figure 50. Inversion results for Problem III, α_x with an 11x11 mesh (a) Analytical value (b) FENN inversion (c) Error between (a) and (b).



(a)



(b)



(c)

Figure 51. Inversion results for Problem III, α_y with an 11x11 mesh (a) Analytical value (b) FENN inversion (c) Error between (a) and (b).

4. Problem IV - Shielded microstrip transmission line: The forward problem is to compute the electric potential due to the shielded microstrip shown in Figure 52 (a). The potentials are zero on the shielding conductor. Since the geometry is symmetric, we can solve the equivalent problem shown in Figure 52 (b), by applying the homogeneous Neumann condition on the plane of symmetry. Accordingly, $\beta = \gamma = q = 0$. The inner conductor (microstrip) is held at a constant potential of V volts. We also assume that the material inside the shielding conductor has a permittivity $\alpha_x = \alpha_y = \varepsilon = \mathrm{K}$, where K is a constant. The corresponding inverse problem is to determine the permittivity everywhere inside the shielding conductor given the potential value everywhere.

The solution to the forward problem is presented in Figure 53, with the FEM solution using 11 nodes in each direction shown in Figure 53 (a) and the corresponding FENN solution in Figure 53 (b). The potential of the microstrip in the forward and inverse problem was set to 10 volts and the permittivity of the medium was $\varepsilon = 1$. The error between the FEM and FENN solutions is presented in Figure 53 (c). Again, the FENN is seen to match the FEM solution accurately. Results of applying the FENN to solve the inverse problem are shown in Figure 54, with Figure 54 (a), (b) and (c) showing the true solution, the FENN prediction, and the error between the two, respectively.

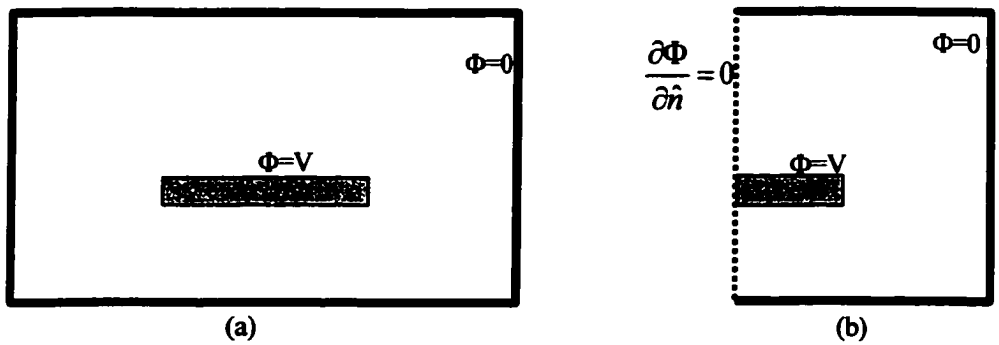


Figure 52. Shielded microstrip geometry (a) complete problem description (b) problem description using symmetry considerations.

All the results presented here indicate that the proposed FENN algorithm is capable of accurately solving both the forward and inverse problems. In addition, the forward problem solution from the FENN is seen to exactly match the FEM solution, indicating that the FENN represents the finite element model exactly in a parallel configuration. However, the result of the inversion process depends on the discretization. In general, increasing the number of elements improves the inversion results at the cost of increasing the number of iterations necessary for convergence. In addition, the convergence time was seen to also depend on the initialization. Most of the results presented in this thesis were obtained using a random initialization for the parameter of interest.

The simulations also showed that the forward problem solutions depend on the discretization, with better results obtained using a higher discretization. Furthermore, the results presented here indicate that the algorithm for inversion may oscillate as it approaches the solution. This issue can be addressed by either using a more sophisticated optimization algorithm (for instance, conjugate gradient or genetic algorithms), or by adding additional regularization constraints to the problem, or a combination of both.

The FENN offers several advantages over conventional neural network based inversion. A major advantage of the FENN forward model is that it does not require any training. Further, the FENN can correctly predict the measurement signal for any defect profile, thus solving the problem of extrapolation. In addition, the FENN structure enables most computations, including gradient computation, to take place in parallel, speeding up the solution process significantly. Sensitivity analysis for the inverse problem also indicates that the network will converge to a solution close to the true solution whenever the SNR is high.

