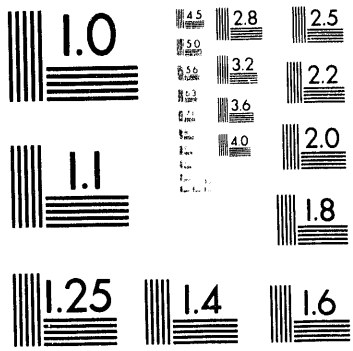

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

1 of 2

NUREG/GR-0010
R5

Hybrid Digital Signal Processing and Neural Networks for Automated Diagnostics Using NDE Methods

Manuscript Completed: October 1993
Date Published: November 1993

Prepared by
B. R. Upadhyaya, W. Yan

Department of Nuclear Engineering
The University of Tennessee
Knoxville, TN 37996-2300

Prepared for
Division of Engineering
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001
FIN G2058

ABSTRACT

The primary purpose of the current research was to develop an integrated approach by combining information compression methods and artificial neural networks for the monitoring of plant components using nondestructive examination data. Specifically, data from eddy current inspection of heat exchanger tubing were utilized to evaluate this technology. The focus of the research was to develop and test various data compression methods (for eddy current data) and the performance of different neural network paradigms for defect classification and defect parameter estimation. Feedforward, fully-connected neural networks, that use the back-propagation algorithm for network training, were implemented for defect classification and defect parameter estimation using a modular network architecture. A large eddy current tube inspection database was acquired from the Metals and Ceramics Division of ORNL. These data were used to study the performance of artificial neural networks for defect type classification and for estimating defect parameters. A PC-based data preprocessing and display program was also developed as part of an expert system for data management and decision making. The results of the analysis showed that for effective (low-error) defect classification and estimation of parameters, it is necessary to identify proper feature vectors using different data representation methods. The integration of data compression and artificial neural networks for information processing was established as an effective technique for automation of diagnostics using nondestructive examination methods.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

This research and development work was made possible by a University Grant sponsored by the U.S. Nuclear Regulatory Commission, Office of Nuclear Regulatory Research. The authors acknowledge the assistance of Dr. Joseph Muscara of the NRC Office of Nuclear Regulatory Research. The assistance of Dr. C. V. Dodd, Metals and Ceramics Division of ORNL, for technical discussion and providing eddy current tube inspection data is gratefully acknowledged.

SUMMARY

The primary purpose of the current research was to develop an integrated approach by combining information compression methods and artificial neural networks for the monitoring of plant components using nondestructive examination (NDE) data. Specifically, data from eddy current inspection of heat exchanger tubing were utilized to evaluate this technology. The focus of the research was to develop and test various data compression methods (for eddy current data) and the performance of different neural network paradigms for defect classification and defect parameter estimation. Feedforward, fully-connected neural networks, that use the back-propagation algorithm for network training, were implemented for defect classification and defect parameter estimation using a modular network architecture. The defect classification was also performed using probabilistic neural networks. A large eddy current tube inspection database was acquired from the Metals and Ceramics Division of ORNL. These data were used to study the performance of artificial neural networks for defect type classification and for estimating defect parameters. Most of the study has been made using the NeuralWorks Professional II/Plus software. A PC-based data preprocessing and display program was also developed as part of an expert system for data management and decision making.

The results of the analysis showed that for effective (low-error) defect classification and estimation of parameters, it is necessary to identify proper feature vectors using different data representation methods. The integration of data compression and artificial neural networks for information processing was established as an effective technique for automation of diagnostics using nondestructive examination methods.

SECTION 1

INTRODUCTION

1.1 Background and Scope of The Research

A nuclear power plant is a complex system with the various components fulfilling the needs of process control and plant safety. Continued operation of these systems has both safety and economic implications. Near-term operation through proper maintenance and long-term plant life with consideration to component aging are important aspects of nuclear plant design and operation. Nondestructive examination (NDE) methods are being increasingly applied to the monitoring of critical components. Eddy current inspection, ultrasonic inspection, and thermographic inspection are the most commonly used methods.

The research performed at the University of Tennessee (during 1991-93) focused on the problem of automating NDE data analysis using an integration of artificial neural networks and digital signal processing techniques. In recent years research in neural networks has been advanced to the point where several real-world applications have been successfully demonstrated [3,6,8,23]. These include automated pattern classification, signal validation, nuclear plant monitoring, plant state identification during transients, estimation of performance related parameters, underwater acoustic signature classification and text recognition. The University of Tennessee Nuclear Engineering Department has been very active in the research and development of neural network architectures and applications to nuclear power plant monitoring, diagnostics, and control. The integration of neural networks and digital signal processing techniques for the automation of NDE signature analysis is a unique feature of this research.

This research will also provide a technology base for the safety assessment of system and subsystem technologies used in nuclear power applications of neural networks.

Nondestructive testing methods include (a) eddy current inspection, (b) ultrasonic inspection, (c) radiographic inspection, (d) thermographic inspection, (e) acoustic emission inspection, (f) penetrant testing, and (g) visual inspection. Among these methods, eddy current inspection has been applied for crack and flaw detection; corrosion, thickness, and coating monitoring; and for monitoring changes in metallurgical properties. Three of the important areas of applications are pressure vessel, steam generator tubes, and turbine-generators. Eddy current technique is used extensively for steam generator tubing inspection [1,4,5]. The primary objective of this research is the development of an automated method for eddy current signature analysis.

1.2 Summary of Approach - General Features

The general approach for automated anomaly classification and defect parameter estimation is shown in Figure 1.1. This integrates data representation and compression, and the development of artificial neural networks for anomaly detection and estimation. The key issues for the development of the automated system for NDE diagnostics include the following: (1) Digital data representation and information compression. (2) Development of robust neural networks with low probability of misclassification. (3) Correlation of neural networks results to failure or fault modes. (4) Correlation of data trending to aging-related problems, and hence forecast incipient behavior in plant components. (5) Provide guidelines for assessing neural networks technology as related to nuclear safety issues. A two-step approach consisting of

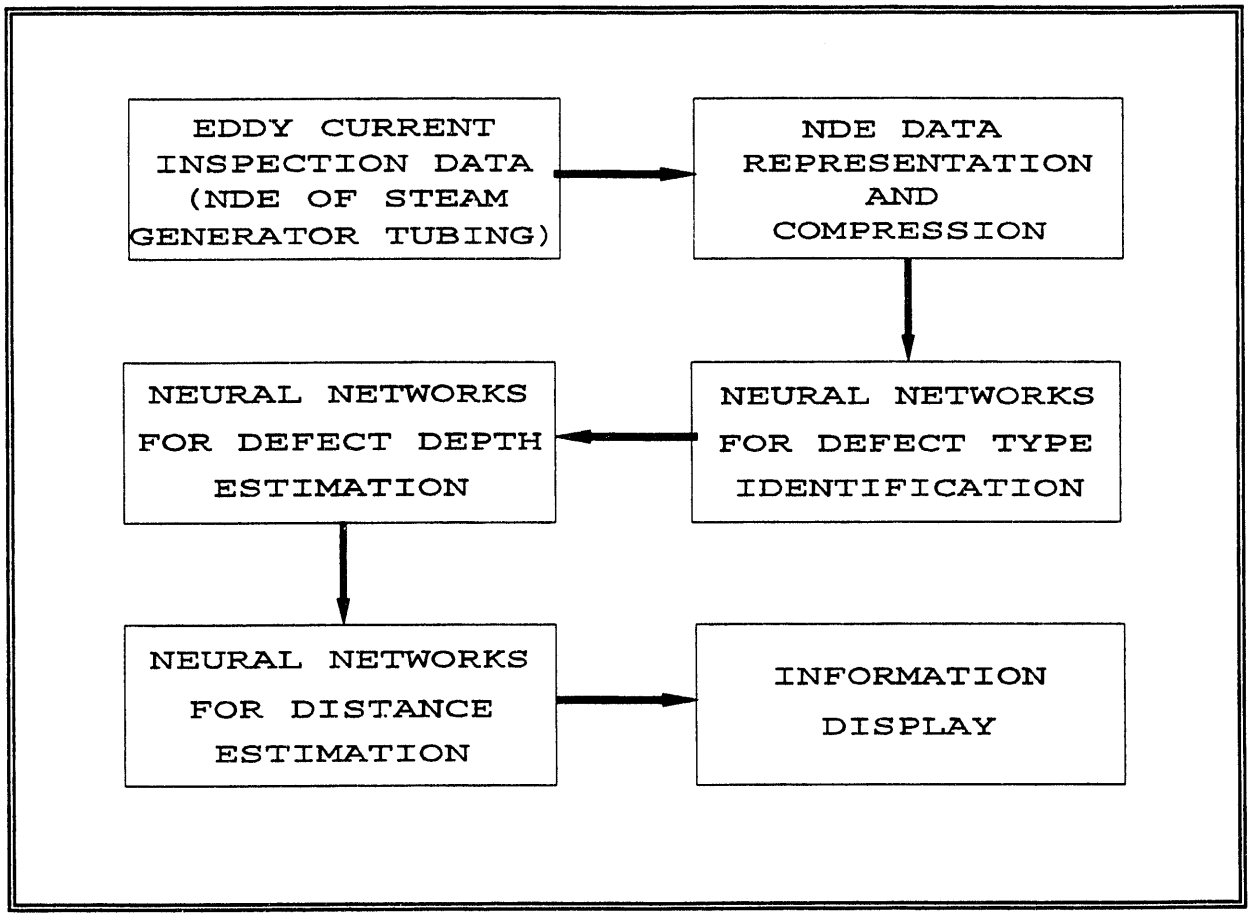


Figure 1.1 Schematic showing eddy current NDE data analysis using artificial neural networks

defect classification and defect parameter estimation has been developed. Because of the large number of defect types, a modular approach was used in establishing neural network models.

During the course of this project, the University of Tennessee worked in cooperation with the Metals and Ceramics Division of ORNL and the Electric Power Research Institute NDE Center in acquiring data from the eddy current inspection of steam generator tubing and for continued technical discussion.

1.3 Definition of Tasks

The following research & development tasks were established in order to accomplish the objectives of this research project. The period of research grant was 9/30/91 - 9/29/93.

Task 1. Review of the NDE data analysis with emphasis on eddy current inspection.

Task 2. Develop a methodology for optimum representation of eddy current data from steam generator tube inspection and develop appropriate signatures using digital data compression methods.

Task 3. Develop and implement artificial neural networks for the estimation of defect characteristics and for defect classification. Quantify the error of misclassification.

Task 4. Evaluate the performance of the developed technology using eddy current inspection data from ORNL.

Task 5. Develop guidelines for automating this technology and provide technology base for NRC safety assessment of technologies used in the application of artificial neural networks to nuclear power plant systems.

1.4 Summary of Significant Results

Several important milestones were accomplished during the course of this research project (9/30/91 - 9/29/93). These are discussed in the sections to follow. A summary of accomplishments is given below.

1. Development of parametric and nonparimetric data representation methods with emphasis on eddy current inspection signatures and data calibration.

2. Development of an optimal neural networks architecture with modular approach for managing various defect types.

3. Integration of defect type classification and estimation of defect parameters.

4. Application of the methodology to the analysis of eddy current tube inspection data. These data were acquired from the Metals and Ceramics Division, ORNL.

5. Development of a data preprocessing, management and decision-making expert system called EDDYANN for implementation in a PC.

6. Presentation of papers at the following conferences

    - EPRI Computer Assisted Technologies for NDE and Plant Monitoring Workshop, Philadelphia, August 1992.

    - International Conference on Artificial Neural Networks, Amsterdam, The Netherlands, September 1993.

SECTION 2

REVIEW OF EDDY CURRENT INSPECTION TECHNIQUE

2.1 Principle of Eddy Current Testing

More than 150 years ago, Michael Faraday and Joseph Henry independently discovered that a changing magnetic field produces an electric current in a conducting material. Ten years earlier, Hans Christian Oversted had observed that an electric current produced its own magnetic field [28]. Interactions between magnetic and electric fields occur in all electromagnetic devices, and they have been applied to many fields.

The interaction between magnetic fields and electrical phenomena is described by three of Maxwell's equations:

$$\nabla \times E = -\partial B / \partial t \tag{2.1}$$

$$\nabla \times H = J \tag{2.2}$$

$$\nabla \cdot B = 0 , \tag{2.3}$$

where **E** is the electric field produced by a change in magnetic flux density **B**, and **H** is the magnetic intensity of the field produced by a current density **J**. The first equation is a form of Faraday's law; the second, Ampere's law; and the third, Gauss's law for magnetism, which reflects the fact that there are no point sources of magnetic flux [28].

Eddy current method of nondestructive examination (NDE) is based on the principle of electromagnetic induction between the inspection coil and the object under study.

When an alternating current is used to excite a coil, an alternating magnetic field is produced and magnetic lines of flux are concentrated at the center of the coil. As this coil is brought near an electrically conductive material, the alternating magnetic field penetrates the material and generates continuous, circular eddy currents as shown in Figure 2.1 [11]. Larger eddy currents are produced near the test surface; as the penetration of the induced field increases, the eddy currents become weaker. The induced eddy currents produce an opposing (secondary) magnetic field. This opposing magnetic field, coming from the material, has a weakening effect on the primary magnetic field and this change can be sensed by the test coil. In effect, the impedance of the coil is reduced proportionally as eddy currents are increased in the test piece.

A crack in the test material obstructs the eddy current flow, lengthens the eddy current path, reduces the secondary magnetic field, and increases the coil impedance. The inductive reactance of the coil continues to increase as the severity of the defect increases. If a test coil is moved over a crack or defect in a metal plate, at a constant clearance and constant rate, a momentary change will occur in coil reactance and coil current. This change can be amplified, detected, and displayed by electronic instruments.

Eddy current testing can be used to identify changes in the physical, structural, and metallurgical conditions in both ferromagnetic and nonferromagnetic conducting materials. This nondestructive method has a wide variety of applications including detection of cracks, voids and inclusions; measurement of the thickness of coating; sorting dissimilar metals, and others.

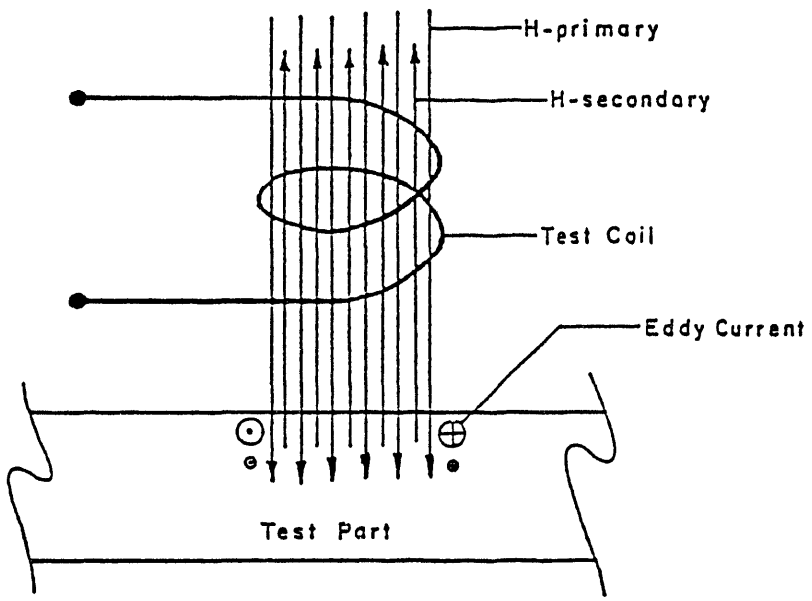


Figure 2.1 Principle of eddy current testing. The primary field of test coil enters test part, generates an eddy current that gives rise to a secondary field. The strength of eddy currents decreases with the depth of penetration.

2.2 Analysis Methods

The eddy current NDE data are related to changes in material properties, probe lift-off, and excitation frequency. This NDE method has been used with success for predicting presence of defects in metal parts such as tubing. The very nature of this technique leads, in general, to three-dimensional, nonlinear, partial-differential equations with very complicated boundary conditions; if the probe is moving, the solutions are functions of both time and position. Experimental, analytical, and numerical models have been used for the solution of electromagnetic field problems [10].

Experimental models are those based on data obtained from measurements on simulated or actual eddy current NDE test rigs. As a result, the models are empirical in nature and cannot be readily extended to the wide variety of test configurations and defect shapes needed for the development of realistic defect characterization schemes.

Analytical models are those derived from the interaction between magnetic field and electrical phenomena described by equations 2.1, 2.2, and 2.3. There are two approaches for solving this system of equations. One is to manipulate the partial differential equations to obtain other differential equations that are easier to solve. Such differential formulations can be very useful. But these partial differential equations have many solutions, and it is not always easy to find the particular solution that fits the boundary conditions for a specific problem. The other approach is to construct integral formulations of the problem. The integral formulations incorporate the boundary conditions in the mathematical description of the system. This can provide closed-form solutions in only a limited number of cases [28]. In order to obtain an analytical result, simplifying approximations are made with regard to the number of dimensions,

linearity of material properties, symmetry, boundary conditions, and defect shape. Even with such simplifying assumptions, the mathematics is very complex and the results tend to be limited to a single geometry.

To overcome the limitations of analytical methods, various numerical calculational methods have been devised. Numerical models and analytical models are both based on the same field equations; however, instead of seeking to solve the equations directly by invoking simplifying assumptions, discretization procedures are used in the numerical model which ultimately lead to a matrix equation whose solution satisfies the original field equations point by point.

Two types of numerical techniques are commonly used. One is the finite difference method (FDM). The other is the finite element method (FEM). The FDM approach focuses on the behavior of the system at specific points, defined by the mesh used to discretize the field region. The drawback to FDM is that it lacks the flexibility needed to accommodate irregular geometries and singularities that do not coincide with constant coordinate surface, and some finite difference techniques require many computational cycles to arrive at a solution. FEM, in contrast, consider the behavior of the system within the discrete elements of the mesh, and then assembles the element equations into a description of overall system. Thus, it can be applied to a much wider class of problem [28].

Several efforts have been made to develop computer modeling tools for EC inspection by many investigators. Pate and Dodd of Oak Ridge National Laboratory have developed several Fortran programs to aid in the design of EC tests and probes. The modeling is based on the Burrows point defect theory. Philipp of Washington State University has developed a 2-D

finite element computer modeling program for EC inspection.

Numerical techniques are not limited by material nonlinearities or awkward defect shapes but rather by the core storage available on computers. The major disadvantage of numerical models is that one does not end up with an actual equation as the solution but rather with flux, current density, and phase plot, or impedance plane trajectories [10].

2.3 Measurement Methods

Three major measurement methods (multi-frequency, pulse, and microwave) may be used to study different aspects of the eddy current phenomena.

Microwave testing is better suited for precise measurement of surface defects because of the higher frequency. But the method has usually been considered too complex for most applications.

Pulsed systems are applied most advantageously to detect defects below the surface. They extract information about the test material by analyzing the shape of the transient waveform. However, pulsed techniques are limited at present by insufficient theoretical and equipment development.

Multiple-frequency techniques are among the most widely used. These techniques benefit from advances in single-frequency analysis, and by their nature present a greater number of measurement variables for use in data analysis. The increased degrees of freedom allow for the elimination of unwanted tests and focus on the characteristics of interest [4].

2.4 Data Analysis

The most difficult part of the eddy current inspection method is the analysis of experimental data. The following methods may be used for experimental data representation and analysis.

- Data transformation and display.
- Pattern recognition methods.
- Digital signal processing methods.
- Statistical models to interpret data.
- Artificial neural networks for defect classification and characterization.

2.5 Test Property Variations in Steam Generator Tubing

A number of variables in a steam generator produce signal changes in an eddy current test. Some of them are shown in Figure 2.2. A steam generator consists of a large number of tubing. The tubes are fixed at each end to a tube sheet and anchored (or restrained) at intervals by tube supports. The tubing is usually thin wall (less than 10 percent of its diameter) and can be inspected from the bore side when the generator head is removed. Nonferromagnetic tubing with high electrical resistivity such as Inconel 600 or Type 304 stainless steel are common in nuclear plant steam generators. The properties that vary and can affect the readings are (a) defect size, (b) defect location in the tube wall, (c) tube wall thickness, (d) tube-to-tube support distance, and (e) coil-to-coil distance [4].

We are primarily concerned with the changes in the first three parameters, but the last two parameters produce the largest signal changes.

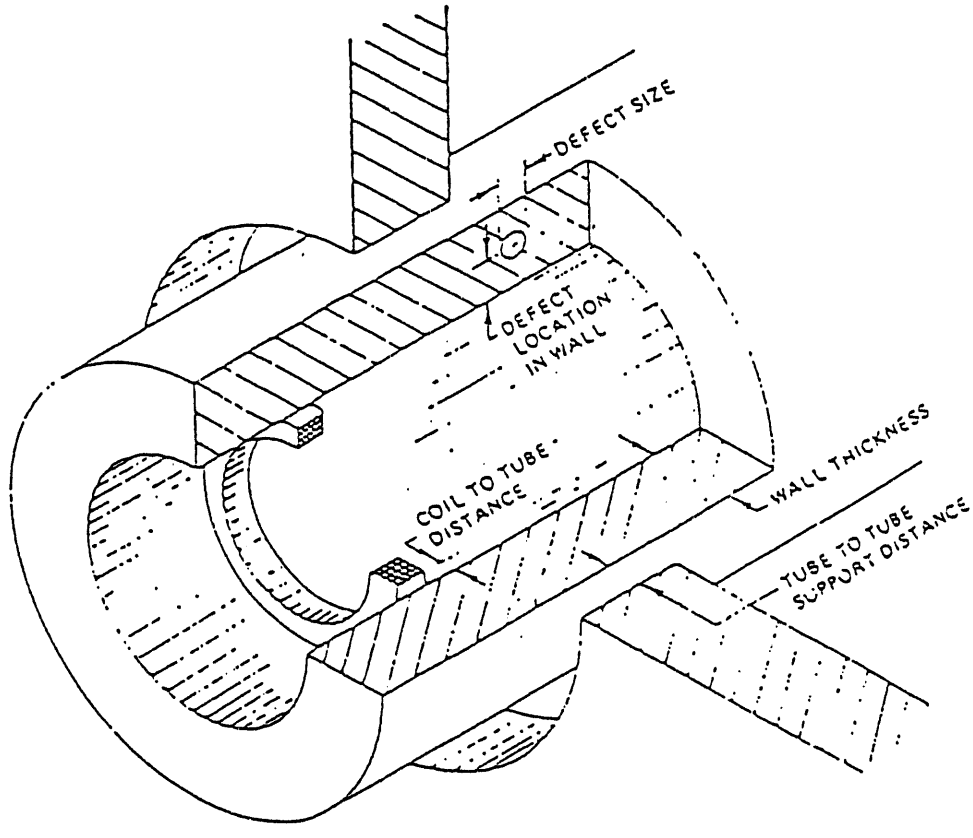


Figure 2.2 Test properties that vary during a steam generator inspection (From Ref. 4).

2.6 Metal Crack Growth Prediction Problem and Application to Component Aging

One of the goals of this research project is to study the possibility of establishing a procedure for the prediction of crack growth using neural networks. Metal crack growth is a basic problem of metal fatigue life study. In nuclear and fossil-fueled power plants, metal fatigue is a potential life limiting mechanism for pressure retaining components. Fatigue cracking may occur at high stress locations, such as steam generators, due to the thermal transients and the accumulation of start-shutdown cycles or load cycles during plant operation [29]. Several studies have been made in the area of crack growth prediction. Some standard procedures have been developed for stress and fatigue analysis, such as the ASME Standard. The following is a brief review of some papers published on the subject of fatigue lifetime estimation and crack growth prediction.

Theoretical Analysis Methods for Fatigue Crack Growth

The fatigue crack growth process in a metallic material is divided in to two phases: crack initiation and crack propagation. Usually the crack initiation stage is defined according to an arbitrarily specified crack length: the crack length ranging from a size of grain diameter to about 50 to 100 μm.

Crack propagation is related to many factors. These relations are formulated in terms of external and internal constraints [30]:

(a). External constraints: the load and displacement boundary conditions, the component and crack geometry, the chemical and thermal environment, and their variation in time.

(b). Internal constraints: the microstructure of the material; chemical composition, the spatial arrangement of the constituent atoms, the type , configuration and distribution of defects, and their variation in time.

The Griffith theory of fracture mechanics and fracture kinetics are the theories for crack growth problem. The Griffith theory (developed in 1920s) is the source of the modern fracture theories. It establishes that crack growth originates at defects which are inherent in all real materials. This growth occurs when the energy needed for the creation of new surfaces (and a plastic zone) is exceeded by the energy that is available from the change in the stored elastic energy. When this condition is satisfied, the crack grows catastrophically, at a significant fraction of the velocity of sound [30].

Since 1950s, fracture mechanics was developed as the theory that defines the mechanical state of a crack. Fracture mechanics is rooted in the Griffith theory, it modified and corrected some shortages of Griffith theory. Fracture mechanics established that a definite condition must exist in the crack-tip zone; extended the linear elastic model to non-linear and to plastic solid continuum; and developed a mathematical formulation that provides the rigorous basis for the practical measurements of the characteristic values which represent the overall, general characteristic fracture resistance of a specific material. However, because fracture mechanics is a continuum mechanics theory, it cannot be used to enquire into the complex physical process at the microscope level where crack growth is controlled. More details of this theory can be found in reference [31].

The theory of fracture kinetics was originated in the 1960s. It was derived from the understanding of the atomic-level physical processes that control crack growth. Essentially, fracture kinetics is the method of establishing (1) the rate of each of the component steps of a crack growth process; (2) the quantitative effects of the associated loading, geometrical boundary conditions, thermal and chemical environment, and material characteristics; and, most

importantly (3) the complete description of the crack growth velocity - that is, the constitutive equation of time-dependent fracture [31, pp. 43]. The fracture kinetics theory includes rate theory, deterministic fracture kinetics theory, deterministic constitutive laws, probabilistic fracture kinetics theory, and probabilistic constitutive laws.

Fatigue Lifetime Predictive Techniques

Fatigue life prediction of components involves the study of creep crack growth under a range of stress and temperature and selection of a suitable parameter to describe the same. Based on this parameter the life of the component can be estimated.

Newman et al [32] discussed the problem in the assessment of residual life prediction. The stress-intensity factor range ΔK is used as the characteristic parameter. Liaw et al [33] analyzed the life estimation of stream pipes of alloy steel based on the Ct energy rate integral approach. Radhakrishnan et al [34] has found that among the various parameters tried to correlate the creep crack growth, the energy rate linear integral C^* gives the best description of the crack growth rate.

Related Applications in the Nuclear Industry

Reference [29] describes the development of an automated fatigue lifetime monitoring technology that utilizes nuclear or fossil-fueled power plant process data to perform a continuous prediction of fatigue damage accumulation in critical components. The methodology has been incorporated into a PC-based computer program called **FatiguePro**. This program acquires plant process computer data (temperature, pressures, flows, valve positions, etc.) and interprets them

to predict local loads and temperatures in the monitored components. Green's functions are then used to determine transient thermal stresses at high stress locations in these components and to account for thermal history effects. A fatigue evaluation is then made based on the computed transient stress history. The fatigue based on Miner's Rule, is accomplished in accordance with ASME Boiler and Pressure Vessel Code stress and fatigue analysis procedures. This research was sponsored by the Electric Power Research Institute (EPRI).

Reference [35] describes some of the state-of-the art engineering techniques utilized for the remaining life assessment and condition improvement of steam turbine/generator components in service. These include the finite element method for thermal and stress analysis, the inelastic strain-based fatigue analysis, and fracture mechanics for remaining life assessment.