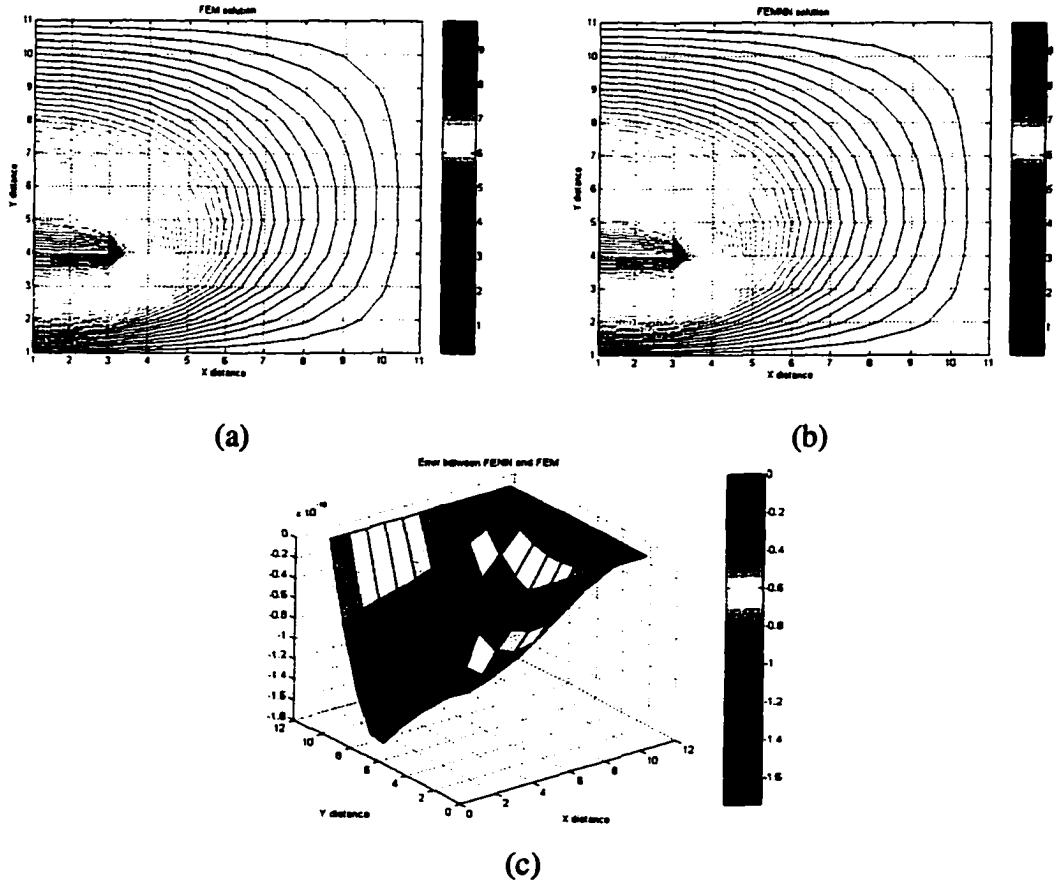
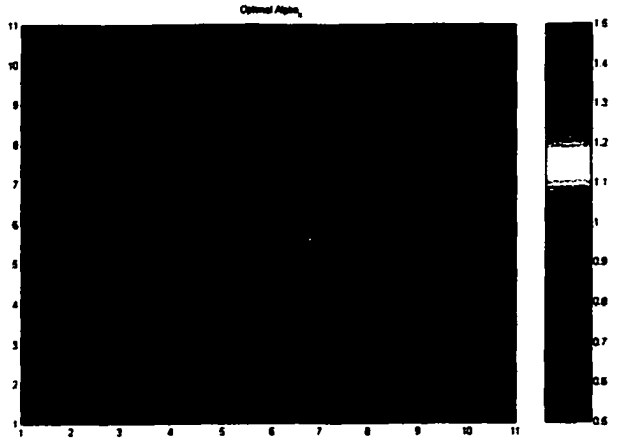
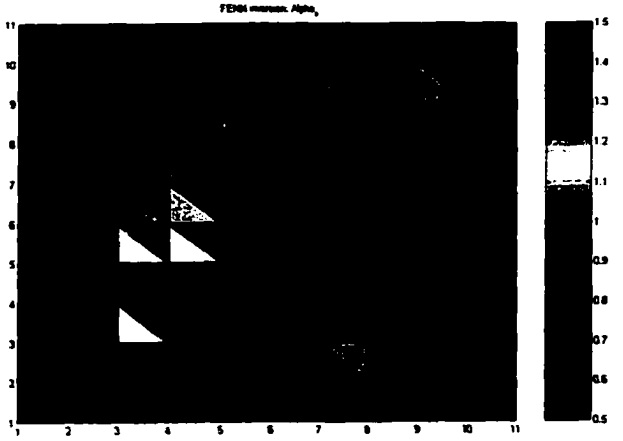