SECTION 3

EDDY CURRENT INSPECTION DATA AND DATA CALIBRATION

3.1 Eddy Current Inspection Data

A large multi-frequency eddy current (EC) inspection database from laboratory testing of typical tube material was acquired from the Metals and Ceramics Division of Oak Ridge National Laboratory. The data were recorded from two series of measurements on an ASME section XI standard specimen, shown in Figure 3.1. The OD artifacts simulate tube support/tube sheet, ferrite and copper. The database structure is shown in Figure 3.2. The OD artifact rings on the standard are moveable, so that the effect of changing their location with respect to tube defect location may be studied. The database was organized into 900 individual files according to artifact type, defect depth, and the distance from the center of the defect to the artifact.

A typical impedance plane (resistance versus inductive reactance) trajectory of data from a differential eddy current probe transducer is shown in Figure 3.3. The artifact is the tube support and the defect is at a distance of 2.0 inches from the artifact and has a depth of 20% of the tube thickness. The phase plane plot shows the trajectory corresponding to an AC source of frequency 60 kHz.

Two computer programs have been developed to display EC tube inspection data. A Fortran program displays impedance plane trajectory, and plots of integration and differentiation of impedance trajectory. A LOTUS 1-2-3™ program displays different descriptors (EC signal compression). These tools are very helpful in providing a "quick-look" at the data as a function of AC source frequency, defect type, and defect parameters. Also, the database management

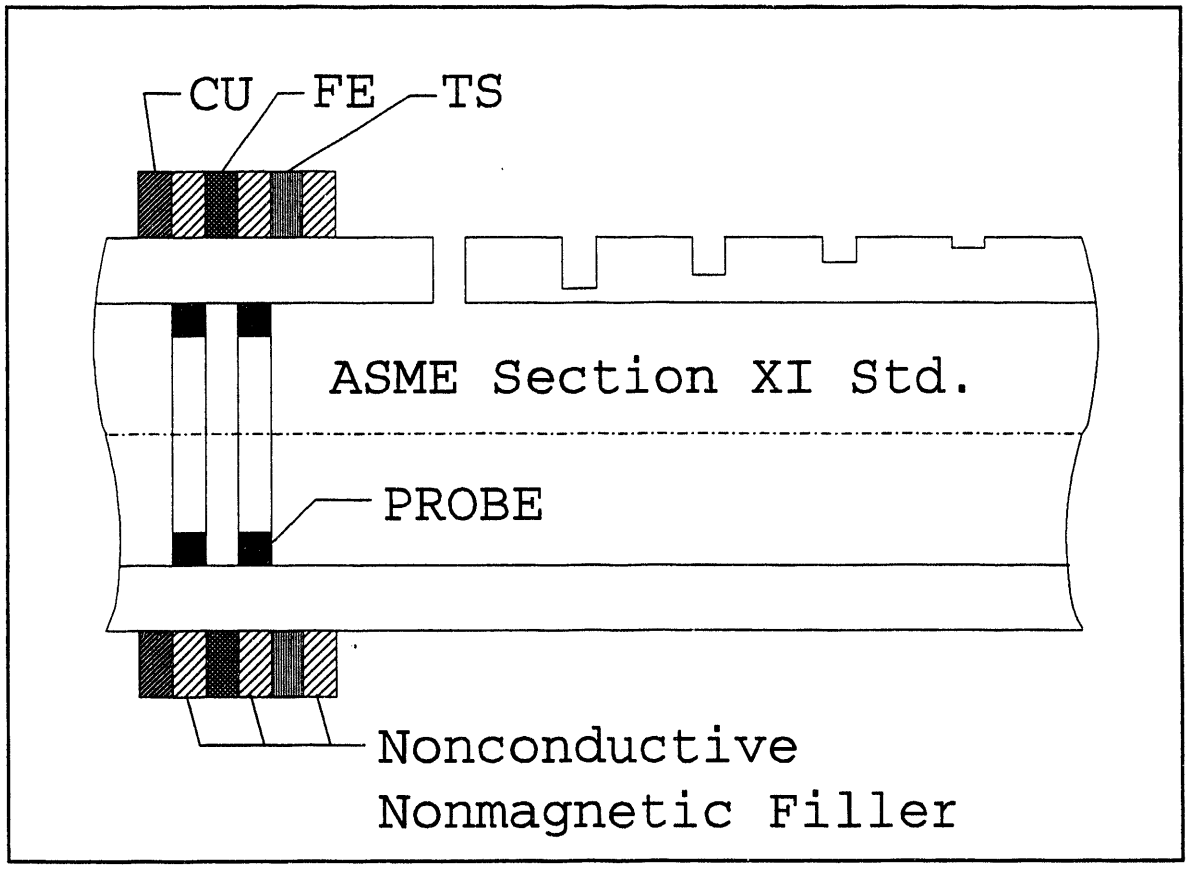


Figure 3.1 ASME Section XI standard test specimen with OD artifacts.

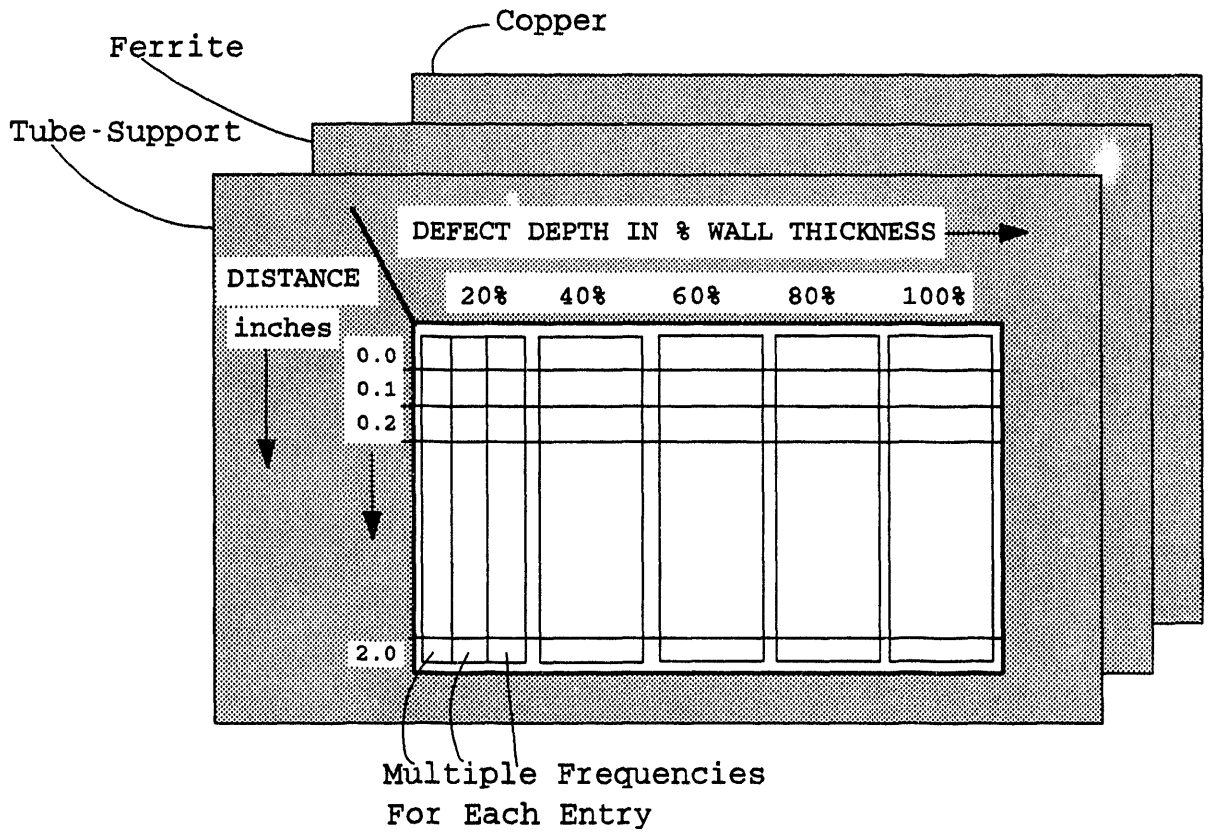


Figure 3.2 Description of multi-frequency eddy current tube inspection data from Oak Ridge National Laboratory (ORNL).

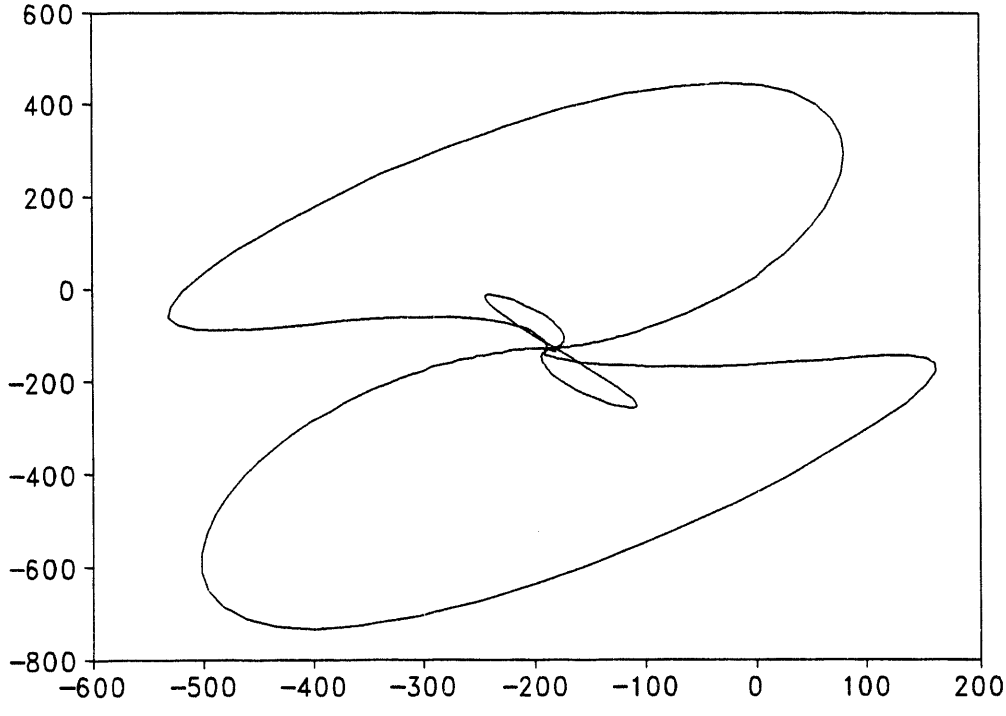


Figure 3.3 A typical impedance plane trajectory of data from an eddy current probe transducer. The defect is in the tube support at a depth of 20% and the distance from the defect to artifact is 2.0 inch. The AC source frequency is 60 kHz. (ORNL).

program EDDYANN may be used to display the data.

3.2 Data Calibration

Two sets of eddy current inspection data have been obtained from ORNL. The data S11RDG8F.dat (C data) were used for neural network training, and the data SEC11RDG.dat (B data) were used for testing the trained neural networks.

The B data and C data were obtained from two series of independent measurements on an ASME Section XI Standard specimen. Since the equipment setup for these two measurements were different, the impedance plane values in the above two data files were also different. Figure 3.4 shows the x component of C data and B data (AC source of frequency 60 kHz) for the tube support artifact, 20% depth, and 0.1 inch distance. It is necessary to calibrate these data to make them compatible.

The measurement data, in general, will have different scaling and orientation (displaced origin) for different equipment setup. The relationship between different measurement data sets can be described using a linear regression model

$$Y = aX + b \tag{6.1}$$

where

Y = one measurement data

X = another measurement data

a = scaling factor

b = shifting factor.

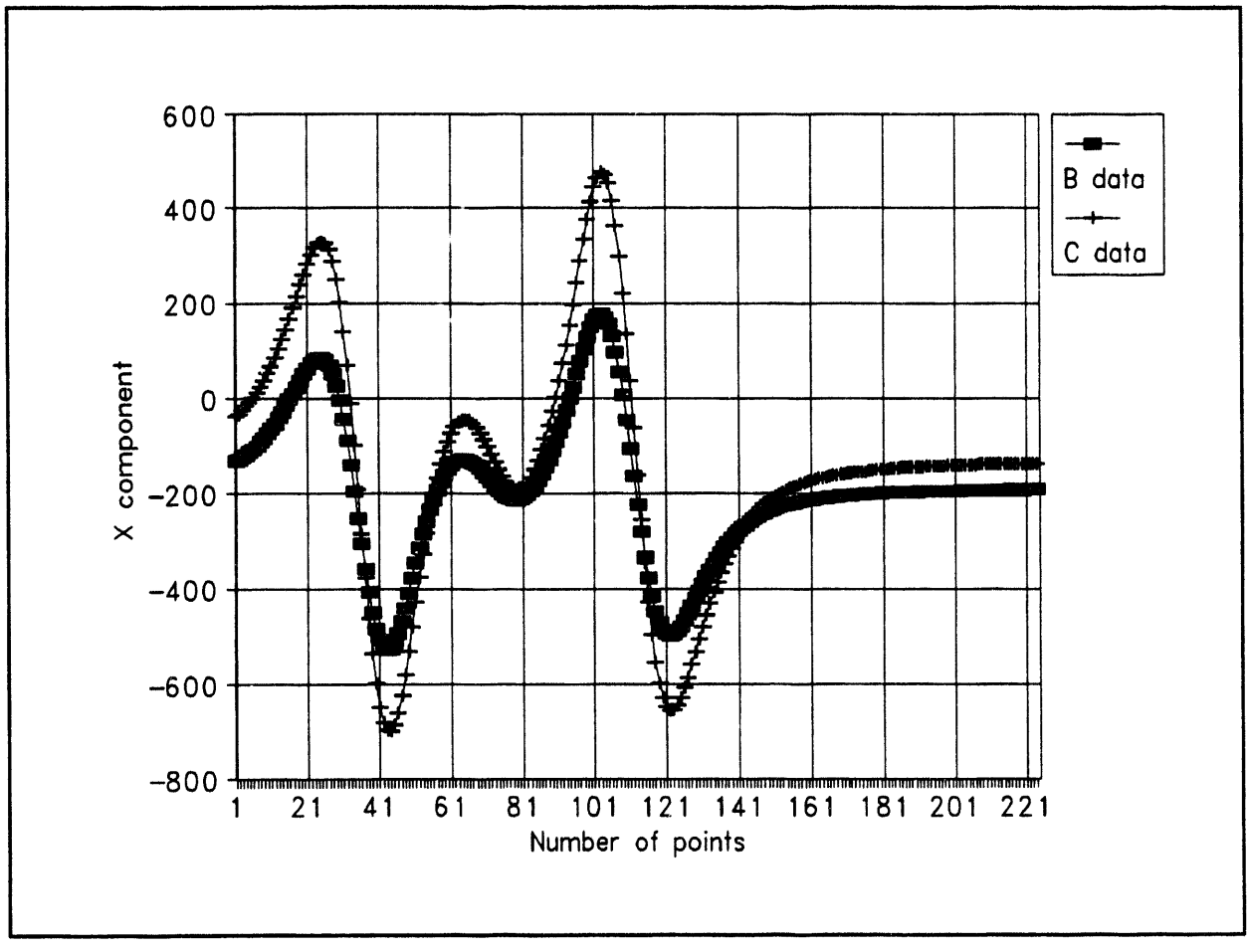


Figure 3.4 X-component of B data and C data before data calibration

For two independent measurements, if parameters a and b can be obtained , then one measurement will be compatible with the other measurement.

Eight data files from C data set and B data set were used to perform the linear regression to determine the parameters a and b. Table 3.1 shows the parameters a, b for x and y component of different frequency signals. The average values of a and b were used to calibrate the B data set. Figure 3.5 shows the comparison of the x components of the C data and the B data (AC source of frequency 60 kHz) for tube support, 20% depth, and 0.1 inch distance after calibration.

Table 3.1(a). Linear regression parameters for x component

	Frequency 1 (600 kHz)		Frequency 2 (200 kHz)		Frequency 3 (60 kHz)	
	a	b	a	b	a	b
c111.dat vs. b111.dat	1.656905	-3960.26	1.695312	784.2246	1.669225	179.9337
c1120.dat vs. b1120.dat	1.614711	-3977.65	1.689191	783.4582	1.666433	182.39488
c1520.dat vs. b1520.dat	1.685385	-3944.38	1.688804	782.3544	1.644389	179.7189
c211.dat vs. b211.dat	1.623865	-3971.84	1.69297	778.1585	1.681263	188.6133
Average	1.645	-3963.5	1.692	782.0	1.665	182.7

Table 3.1(b). Linear regression parameters for y component

	Frequency 1 (600 kHz)		Frequency 2 (200 kHz)		Frequency 3 (60 kHz)	
	a	b	a	b	a	b
c111.dat vs. b111.dat	1.670121	1098.628	1.674005	-1486.04	1.679496	-28.5989
c1120.dat vs. b1120.dat	1.640697	1091.948	1.671087	-1486.35	1.680279	-28.1667
c1520.dat vs. b1520.dat	1.68945	1105.464	1.655070	-1498.05	1.661505	-27.8862
c211.dat vs. b211.dat	1.577619	1073.787	1.679900	-1476.59	1.680279	-28.1667
Average	1.645	1092.4	1.670	-1486.8	1.674	28.3

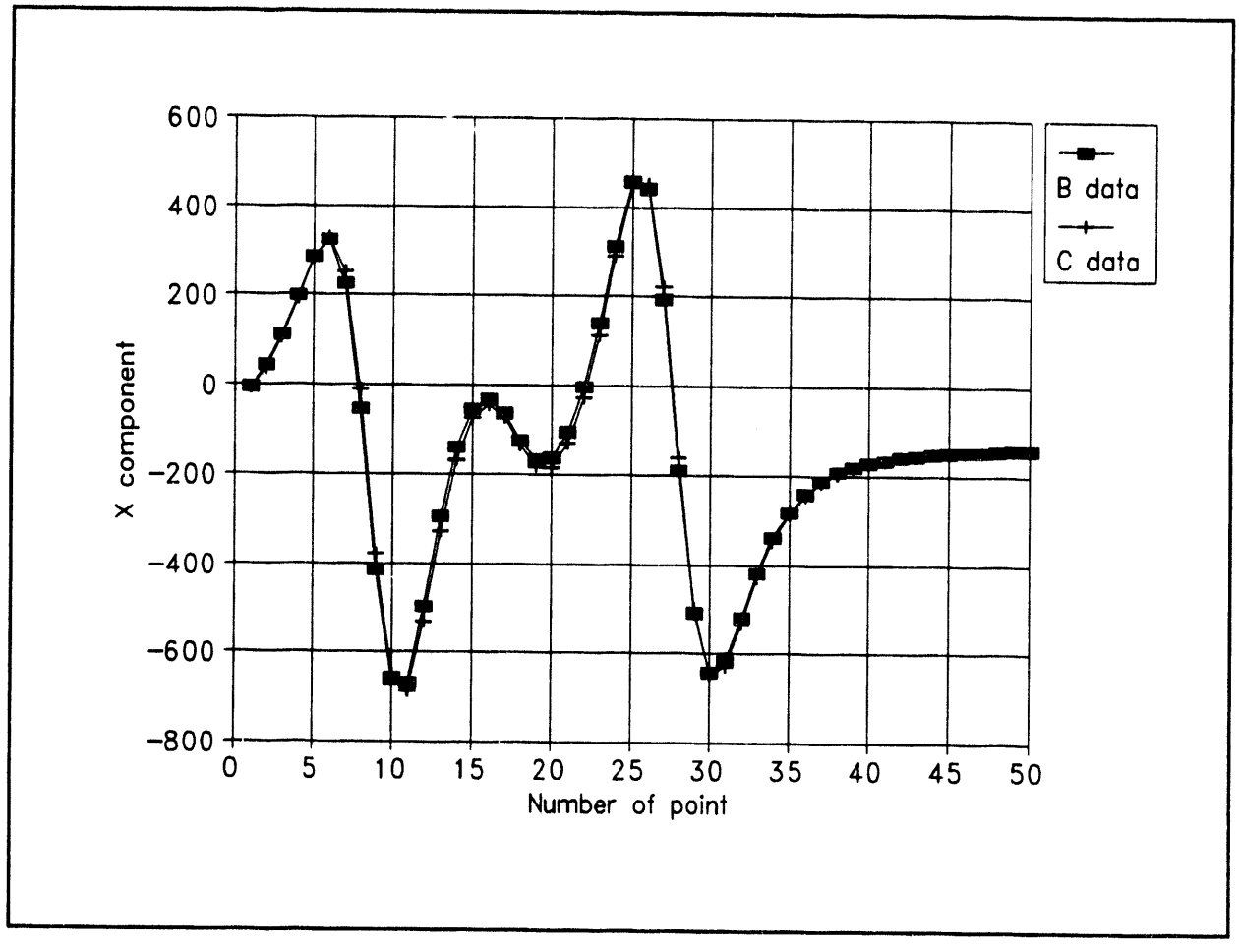


Figure 3.5 X-component of B data and C data after data calibration

SECTION 4

REPRESENTATION OF EDDY CURRENT DATA

For the artificial neural network approach to be effective in defect classification and defect parameter estimation, the information input to the network must have certain features. These are (a) size of data vector, (b) invariance to data scaling, (c) invariance to data orientation, and (d) sensitivity of the defect type and defect size parameters to input signatures. The data representation (and compression) may be classified into nonparametric and parametric techniques.

4.1 Nonparametric Data Representation Methods

Nonparametric representation methods involve reorganizing the raw measurement data using (1) direct compression of raw data, (2) subtraction from a reference data, (3) magnitude and phase of the raw data, (4) integral value of the raw data, and (5) sequence of radii from the center of gravity to the closed contour of the shape.

4.1.1 Direct Compressed Raw Data Representation (RAW)

The current neural networks (NNET) software has the following requirements.

(a) All data files must have the same number of elements in the input vector.

(b) The total number of elements in the input vector, hidden layer and output layer vector must not exceed 350.

(c) All training and testing data files must be normalized.

The size of the EC inspection raw data changes from 225 lines to 421 lines. Therefore, the raw data files must be compressed to yield small and equal size data files. The information in the original data files, must be reflected in the compressed data files.

A direct data compression method has been used. Based on the NNET requirement, the size of the compressed data vector is set as 50. The new vectors are determined by skipping n points, where n is an integer (total length of the raw data divided by 50). Figure 4.1(a) shows the eddy current impedance plane trajectory of raw data D111.dat (frequency 1). Figure 4.1(b) shows the corresponding compressed data plot. By comparing them, we can see that the compressed data file contains sufficient information to describe the impedance plane trajectory.

The direct compressed raw data is not very effective for characterization using neural networks, because the experimental eddy current signals suffer from instrument gain drift and zero fluctuations. However, this approach can be used to compress the size of the raw data to satisfy the requirements of neural networks. This is the preliminary step in undertaking other nonparametric data representation methods.

4.1.2 Subtraction From a Reference Data (SRAW)

The difference in the complex impedance plots of eddy current data between normal and defect cases shows only in portions of the two signal. The raw data subtraction method is the subtracting of test data from a normal reference data. Thus, the common portions are removed, and the defect is enhanced. Figure 4.2(a) shows the impedance plane plot of the reference data. Figure 4.2(b) shows the plot of compressed data D111.dat. The subtraction of reference data from D111.dat is shown in Figure 4.2(c). Generally, this treatment can increase the sensitivity

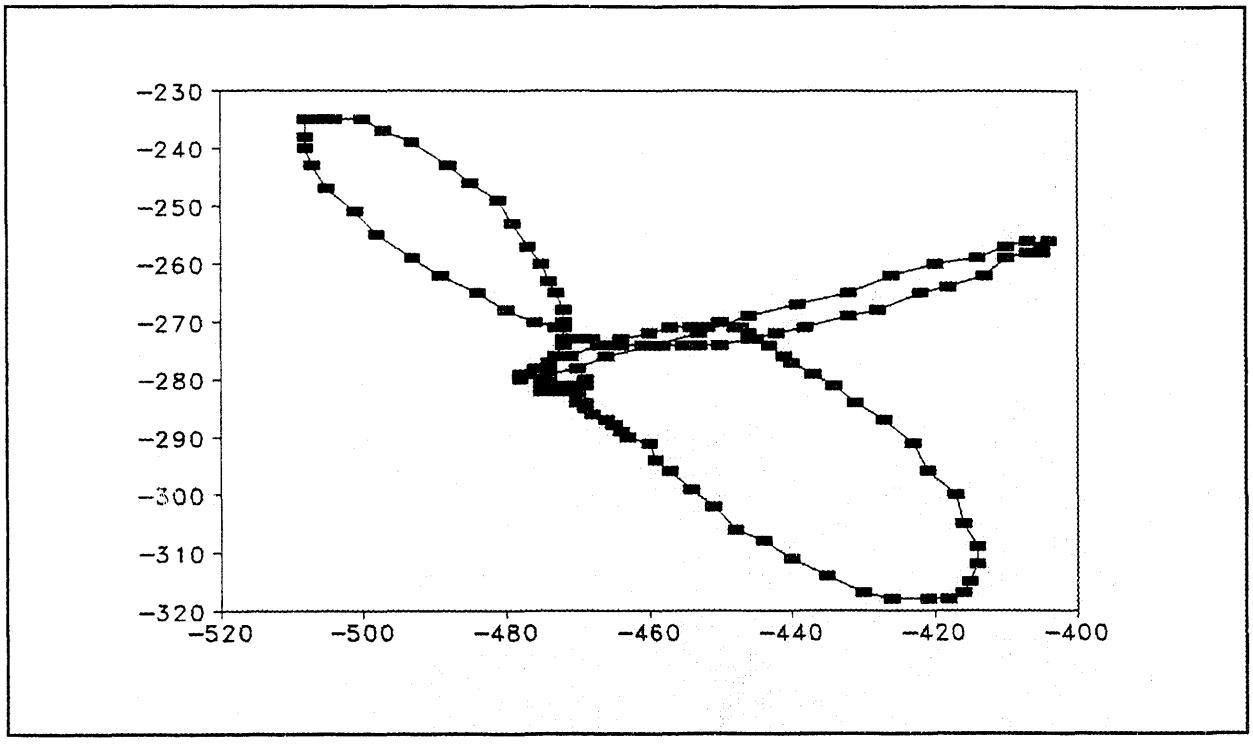


Figure 4.1(a) The eddy current impedance plane plot of raw data D111.dat (frequency 1).

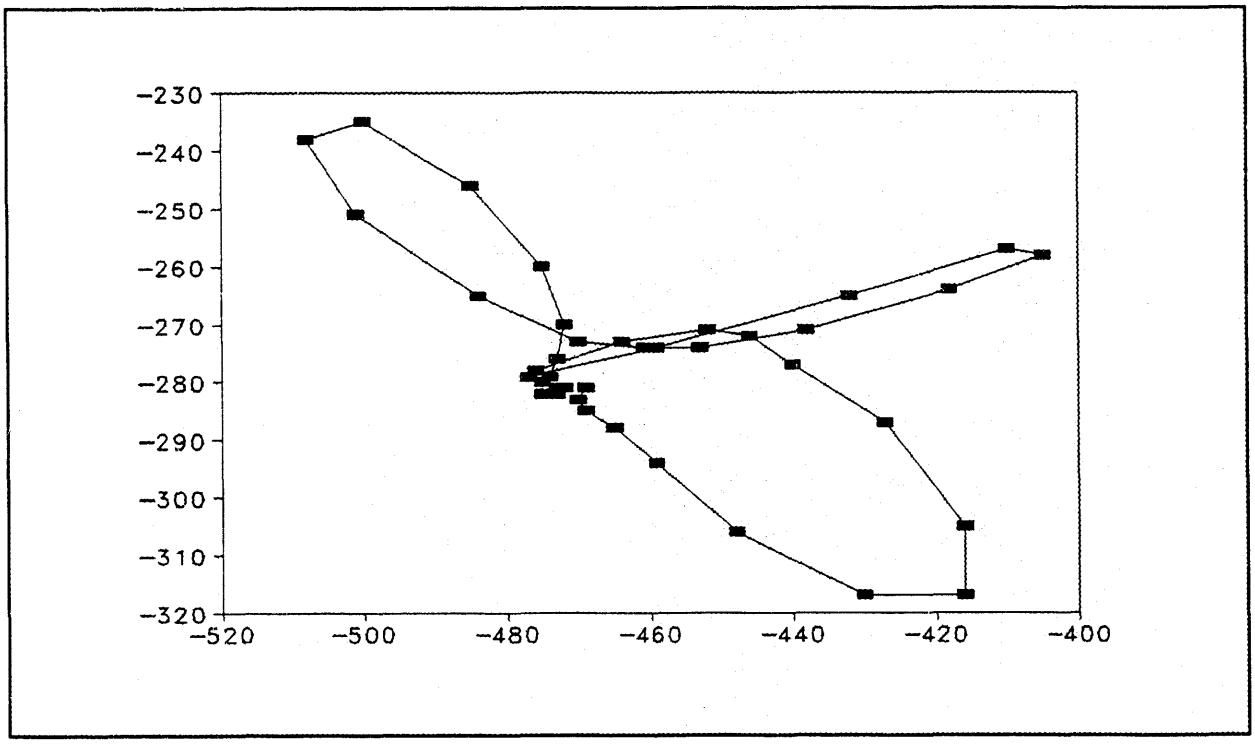


Figure 4.1(b) The eddy current impedance plane plot of compressed data D111.dat (freq. 1).

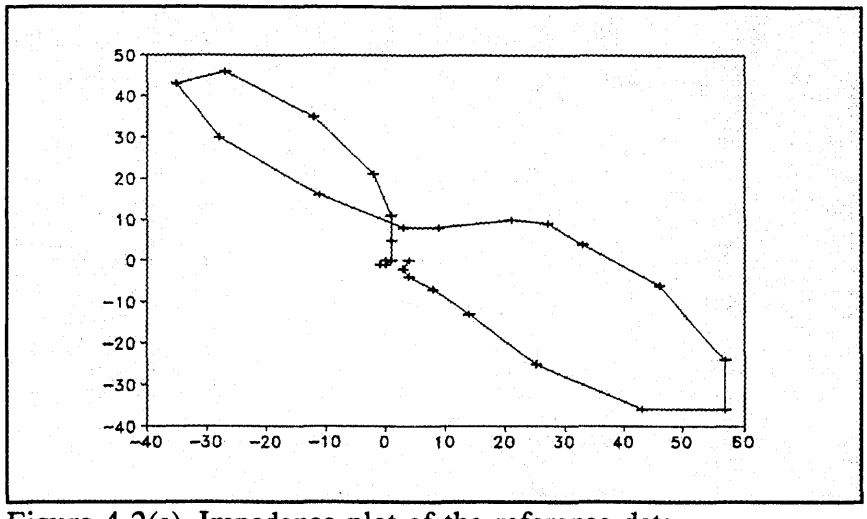


Figure 4.2(a) Impedance plot of the reference data

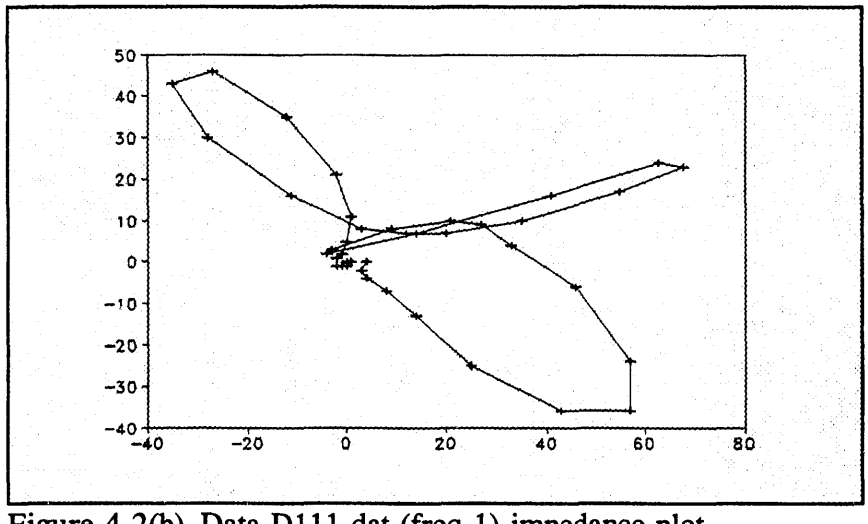


Figure 4.2(b) Data D111.dat (freq.1) impedance plot

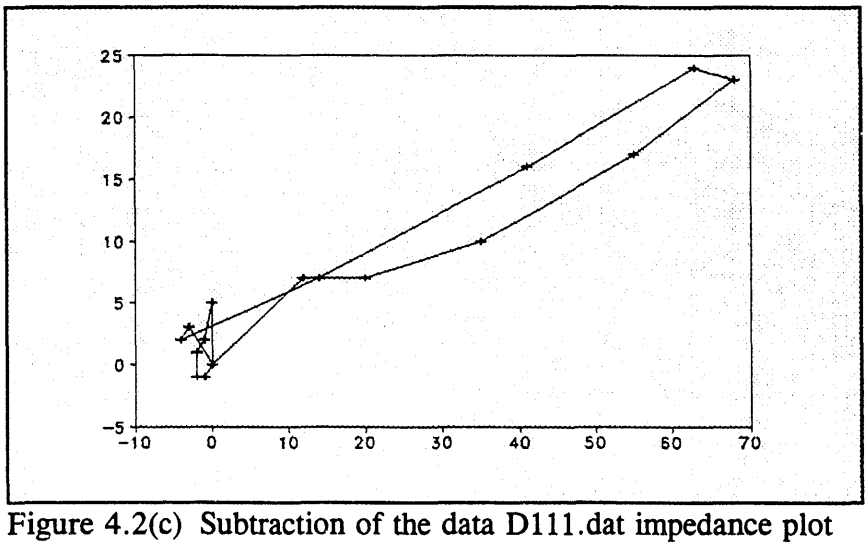


Figure 4.2(c) Subtraction of the data D111.dat impedance plot

of the test signal.

The drawback of this method and other nonparametric representation techniques is that they suffer from the effects of instrument gain drift, zero fluctuation, and the starting point of the data. Therefore, it is necessary to ascertain the accuracy of raw data. Deeds and Dodd [1] have developed a procedure to calibrate the measurement data with a standard data. Therefore, the calibration data can be considered as the correct data.

4.1.3 Compressed Magnitude and Phase Representation (MP)

This method converts the compressed complex impedance values to magnitudes and phases so that magnitude and phase can be normalized separately.

4.1.4 Compressed Integral Signal Representation (CINT)

By observing the plots of integrated raw data, it is seen that the line integration of impedance data is sensitive to certain frequencies, and may be used to identify the defect type. Figures 4.3(a), 4.3(b), and 4.3(c) show integrated data plots for files D111.dat, D211.dat, and D311.dat. These correspond to the artifact types of tube support, ferrite magnetite, and copper, respectively, and for a 20% defect depth and 0.1 inch defect size.

4.1.5 Radii From the Center of Gravity (CG)

Since the defect parameters will influence the center of gravity of the complex impedance plot, a sequence of radii from the center of gravity to the contour of the shape is used to train the neural networks to estimate defect parameters. Figure 4.4 illustrates the definition of radii

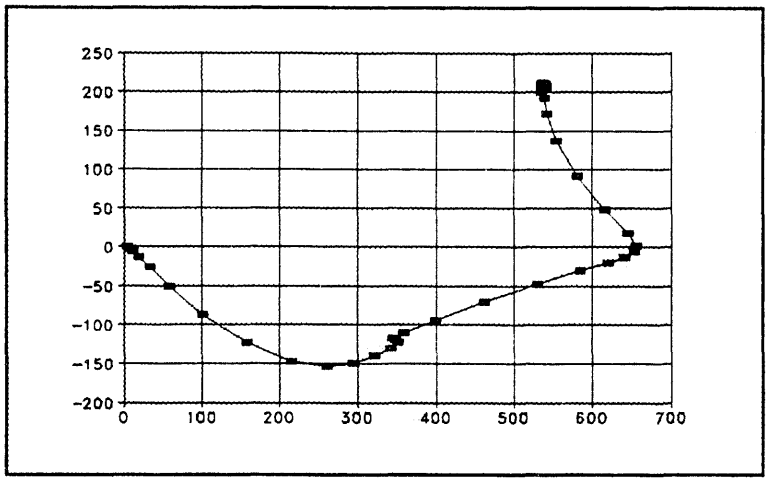


Figure 4.3(a) Data D111.dat (freq. 2) integral plot

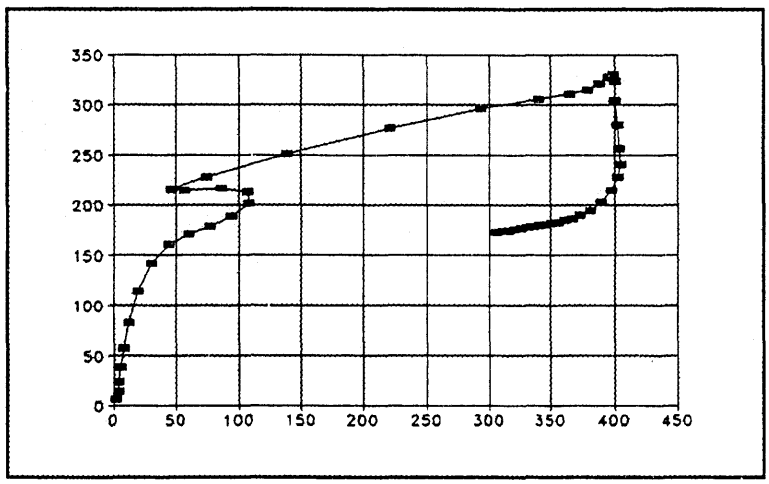


Figure 4.3(b) Data D211.dat (freq.2) integral plot

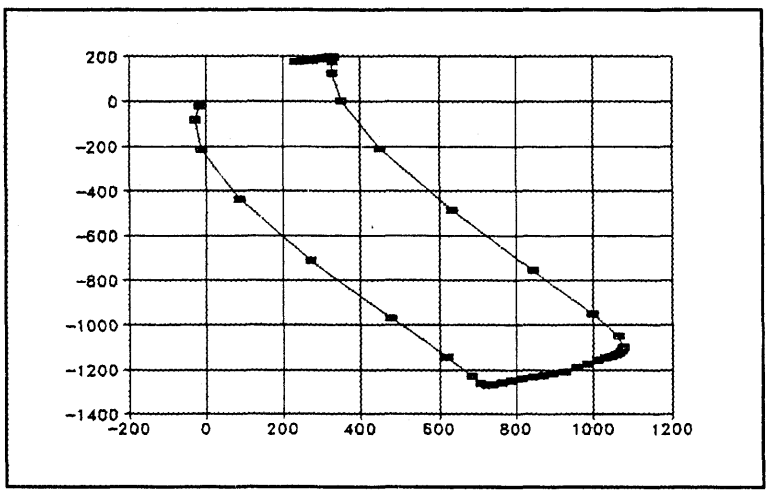


Figure 4.3(c) Data D311.dat (freq.2) integral plot

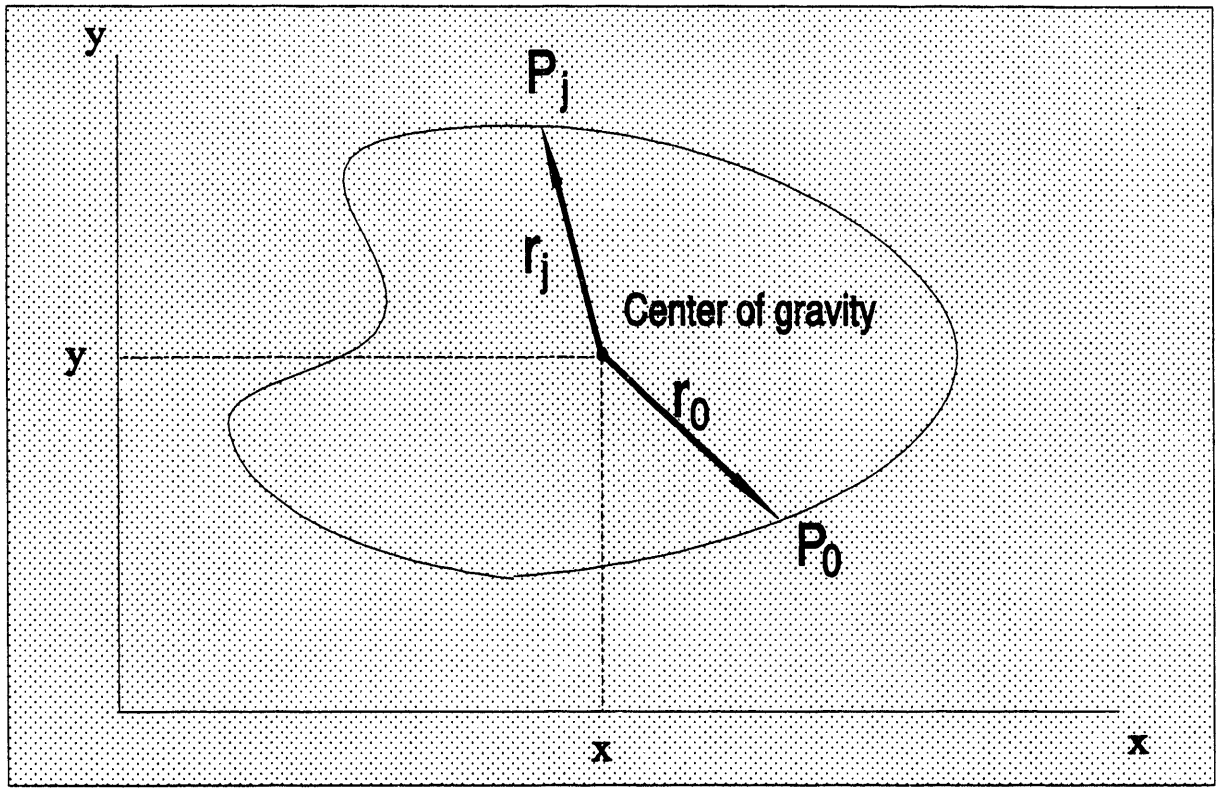


Figure 4.4. Radii from the center of gravity of a closed contour.

from the center of gravity of the closed contour.

4.2 Parametric Representation Methods

Parametric representation methods compress the data using certain "feature vectors" and searching the parameter space for the best fit between the measurement and the computed signature. These techniques include (1) Fourier descriptor method, and (2) autoregression modeling.

4.2.1 Fourier Descriptor Representation (FD)

The Fourier descriptor method is widely used in pattern recognition and applied artificial intelligence. This technique generates a feature vector called the Fourier descriptors whose elements are a function of the shape of the signal. These descriptors have the property of being invariant under scaling, rotation, and translation operations. In addition, they offer a significant amount of data compression. Udpa and Udpa [6] applied this method for the classification of EC signals.

The Fourier descriptor technique is based on the fact that the eddy current defect signal represents a closed contour. It involves the representation of the eddy current signal as a periodic function, with the arc length l serving as the independent variable, as shown in Figure 4.5. Consider a simple closed clockwise-oriented smooth curve γ and let $[x(l), y(l)]$ be the coordinates of a point P that is l arc length units away from an arbitrary starting point P_0. Representing the point P by the complex contour function $u(l)$ and expanding it in a Fourier series gives

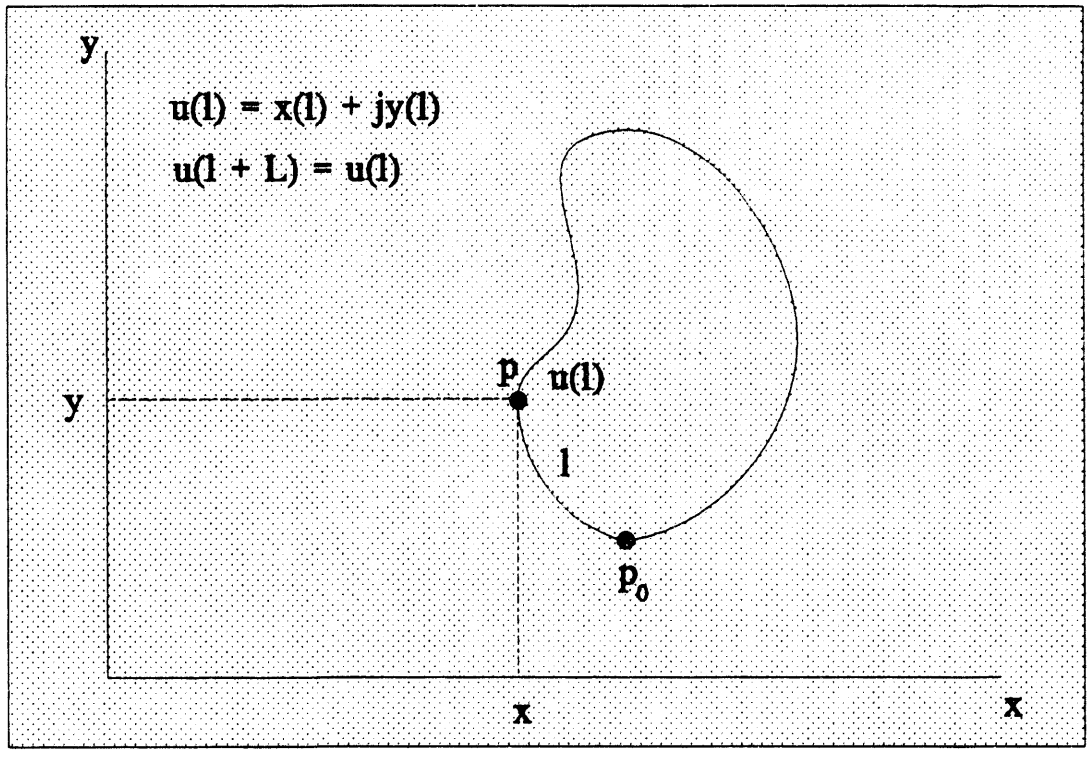


Figure 4.5. Reprentation of a closed curve by a complex contour function u(l) defined as a function of the arc length l

$$u(l) = \sum_{n=-\infty}^{\infty} c_n \exp(\frac{j2\pi nl}{L}) \tag{4.1}$$

where

$$c_n = \frac{1}{L} \int_0^L u(l)\exp(\frac{j2\pi nl}{L}) \tag{4.2}$$

and L is the total arc length of the curve.

By approximating the curve by a polygon of m sides with vertices at V_0, V_1, ..., V_{m-1}, the Fourier coefficients can be computed as

$$c_n = \frac{L}{4\pi^2 n^2} \sum_{k=1}^{m} (b_{k-1} - b_k) \exp(\frac{-j2\pi nl_k}{L}) \tag{4.3}$$

where

$$l_k = \sum_{i=1}^{k} | V_i - V_{i-1} | \qquad k > 0,\ l_0 = 0 \tag{4.4}$$

and

$$b_k = \frac{V_{k+1} - V_k}{| V_{k+1} - V_k |} \tag{4.5}$$

The Fourier descriptor set $\{d_n\}$ is defined as

$$d_n = \frac{c_{1+n}\, c_{1-n}}{c_1^2} \tag{4.6}$$

Generally, 8 to 10 Fourier descriptors are sufficient to represent the information of a closed contour. We chose eight descriptors to represent an original eddy current signal.

Based on our experience in training neural networks, the Fourier descriptors were found to be not very sensitive in identifying the difference among EC signatures for parameter estimation for different defect types. This may be because (1) only a limited number of terms are used in Fourier series to approximate the periodic signal, (2) portions of the impedance plot do not bear useful information, and (3) the Fourier descriptors have difficulties in describing local information and discriminating symmetrical shapes [2].

A subtraction technique similar to the raw data subtraction from a reference data is used to increase the sensitivity of the Fourier descriptors. The Fourier descriptor d_n can be divided into two parts: d_n(defect) indicates the information from the defect data, and d_n(normal) represents the information from the non-defect reference data. The d_n(defect) can be obtained by subtracting the d_n(normal) from d_n. The requirement of this treatment is that all the data set should have the same starting point.

4.2.2 Autoregression (AR) Modeling of Object Contours

This approach is to represent the EC defect data using autoregression model parameters. The procedure is described briefly.

The approach to represent a 2-D shape with a 1-D signal is to use a sequence of radii from the center of gravity to the closed contour. The radii sequence $r_t(l)$ is:

$$r_t(l) = r[(t-l)L + l], \quad l = -(M-1),...,L. \tag{4.7}$$

The autoregression model is a simple prediction of the current radius by a linear combination of M previous radii plus a constant term and an error term:

$$r(l) = \alpha + \sum_{j=1}^{M} \theta_j r(l-j) + \omega_l \qquad l=1,...,L, \tag{4.8}$$

The feature vector is

$$X = [\theta_1,.....,\theta_M,\alpha]^t \tag{4.9}$$

with dimension M+1.

The use of AR parameters as a feature vector can characterize shapes in a very systematic manner, and the feature vectors are proved to be translation and scale invariant. The major disadvantage is that these schemes are very sensitive to shape occlusion [2].

SECTION 5

ARTIFICIAL NEURAL NETWORK MODELS FOR DEFECT CLASSIFICATION AND PARAMETER ESTIMATION

5.1 Description of Artificial Neural Networks

Artificial neural networks provide general mapping between two sets of information. This nonlinear mapping from data to data is very useful in associating information pairs where a clear mathematical relationship is not available. Artificial neural networks have been applied to the problems of pattern classification, signal validation, plant monitoring, transient state identification in power plants, underwater acoustic signature recognition, and many others [24].

Artificial neural networks are developed to simulate the most elementary functions of neurons in the human brain, based on the present understanding of biological nervous systems. These network models attempt to achieve good human-like performance such as: learning from experiments and generalization from previous samples. The network models are composed of many nonlinear computational units which are called as processing elements (PE) or nodes that operate in a parallel distributed processing architecture.

A typical nonlinear processing element is shown in Figure 5.1. A processing element is analogous to a neuron in that it has many inputs from input signals or from other PEs and combines (sum up) the values of the inputs, adjusted by their weights. This sum is then subjected to a nonlinear transformation, often called a transfer function, that controls the output in accordance with the prescribed nonlinear relationship. If the transfer function is a threshold function, output signals are generated only if the sum of the weighted inputs exceeds the

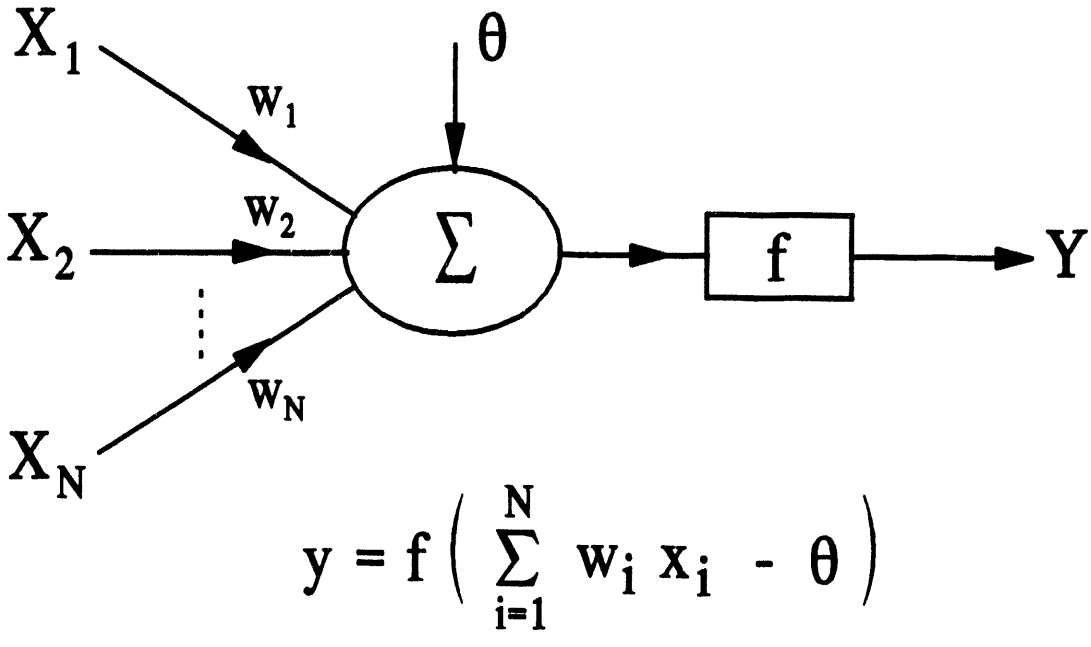


Figure 5.1 A typical nonlinear processing element of a neural network.

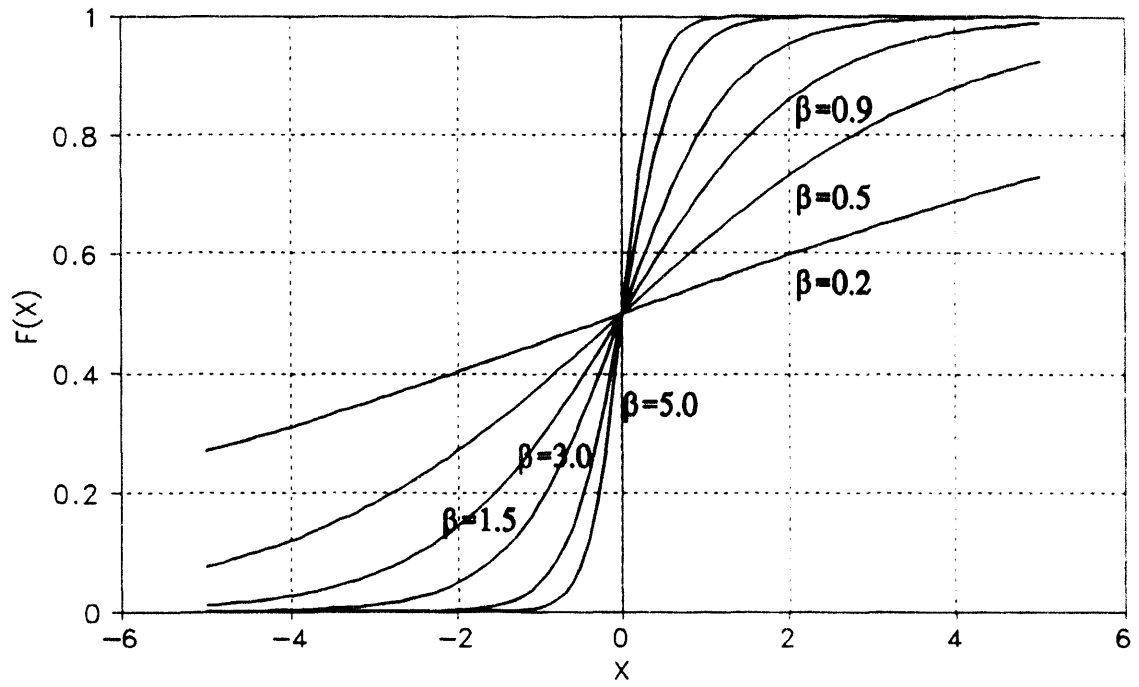
threshold value. If the transfer function is a continuous nonlinear (or linear) relationship, the output is a continuous function of the combined input. The most commonly used transfer function is the sigmoid function with the form

$$F(x) = \frac{1}{1 + e^{-\beta x}}, \quad \beta > 0 \tag{5.1}$$

The sigmoid function changes smoothly from zero (for large negative values) to one (for large positive values) with a value of 1/2 for zero input. This sigmoidal relationship is shown in Figure 5.2.