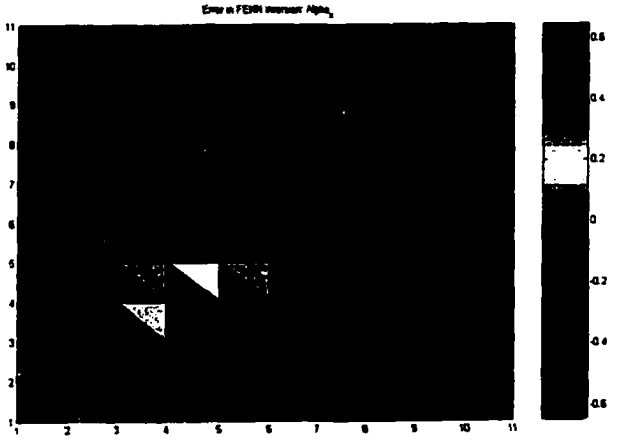
Figure 53. Forward problem solutions for shielded microstrip problem (a) FEM (b) FENN (c) error between (a) and (b).



(a)



(b)



(c)

Figure 54. Inversion result for a shielded microstrip (a) True solution for α (b) FENN inversion (c) error between (a) and (b).

6. CONCLUSIONS AND FUTURE WORK

This study proposed the use of neural network based forward models in iterative algorithms for inversion of NDE signals. The use of neural network based forward models offers several advantages over numerical models in terms of both implementation of gradient calculations in the updates of the defect profiles and overall computational cost. Two different types of neural networks – radial basis function neural networks and wavelet basis function neural networks – were initially used to represent the forward model. These forward models were used, in a simple iterative scheme, or in combination with an inverse model in feedback configuration, to solve the inverse problem. The results presented on inversion of MFL data indicate that these algorithms are capable of accurately solving the inverse problem with a relatively low computational effort, even in the presence of noise. The results obtained with the feedback neural network configuration also indicate that this approach can provide a measure of confidence in its prediction. This feature is especially useful when the inversion process must be performed in the presence of noise.

One drawback of these approaches is that the forward models are not accurate when the input signals are not similar to those used in the training database. This thesis proposed the design of neural networks that are capable of solving differential equations and hence does not depend on training data. This specialized neural network – the finite element model neural network (FENN) – has a weight structure that allows both the forward and inverse problems to be solved using simple gradient-based algorithms. Initial results of applying the FENN to one- and two-dimensional problems were presented and show that the proposed FENN accurately models the forward problem. Application of this neural network for inverse problem solutions indicates that the solution closely matches the analytical solution. The

structure of the FENN also allows easy application of constraints on the inverse problem solution, in the form of clamped input nodes. In addition, the FENN is easily amenable to parallel implementation, in both hardware and software. An analysis of the sensitivity of the FENN to measurement noise indicates that the corresponding inversion result is bounded if the measurement error is bounded. Under typical conditions, where the SNR is high, the error in the inverse problem solution is fairly small, and goes to zero as the noise goes to zero.

Future work will concentrate on extending the FENN to three-dimensional electromagnetic NDE problems. One advantage of the FENN is that it can be used as the forward model in both the neural network inversion approach (Approach I) and the feedback neural network approach (Approach II), with very little change in the algorithm. Results of applying the FENN in Approach I have been shown in Section 5. Its use in Approach II will be investigated. Furthermore, all the approaches use simple gradient-based methods in the optimization process. The use of better optimization algorithms, such as conjugate gradient methods, can improve the solution speed even further.

An alternative approach to deriving specialized neural networks involves the solution of the integral equations that describe the physical process (as opposed to the differential equations that were used to derive the FENN). The feasibility of this network, called the Green's function neural network (GFNN), and its application to electromagnetic NDE will also be investigated.

The approaches described in this proposal are very general in that they can be applied to a variety of inverse problems in fields other than electromagnetic NDE. Some of these other applications will also be investigated to show the general nature of the proposed methods.

APPENDIX. MAGNETIC FLUX LEAKAGE METHODS

The magnetic flux leakage method, frequently used in the inspection of ferromagnetic materials, employs permanent magnets or currents to magnetize the sample and a set of flux-sensitive sensors to record the leakage flux for analysis.

Leakage flux arises because the presence of a defect causes an increase in magnetic flux density in the vicinity of the flaw. This causes a shift in the operating point on the hysteresis curve and a corresponding decrease of local permeability, resulting in a leakage of flux into the surrounding medium (air) [35]. The leakage flux is recorded using either hall probes or a coil.

The governing equations for magnetic flux leakage can be derived from the static form of Maxwell's equations. Considering only source-free regions of the material, we have [36]

$$\nabla \times \mathbf{H} = 0 \tag{A-1}$$

$$\nabla \bullet \mathbf{B} = 0 \tag{A-2}$$

where **H** is the magnetic field strength and **B** is the magnetic flux density. Therefore, the magnetic field strength **H** can be represented by the gradient of a magnetic scalar potential U:

$$\mathbf{H} = -\nabla U \tag{A-3}$$

Substituting back in (A-2), and assuming the region is homogeneous and isotropic, we get Laplace's equation

$$\nabla^2 U = 0 \tag{A-4}$$

Although analytical models involving the corresponding Green's function provide exact solutions to the problem, solution of the forward problem in the case of realistic inspection geometries with complex defect shapes necessitates the use of numerical techniques such as a

finite difference or finite element model (FEM) [6, 7, 8] where the models predict all three components of the magnetic leakage flux density **B** as a function of spatial location.

REFERENCES

[1] H. A. Sabbagh, L. D. Sabbagh and T. M. Roberts, "An eddy-current model and algorithm for three-dimensional nondestructive evaluation of advanced composites," *IEEE Transactions on Magnetics*, Vol. 24, No. 6, pp. 3201-3212, 1988.

[2] L. Udpa and S. S. Udpa, "Application of signal processing and pattern recognition techniques to inverse problems in NDE," *Int. J. Applied Electromagnetics and Mechanics*, Vol. 8, pp. 99-117, 1997.

[3] M. Yan, M. Afzal, S. Udpa, S. Mandayam, Y. Sun, L. Udpa and P. Sacks, "Iterative Algorithms for Electromagnetic NDE Signal Inversion," *ENDE '97*, September 14-16, 1997, Reggio Calabria, Italy.

[4] S. Hoole, S. Subramaniam, R. Saldanha, J. Coulomb, and J. Sabonnadiere, "Inverse Problem Methodology and Finite Elements in The Identification of Cracks, Sources, Materials and Their Geometry in Inaccessible Locations," *IEEE Transactions on Magnetics*, Vol.27, No.3, pp. 3433-3443, 1991.

[5] T. M. Roberts, H. A. Sabbagh and L. D. Sabbagh, "Electromagnetic interactions with an anisotropic slab," *IEEE Transactions on Magnetics*, Vol. 24, No. 6, pp. 3193-3200, 1988.

[6] P. P. Sylvester and R. L. Ferrari, *Finite elements for electrical engineers*, Cambridge University Press, Cambridge, England, 1990.

[7] P. Zhou, *Numerical analysis of electromagnetic fields*, Springer-Verlag, Berlin, 1993.

[8] J. Jin, *The finite element method in electromagnetics*, John Wiley & Sons, Inc., New York, NY, 1993.

[9] K. Hwang, W. Lord, S. Mandayam, L. Udpa, S. Udpa, "A Multiresolution Approach for Characterizing MFL Signatures from Gas Pipeline Inspections," *Review of Progress in Quantitative Nondestructive Evaluation*, Plenum Press, New York, Vol.16, pp.733-739, 1997.

[10] C. A. Jensen, et al, "Inversion of feedforward neural networks: algorithms and applications," *Proc. IEEE*, Vol. 87, No. 9, pp. 1536-1549, 1999.

[11] D. C. Youla, "Generalized image restoration by the method of alternating orthogonal projections," *IEEE Trans. Circuits and Systems*, Vol. CAS-25, No. 9, pp. 694-702, 1978.

[12] D. C. Youla and H. Webb, "Image restoration by the method of convex projections: Part I – Theory," *IEEE Trans. Medical Imaging*, Vol. MI-1, No. 2, pp. 81-94, 1982.

[13] A. Lent and H. Tuy, "An iterative method for the extrapolation of band-limited functions," *J. Math. Analysis and Applications*, Vol. 83, pp. 554-565, 1981.