Artificial neural systems are usually organized into layers of PE's. The layer which receives the input signals from the environment is called the input layer, or input buffer. The layer which emits the signals to the environment is called the output layer. The output signals from the output layer maybe in the form of a signature vector, or a pattern classification index. Any layer between the input layer and the output layer is called a hidden layer. Hidden layers play an important role in processing signals. These layers are used to represent the nonlinear relationships between the two sets of information. A general architecture of a multi-layer neural network is shown in Figure 5.3.

The operation of an artificial neural network involves two processes: learning and recall. Learning is the process of adapting the connection weights in response to external stimuli presented at the input layer. During the learning process, each processing element receives weighted inputs, it combines them and computes an output. If the calculated output of the network is equal to the target output, then learning will stop; if there is a significant difference between those two values, an adjustment of weights must be made to minimize the difference



$$F(x) = 1/(1+ e^{-\beta x})$$

Figure 5.2 Sigmoidal function used for transfer function of neural processing element

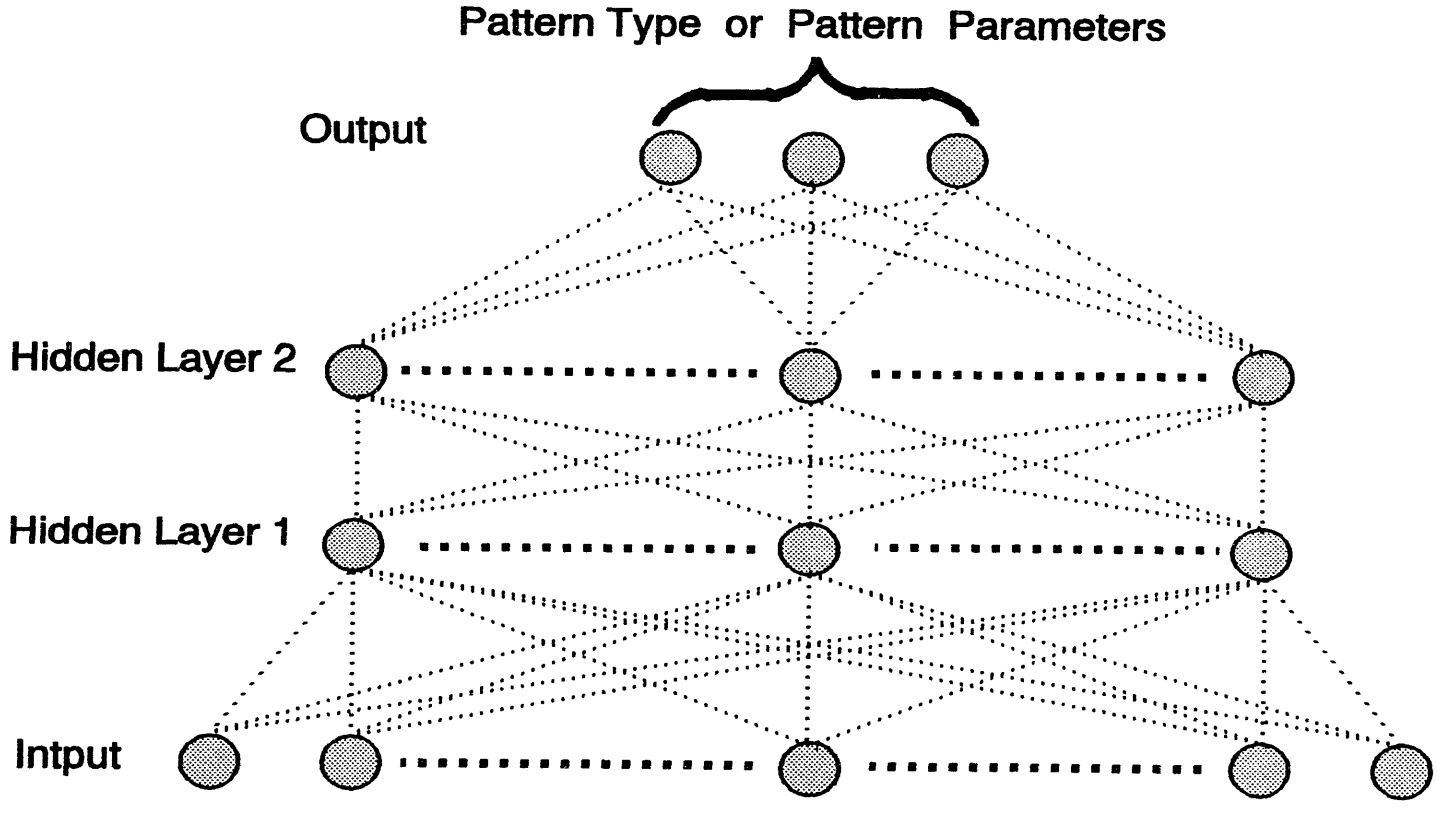


Figure 5.3 Architecture of a multilayer neural network used in eddy current pattern classification and for estimating defect parameters.

[36]. Recall is the process of using or testing the trained network.

There is a variety of neural network architectures: Hopefield network, multi-layer feedforward network, perceptron network, bi-directional associative memory networks, adaptive resonance theory networks, Kohonen self-organization feature map, probabilistic neural network, counter-propagation network, and others. In this study, multi-layer feedforward networks and probabilistic neural networks were used. A detailed discussion of these two algorithms is given in the following sections.

5.2 Learning Methods

Several different kinds of learning methods are used for establishing ANN's. These learning methods may be classified into three categories: supervised learning, unsupervised learning, and random learning.

Supervised learning is the most commonly used method. In supervised learning, a stimulus presented at the input buffer of the network generates an output at the output buffer and is then compared with the known desired output to produce an error signal. The system then uses a learning algorithm to convert the difference (error signal) into an adjustment of the connection weights. The key to the development of a successful supervised neural network is to provide supervised information to the network and estimate the connection weights such that the error between the desired output and the network output is minimized. In this study, only supervised learning is used.

In unsupervised learning, only the input stimuli are applied to the input layer of the network. The network then organizes itself internally so that a particular hidden processing

element responds strongly to a particular type of input stimulus [37].

Random learning introduces random incremental changes into the weighting functions, then either retains or drops the changes depending whether or not the output is improved or not (based on arbitrary criteria set by the user). This process continues until the error approaches a minimum value.

5.3 Learning Algorithms

The most common learning algorithms are: Hebbian learning, Delta-rule learning, and competitive learning.

The Hebbian learning occurs when a connection weight on an input path to a PE is incremented if both the input and the desired output are high. This is analogous to the biological process in which a neural pathway is strengthened each time it is used.

The Delta-rule learning occurs when the error signal (difference between the desired output and the actual output) is minimized using a least-squares process. The back-propagation is the most common implementation of the Delta-rule.

The competitive learning occurs when the processing elements compete; only the processing element yielding the strongest response to a given input can modify itself, and become more like the input.

5.4 Training of Neural Networks Using Back-Propagation Algorithm

Back-propagation algorithm is the most widely used systematic method for supervised learning in multiple (three or more) layer artificial neural networks. The application of this

training algorithm in 1986 by Rumelhart, Hinton, and Williams [25] was the key step in making neural networks practical in many real world situations. The mathematical basis for the back-propagation training of ANN's is straightforward but involves several steps. It is well described in Ref. [25].

The goal of the back-propagation algorithm is to teach the network to associate specific output patterns (target patterns) by adjusting the connection weights in order to minimize the error between the target output and the actual output of the network. To accomplish this, the network is usually trained with a large number of input/output pairs. A gradient descent algorithm is generally used to perform the optimization.

The back-propagation training process is composed of two types of passes: the forward pass and the reverse pass. In the forward pass the input signals propagate from the network input to the output. In the reverse pass, the calculated error signals propagate backward through the network where they are used to adjust the weights. The calculation of the output is carried out, layer by layer, in the forward direction. The output of one layer is the input to the next layer. In the reverse pass, the weights of the output layer are adjusted first since the target value of each output node is available to guide the adjustment of the associated weights, using a modification of the Delta Rule. Next, the weights of the middle layers are adjusted. Since the middle layers have no target values, the error must be propagated back through the network, layer by layer.

Consider a typical PE as shown in Figure 5.1, with inputs x_i, weight w_i. The summation of the weighted inputs designated by I is given by

$$I = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n = \sum_{i=1}^{n} x_i w_i \tag{5.2}$$

The nonlinear function used is the typical sigmoid and the output of the processing element is given by

$$y(I) = \frac{1}{1 + e^{-\alpha(I+\theta)}} \tag{5.3}$$

where θ is a nodal bias.

In Equation (5.3) the parameter α describes the shape of the sigmoidal function. The effect of different values of α is illustrated in Figure 5.2. The bias is used to shift the activation function along the X-axis. As the forward pass completes, the error between the network output and the target values are calculated.

In the reverse pass, the connection weights are corrected to reduce the error found after the forward pass. This error-correction procedure is made from the output layer to the hidden layer. The Generalized Delta Rule is utilized to adjust the interconnection weights so as to reduce the square of the error for each pattern. In this process, the local error δ is calculated for each node. It reflects the amount of error associated with that unit. The local error for node q in the output layer k is defined as:

$$\delta_{q,k} = -2\alpha \, [T_q - f_{q,k}] \, f_{q,k} \, [1 - f_{q,k}] \tag{5.4}$$

where

T_q = desired output of node q

$f_{q,k}$ = actual output of node q.

The local error for node p in the hidden layer j is defined as:

$$\delta_{pj} = \sum_{q=1}^{n} \alpha \, \delta_{q,k} \, W_{pq,k} \, f_{pj} \, [1 - f_{pj}] \tag{5.5}$$

where

$W_{pq,k}$ = the weight between the node p in the hidden layer j and the node q in the output layer k

$f_{p,j}$ = output of node p.

Then, according to the General Delta Rule, the weight adjustments are made as

$$\Delta W_{ab} = \eta \, \delta_b \, f_a \tag{5.6}$$

where

ΔW_{ab} = the weight adjustment between lower layer node a and upper layer node b

δ_b = local error of node b

f_a = active value of lower layer node a

η = learning rate.

The learning rate defines the step size of training.

A common technique to reduce training time and reduce the probability of being trapped in a local minimum is to use a momentum term which enhances the stability of the training process. This technique involves adding to the weight adjustment a term which is proportional to the amount of the previous weight change.

The step-by-step procedure of back-propagation training is as follows:

(1) Randomize the weights to small random values (both positive and negative) to assure that the network is not saturated by large values of weights.

(2) Select a training pair from the training set.

(3) Apply the input vector to network input.

(4) Propagate the input vector in a forward fashion through the network using Equations (5.2) and (5.3) until the final network outputs are calculated.

(5) Calculate the network output and the error (the difference between the network output and the desired output).

(6) Calculate the local errors using Equations (5.4) and (5.5).

(7) Adjust the weights of the network using Equation (5.6) to minimize the error.

(8) Repeat steps 2-7 for each pair of input/output vectors in the training set until the error for the entire system is acceptably low.

5.5 Optimal Back-propagation Networks for Eddy Current Inspection Data Analysis

Back-propagation neural networks (BPN) were used to develop the neural network models for artifact classification and defect parameters estimation. The preprocessed eddy current impedance data were used as input feature to back-propagation neural networks, with the output map providing artifact type and estimates of defect parameters (depth and distance). Different data compression methods are implemented here and are ranked according to their effectiveness in producing a proper mapping. Separate networks for artifact classification and parameter estimation have been developed. A PC-based software called the NeuralWorks Professional II/Plus is used in most of the implementation.

There are several issues that need to be considered when utilizing the back-propagation algorithm to train a neural network. The following discusses the selection of hidden layers and nodes, and the learning options.

5.5.1 Selection of Number of Hidden Layers and Nodes

The selection of number of hidden layers and hidden nodes is one of the most important issues in back-propagation network applications. There have been various studies related to this topic, but there is no definite solution to this problem. However, it has been concluded that using only one hidden layer is sufficient to solve the problems in the area of signal processing, plant monitoring, parameter estimation, and sensor validation [23]. In this project, it has been found that one hidden layer is sufficient for the training of the center of gravity data and integral data for defect parameter estimation.

The selection of hidden nodes for a fully-connected, feedforward networks with one hidden layer is based on the following rules [16]:

Rule-of-thumb 1: The more complex the relationship between the input data and the desired output, the more nodes are normally required in the hidden layer.

Rule-of thumb 2: The upper bound for the number of nodes in the hidden layer is

$$\frac{cases}{10\times(m + n)} = h \tag{5.7}$$

where

cases is the number of rows or vectors in the training file.

m is the number of nodes in the output layer.

n is the number of nodes in the input layer.

h is the number of nodes in the hidden layer.

5.5.2 Selection of Learning Options

The most important learning options for back-propagation network training is selected as follows.

1. Learning coefficient

Learning coefficient is the rate at which weights adjust to correct for errors. In our application, it is set to 0.4 for the first 10000 iterations, and 0.1 for other iterations.

2. Momentum Term

Momentum term is a factor used to smooth the learning. Here it is set to 0.5.

3. The nonlinear transfer function

The nonlinear transfer function transfers the internally generated sum for each node to an output value. Available transfer functions in back-propagation network are: linear, sigmoid, hyperbolic tangent, and Gaussian cumulative distribution. Through several trials of network training, it is found that hyperbolic tangent transfer function facilitates faster and more accurate network training. This is because the output range of hyperbolic tangent is from -1 to +1, as compared to the sigmoid range of 0 to 1. The output of the transfer function is used as a multiplier in the weight update equation, a range of 0 to 1 means a smaller multiplier when the summation is a low value, and a higher multiplier for higher summation. This could lead to a bias in learning higher desired output. The hyperbolic tangent gives equal weight to low and high end values [16].

4. The learning rule

The learning rule in back-propagation network specifies how connection weights are changed during the learning process. Three learning rules are commonly used in BPN: Delta-

Rule, Cumulative Delta-Rule, and Normalized-Cumulative Delta-Rule.

Some networks have been trained using Delta-Rule and the Normalized-Cumulative Delta-Rule. The results are listed in Table 5.1. It is seen that the network using the Delta-Rule takes more iterations to converge and the RMS error can only be reduced to 0.003. The Normalized-Cumulative Delta-Rule is more effective and should be selected.

5. Gaussian noise and RMS threshold value

The Gaussian noise function adds a random number within a special range to each node summation value in the layer. Table 5.2 illustrates the network performance for estimating the distances of the defect from the center of the OD artifact.. It is seen that the network with noise converges faster than the network without noise during training. However, the network with noise does not perform as well as the noise-free network in the recall phase.

The root mean-square (RMS) error threshold is the convergence threshold for BPN training. The training terminates when the RMS is smaller than this threshold. The accuracy of the network can be controlled by setting a value for this threshold. In neural network training, it is not desirable to overtrain the network because of poor interpolation or prediction caused by memorization. Table 5.2 shows the network with different RMS threshold values. It has been found that the network with an RMS threshold of 0.01 has the best recall result.

5.6 Probabilistic Neural Networks (PNN)

The probabilistic neural network (PNN) is very well suited for feature classification. The PNN is based on the Bayesian decision boundaries. The "key advantages of the PNN are that

Table 5.1. Training Networks Using Delta-Rule or Non-Cum Delta-Rule

	RMS	Recall Results	Error (%)	Iterations
Delta Rule	0.05	0.764173	1.89	993
	0.01	0.774499	3.27	9630
	0.005	0.800754	6.77	28121
	0.003	0.806697	7.56	102659
Nor-Cum Delta-Rule	0.05	0.743972	0.80	431
	0.01	0.745801	0.60	6549
	0.005	0.741182	1.18	8272
	0.001	0.743269	0.90	10039

Table 5.2. Training Networks with Noise or without Noise

	RMS	Recall Results	Error (%)	Iterations
Noise	0.05	0.656779	12.43	449
	0.01	0.685313	8.62	2065
	0.005	0.689318	8.09	3939
	0.003	0.694373	7.42	6489
No Noise	0.05	0.743972	0.80	431
	0.01	0.745801	0.60	6549
	0.005	0.741182	1.18	8272
	0.001	0.743269	0.90	10039

the training requires only one single pass and that the decision surface is guaranteed to approach the Bayes-optimal decision boundary as the number of training samples grows" [26]. The main "disadvantage of PNN is that all training sample must be stored and used in classifying new patterns."

Let us assume that there are M patterns to be classified. Let P_i, i=1,2,..., M be the a priori probability of these classes. Let $p(\underline{X}/\theta_i)$, i=1,2,..., M be the probability density function of feature vector X belonging to class i. Then the Bayes decision rule states that class i is chosen over class j if

$$P_i\, p\,(\,\underline{X}\,/\,\theta_i\,) \geq P_j\, p(\,\underline{X}\,/\,\theta_j\,) \tag{5.8}$$

Thus the best choice among the M features is made such that class i satisfies the condition

$$\max_{i=1,2,..,M} \; P_i\, p\,(\,\underline{X}\,/\,\theta_i\,) \tag{5.9}$$

In most cases the feature vector $\underline{X}$ has a Gaussian distribution and has the form

$$p(\underline{X}/\theta_i) = \frac{1}{(2\pi)^{M/2}(detQ_i)^{1/2}} \exp[-\frac{1}{2}(X-X_{mi})^T Q_i^{-1}(X-X_{mi})] \tag{5.10}$$

where Q is the covariance matrix, or in the present context called the smoothing parameter.

The PNN reflects the Bayesian decision making in a feedforward network fashion. Figure 5.4 shows a typical probabilistic neural network. The input units are components of each training feature vector. The pattern unit performs a weighted sum of feature vector components. The number of pattern units for each class corresponds to the number of training patterns for that class. The density functions of class A and class B (in this case) are formed at the

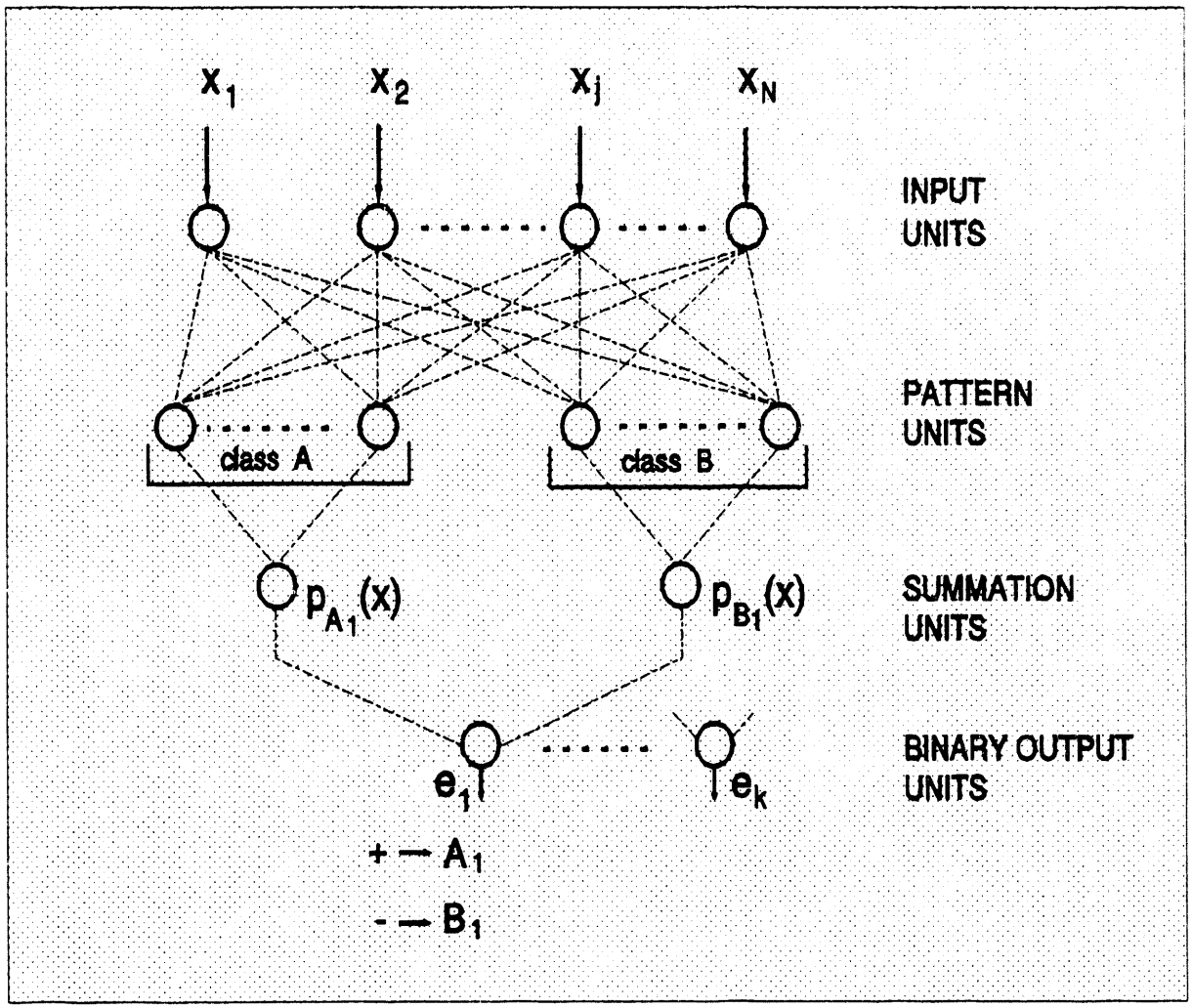


Figure 5.4 Architecture of the Probabilistic Neural Network (PNN) [26].

summation units. A binary output unit is used to decide whether the i-th pattern belongs to class A or class B.

The training of PNN takes only one pass and is a very effective network for feature classification problems. One of the disadvantages of the PNN is that it requires all the patterns for updating the network when new patterns are included. For further details see Ref. [26].

SECTION 6

RESULTS OF DEFECT DIAGNOSTICS USING EDDY CURRENT DATA

6.1 Estimation of Tube Defect Parameters

The defect parameters include defect depth and the distance from the defect to the center of the artifact. In order to make the networks perform effectively, separate networks were established to estimate defect depth and distance from the artifact.

For distance estimation, in order to compare the effects of the different data representation techniques, the data corresponding to defect in tube-support, 20% depth, 0.1 to 2.0 inch sizes were used. In defect depth estimation, the defect in tube-support, at 20% to 100% depths were used.

Generally, the Mean-Squared Error (MSE) is used to evaluate the accuracy of the network. The MSE is defined as

$$MSE = \frac{1}{N} \sum_{k=1}^{N} [x_p(k) - x_m(k)]^2 \qquad (6.1)$$

where

N = Number of patterns.

$x_p(k)$ = the network predicted value for pattern k, and

$x_m(k)$ = the measured value for pattern k.

In parameter estimation, an Averaged Absolute Scale Error (AASE) was also used to judge the accuracy of the network. This is defined as

$$AASE(\%) = \frac{1}{N}\sum_{k=1}^{N}\left|\frac{100(x_p(k) - x_m(k))}{x_m(k)}\right| \qquad (6.2)$$

where N, $x_p(k)$, and $x_m(k)$ are the same as in Equation (6.1).

For EC distance estimation, different back-propagation networks (BPN) have been tried using different number of hidden layers, hidden elements, number of iterations, and training coefficients. Through the training, the trained networks with center of gravity (CG) and compressed raw subtraction data (CRS) were found to be the most for distance estimation. Table 6.1 lists the information for the back-propagation networks obtained from training for these two data representation techniques. Figure 6.1(a) shows the recall result of distance estimation using the center of gravity training data. Figure 6.1(b) shows the recall result of distance estimation using compressed raw subtraction training data.

From the above results, it can be seen that the AASE for the compressed raw substraction data is 0.998%, and for the center of gravity data is 0.156%. Therefore, both of them can be used to estimate the distance from the EC defect to the artifact.

Many networks were trained for the eddy current defect depth estimation. Through training, it was found that only the networks trained using phase angle (PHS) data and the center of gravity data have low mean-squared errors (MSE). Table 6.2 lists the information for the back-propagation networks obtained from training for these two data representation techniques. Figure 6.2 shows the recall results from one of the BPN's trained using phase angle training data. Figure 6.3 shows the recall results from one of the BPN's trained using the radii from the center of gravity data.

From the results, it can be seen that the estimated depths are very close to the desired

Table 6.1. BPN's for EC Defect Distance Estimation

Input Signal	CG	CRS
No. of Input	50	100
No. of Hidden Layers	1	1
No. of Hidden Elements	35	60
Learning Coefficient	0.4	0.4
Transfer Function's Shape	0.7	0.7
Momentum Term	0.5	0.5
No. of Iterations	12051	29987
Mean Squared Error (MSE)	0.0027	0.0041
Avg. Abso. Scale Error (AASE) (%)	0.156	0.998

CG: center of gravity training data
CRS: compressed raw data subtraction training data

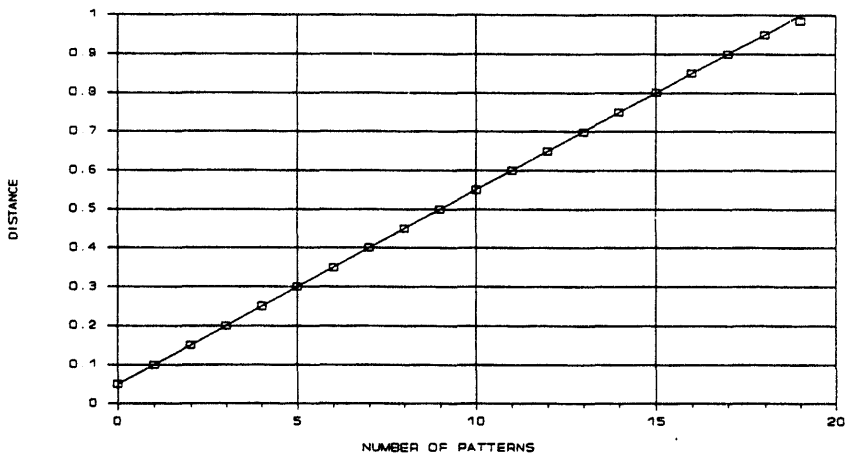


Figure 6.1(a). Recall result of BPN using CG testing data for distance estimation

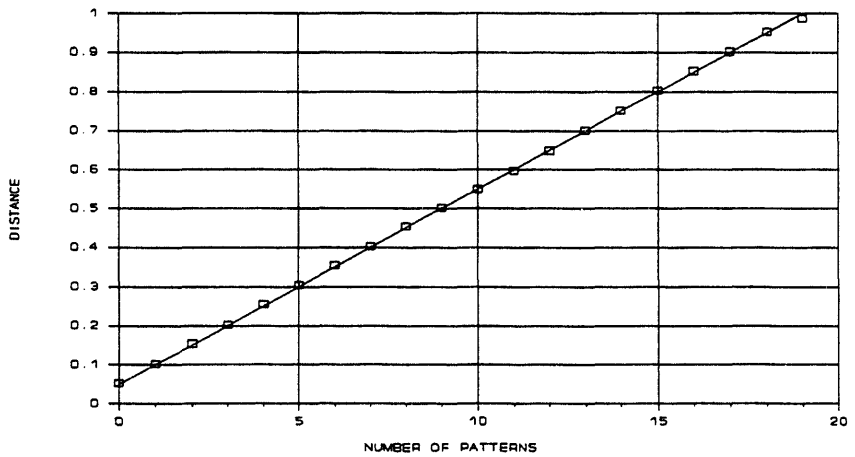


Figure 6.1(b). Recall result of BPN using CRS testing data for distance estimation

Table 6.2. BPN's for EC Defect Depth Estimation

Input Signal	CG	PHS
No. of Input	50	50
No. of Hidden Layers	1	1
No. of Hidden Elements	35	35
Learning Coefficient	0.4	0.4
Transfer Function's Shape	0.7	0.7
Momentum Term	0.5	0.5
No. of Iterations	22051	19889
Mean Squared Error (MSE)	0.0078	0.0054
Avg. Abso. Scale Error (AASE) (%)	0.898	0.595

CG: center of gravity training data
PHS: phase angle training data

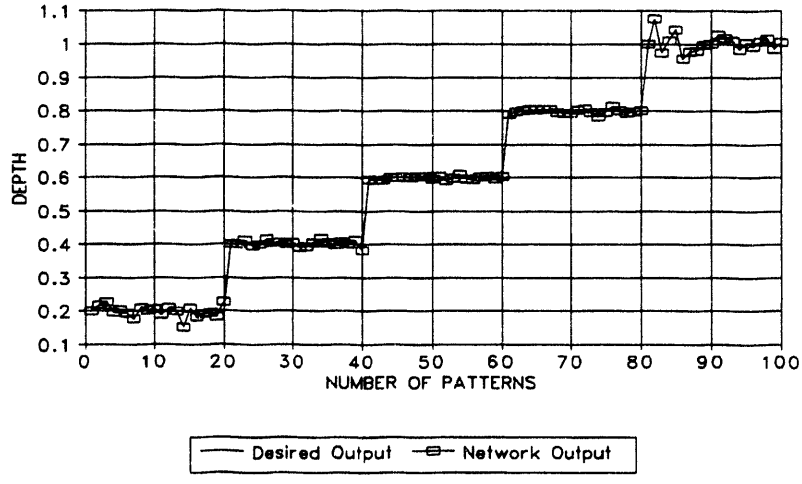


Figure 6.2 Recall result of BPN using phase angle testing data for depth estimation

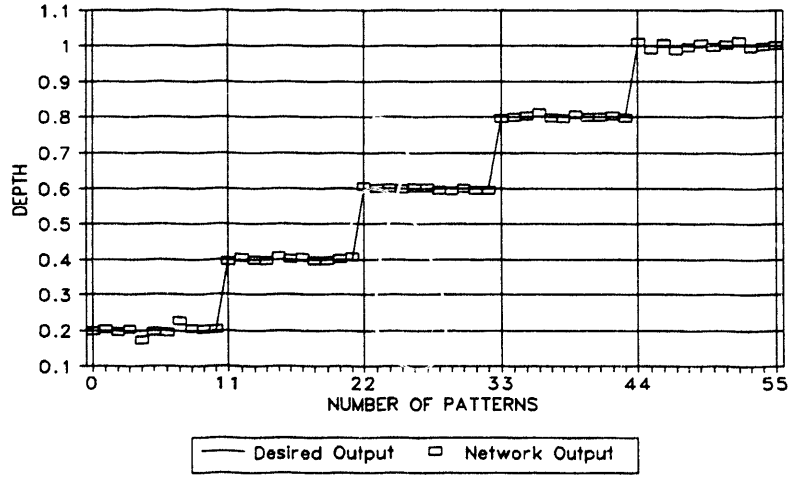


Figure 6.3 Recall result of BPN using CRS testing data for depth estimation

values. Therefore, phase angle data representation method and the center of gravity radii signatures are good for defect depth estimation. The BPN's trained using other data representation methods did not work very well for defect depth estimation. This is because only phase angle representation and radii from the center of gravity representation emphasize the phase angle information about the EC inspection data. A fact to be noted is that the accuracy of defect depth estimation using phase angle data and CG data is very sensitive to the data orientation and scaling. Hence other data representation methods are expected to be tried in the future.

6.2 EC Defect Type Identification

The three defect type or artifacts used in this study were tube support, ferrite magnetite, and copper. Some back-propagation networks were developed to identify them using (1) Fourier descriptors (FD), (2) compressed magnitude and phase signatures (MP), (3) radii from the center of gravity (CG), and (4) compressed integral signal (CINT). One probabilistic neural network (PNN) was also developed using compressed integral signal(CINT).

The networks trained using center of gravity data and compressed integral data were found to be effective for defect type identification. Table 6.3 lists the information for the best networks obtained for these two signatures. In the training data, the tube support, ferrite magnetite, and copper are presented as (0.1, 0.1, 0.9), (0.1, 0.9, 0.1), and (0.9, 0.1, 0.1) respectively. Figures 6.4(a), 6.4(b), and 6.4(c) show the recall results of the BPN using the compressed integral signal. Figures 6.5(a), 6.5(b), and 6.5(c) show the recall results of the BPN using the center of gravity signal. Figure 6.6 shows the recall results using the probabilistic

Table 6.3. BPN's for EC Defect Type Identification

Input Signal	CG	CINT
No. of Input	50	100
No. of Hidden Layers	1	1
No. of Hidden Elements	35	50
Learning Coefficient	0.4	0.4
Transfer Function's Shape	1.0	0.7
Momentum Term	0.5	0.5
No. of Iterations	51241	12561
Mean-Squared Error for Output 1	0.02	0.04
Mean-Squared Error for Output 2	0.02	0.15
Mean-Squared Error for Output 3	0.02	0.18

CINT: compressed integral training data
CG: radii from the center of gravity data

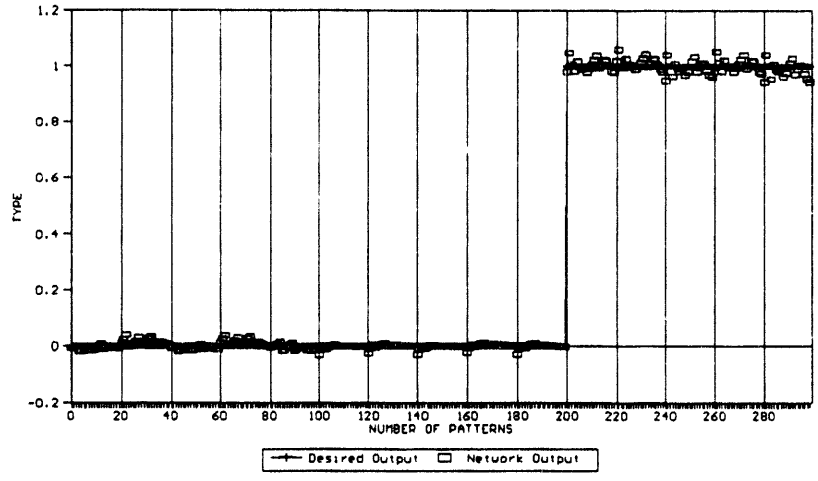


Figure 6.4(a). Output 1 recall result of BPN using CINT testing data

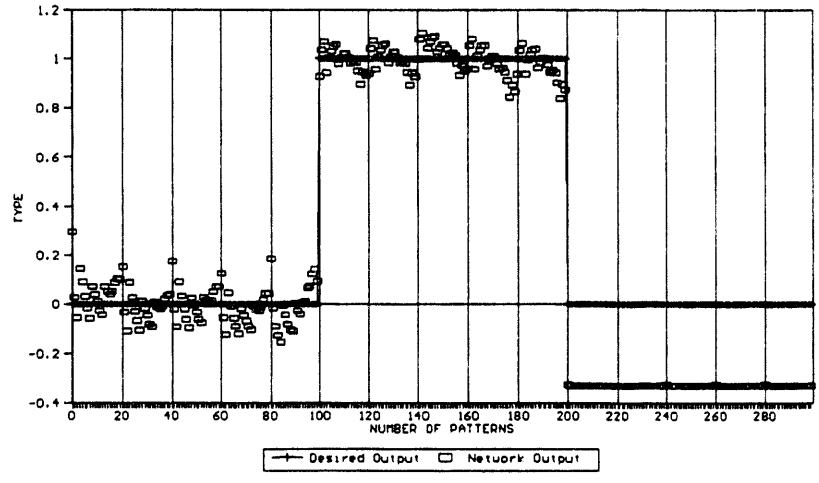


Figure 6.4(b). Output 2 recall result of BPN using CINT testing data

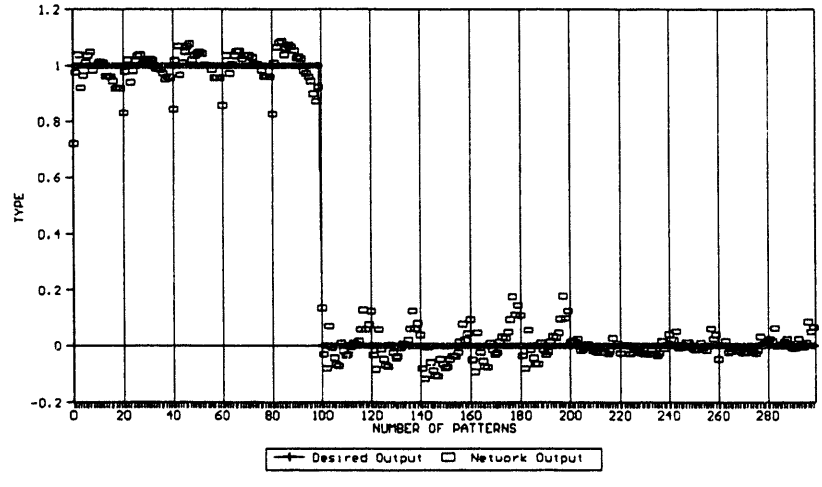


Figure 6.4(c). Output 3 recall result of BPN using CINT testing data

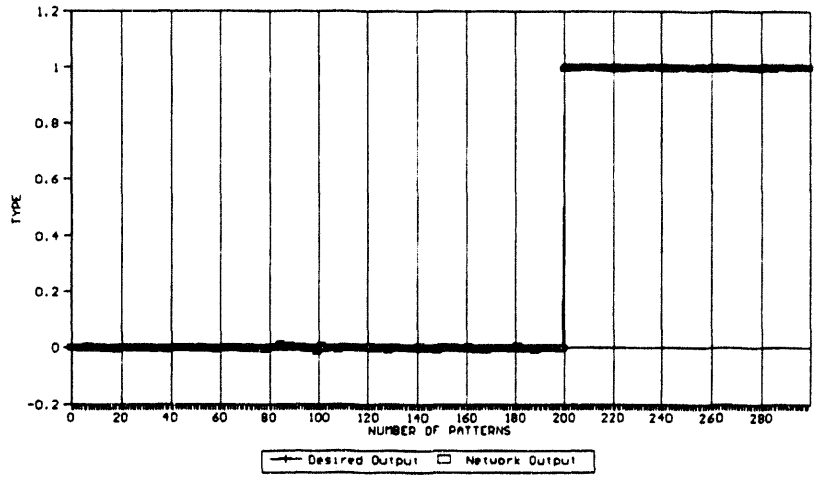


Figure 6.5(a). Output 1 recall result of BPN using CG testing data

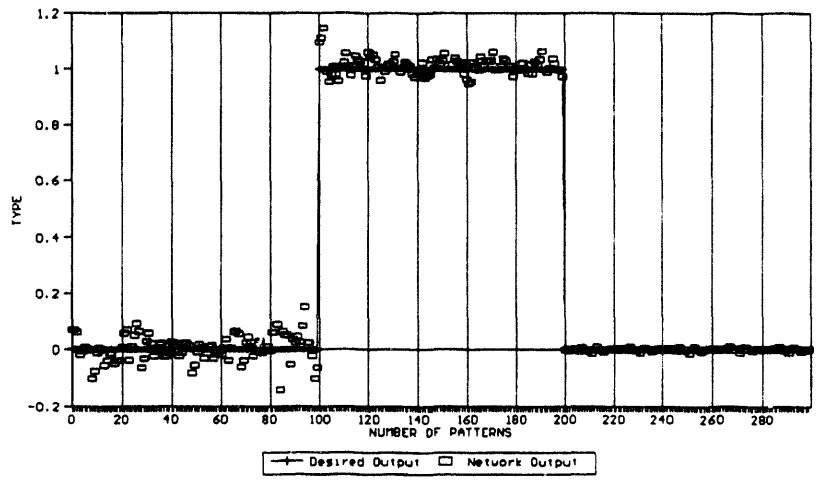


Figure 6.5(b). Output 2 recall result of BPN using CG testing data

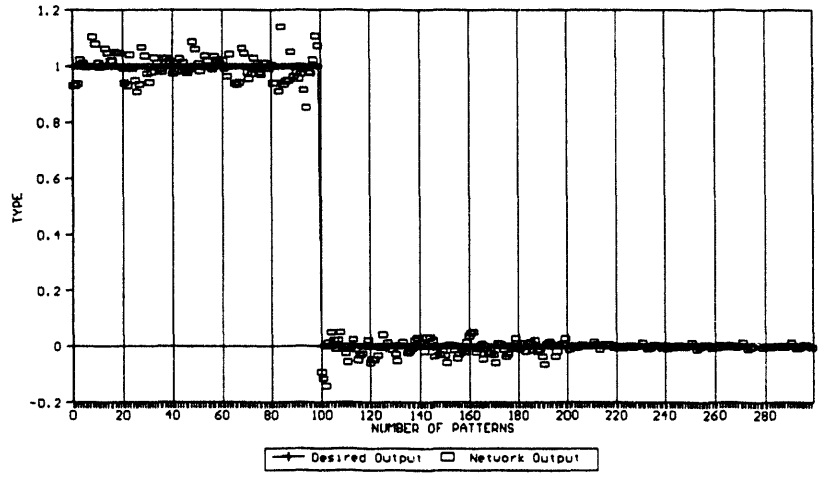


Figure 6.5(c). Output 3 recall result of BPN using CG testing data

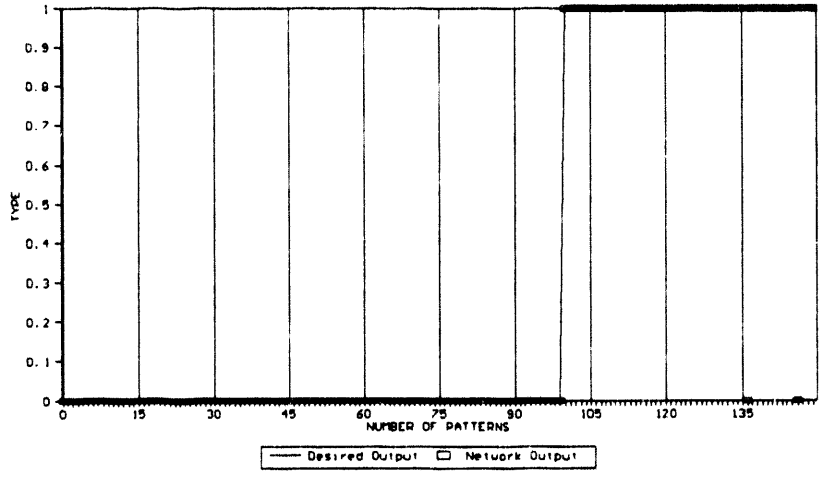


Figure 6.6(a). Output 1 recall result of PNN using CINT testing data

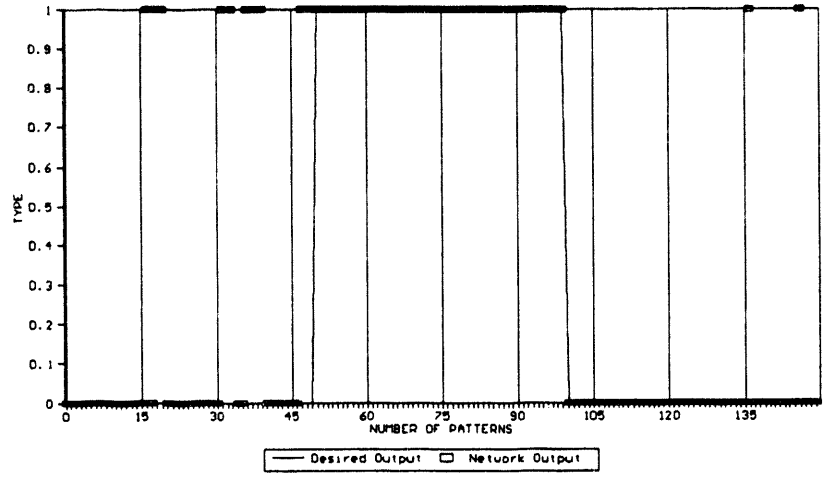


Figure 6.6(b). Output 2 recall result of PNN using CINT testing data

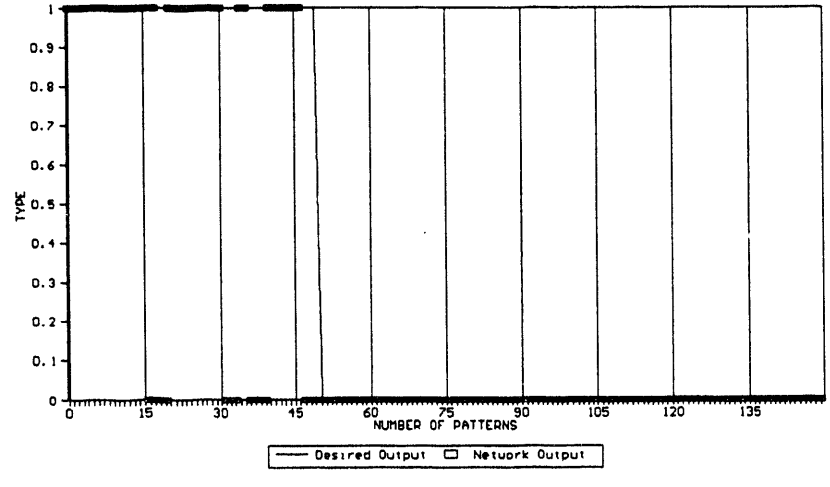


Figure 6.6(c). Output 3 recall result of PNN using CINT testing data

neural networks.

The results indicated that both the BPN and the PNN networks are efficient in classifying tube defect types, using either the CG or the CINT data.

SECTION 7

DEVELOPMENT OF THE EXPERT SYSTEM "EDDYANN"

7.1 Introduction

One of the objectives of the research under this project is to develop a neural network-based expert system. The system developed for the PC platform is called "EDDYANN". The software combines all the analysis into a user-friendly, PC-based diagnostics system. EDDYANN consists of a user interface, a rule base, a knowledge base, and supporting modules. The user interface provides choice of eddy current inspection data file, display of related information, and presentation of data analysis results. The knowledge base consists of trained neural networks for defect type identification and defect parameter estimation. The rule base consists of logical steps for data analysis and rules for decision making. The overall system would perform the analysis of multi-frequency eddy current testing (ECT) data analysis automatically. Figure 7.1 shows the major functional blocks of the EDDYANN expert system. These are

- Multi-frequency, tube inspection ECT data base.
- Data representation.
- Knowledge base.
- Rule base.
- User interface.

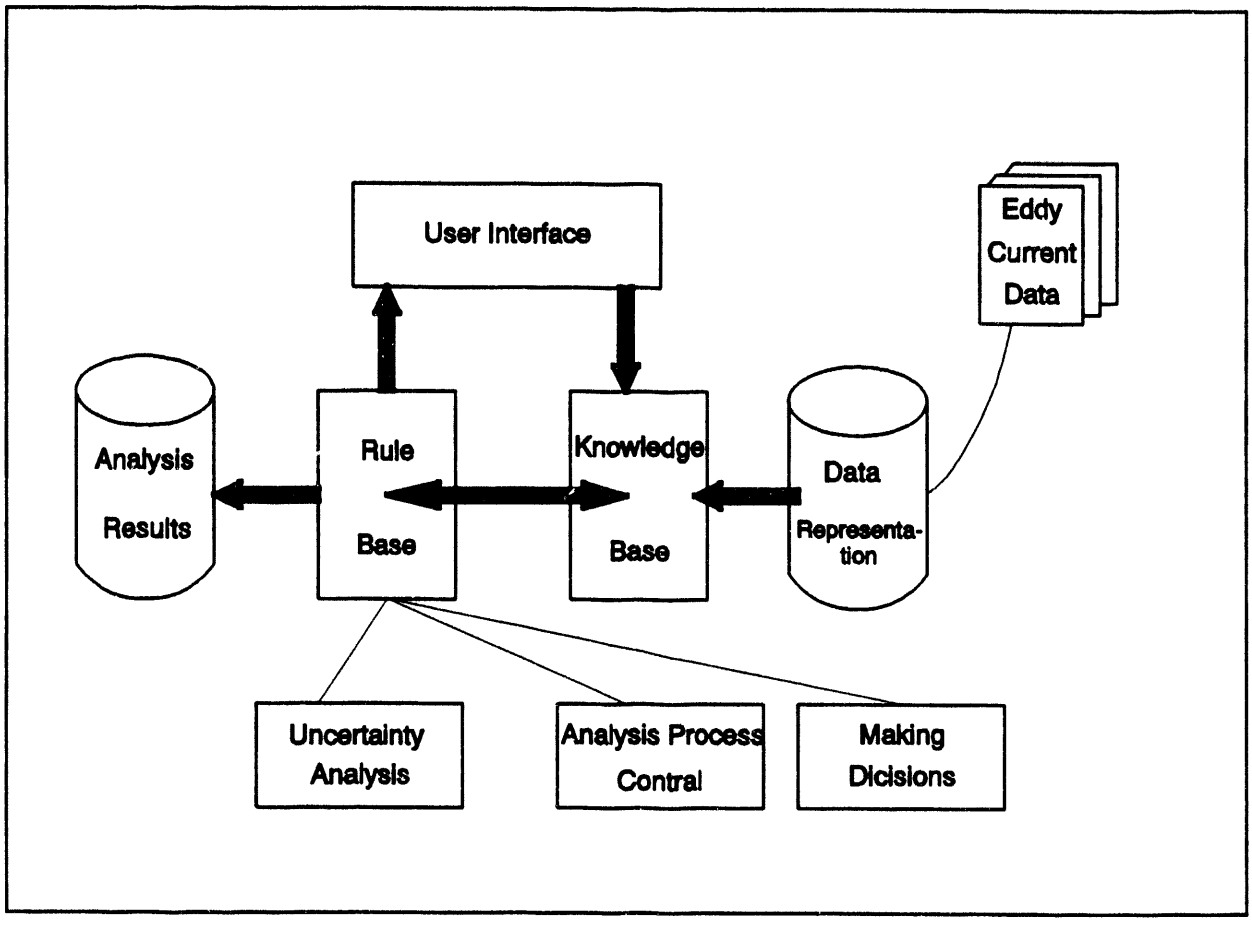


Figure 7.1 Architecture of the EDDYANN expert system

7.2 Knowledge Basc

The knowledge base is the heart of an expert system. EDDYANN uses the trained neural networks as its knowledge base. These trained neural networks embody the knowledge contained in the eddy current (EC) data. When a new set of EC inspection data were passed through these networks, the information about defect type, defect depth, and the distance from defect to artifact is obtained.

The knowledge base of EDDYANN has three levels of neural networks as shown in Figure 7.2. The first level is a network for artifact type identification. This network identifies the artifact type (tube support, ferrite, or copper). The second level contains networks for defect depth estimation for each artifact type. The depth varies from 20 to 100 percent through wall thickness. The third level consists of networks for distance estimation. The distance from a defect (at certain depth) to an artifact is between 0.1 to 2.0 inch.

Since the trained networks from NeuralWorks software cannot be used outside the NeuralWorks platform, it is necessary to convert a trained network into a general C source code. This external code is then used in the EDDYANN knowledge base. One of the accomplishments of this project is to develop such a general C code. This code has two parts: (1) a C source code callable function, and (2) a main program. The C source code callable function is generated using the Flash Code function of the NeuralWorks software. It contains the weight values of the trained network. The main program can read the recall input data file, execute the callable function, and write the recall results into an output file.

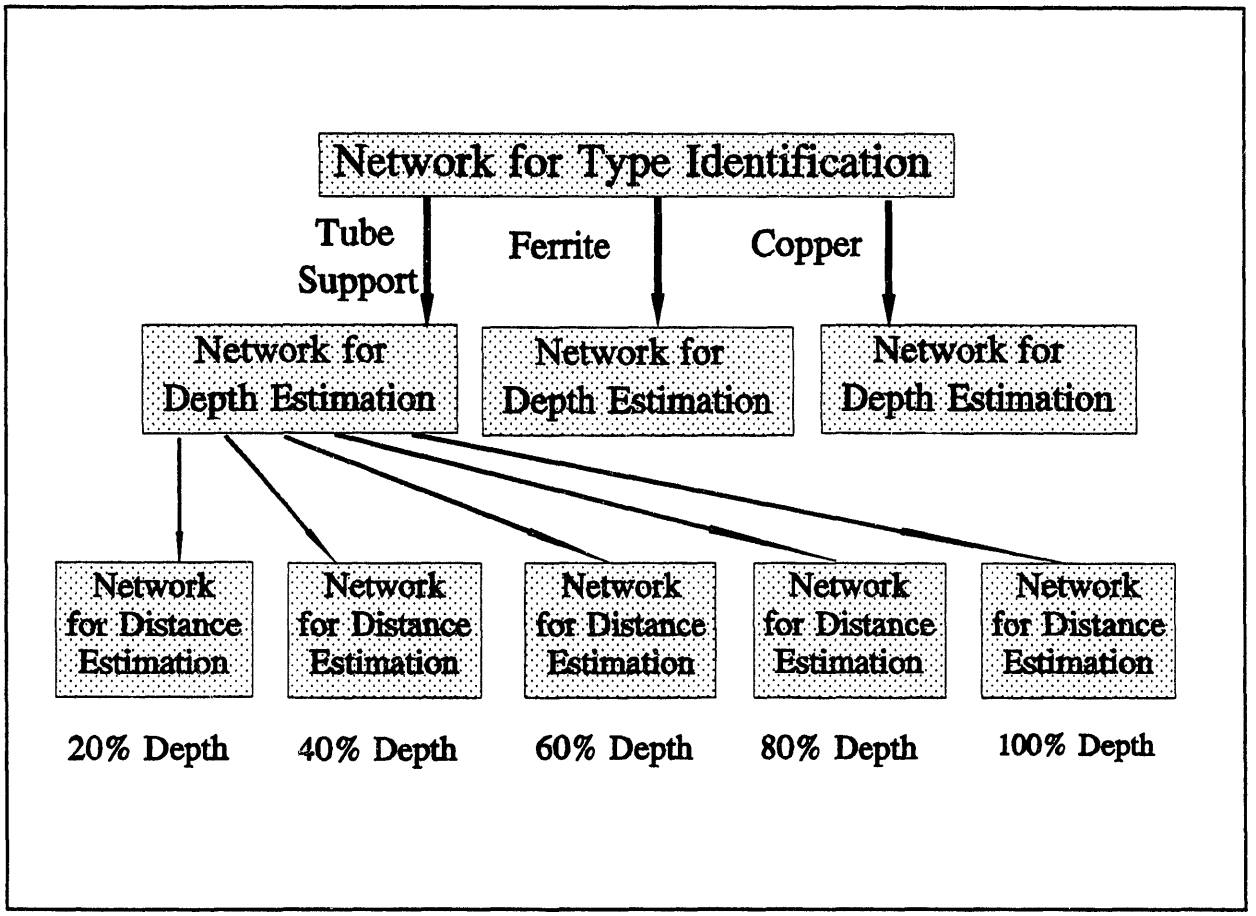


Figure7.2 The structure of neural networks in the knowledge base

7.3 The Expert System Rule Base

The EDDYANN rule base has the rules to control the data analysis process and to make decisions. These rules are written in the "if..., then..." forms. The rules can be loosely classified into two categories: type identification and depth estimation rules.

Artifact Type Identification Rules

Three parameters atype, btype, and ctype are used to describe the artifact types. There are three rules for artifact type identification and data flow.