[14] W. Chen, "A new extrapolation algorithm for band-limited signals using the regularization method," *IEEE Trans. Signal Proc.*, Vol. 41, No. 3, pp. 1048-1060, 1993.

[15] C. H. Barbarosa, A. C. Bruno, M. Vellasco, M. Pacheco, J. P. Wikswo Jr. and A. P. Ewing, "Automation of SQUID nondestructive evaluation of steel plates by neural networks," *IEEE Trans. Applied Superconductivity*, Vol. 9, No. 2, pp. 3475-3478, 1999.

[16] W. Qing, S. Xueqin, Y. Qingxin and Y. Weili, "Using wavelet neural networks for the optimal design of electromagnetic devices," *IEEE Trans. Magnetics*, Vol. 33, No. 2, pp. 1928-1930, 1997.

[17] J. Takeuchi and Y. Kosugi, "Neural network representation of the finite element method," *Neural networks*, Vol. 7, No. 2, pp. 389-395, 1994.

[18] R. Sikora, J. Sikora, E. Cardelli and T. Chady, "Artificial neural network application for material evaluation by electromagnetic methods," *Proc. Int'l. Joint Conf. Neural Networks*, Vol. 6, pp. 4027-4032, 1999.

[19] G. Xu, G. Littlefair, R. Penson and R. Callan, "Application of FE-based neural networks to dynamic problems," *Proc. Int'l. Conf. Neural Information Processing*, Vol. 3, pp. 1039-1044, 1999.

[20] F. Guo, P. Zhang, F. Wang, X. Ma and G. Qiu, "Finite element analysis-based Hopfield neural network model for solving non-linear electromagnetic field problems," *Proc. Int'l. Joint Conf. Neural Networks*, Vol. 6, pp. 4399-4403, 1999.

[21] I. E. Lagaris, A. Likas and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Trans. Neural Networks*, Vol. 9, No. 5, pp. 987-1000, 1998.

[22] R. P. Lippman, "An introduction to computing with neural networks," *IEEE ASSP Magazine*, Vol. 4, pp. 4-22, 1987.

[23] S. Haykin, *Neural networks: A comprehensive foundation*, Prentice-Hall, Upper Saddle River, NJ, 1994.

[24] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, Vol. 78, No. 9, pp. 1481-1497, 1990.

[25] A. N. Tikhonov and V. Y. Arsenin, *Solutions of ill-posed problems*, W. H. Winston, Washington D. C., 1977.

[26] W. A. Light, "Some aspects of radial basis function approximation," In *Approximation Theory, Spline Functions and Applications* (S. P. Singh, ed.), NATO ASI Series, Vol. 256, pp. 163-190, 1992.

[27] J. A. Hartigan, *Clustering algorithms*, John Wiley & Sons, New York, NY, 1975.

[28] J. T. Tou and R. C. Gonzales, *Pattern Recognition Principles*, Addison-Wesley, MA, 1974.

[29] R. Polikar, "The Engineer's ultimate guide to wavelet analysis: The Wavelet Tutorial," at http://www.public.iastate.edu/~rpolikar/WAVELETS/WTtutorial.html

[30] B. R. Bakshi and G. Stephanopoulos, "Wave-Net: A multiresolution, hierarchical neural network with localized learning," *AIChE J.*, Vol. 39, No. 1, pp. 57-81, 1993.

[31] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Analysis Mach. Intelligence*, Vol. 11, No. 7, pp. 674-693, 1989.

[32] T. Kugarajah and Q. Zhang, "Multidimensional wavelet frames," *IEEE Transactions on Neural Networks*, Vol. 6, No. 6, pp. 1552-1556, 1995.

[33] K. Hwang, *3-D defect profile reconstruction from magnetic flux leakage signatures using wavelet basis function neural networks*, Ph.D. Dissertation, Iowa State University, Ames, IA, 2000.

[34] S. Russell and P. Norvig, *Artificial Intelligence: A modern approach*, Prentice-Hall, Upper Saddle River, NJ 1995.

[35]. W. Lord and D. J. Oswald, "Leakage field methods of defect detection," *Int. J. NDT*, Vol. 4, pp. 249-274, 1972.

[36]. L. Udpa, *Imaging of electromagnetic NDT phenomena*, Ph.D. Dissertation, Colorado State University, Fort Collins, CO, 1985.