Rule 1: If atype is larger than 0.5, and btype is smaller than 0.5, and ctype is smaller than 0.5,
Then the artifact type is "Tube Support", and program goes to tube-support category to execute networks for depth estimation.

Rule 2: If btype is larger than 0.5, and atype is smaller than 0.5, and ctype is smaller than 0.5,
Then the artifact type is "Ferrite", and program goes to ferrite category to execute networks for depth estimation.

Rule 3: If ctype is larger than 0.5, and atype is smaller than 0.5, and btype is smaller than 0.5,
Then the artifact type is "Copper", and program goes to copper category to execute networks for depth estimation.

Defect Depth Estimation Rules

The parameter "depth" is used to describe the defect depth. There are five rules for defect depth estimation.

Rule 1: If depth is between 0 and 0.29,
Then the depth is 20%, and program goes to 20% depth category to execute networks for distance estimation.

Rule 2: If depth is between 0.5 and 0.49,
Then the depth is 40%, and program goes to 40% depth category to

execute networks for distance estimation.

Rule 3: If depth is between 0.5 and 0.69,
Then the depth is 60%, and program goes to 60% depth category to execute networks for distance estimation.

Rule 4: If depth is between 0.7 and 0.89,
Then the depth is 80%, and program goes to 20% depth category to execute networks for distance estimation.

Rule 5: If depth is between 0.9 and 1.0,
Then the depth is 100%, and program goes to 100% depth category to execute networks for distance estimation.

7.4 User Interface

The user interface allows the user to enter the EC data file name, displays related information, and summarizes final data analysis results. Figure 7.3 shows the first monitor screen of the program. It has the data input interface and provides some general information. The second screen displays the final results from network recall and the waveforms of each data pattern. This display is shown in Figure 7.4.

7.5 Executing EDDYANN

This report also includes disks containing the EDDYANN expert system program and three test data files. The test data files are : d111.dat, d2310.dat, and d3520.dat.

The executing steps are:

1. Load the program from the system prompt, type "eddyann" and press the "Return" key. Then the first screen will appear.
2. Enter the data file name, for example, "d111.dat", and then press the "Return" key.

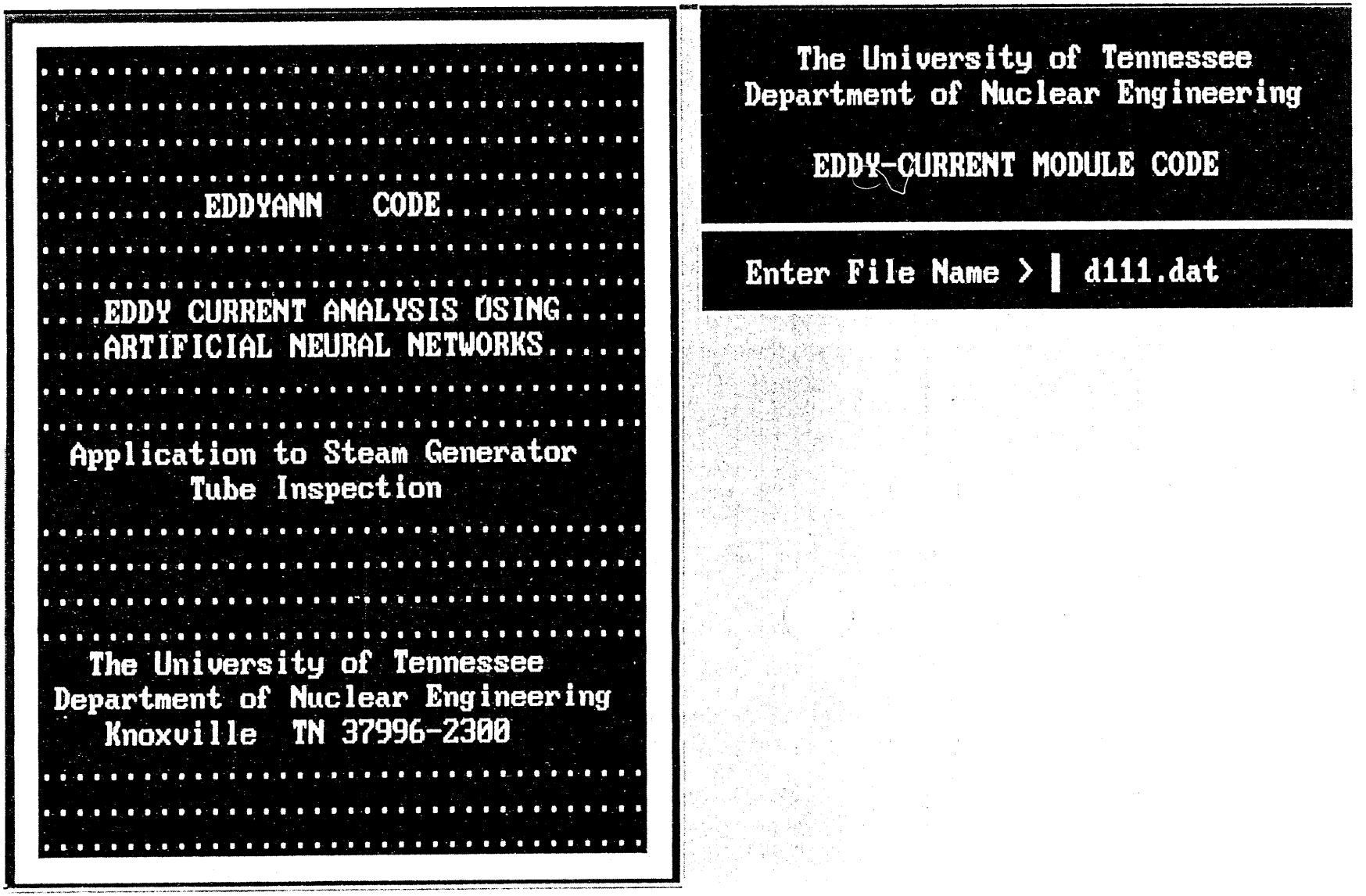
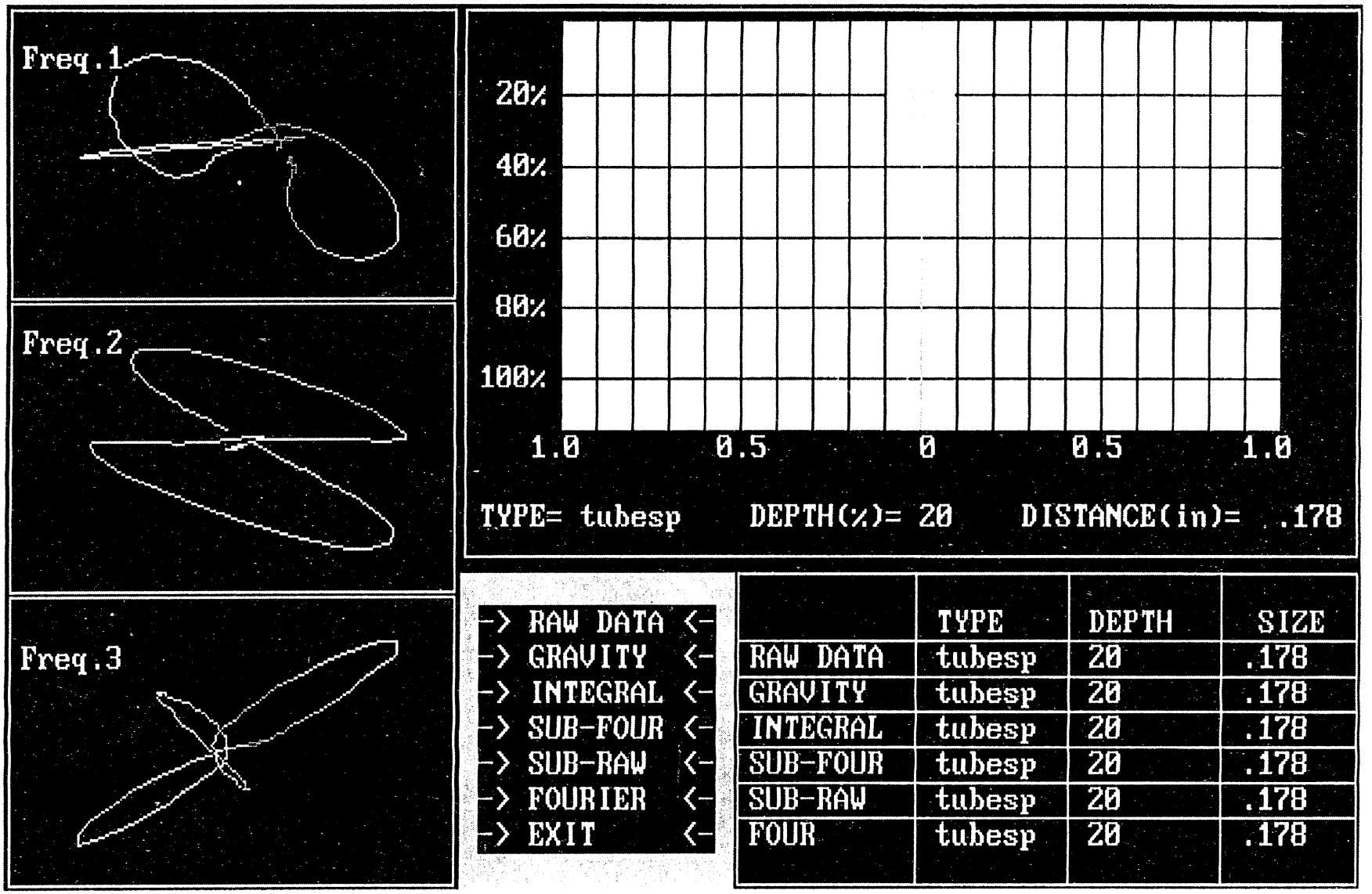


Figure 7.3 The first monitor screen of the EDDYANN system (Enter data file information)



	TYPE	DEPTH	SIZE
RAW DATA	tubesp	20	.178
GRAVITY	tubesp	20	.178
INTEGRAL	tubesp	20	.178
SUB-FOUR	tubesp	20	.178
SUB-RAW	tubesp	20	.178
FOUR	tubesp	20	.178

Figure 7.4 The results and information display screen of the EDDYANN system

After receiving the name of the data file, EDDYANN will perform data compression and representation, form the recall files for network recall, consult the knowledge base, and find the final results. This process will take about 10 to 20 seconds.

SECTION 8

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

8.1 Concluding Remarks

The research performed under this NRC grant demonstrated the effectiveness of integrating digital signal processing and artificial neural networks for automating the diagnostics of defects in plant components. The defect parameters were estimated using modular neural networks. The following is a summary of the significant results achieved under this project.

- Eddy current data representation techniques and signature transformation to minimize the effects of scaling and translation.
- Development of modular, multi-layer perceptron networks for defect type classification and defect parameter estimation.
- Development of a neural network based PC expert system computer code EDDYANN.
- Application of the technology to the analysis of eddy current tube inspection data.

8.2 Recommendations for Future Research

The research and development performed under this NRC University Grant established the feasibility of automating diagnostics of nuclear plant component anomalies using the integration of data processing and artificial neural networks. Further research and development is necessary for the analysis of large NDE database and for increasing the reliability of decision making for anomaly detection and quantification. The following topics are recommended for further research that could result in an industry implementable system.

- Principal component analysis (PCA) for data compression and dimensionality reduction. This is an information preserving transformation.

- Hybrid neural networks estimation and fuzzy-logic uncertainty analysis.

- A PC-based expert system development for managing large databases and for interactive analysis.

- A large-scale testing of the technology using steam generator tube inspection data from commercial power plants. (Data acquisition from EPRI NDE Center)

- Development of detailed guidelines for in-plant implementation.

LIST OF REFERENCES

1. W. E. Deeds and C. V. Dodd, "Eddy Current Inspection of Steam Generator Tubing," Electromagnetic Methods of Nondestructive Testing, Vol. 3 of Nondestructive Testing Monographs and Tracts, W. Lord, Ed., Gordon and Breach, New York, 1985.

2. Y. He and A. Kundu, "2-D Shape Classification Using Hidden Markov Model," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 11, November 1991.

3. Y. Yao et al, "Pattern Recognition by a Distributed Neural Network: An Industrial Application," Neural Networks, Vol. 4, pp 103-121, 1991.

4. C. V. Dodd and W. E. Deeds, "In-Service Inspection of Steam Generator Tubing Using Multiple-Frequency Eddy-Current Techniques," Special Technical Publication, American Society for Testing and Material, Philadelphia, PA., 1981.

5. C. V. Dodd et al, "Three-Frequency Eddy-Current Instrument," Oak Ridge National Laboratory, May 1988.

6. L. Udpa and S.S. Udpa, "Neural Networks for the Classification of Nondestructive Evaluation Signals," IEE Proceedings-F, Vol. 138, No. 1, February 1991.

7. T. Pavlidis, "Algorithms for Shape Analysis of Contours and Waveforms," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 4, July 1980.

8. Y. Liu, B. R. Upadhyaya, and M. Naghedolfeizi, "Chemometric Data Analysis Using Artificial Neural Networks," Applied Spectroscopy, Vol. 47, No. 1, pp 12-23,1993.

9. "Eddy Current Characterization of Materials and Structures," A Symposium sponsored by ASTM, American Society for Testing and Materials, Gaithersburg, MD, 5-7 Sept. 1979.

10. "Eddy Current Nondestructive Testing," U.S. National Bureau of Standards Special Publication 589, 1981.

11. P. E. Mix, "Introduction to Nondestructive Testing," John Wiley, New York, 1987.

12. "Transactions of the Eighteenth Water Reactor Safety Information Meeting," Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, Washington, 1990.

13. C. V. Dodd et al, "An Eddy Current Laboratory Test System Using Commercial Equipment," Oak Ridge National Laboratory, April 1987.

14. E. Persoon, K. S. Fu, "Shape Discrimination Using Fourier Descriptors," IEEE Trans.,

SMC-7, pp 170-179, March 1977.

15. G. H. Granlund, "Fourier Preprocessing for Hand Print Character Recognition," IEEE Trans., C-21, pp 195-201, February 1972.

16. "Using Nworks," NeuralWorks Professional II/PLUS and NeuralWorks Explorer Softerware, Neural Ware, Inc., 1991.

17. "Eddy-Current Characterization of Materials and Structures," American Society for Testing and Materials, 1981.

18. "Eddy Current Nondestructive Testing," NBS Special Publication 589, U.S. Department of Commerce / National Bureau of Standards, 1977. Special Technical Publication, American Society for Testing and Material, Philadelphia, 1981.

19. "Neural Computing," NeuralWorks Professional II/PLUS and Neural Works Explorer Software, NeuralWare, Inc., 1991.

20. "Nondestructive Evaluation Program," EPRI NP-3347-NR, March 1984.

21. R. C. McMaster, P. McIntire, and M. L. Mester (Eds), "Nondestructive Testing Handbook," Vol. 4 (Electromagnetic Testing), Am. Soc. for Nondestructive Testing, 1987.

22. D. F. Specht, "Probalilistic Neural Networks," Neural Networks, Vol. 3, No. 1, pp 109-118, Jan. 1990.

23. B. R. Upadhyaya, E. Eryurek, "Application of Neural Networks for Sensor Validation and Plant Monitoring," Nuclear Technology, Vol. 27, No. 2, pp 170-176, 1992.

24. R. P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, Vol. 4, No. 2, pp 4-22, April 1987.

25. D. Rumelhart and J. McClelland, Paralled Distributed Processing, Vol. 2, Bradford Books/MIT Press, Cambridge, MA, 1986.

26. D. F. Specht, "Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification," IEEE Trans. Neural Networks, Vol. 1, No. 1, pp 111-121, March 1990.

27. W. Yan and B. R. Upadhyaya, "Hybrid Digital Signal Processing and Neural Networks for Automated Diagnostics Using Eddy Current Inspection," Proc. Intl. Conf. Artificial Neural Networks, ICANN '93, Amsterdam, The Netherlands, pp 799-804, September 1993.

28. E. E. Kriezis et al, "Eddy Current: Theory and Applications," Proceedings of IEEE, Vol. 80, No. 10, October 1992.

29. P. C. Riccardella, A. F. Deardorff, and Timothy J. Griesbach, "Fatigue Lifetime Monitoring in Power Plants," Advances in Fatigue Lifetime Predictive Techniques, ASTM STP 1122, American Society for Testing and Materials, Philadelphia, pp 460-473, 1992.

30. A. S. Krausz and K. Krausz, "Fracture Kinetics of Crack Growth," Kluwer Academic Publishers, The Netherlands, 1988.

31. K. Hellan, "Introduction to Fracture Mechanics," McGraw-Hill, New York, 1984.

32. J. C. Newman, Jr., E. P. Phillips, M. H. Swain, and R. A. Everett, Jr., "Fatigue Mechanics: An Assessment of a Unified Approach to Life Prediction," Advances in Fatigue Lifetime Predictive Techniques, ASTM STP 1122, American Society for Testing and Materials, Philadelphia, pp 5-27, 1992.

33. P. K. Liaw, A. Saxena, and J. Schaffer, "Estimating Remaining Life of Elevated Temperature Steam Pipes. Part I and Part II," Engineering Fract. Mech., Vol. 32, No. 5, pp 675-722, 1989.

34. V. M. Radhakrishnan, M. Kamaraj, and V. V. Balasubramaniam, "Life Estimation of Cracked Stainless Steel Components Under Creep Conditions," Engineering Materials and Technology, Vol. 113, pp 303-306, July 1991.

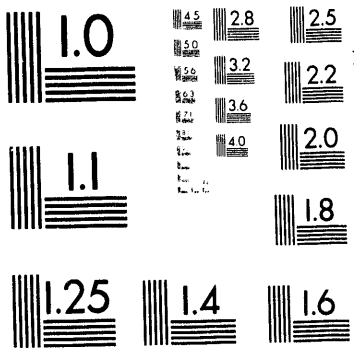
35. H. R. Jhansale and D. R. McCann, "Fatigue Analysis Techniques for Vintage Steam Turbine/Generator Components," Advances in Fatigue Lifetime Predictive Techniques, ASTM STP 1122, American Society for Testing and Materials, Philadelphia, pp 474-489, 1992.

36. P. K. Simpson, "Artificial Neural Systems," Pergamonn Press, New York, 1990.

37. R. E. Uhrig et al, "Application of Neural Networks to Determine the Operability of Check Valves," Final Report prepared for Electric Power Research Institute, EPRI RP-8010-12, 1993.

APPENDIX A

EDDYANN COMPUTER CODE


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1

2 of 2

```
c***********************************************c
c                                               c
c            EDDYANN.for PROGRAM                c
c                                               c
c            The University of Tennessee        c
c            Nuclear Engineering Department     c
c            Knoxville TN 37996-2300            c
c            September 1993                     c
c                                               c
c***********************************************c

      interface to integer*2 function cur_rd [c] ()
      end

      interface to integer*2 function kb_chk [c] ()
      end

      interface to integer*2 function system [c]
     +  (string[reference])
      character*1 string
      end

      include 'fgraph.fi'
      include 'fgraph.fd'

      character*1 y,sy
      character*10 fname,fname1,fname2,fname3,fname4,fname5
      character*10 fname6,g6,mtype
      character*10 g14
      character*5 g5
      character*3 g3
      character*25 type
      character*40 adummy
      integer*2 dummy,cur_rd,ktus,kb_chk,inum
      integer*4 dummy4
      integer idepth,x1(1000),y1(1000),x2(1000),y2(1000)
      integer x3(1000),y3(1000),jtype
      integer x11(50),y11(50),x22(50),y22(50),x33(50),y33(50)
      real xr1(1000),xr2(1000),xr3(1000),yr1(1000),yr2(1000)
      real yr3(1000)
      integer*2 i,system
      character*6 mt(6)
      integer md(6)
      real as(6)

      record /rccoord/ s
      dummy=setvideomode($ERESCOLOR)
```

```
10      idum=0
        do j=1,1000
        x1(j)=99999
        y1(j)=99999
        x2(j)=99999
        y1(j)=99999
        x3(j)=99999
        y3(j)=99999
        end do
c--Environment--first screen-----

        dummy4=setbkcolor($BLUE)

        dummy=setcolor(8)
        dummy=rectangle($gfillinterior,0,0,640,350)

c--information box on the left--

        dummy=setcolor(1)
        dummy=rectangle($gfillinterior,0,0,320,350)
        dummy=setcolor(9)
        dummy=rectangle($gfillinterior,5,5,315,345)
        dummy=setcolor(15)
        dummy=rectangle($gborder,0,0,320,348)

        dummy=settextcolor(14)
        call settextposition(2,3,s)
        call outtext('.....')
        call settextposition(3,3,s)
        call outtext('.....')
        call settextposition(4,3,s)
        call outtext('.....')
        call settextposition(5,3,s)
        call outtext('.....')
        call settextposition(6,3,s)
        call outtext('.....EDDYANN  CODE.....')
        call settextposition(7,3,s)
        call outtext('.....')
        call settextposition(8,3,s)
        call outtext('.....')
        call settextposition(9,3,s)
        call outtext('....EDDY CURRENT ANALYSIS USING.....')
        call settextposition(10,3,s)
        call outtext('....ARTIFICIAL NEURAL NETWORKS.....')
        call settextposition(11,3,s)
        call outtext('.....')
        call settextposition(12,3,s)
```

```
      call outtext('.....')
      call settextposition(13,3,s)
      call outtext(' Application to Steam Generator  ')
      call settextposition(14,3,s)
      call outtext('      Tube Inspection      ')
      call settextposition(15,3,s)
      call outtext('.....')
      call settextposition(16,3,s)
      call outtext('.....')
      call settextposition(17,3,s)
      call outtext('.....')
      call settextposition(18,3,s)
      call outtext('.....')
      dummy=settextcolor(11)
      call settextposition(19,3,s)
      call outtext(' The University of Tennessee  ')
      call settextposition(20,3,s)
      call outtext(' Department of Nuclear Engineering ')
     dummy=settextcolor(10)
      call settextposition(21,3,s)
      call outtext('  Knoxville TN 37996-2300  ')
      call settextposition(22,3,s)
      call outtext('.....')
      call settextposition(23,3,s)
      call outtext('.....')
      call settextposition(24,3,s)
      call outtext('.....')

c--DRAW FANCY STAFF --

c----connections----------

      ix=435
      iy=190
      dummy=setcolor(11)
      call moveto(ix,iy,s)
      dummy=lineto(ix-50,iy+50)

      call moveto(ix,iy,s)
      dummy=lineto(ix+50,iy+50)

      call moveto(ix,iy,s)
      dummy=lineto(ix+150,iy+50)
c--
      ix=535
      iy=190
      call moveto(ix,iy,s)
```

```
      dummy=lineto(ix-150,iy+50)

      call moveto(ix,iy,s)
      dummy=lineto(ix-50,iy+50)

      call moveto(ix,iy,s)
      dummy=lineto(ix+50,iy+50)
c-----
      ix=435
      iy=290
      dummy=setcolor(11)
      call moveto(ix,iy,s)
      dummy=lineto(ix-50,iy-50)

      call moveto(ix,iy,s)
      dummy=lineto(ix+50,iy-50)

      call moveto(ix,iy,s)
      dummy=lineto(ix+150,iy-50)
c--
      ix=535
      iy=290
      call moveto(ix,iy,s)
      dummy=lineto(ix-150,iy-50)

      call moveto(ix,iy,s)
      dummy=lineto(ix-50,iy-50)

      call moveto(ix,iy,s)
      dummy=lineto(ix+50,iy-50)

c--arrows-----

      call moveto(435,180,s)
      dummy=lineto(435,140)
      call moveto(435,140,s)
      dummy=lineto(430,145)
      call moveto(435,140,s)
      dummy=lineto(440,145)


      call moveto(535,180,s)
      dummy=lineto(535,140)
      call moveto(535,140,s)
      dummy=lineto(530,145)
      call moveto(535,140,s)
      dummy=lineto(540,145)
```

```
        call moveto(435,300,s)
        dummy=lineto(435,330)
        call moveto(435,310,s)
        dummy=lineto(430,315)
        call moveto(435,310,s)
        dummy=lineto(440,315)

        call moveto(535,300,s)
        dummy=lineto(535,330)
        call moveto(535,310,s)
        dummy=lineto(530,315)
        call moveto(535,310,s)
        dummy=lineto(540,315)

c-------ellipses----------

        ix=420
        iy=180

        dummy=setcolor(13)
        dummy=ellipse($gfillinterior,ix,iy,ix+30,iy+22)
        dummy=ellipse($gfillinterior,ix+100,iy,ix+130,iy+22)

        iy=230
        dummy=ellipse($gfillinterior,ix-50,iy,ix-20,iy+22)
        dummy=ellipse($gfillinterior,ix+50,iy,ix+80,iy+22)
        dummy=ellipse($gfillinterior,ix+150,iy,ix+180,iy+22)

        iy=280

        dummy=ellipse($gfillinterior,ix,iy,ix+30,iy+22)
        dummy=ellipse($gfillinterior,ix+100,iy,ix+130,iy+22)

        iy=180
        dummy=setcolor(15)
        dummy=ellipse($gborder,ix,iy,ix+30,iy+22)
        dummy=ellipse($gborder,ix+100,iy,ix+130,iy+22)

        iy=230
        dummy=ellipse($gborder,ix-50,iy,ix-20,iy+22)
        dummy=ellipse($gborder,ix+50,iy,ix+80,iy+22)
        dummy=ellipse($gborder,ix+150,iy,ix+180,iy+22)

        iy=280

        dummy=ellipse($gborder,ix,iy,ix+30,iy+22)
        dummy=ellipse($gborder,ix+100,iy,ix+130,iy+22)
```

```
      dummy=setcolor(1)
      dummy=rectangle($gfillinterior,330,0,639,85)

      dummy=setcolor(9)
      dummy=rectangle($gborder,330,0,639,85)

c--LABELS----

      dummy=settextcolor(11)
      call settextposition(2,45,s)
      call outtext('  The University of Tennessee  ')
      call settextposition(3,45,s)
      call outtext('Department of Nuclear Engineering')
      dummy=settextcolor(15)
      dummy=settextcolor(14)
      call settextposition(5,45,s)
      call outtext('  EDDY-CURRENT MODULE CODE    ')

      dummy=setcolor(1)
      dummy=rectangle($gfillinterior,330,87,639,120)

      dummy=setcolor(13)
      dummy=rectangle($gborder,330,87,639,120)

      dummy=settextcolor(13)
      call settextposition(8,45,s)
      call outtext('Enter File Name >              ')

      fname="x        "
      y=" "

c--------keyboard check ----------------------------
      k=0
20    ktus=kb_chk()
      idelay=idelay+1

c--key hit----------------------
        if(ktus.eq.1) then
            ktus=cur_rd()

c---exit----
            if(ktus.eq.27) go to 10000

c---enter----

      if((ktus.eq.13).and.(k.eq.0)) go to 20
```

```
        if((ktus.eq.13).and.(k.eq.1)) then
        k=0
        go to 20
        end if

                if(ktus.eq.13) then
                k=0
                go to 30
                end if

c--backspace---
                if(ktus.eq.8) then
                fname="          "
                dummy=settextcolor(13)
                call settextposition(8,45,s)
                call outtext('Enter File Name >              ')
                y=" "
                k=0
                go to 20
                end if

                k=k+1
                y=char(ktus)
                dummy=settextcolor(15)
                call settextposition(8,64+k,s)
                write(sy,'(a1\)') y
                call outtext(sy)

                fname(k:k)=y
         end if
c---key hit end---------

c--blink---
        if(idelay.eq.500) then
        dummy=setcolor(15)
        dummy=rectangle($gfillinterior,496,98,499,112)
        end if

        if(idelay.eq.1000) then
        dummy=setcolor(8)
        dummy=rectangle($gfillinterior,496,98,499,112)
        idelay=0
        end if

        go to 20

        I=SYSTEM('del *.cog'C)
```

```
      I=SYSTEM('del *.int'C)
      I=SYSTEM('del *.frd'C)
      I=SYSTEM('del compress.dat'C)
      I=SYSTEM('del *.txt'C)
      I=SYSTEM('del result.dat'C)
      I=SYSTEM('del result1.dat'C)
      I=SYSTEM('del size1.dat'C)
      I=SYSTEM('del size2.dat'C)
      I=SYSTEM('del size3.dat'C)
      I=SYSTEM('del cog1.dat'C)
      I=SYSTEM('del cog2.dat'C)
      I=SYSTEM('del cog3.dat'C)

c--open-read data files-------

30    open(100,file=fname,status='old',err=70066)

      read(100,*)
      read(100,'(i2)',err=7001) jtype
      read(100,*)
      read(100,'(i4)',err=7001) idepth
      read(100,*)
      read(100,'(f6.4)',err=7001) size

      m=1
50    read(100,'(6i7)',end=200,err=7001)
     + x1(m),y1(m),x2(m),y2(m),x3(m),y3(m)
      m=m+1
      go to 50
200   close(100)

      open(10,file='name.txt',status='unknown')
      write(10,'(i2)') jtype
      write(10,'(i4)') idepth
      close(10)

      m=m-1
      go to 2002

c---cannot find file---

70066       fname="          "
            dummy=settextcolor(13)
            call settextposition(8,45,s)
            call outtext('Enter File Name >              ')
            y="x"
            k=0
```

```
                go to 20

c2002    inum=1
c        call child(inum)
c
c        inum=2
c        call child(inum)
c
c        inum=3
c        call child(inum)


c--draw new screen
2002     dummy=setcolor(0)
         dummy=rectangle($gfillinterior,0,0,640,350)

         dummy=setcolor(0)
         dummy=rectangle($gfillinterior,0,0,213,350)

c-- FILE STRUCTURE ------
c
c 1-fname1  xxx.dat  raw data
c 2-fname2  xxx.cog  center of gravity data
c 3-fname3  xxx.int  integral data
c 4-fname4  xxx.sfd  substracted Fourier Desc.
c 5-fname5  xxx.srd  substracted raw data
c 6-fname6  xxx.frd  fourier descriptor
c----------------------------------------

c--create filenames --
c- file1
         do j=1,6
           if(fname(j:j).eq.".") then
         fname1(j:j)=fname(j:j)
           jkeep=j+1
           go to 9000
           end if
         fname1(j:j)=fname(j:j)
         end do

9000       fname1(jkeep:jkeep)="d"
           jkeep=jkeep+1
           fname1(jkeep:jkeep)="a"
           jkeep=jkeep+1
           fname1(jkeep:jkeep)="t"
           jkeep=0
c--file2
```

```
        do j=1,6
          if(fname(j:j).eq.".") then
         fname2(j:j)=fname(j:j)
          jkeep=j+1
          go to 9001
          end if
        fname2(j:j)=fname(j:j)
        end do

9001      fname2(jkeep:jkeep)="c"
          jkeep=jkeep+1
          fname2(jkeep:jkeep)="o"
          jkeep=jkeep+1
          fname2(jkeep:jkeep)="g"
          jkeep=0

c--file3

        do j=1,6
          if(fname(j:j).eq.".") then
         fname3(j:j)=fname(j:j)
          jkeep=j+1
          go to 9003
          end if
        fname3(j:j)=fname(j:j)
        end do

9003      fname3(jkeep:jkeep)="i"
          jkeep=jkeep+1
          fname3(jkeep:jkeep)="n"
          jkeep=jkeep+1
          fname3(jkeep:jkeep)="t"
          jkeep=0
c--file4

        do j=1,6
          if(fname(j:j).eq.".") then
          fname4(j:j)=fname(j:j)
          jkeep=j+1
          go to 9004
          end if
          fname4(j:j)=fname(j:j)
        end do

9004      fname4(jkeep:jkeep)="s"
          jkeep=jkeep+1
          fname4(jkeep:jkeep)="f"
```

```
          jkeep=jkeep+1
          fname4(jkeep:jkeep)="d"
          jkeep=0
c--file5

        do j=1,6
          if(fname(j:j).eq.".") then
          fname5(j:j)=fname(j:j)
          jkeep=j+1
          go to 9005
          end if
          fname5(j:j)=fname(j:j)
        end do

9005      fname5(jkeep:jkeep)="s"
          jkeep=jkeep+1
          fname5(jkeep:jkeep)="r"
          jkeep=jkeep+1
          fname5(jkeep:jkeep)="d"
          jkeep=0

c--file6
        do j=1,6
          if(fname(j:j).eq.".") then
          fname(j:j)=fname(j:j)
          jkeep=j+1
          go to 9006
          end if
          fname6=fname
        end do
c
9006      fname6(jkeep:jkeep)="f"
          jkeep=jkeep+1
          fname6(jkeep:jkeep)="r"
          jkeep=jkeep+1
          fname6(jkeep:jkeep)="d"
          jkeep=0
c-------------------------

        open(500,file='name1.txt',status='unknown')
        write(500,'(a10)') fname2
        close(500)

c-+++++ Data Representation +++++++-
c
    call four(x1,y1,x2,y2,x3,y3,m,fname6)
c
```

```
      call compress(x1,y1,x2,y2,x3,y3,m,x11,y11,x22,y22,x33,y33)
c
      call cg(x11,y11,x22,y22,x33,y33,fname2)
c
      call int(x11,y11,x22,y22,x33,y33,fname3)


         I=SYSTEM('CONVERT.EXE'C)

c-+++++ calculate results-------

         I=SYSTEM('ANALYSIS.EXE'C)

c-------------------------------

         iselect=1

         dummy=setcolor(2)
         dummy=rectangle($gfillinterior,215,0,640,220)
         dummy=setcolor(15)
         dummy=rectangle($gborder,217,2,638,218)

         dummy=setcolor(14)
         dummy=rectangle($gfillinterior,262,7,602,175)

c--grids
         dummy=setcolor(0)
         iy=35
         do k=1,5
         call moveto(252,iy,s)
         dummy=lineto(602,iy)
         iy=iy+28
         end do

         ix=262
         do k=1,21
         call moveto(ix,180,s)
         if(k.eq.11) dummy=setcolor(12)
         if(k.gt.11) dummy=setcolor(0)
         dummy=lineto(ix,7)
         ix=ix+17
         end do

         dummy=settextcolor(15)
         call settextposition(3,30,s)
         call outtext('20%')
         call settextposition(5,30,s)
```

```
      call outtext('40%')
      call settextposition(7,30,s)
      call outtext('60%')
      call settextposition(9,30,s)
      call outtext('80%')
      call settextposition(11,29,s)
      call outtext('100%')

      call settextposition(13,32,s)
      call outtext('1.0      0.5      0      0.5      1.0')
      call settextposition(15,29,s)
      call outtext
     & ('TYPE=          DEPTH(%)=      DISTANCE(in)=      ')


c--read from results file

      open(350,file='result.dat',status='unknown')
      read(350,'(f5.3)') asize
      read(350,'(i3)') mdepth
      read(350,'(a10)') mtype
      close(350)

      dummy=settextcolor(14)
      call settextposition(15,35,s)
      write(g14,'(a10)') mtype
      call outtext(g14)

      call settextposition(15,54,s)
      write(g3,'(i3)') mdepth
      call outtext(g3)

      call settextposition(15,75,s)
      write(g5,'(f5.3)') asize
      call outtext(g5)

c--draw-defect

      asize=asize*170./2.0
      ix1=432-asize
      ix2=432+asize
      iy1=28*mdepth/20-7
      iy2=iy1+28
      dummy=setcolor(12)
      dummy=rectangle($gfillinterior,ix1,iy1,ix2,iy2)

c--message box----
```

```
        dummy=setcolor(15)
        dummy=rectangle($gborder,346,225,638,348)

        dummy=settextcolor(10)
        call settextposition(18,55,s)
        call outtext(' TYPE     DEPTH     SIZE')

        dummy=settextcolor(11)
        call settextposition(19,45,s)
        call outtext('RAW DATA')
        call settextposition(20,45,s)
        call outtext('GRAVITY ')
        call settextposition(21,45,s)
        call outtext('INTEGRAL')
        call settextposition(22,45,s)
        call outtext('SUB-FOUR')
        call settextposition(23,45,s)
        call outtext('SUB-RAW ')
        call settextposition(24,45,s)
        call outtext('FOUR    ')


c- read
        open(360,file='result1.dat',status='unknown')
        do j=1,6
        read(360,'(a6,i3,f5.3)') mt(j),md(j),as(j)
        end do
        close(360)


        dummy=settextcolor(14)
        do k=1,6
        call settextposition(18+k,56,s)
        write(g6,'(a6)') mt(k)
        call outtext(g6)
        end do

        do k=1,6
        call settextposition(18+k,64,s)
        write(g3,'(i3)') md(k)
        call outtext(g3)
        end do

        do k=1,6
        call settextposition(18+k,73,s)
        write(g5,'(f5.3)') as(k)
        call outtext(g5)
```

```
        end do

        dummy=setcolor(15)
        iyi=0
        do k=1,6
        call moveto(346,252+iyi,s)
        dummy=lineto(638,252+iyi)
        iyi=iyi+14
        end do

        ixi=0
        do j=1,3
        call moveto(430+ixi,225,s)
        dummy=lineto(430+ixi,348)
        ixi=ixi+72
        end do

c-----control box

     dummy=setcolor(11)
     dummy=rectangle($gfillinterior,215,225,344,350)

        dummy=settextcolor(12)
        call settextposition(18,29,s)
        call outtext('-> RAW DATA <-')

        dummy=settextcolor(11)
        call settextposition(19,29,s)
        call outtext('-> GRAVITY  <-')

        call settextposition(20,29,s)
        call outtext('-> INTEGRAL <-')

        call settextposition(21,29,s)
        call outtext('-> SUB.FOUR <-')

        call settextposition(22,29,s)
        call outtext('-> SUB.RAW  <-')

        call settextposition(23,29,s)
        call outtext('-> FOURIER  <-')

        call settextposition(24,29,s)
        call outtext('-> EXIT     <-')

c-- H E A D -------
```

```
10011    ktus=kb_chk()
         if(ktus.eq.1) then
         ktus=cur_rd()

         if(ktus.eq.72) iselect=iselect-1
         if(ktus.eq.80) iselect=iselect+1
         if(iselect.lt.1) iselect=7
         if(iselect.gt.7) iselect=1

        dummy=settextcolor(11)
        if(iselect.eq.1) dummy=settextcolor(12)
        call settextposition(18,29,s)
        call outtext('-> RAW DATA <-')

        dummy=settextcolor(11)
        if(iselect.eq.2) dummy=settextcolor(12)
        call settextposition(19,29,s)
        call outtext('-> GRAVITY  <-')

        dummy=settextcolor(11)
        if(iselect.eq.3) dummy=settextcolor(12)
        call settextposition(20,29,s)
        call outtext('-> INTEGRAL <-')

        dummy=settextcolor(11)
        if(iselect.eq.4) dummy=settextcolor(12)
        call settextposition(21,29,s)
        call outtext('-> SUB-FOUR <-')

        dummy=settextcolor(11)
        if(iselect.eq.5) dummy=settextcolor(12)
        call settextposition(22,29,s)
        call outtext('-> SUB-RAW  <-')

        dummy=settextcolor(11)
        if(iselect.eq.6) dummy=settextcolor(12)
        call settextposition(23,29,s)
        call outtext('-> FOURIER  <-')

        dummy=settextcolor(11)
        if(iselect.eq.7) dummy=settextcolor(12)
        call settextposition(24,29,s)
        call outtext('-> EXIT     <-')

c--draw the selected data---

     if((ktus.eq.13).and.(iselect.eq.7)) go to 10009
```

```
      if((ktus.eq.13).and.(iselect.eq.1)) go to 20001
      if((ktus.eq.13).and.(iselect.eq.2)) go to 20002
      if((ktus.eq.13).and.(iselect.eq.3)) go to 20003
      if((ktus.eq.13).and.(iselect.eq.4)) go to 20004
      if((ktus.eq.13).and.(iselect.eq.5)) go to 20005
      if((ktus.eq.13).and.(iselect.eq.6)) go to 20006
      go to 6009

c-----------------------------------------
c--EXIT TO MAIN MENU---

10009  ik=0
       go to 10

c--draw fname1--
20001  open(300,file=fname1,status='old',err=7000)
       do k=1,7
       read(300,'(a40)',err=7001) adummy
       end do

       m1=0
       do j=1,1000
       m1=m1+1
       read(300,'(6i7)',end=6001) x1(j),y1(j),x2(j),y2(j),
   &  x3(j),y3(j)
       end do
6001   m1=m1-1
       close(300)
       call draw(x1,x2,x3,y1,y2,y3,m1)
       ktus=0
       go to 10011

c--draw fname2----------------------------------------

20002  open(300,file=fname2,status='old',err=7000)
       m1=0
       do j=1,100
       m1=m1+1
       read(300,'(6f14.6)',end=6002,err=7001)
   &  xr1(j),yr1(j),xr2(j),yr2(j),xr3(j),yr3(j)
       end do
6002   m1=m1-1
       close(300)
       call drawr(xr1,xr2,xr3,yr1,yr2,yr3,m1)
       ktus=0
       go to 10011
```

```
c--draw fname3--------------------------------------

20003   open(300,file=fname3,status='old',err=7000)
        m1=0
        do j=1,100
        m1=m1+1
        read(300,'(6i7)',end=6003,err=7001) x1(j),y1(j),x2(j),y2(j),
    &   x3(j),y3(j)
        end do
6003    m1=m1-1
        close(300)
        call draw(x1,x2,x3,y1,y2,y3,m1)
        ktus=0
        go to 10011

c--draw fname4--------------------------------------

20004   open(300,file=fname4,status='old',err=7000)
        m1=0
        do j=1,100
        m1=m1+1
        read(300,'(6f14.6)',end=6004,err=7001)
    &   xr1(j),yr1(j),xr2(j),yr2(j),
    &   xr3(j),yr3(j)
        end do
6004    m1=m1-1
        close(300)
        call drawr(xr1,xr2,xr3,yr1,yr2,yr3,m1)
        ktus=0
        go to 10011

c--draw fname5--------------------------------------

20005   open(300,file=fname5,status='old',err=7000)
        m1=0
        do j=1,100
        m1=m1+1
        read(300,'(6f12.5)',end=6005,err=7001)
    &   xr1(j),yr1(j),xr2(j),yr2(j),
    &   xr3(j),yr3(j)
        end do
6005    m1=m1-1
        close(300)
        call drawr(xr1,xr2,xr3,yr1,yr2,yr3,m1)
        ktus=0
        go to 10011
```

```
c--draw fname6----------------------------------------

20006   open(300,file=fname6,status='old',err=7000)
        m1=0
        do j=1,100
        m1=m1+1
        read(300,'(6f14.6)',end=6006,err=7001)
     &  xr1(j),yr1(j),xr2(j),yr2(j),xr3(j),yr3(j)
        end do
6006    m1=m1-1
        close(300)
        call drawr(xr1,xr2,xr3,yr1,yr2,yr3,m1)
        ktus=0
        go to 10011

6009    ktus=0
        end if

        go to 10011

c---------------------GO BACK TO THE CONTROL BOX----

c--ERRORS----

7000    dummy=settextcolor(14)

        call settextposition(14,2,s)
        call outtext('                    ')
        call settextposition(15,2,s)
        call outtext('CANNOT OPEN DATA FILE !')
        call settextposition(16,2,s)
        call outtext('                    ')
        call bello()
        call settextposition(17,2,s)
        call outtext('Hit RETURN to Continue ')
        call settextposition(18,2,s)
        call outtext('                    ')
        read(*,*)
        go to 10000

7001    dummy=settextcolor(14)
        call settextposition(14,2,s)
        call outtext('                    ')
        call settextposition(15,2,s)
        call outtext('CANNOT READ DATA FILE !')
        call settextposition(16,2,s)
        call outtext('                    ')
```

```
      call bello()
      call settextposition(17,2,s)
      call outtext('Hit RETURN to Continue ')
      call settextposition(18,2,s)
      call outtext('                    ')
      read(*,*)
      go to 10

10000 idum=0
      dummy=setvideomode($defaultmode)
      end

cXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

      subroutine draw(x1,x2,x3,y1,y2,y3,m)

      include 'fgraph.fd'
      integer x1(1000),x2(1000),x3(1000),y1(1000),y2(1000),y3(1000)
      integer*2 dx1,dy1,dx2,dy2,dx3,dy3
      real x1inc,x2inc,x3inc,y1inc,y2inc,y3inc,size
      integer x1max,x2max,x3max,y1max,y2max,y3max
      integer x1min,x2min,x3min,y1min,y2min,y3min
      integer*2 dummy

      record /rccoord/ s

      dummy=setcolor(1)
      dummy=rectangle($gfillinterior,2,2,212,348)
      dummy=setcolor(15)
      dummy=rectangle($gborder,2,2,212,115)
      dummy=rectangle($gborder,2,117,212,232)
      dummy=rectangle($gborder,2,234,212,348)

      dummy=settextcolor(11)
      call settextposition(2,2,s)
      call outtext('Freq.1')

      dummy=settextcolor(11)
      call settextposition(10,2,s)
      call outtext('Freq.2')

      dummy=settextcolor(11)
      call settextposition(19,2,s)
      call outtext('Freq.3')


      x1max=x1(1)
```

```
      x1min=x1(1)
      x2max=x2(1)
      x2min=x2(1)
      x3max=x3(1)
      x3min=x3(1)

      y1max=y1(1)
      y1min=y1(1)
      y2max=y2(1)
      y2min=y2(1)
      y3max=y3(1)
      y3min=y3(1)

      do j=2,m
      x1max=max(x1(j),x1max)
      x1min=min(x1(j),x1min)
      x2max=max(x2(j),x2max)
      x2min=min(x2(j),x2min)
      x3max=max(x3(j),x3max)
      x3min=min(x3(j),x3min)
      y1max=max(y1(j),y1max)
      y1min=min(y1(j),y1min)
      y2max=max(y2(j),y2max)
      y2min=min(y2(j),y2min)
      y3max=max(y3(j),y3max)
      y3min=min(y3(j),y3min)

      end do

c--increments

      y1inc=80./(y1max-y1min)
      y2inc=80./(y2max-y2min)
      y3inc=80./(y3max-y3min)
      x1inc=150./(x1max-x1min)
      x2inc=150./(x2max-x2min)
      x3inc=150./(x3max-x3min)

c-first drawing------

      dummy=setcolor(14)
      do j=1,m-1
      dy1=100-(y1max-y1(j))*y1inc
      dx1=(x1max-x1(j))*x1inc+35
      call moveto(dx1,dy1,s)
      dy1=100-(y1max-y1(j+1))*y1inc
      dx1=(x1max-x1(j+1))*x1inc+35
```

```
        dummy=lineto(dx1,dy1)
        end do

c--second drawing

        dummy=setcolor(14)
        do j=1,m-1
        dy2=100+115-(y2max-y2(j))*y2inc
        dx2=(x2max-x2(j))*x2inc+40
        call moveto(dx2,dy2,s)
        dy2=100+115-(y2max-y2(j+1))*y2inc
        dx2=(x2max-x2(j+1))*x2inc+40
        dummy=lineto(dx2,dy2)
        end do

c--third drawing

        dummy=setcolor(14)
        do j=1,m-1
        dy3=232+100-(y3max-y3(j))*y3inc
        dx3=(x3max-x3(j))*x3inc+35
        call moveto(dx3,dy3,s)
        dy3=232+100-(y3max-y3(j+1))*y3inc
        dx3=(x3max-x3(j+1))*x3inc+35
        dummy=lineto(dx3,dy3)
        end do

        dummy=setcolor(15)
        dummy=rectangle($gborder,2,2,212,115)
        dummy=rectangle($gborder,2,117,212,232)
        dummy=rectangle($gborder,2,234,212,348)

        dummy=settextcolor(11)
        call settextposition(2,2,s)
        call outtext('Freq.1')

        dummy=settextcolor(11)
        call settextposition(10,2,s)
        call outtext('Freq.2')

        dummy=settextcolor(11)
        call settextposition(19,2,s)
        call outtext('Freq.3')

        return
        end
```

```
cXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

      subroutine drawr(xr1,xr2,xr3,yr1,yr2,yr3,m)

      include 'fgraph.fd'
      real xr1(1000),xr2(1000),xr3(1000),yr1(1000),yr2(1000),yr3(1000)
      integer*2 dx1,dy1,dx2,dy2,dx3,dy3
      real x1inc,x2inc,x3inc,y1inc,y2inc,y3inc,size
      real x1max,x2max,x3max,y1max,y2max,y3max
      real x1min,x2min,x3min,y1min,y2min,y3min
      integer*2 dummy

      record /rccoord/ s

      dummy=setcolor(1)
      dummy=rectangle($gfillinterior,2,2,212,348)
      dummy=setcolor(15)
      dummy=rectangle($gborder,2,2,212,115)
      dummy=rectangle($gborder,2,117,212,232)
      dummy=rectangle($gborder,2,234,212,348)

      dummy=settextcolor(11)
      call settextposition(2,2,s)
      call outtext('Freq.1')

      dummy=settextcolor(11)
      call settextposition(10,2,s)
      call outtext('Freq.2')

      dummy=settextcolor(11)
      call settextposition(19,2,s)
      call outtext('Freq.3')


      x1max=xr1(1)
      x1min=xr1(1)
      x2max=xr2(1)
      x2min=xr2(1)
      x3max=xr3(1)
      x3min=xr3(1)

      y1max=yr1(1)
      y1min=yr1(1)
      y2max=yr2(1)
      y2min=yr2(1)
      y3max=yr3(1)
      y3min=yr3(1)
```

```
      do j=2,m
      x1max=max(xr1(j),x1max)
      x1min=min(xr1(j),x1min)
      x2max=max(xr2(j),x2max)
      x2min=min(xr2(j),x2min)
      x3max=max(xr3(j),x3max)
      x3min=min(xr3(j),x3min)
      y1max=max(yr1(j),y1max)
      y1min=min(yr1(j),y1min)
      y2max=max(yr2(j),y2max)
      y2min=min(yr2(j),y2min)
      y3max=max(yr3(j),y3max)
      y3min=min(yr3(j),y3min)

      end do
      open(1000,file='max.dat',status='unknown')
      write(1000,*)x1max,x1min,x2max,x2min,x3max,x3min,
     &          y1max,y1min,y2max,y2min,y3max,y3min
c--increments

      y1inc=80./(y1max-y1min)
      y2inc=80./(y2max-y2min)
      y3inc=80./(y3max-y3min)
      x1inc=150./(x1max-x1min)
      x2inc=150./(x2max-x2min)
      x3inc=150./(x3max-x3min)

c-first drawing------

      dummy=setcolor(14)
      do j=1,m-1
      dy1=100-(y1max-yr1(j))*y1inc
      dx1=(x1max-xr1(j))*x1inc+35
      call moveto(dx1,dy1,s)
      dy1=100-(y1max-yr1(j+1))*y1inc
      dx1=(x1max-xr1(j+1))*x1inc+35
      dummy=lineto(dx1,dy1)
      end do

c--second drawing

      dummy=setcolor(14)
      do j=1,m-1
      dy2=100+115-(y2max-yr2(j))*y2inc
      dx2=(x2max-xr2(j))*x2inc+40
      call moveto(dx2,dy2,s)
      dy2=100+115-(y2max-yr2(j+1))*y2inc
```

```
        dx2=(x2max-xr2(j+1))*x2inc+40
        dummy=lineto(dx2,dy2)
        end do

c--third drawing

        dummy=setcolor(14)
        do j=1,m-1
        dy3=232+100-(y3max-yr3(j))*y3inc
        dx3=(x3max-xr3(j))*x3inc+35
        call moveto(dx3,dy3,s)
        dy3=232+100-(y3max-yr3(j+1))*y3inc
        dx3=(x3max-xr3(j+1))*x3inc+35
        dummy=lineto(dx3,dy3)
        end do

        dummy=setcolor(15)
        dummy=rectangle($gborder,2,2,212,115)
        dummy=rectangle($gborder,2,117,212,232)
        dummy=rectangle($gborder,2,234,212,348)

        dummy=settextcolor(11)
        call settextposition(2,2,s)
        call outtext('Freq.1')

        dummy=settextcolor(11)
        call settextposition(10,2,s)
        call outtext('Freq.2')

        dummy=settextcolor(11)
        call settextposition(19,2,s)
        call outtext('Freq.3')

        return
        end

        subroutine bello()
        character*1 k
        ik=7
        do j=1,3
        k=char(ik)
        write(*,'(a1\)') k
        end do
        return
        end
```

******** FOURIER DESCRIPTOR **********

```
      subroutine four(x1,y1,x2,y2,x3,y3,m,fname6)

      integer m,n
      integer x1(450),x2(450),x3(450),y1(450),y2(450),y3(450)
      real l1(450),l2(450),l3(450),d1(25),d2(25),d3(25)
      complex b1(450),b2(450),b3(450),f1(450),f2(450),f3(450)
      real s1,s2,s3,T1,T2,T3,pi,fn
      complex v1(450),v2(450),v3(450),w1(50),w2(50),w3(50)
      complex c1(50),c2(50),c3(50)
      character fname6*(*)

c     open the fourier descriptor data output file

      open(unit=40,file=fname6,status='unknown')

c     calculate the fourier descriptors

      pi=3.14159
      do 10 j=1,m+1
           if(j.ne.(m+1)) then
              v1(j)=cmplx(x1(j),y1(j))
              v2(j)=cmplx(x2(j),y2(j))
              v3(j)=cmplx(x3(j),y3(j))
              else
              v1(j)=cmplx(x1(1),y1(1))
              v2(j)=cmplx(x2(1),y2(1))
              v3(j)=cmplx(x3(1),y3(1))
              endif
10    continue
c
      l1(0)=0
      l2(0)=0
      l3(0)=0
c
      do 120 k=1,m
              f1(k)=v1(k+1)-v1(k)
              f2(k)=v2(k+1)-v2(k)
              f3(k)=v3(k+1)-v3(k)
c
         if(f1(k).eq.0.0) then
           b1(k)=0.0
         else
              b1(k)=f1(k)/cabs(f1(k))
         endif
c
         if(f2(k).eq.0.0) then
           b2(k)=0.0
```

```
      else
           b2(k)=f2(k)/cabs(f2(k))
      endif
c
      if(f3(k).eq.0.0) then
        b3(k)=0.0
      else
           b3(k)=f3(k)/cabs(f3(k))
      endif
c
           s1=0
           s2=0
           s3=0
c
      do 130 i=1,k
             l1(k)=s1+cabs(v1(i+1)-v1(i))
            l2(k)=s2+cabs(v2(i+1)-v2(i))
            l3(k)=s3+cabs(v3(i+1)-v3(i))
             s1=l1(k)
             s2=l2(k)
             s3=l3(k)
130     continue
120   continue
c
      T1=s1
      T2=s2
      T3=s3
c
      do 140 n=-25,25
        if(n.ne.0) then
          s1=0
          s2=0
          s3=0
        do 150 k=2,m+1
           w1(n)=cmplx(cos(-2*pi*n*l1(k-1)/T1),sin(-2*pi*n*l1(k-1)/T1))
           w2(n)=cmplx(cos(-2*pi*n*l2(k-1)/T2),sin(-2*pi*n*l2(k-1)/T2))
           w3(n)=cmplx(cos(-2*pi*n*l3(k-1)/T3),sin(-2*pi*n*l3(k-1)/T3))
           c1(n)=s1+(T1/(4*pi*pi*n*n))*(b1(k-1)-b1(k))*w1(n)
           c2(n)=s2+(T2/(4*pi*pi*n*n))*(b2(k-1)-b2(k))*w2(n)
           c3(n)=s3+(T3/(4*pi*pi*n*n))*(b3(k-1)-b3(k))*w3(n)
           s1=c1(n)
           s2=c2(n)
           s3=c3(n)
150     continue
        else
        goto 140
        endif
```

```
140     continue
c
        do 160 n=2,9
              d1(n)=(c1(1+n)*c1(1-n))/(c1(1)**2)
              d2(n)=(c2(1+n)*c2(1-n))/(c2(1)**2)
              d3(n)=(c3(1+n)*c3(1-n))/(c3(1)**2)
c
              fn=float(n-1)
              write(40,110)fn,d1(n),fn,d2(n),fn,d3(n)
110           format(6f14.6)
160     continue
        close(40)
        return
        end

******** COMPRESS ****************************************************
        subroutine compress(x1,y1,x2,y2,x3,y3,m,x11,y11,x22,y22,x33,y33)
c
        integer m,n,k
        integer x1(500),y1(500),x2(500),y2(500),x3(500),y3(500)
        integer x11(50),y11(50),x22(50),y22(50),x33(50),y33(50)
c
c       open the compressed output data file
c
        open(20,file='compress.dat',status='unknown')
c
c       output the results to the output file
c
        n=m/50
        k=1
c
        do 60 j=1,m
          if(j.eq.(k*n).and.k.le.50) then
          x11(k)=x1(j)
          y11(k)=y1(j)
          x22(k)=x2(j)
          y22(k)=y2(j)
          x33(k)=x3(j)
          y33(k)=y3(j)
          write(20,110) x11(k),y11(k),x22(k),y22(k),x33(k),y33(k)
110       format(6i7)
          k=k+1
c
          end if
60        continue
c
        close(20)
```

```
      return
      end

******** CENTER OF GRAVITY ********************************************

      subroutine cg(x11,y11,x22,y22,x33,y33,fname2)

      character fname2*(*)
      real xx1,yy1,r1(100),fj
      real xx2,yy2,r2(100)
      real xx3,yy3,r3(100)
      integer x22(100),y22(100),x33(100),y33(100),m
      integer sumx2(100),sumy2(100),sumx3(100),sumy3(100)
      integer x11(100),y11(100),sumx1(100),sumy1(100)
c
c     open the output data file
c
      open(30,file=fname2,status='unknown')
c
      sumx1(0)=0
      sumy1(0)=0
      sumx2(0)=0
      sumy2(0)=0
      sumx3(0)=0
      sumy3(0)=0
c
      do 70 m=1,50
      sumx1(m)=x11(m)+sumx1(m-1)
      sumy1(m)=y11(m)+sumy1(m-1)
      sumx2(m)=x22(m)+sumx2(m-1)
      sumy2(m)=y22(m)+sumy2(m-1)
      sumx3(m)=x33(m)+sumx3(m-1)
      sumy3(m)=y33(m)+sumy3(m-1)
70    continue
c
      xx1=sumx1(50)/50
      yy1=sumy1(50)/50
      xx2=sumx2(50)/50
      yy2=sumy2(50)/50
      xx3=sumx3(50)/50
      yy3=sumy3(50)/50
c
c     creat the 1-D radiis data file
c
      do 80 j=1,50
        r1(j) = sqrt((x11(j)-xx1)**2 + (y11(j)-yy1)**2)
        r2(j) = sqrt((x22(j)-xx2)**2 + (y22(j)-yy2)**2)
```

```
          r3(j) = sqrt((x33(j)-xx3)**2 + (y33(j)-yy3)**2)
          fj=float(j)
          write(30,110) fj,r1(j),fj,r2(j),fj,r3(j)
110       format(6f14.6)
80      continue
c
        close(30)
        return
        end

******** INTERGRAL REPRESENTATION ************************************

        subroutine int(x11,y11,x22,y22,x33,y33,fname3)
        integer x11(50),y11(50),x22(50),y22(50)
        integer x33(50),y33(50)
        integer xx1(50),xx2(50),xx3(50)
        integer yy1(50),yy2(50),yy3(50)
        character fname3*(*)
c
c       open the output file
c
        open(40,file=fname3, status='unknown')
c
c       integrals
c
        xx1(1)=x11(1)
        xx2(1)=x22(1)
        xx3(1)=x33(1)
        yy1(1)=y11(1)
        yy2(1)=y22(1)
        yy3(1)=y33(1)
c
        write(40,'(6i7)') xx1(1),yy1(1),xx2(1),yy2(1),xx3(1),yy3(1)
c
        do j=2,50

        xx1(j)=xx1(j-1)+x11(j)
        xx2(j)=xx2(j-1)+x22(j)
        xx3(j)=xx3(j-1)+x33(j)
        yy1(j)=yy1(j-1)+y11(j)
        yy2(j)=yy2(j-1)+y22(j)
        yy3(j)=yy3(j-1)+y33(j)
c
        write(40,'(6i7)') xx1(j),yy1(j),xx2(j),yy2(j),xx3(j),yy3(j)
c
        end do
        close(40)
```

```
      return
      end

c-------CONVERT.FOR PROGRAM-----------

      real r1(50),r2(50),r3(50)
      character*10 fname
      real rr(50),max1,min1
      real max2,min2,max3,min3

      open(10,file='name1.txt',status='unknown')
      read(10,'(a10)')fname
      close(10)

c
c---- OPEN OUTPUT FILE
c
      open(40,file='cog1.dat',status='unknown')
      open(41,file='cog2.dat',status='unknown')
      open(42,file='cog3.dat',status='unknown')
c
c---- READ TARGET INPUT ------------------------
c
      open(20,file=fname,status='old')

        max1=-1000000
        min1=1000000
        max2=-1000000
        min2=1000000
        max3=-1000000
        min3=1000000
c
            do kj=1,50
c
            read(20,'(3f12.5)') r1(kj),r2(kj),r3(kj)
c
            if(r1(kj).ge.max1) then
            max1=r1(kj)
            endif
            if(r1(kj).le.min1) then
            min1=r1(kj)
            endif

            if(r2(kj).ge.max2) then
            max2=r2(kj)
            endif
            if(r2(kj).le.min2) then
```

```
                min2=r2(kj)
                endif

                if(r3(kj).ge.max3) then
                max3=r3(kj)
                endif
                if(r3(kj).le.min3) then
                min3=r3(kj)
                endif

                end do

        close(20)
c
c---- NORMALIZW & PREPARE TRAINING DATA FILE ----------
c
        call NORM(r1,max1,min1,rr)
        write(40,80) (rr(jj),jj=1,50)
80    format (50f12.8)

        call NORM(r2,max2,min2,rr)
        write(41,81) (rr(jj),jj=1,50)
81    format (50f12.8)

        call NORM(r3,max3,min3,rr)
        write(42,82) (rr(jj),jj=1,50)
82    format (50f12.8)

        close(40)
        close(41)
        close(42)

        end

        subroutine NORM(r,max,min,rr)

        real rr(50),MM,BM,max,min,r(50)

        MM = .8/(max-min)
        BM = .9-MM*max

        do jk=1,50
        rr(jk)=(MM*r(jk))+BM
        end do

        end
```

```
C-------ANALYSIS.FOR PROGRAM------------

      INTERFACE TO INTEGER*2 FUNCTION SYSTEM [C]
     +  (STRING[REFERENCE])
      CHARACTER*1 STRING
      END

      character*6 atype
      real asize,bsize,csize,size
      integer idepth,type
      integer*2 I,SYSTEM

c****** Pattern Recognition

      open(100,file='name.txt',status='old')

      read(100,'(i2)') type
      read(100,'(i4)') idepth

      close(100)

      if(type.eq.1) then
      atype='tubesp'
      go to 100
      end if

      if(type.eq.2) then
      atype='FE'
      go to 200
      end if

      if(type.eq.3) then
      atype='copper'
      go to 300
      end if

100   if(idepth.eq.20) then
     I=SYSTEM ('TSIZE21.EXE'C)
     I=SYSTEM ('TSIZE22.EXE'C)
     I=SYSTEM ('TSIZE23.EXE'C)
      end if

      if(idepth.eq.40) then
     I=SYSTEM ('TSIZE41.EXE'C)
     I=SYSTEM ('TSIZE42.EXE'C)
     I=SYSTEM ('TSIZE43.EXE'C)
      end if
```

```
      if(idepth.eq.60) then
     I=SYSTEM ('TSIZE61.EXE'C)
     I=SYSTEM ('TSIZE62.EXE'C)
     I=SYSTEM ('TSIZE63.EXE'C)
      end if

      if(idepth.eq.80) then
     I=SYSTEM ('TSIZE81.EXE'C)
     I=SYSTEM ('TSIZE82.EXE'C)
     I=SYSTEM ('TSIZE83.EXE'C)
      end if

      if(idepth.eq.100) then
     I=SYSTEM ('TSIZE11.EXE'C)
     I=SYSTEM ('TSIZE12.EXE'C)
     I=SYSTEM ('TSIZE13.EXE'C)
      end if

200   if(idepth.eq.20) then
     I=SYSTEM ('FSIZE21.EXE'C)
     I=SYSTEM ('FSIZE22.EXE'C)
     I=SYSTEM ('FSIZE23.EXE'C)
      end if

      if(idepth.eq.40) then
     I=SYSTEM ('FSIZE41.EXE'C)
     I=SYSTEM ('FSIZE42.EXE'C)
     I=SYSTEM ('FSIZE43.EXE'C)
      end if

      if(idepth.eq.60) then
     I=SYSTEM ('FSIZE61.EXE'C)
     I=SYSTEM ('FSIZE62.EXE'C)
     I=SYSTEM ('FSIZE63.EXE'C)
      end if

      if(idepth.eq.80) then
     I=SYSTEM ('FSIZE81.EXE'C)
     I=SYSTEM ('FSIZE82.EXE'C)
     I=SYSTEM ('FSIZE83.EXE'C)
      end if

      if(idepth.eq.100) then
     I=SYSTEM ('FSIZE11.EXE'C)
     I=SYSTEM ('FSIZE12.EXE'C)
     I=SYSTEM ('FSIZE13.EXE'C)
```

```
        end if

300     if(idepth.eq.20) then
      I=SYSTEM ('CSIZE21.EXE'C)
      I=SYSTEM ('CSIZE22.EXE'C)
      I=SYSTEM ('CSIZE23.EXE'C)
        end if

        if(idepth.eq.40) then
      I=SYSTEM ('CSIZE41.EXE'C)
      I=SYSTEM ('CSIZE42.EXE'C)
      I=SYSTEM ('CSIZE43.EXE'C)
        end if

        if(idepth.eq.60) then
      I=SYSTEM ('CSIZE61.EXE'C)
      I=SYSTEM ('CSIZE62.EXE'C)
      I=SYSTEM ('CSIZE63.EXE'C)
        end if

        if(idepth.eq.80) then
      I=SYSTEM ('CSIZE81.EXE'C)
      I=SYSTEM ('CSIZE82.EXE'C)
      I=SYSTEM ('CSIZE83.EXE'C)
        end if

        if(idepth.eq.100) then
      I=SYSTEM ('CSIZE11.EXE'C)
      I=SYSTEM ('CSIZE12.EXE'C)
      I=SYSTEM ('CSIZE13.EXE'C)
        end if

c******* Distance Estimation

        open(10,file='size1.dat',status='unknown')
        read(10,'(f8.6)') asize
        close(10)

        open(20,file='size2.dat',status='unknown')
        read(20,'(f8.6)') bsize
        close(20)

        open(30,file='size3.dat',status='unknown')
        read(30,'(f8.6)') csize
        close(30)

        if(asize.lt.0.0) then
```

```
      asize=0.0
      end if

      if(bsize.lt.0.0) then
      bsize=0.0
      end if

      if(csize.lt.0.0) then
      csize=0.0
      end if

      size=(asize+bsize+csize)/1.5

c     write(*,*) asize, bsize, csize, size

      open(20,file='result.dat',status='unknown')
      write(20,'(f5.3)')size
      write(20,'(i3)')idepth
      write(20,'(a6)')atype
      close(20)

    open(30,file='result1.dat',status='unknown')
    do j=1,6
    write(30,'(a6,i3,f5.3)')atype,idepth,size
      end do

      END
```

APPENDIX B

GUIDELINES FOR THE IMPLEMENTATION OF ARTIFICIAL NEURAL NETWORKS

B.1. Introduction

The primary purpose of this research project was to develop an integrated approach by combining information compression methods and artificial neural networks for the monitoring of plant components using NDE data. Specifically, data from eddy current (EC) inspection of heat exchanger tubing were utilized to develop this technology. The focus of the research was to develop and test various data compression methods (for eddy current data) and the performance of different neural network paradigms for artifact classification and defect parameter estimation. Feedforward fully-connected neural networks, that use the back-propagation algorithm for network training, were implemented for artifact classification and defect parameter estimation using modular networks. The artifact classification was also performed using probabilistic neural networks. A large database from eddy current tube inspection was acquired from the Metals and Ceramics Division of ORNL. These data were used to study the performance of artificial neural networks for artifact type classification and for estimating defect parameters. Most of the study was made using the NeuralWare Professional II/Plus software. A PC based data preprocessing and display program was also developed as part of an expert system for data management and decision making.

As part of this research project a detailed set of guidelines were developed for the implementation of neural networks for diagnostics using eddy current testing. These guidelines were demonstrated for a typical application of eddy current inspection of steam generator tubing. The guidelines include: (1) selection of artificial neural networks, (2) EC data collection and data representation, and (3) design of back-propagation neural networks.

B.2. Selection of Artificial Neural Networks

Artificial neural networks provide general mapping between two sets of information. This nonlinear mapping from data to data is very useful in associating information pairs where a clear mathematical relationship is not available. Artificial neural networks are developed to simulate the most elementary functions of neurons in the human brain, based on the present understanding of biological nervous systems. These network models attempt to achieve good human-like performance such as: learning from experiments and generalization from previous samples. The network models are composed of many nonlinear computational units which are called processing elements (PE) or nodes that operate in a parallel distributed processing architecture.

Selecting a network architecture is the first step for a neural network application. This step is accomplished by: critically evaluating the application, considering the various architectures, and reviewing similar applications. A list of neural network applications and related neural networks is given bellow

Prediction: Use input values to predict output values

(a) Back-propagation,
(b) Digital Neural Network Architecture,
(c) Adaline & Madline,
(d) Perceptron.

Classification: Use input values to predict a categorical output

(a) Back-propagation,
(b) Categorical Learning,
(c) Counter-propagation,
(d) Probabilistic Neural Network,
(e) Learning Vector Quantization.

Data Association: Networks learn associations of error-free or ideal data, then classify or associate data that contain error

(a) Bidirectional Associative Memory,
(b) Hopfield Network,
(c) Hamming Network,
(d) Boltzmann Pattern Completion.

Data Conceptualization: Analyze data and determine conceptual relationships

(a) Adaptive Resonance Theory I,
(b) Self-Organizing Map.

Data Filtering: Smooth an input signal

(a) Recirculation Network.

Optimization: Determine optimal value

(a) Hopfiled Network.

The selection of the type of network is dependent on different applications. In this research project, back-propagation neural networks and probabilistic neural networks were chosen. A PC-based software called the NeuralWare Professional II/Plus was used in the implementation.

B.3. EC Data Collection and Data Representation

B.3.1 Collect the Data

Once the neural network architecture has been identified, the next step is to collect and prepare the data. For this project, a large multi-frequency eddy current (EC) inspection database from laboratory testing of typical tube material was acquired from the Metals and Ceramics Division of Oak Ridge National Laboratory. The data were recorded from two series of measurements on an ASME Section XI standard specimen, shown in Figure B.1. The OD artifacts simulate tube support/tube sheet, ferrite and copper. The OD artifact rings on the

standard are moveable, so that the effect of changing their location with respect to tube defect location may be studied. The database was organized into 900 individual files according to artifact type, defect depth, and the distance from the center of the defect to the artifact.

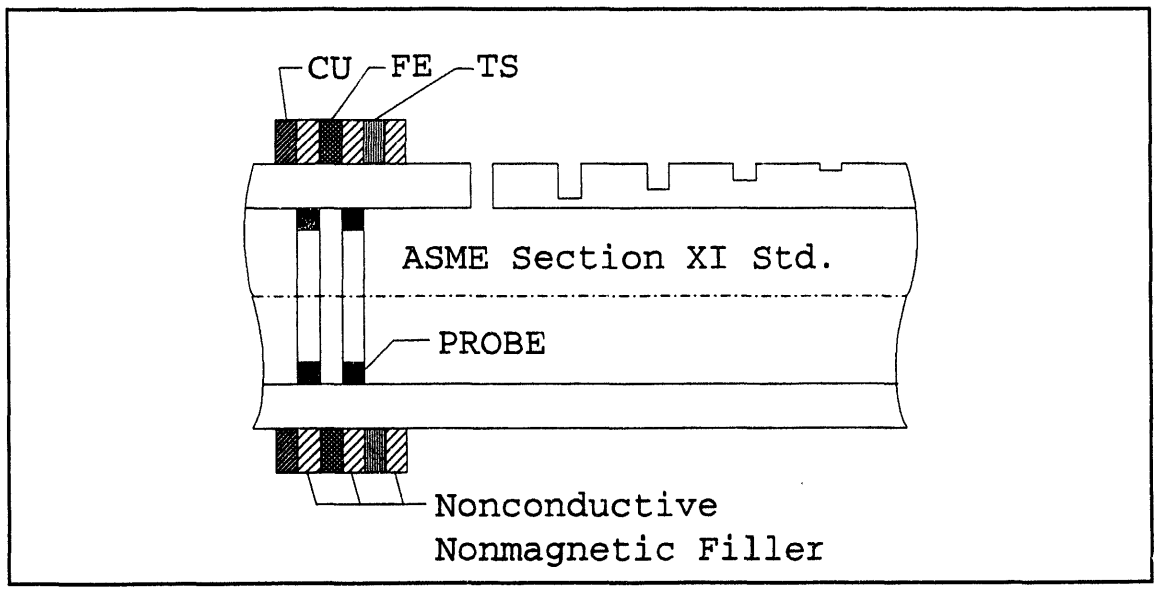


Figure B.1. ASME Section XI standard test specimen with OD artifacts.

B.3.2 Divide the Data into Two Sets: Training and Testing

The collected data set should be divided into two sets: training data and testing data. The training set is designed to maximize the learning process. The testing set is designed to provide the trained network with some real data and test the performance of the network. The training set should contain only "good" data. For best results, it should be evenly divided between the various outcomes and be reasonably representative of the entire universe. In this project, one of the two series EC measurement data set was used as training data and the other was used as the testing data.

B.3.3 Data Representation

For the neural network approach to be effective in artifact classification and defect parameter estimation, the information input to the network must have certain features. These are (a) size of data vector, (b) invariance to data scaling, (c) invariance to data orientation, and (d) sensitivity of the defect type and size parameters to input signature. The data representation may be classified into nonparametric and parametric techniques.

Nonparametric data representation technique involves reorganizing the raw measurement data using (1) direct compression of raw data, (2) subtraction from a reference data, (3) magnitude and phase of the raw data, (4) integral value of the raw data, and (5) sequence of radii from the center of gravity to the closed contour of the shape. Impedance plane integration, radii from the center of gravity and phase angle representation were found to be effective in data representation.

Compressed Magnitude and Phase Representation (MP). This method converts the compressed complex impedance values to magnitudes and phases so that magnitude and phase can be normalized separately.

Compressed Integral Signal Representation (CINT). The line integration of impedance data is sensitive to certain frequencies, and may be used to identify artifact types.

Radii From the Center of Gravity (CG). Since the defect parameters will influence the center of gravity of the complex impedance plot, a sequence of radii from the center of gravity to the contour of the shape is used to train the neural networks to estimate defect parameters and identify artifact types.

The parametric representation technique compresses the data using certain "feature

vectors" and searching the parameter space for the best fit between the measurement and the computed signature. Fourier descriptor method was tried in this project. This technique generates a feature vector called Fourier descriptors whose elements are a function of the shape of the signal. These descriptors have the property of being invariant under scaling, rotation, and translation operations. In addition, they offer a significant amount of data compression. However, the Fourier descriptors were found to be not sensitive enough to describe the differences in the EC defects for parameter estimation in our neural network training. This may be because (1) only limited terms are used in Fourier series to approximate the periodic signal, (2) portions of the impedance plot do not bear useful information, and (3) the Fourier descriptors have difficulties in describing local information and discriminating symmetrical shapes.

Some other parametric techniques such as autoregression modeling of object contours, auto-power spectral densities (APSDs), cross-power spectral densities (CPSDs), coherence functions, and wavelet transform signals were also used in some other applications.

The results of analysis show that for effective (low-error) defect classification and estimation of parameters, it is necessary to identify proper feature vectors using various data compression methods. The center of gravity and phase angle representation methods were found to be sensitive to defect parameter estimation. The center of gravity and the compressed integral signature were very effective in artifact type identification.

B.3.4 Scale the Data

Neural networks are very sensitive to absolute magnitudes. If one input ranges from 1,000 to one million and a second from zero to one, fluctuations in the first input will tend to

swamp any importance given to the second, even if the second input is much more important to predict the desired output. To minimize the influence of the absolute scaling, all inputs to a neural network should be scaled so that they correspond to roughly the same range of values. Commonly chosen ranges are 0 to 1 or -1 to +1.

Data scaling can be done by writing a simple computer program to map the desired range of a variable (with a range between the minimum and the maximum values) to the full range of the network. An example of such a program can be found in the EDDYANN Code.

Many neural network simulation software systems perform data scaling automatically. NeuralWare Professional II/Plus software provides the MinMax table function to perform the data scaling automatically.

B.4. Design of Back-Propagation Neural Networks

Back-propagation neural networks (BPN) were used in this research to develop the neural network models for artifact classification and defect parameter estimation. The preprocessed eddy current impedance data were used as input feature to back-propagation neural networks, with the output map providing artifact type and estimates of defect parameters (depth and distance). Separate networks for artifact classification and parameter estimation were developed. The NeuralWorks Professional II/Plus was used in most of the implementation.

There are several issues that need to be considered when utilizing the back-propagation algorithm to train a neural network. The following discusses the selection of hidden layers and nodes, the selection of learning options, training and testing, and deploying of networks.

B.4.1 Selection of Number of Hidden Layers and Nodes

The selection of the number of hidden layers and the hidden layer nodes is one of the most important issues in back-propagation network applications. There have been various studies related to this topic, but there is no definite solution to this problem. However, it has been concluded that using only one hidden layer is sufficient to solve the problems in the area of signal processing, plant monitoring, parameter estimation, and sensor validation. In this project, it has been found that one hidden layer is sufficient for the training of the center of gravity data and integral data for defect parameter estimation.

The selection of hidden nodes for a fully-connected, feedforward network with one hidden layer is based on the following rules:

Rule-of-thumb 1: The more complex the relationship between the input data and the desired output, the more nodes are normally required in the hidden layer.

Rule-of thumb 2: The upper bound for the number of nodes in the hidden layer is

$$\frac{cases}{10\times(m + n)} = h$$

where

cases is the number of rows or vectors in the training file.

m is the number of nodes in the output layer.

n is the number of nodes in the input layer.

h is the number of nodes in the hidden layer.

B.4.2 Set the Network Parameters

The most important learning options for back-propagation network training are selected as follows.

(1) Learning coefficient

The learning coefficient is the rate at which weights adjust to correct for errors. In our application, it is set to 0.4 for the first 10000 iterations, and 0.1 for the remaining iterations.

(2) Momentum Term

Momentum term is a factor used to smooth the learning. Here it is set to 0.5.

(3) The nonlinear transfer function

The nonlinear transfer function transfers the internally generated sum for each node to an output value. Available transfer functions in back-propagation network are: linear, sigmoid, hyperbolic tangent, and Gaussian cumulative distribution. Through several trials of network training, it is found that the hyperbolic tangent transfer function facilitates faster and more accurate network training. This is because the output range of hyperbolic tangent is from -1 to +1, as compared to the sigmoid range of 0 to 1. The output of the transfer function is used as a multiplier in the weight update equation, a range of 0 to 1 means a smaller multiplier when the summation is a low value, and a higher multiplier for higher summation. This could lead to a bias in learning higher desired output. The hyperbolic tangent gives equal weight to low and high end values.

(4) The learning rule

The learning rule in back-propagation network specifies how connection weights are changed during the learning process. Three learning rules are commonly used in BPN: Delta-

Rule, Cumulative Delta-Rule, and Normalized Cumulative Delta-Rule. Usually, the Normalized Cumulative Delta-Rule is more effective and should be selected.

B.4.3 Training and Testing

The goal of the back-propagation algorithm is to teach the network to associate specific output patterns (target patterns) by adjusting the connection weights in order to minimize the error between the target output and the actual output of the network. To accomplish this, the network is usually trained with a large number of input/output pairs. A gradient descent algorithm is generally used to perform the optimization.

The back-propagation training process is composed of two types of passes: the forward pass and the reverse pass. In the forward pass the input signals propagate from the network input to the output. In the reverse pass, the calculated error signals propagate backward through the network where they are used to adjust the weights. The calculation of the output is carried out, layer by layer, in the forward direction. The output of one layer is the input to the next layer. In the reverse pass, the weights of the output layer are adjusted first since the target value of each output node is available to guide the adjustment of the associated weights, using a modification of the Delta Rule. Next, the weights of the middle layers are adjusted. Since the middle layers have no target values, the error must be propagated back through the network, layer by layer.

A common technique to reduce training time and reduce the probability of being trapped in a local minimum is to use a momentum term which enhances the stability of the training process. This technique involves adding to the weight adjustment a term which is proportional

to the amount of the previous weight change.

The step-by-step procedure of back-propagation training is as follows:

(1) Randomize the weights to small random values (both positive and negative) to assure that the network is not saturated by large values of weights.

(2) Select a training pair from the training set.

(3) Apply the input vector to network input.

(4) Propagate the input vector in a forward fashion through the network until the final network outputs are calculated.

(5) Calculate the network output and the error (the difference between the network output and the desired output).

(6) Calculate the local errors.

(7) Adjust the weights of the network to minimize the error.

(8) Repeat steps 2-7 for each pair of input/output vectors in the training set until the error for the entire system is acceptably low.

B.4.4 Deploy the Networks

Since the trained networks from NeuralWorks software cannot be used outside the NeuralWorks platform, it is necessary to convert a trained network into a general C source code. This code has two parts: (1) a C source code callable function, and (2) a main program. The C source code callable function is generated using the Flash Code function of NeuralWorks software. It contains the weight values of the trained network. The main program can read the recall input data file, execute the callable function, and write the recall results into an output file. The following is an example of a main program. In this program, the input data file is "COG1.DAT", the output file is "SIZE1.DAT", and the callable function is "NN_Recall( )".

```
#include "stdio.h"
#include "math.h"

main()

/*prepare INPUT and OUTPUT files */
{
        float NN_Recall();
        float Yin[50],Yout;
        int i;

        FILE *file1,*file2,*fopen();

        /* open both files */
        file1=fopen("COG1.DAT","r");
        file2=fopen("SIZE1.DAT","w");

        for(i=0; i<50; i++)
        fscanf(file1,"%f", &Yin[i]);

        Yout=NN_Recall(Yin);

        fprintf(file2,"%f", Yout);

        /* close files */
        fclose(file1);
        fclose(file2);
        exit(0);
}
```

DATE FILMED

1 / 24 / 94

END