8-2003

Advanced eddy current test signal analysis for steam generator tube defect classification and characterization

James Patrick McClanahan

To the Graduate Council:

I am submitting herewith a dissertation written by James Patrick McClanahan entitled "Advanced eddy current test signal analysis for steam generator tube defect classification and characterization." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

B. R. Upadhyaya, Major Professor

We have read this dissertation and recommend its acceptance:

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by James Patrick McClanahan entitled "Advanced Eddy Current Test Signal Analysis for Steam Generator Tube Defect Classification and Characterization." I have examined the final paper copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

Belle R Upadhyaya

B. R. Upadhyaya, Major Professor

We have read this dissertation and recommend its acceptance:

Accepted for the Council:

Vice Provost and Dean of Graduate Studies

Advanced Eddy Current Test Signal Analysis for Steam Generator Tube Defect Classification and Characterization

A Dissertation
Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

James Patrick McClanahan
August 2003

ACKNOWLEDGMENTS

The first person that I would like to thank would be Dr. B. R. Upadhyaya. Dr. Upadhyaya's guidance and support throughout my graduate career was invaluable. I would also like to thank my dissertation committee members, Dr. L. W. Townsend, Dr. J. W. Hines both of the Nuclear Engineering Department, and Dr. H. Qi of the Electrical and Computer Engineering Department. Their suggestions and guidance contributed to this work significantly. Also I would like to thank Dr. H. L. Dodds, Head of the Nuclear Engineering Department for his guidance and support.

Through the past five years, my funding has originated from three sources. For the first three years, my funding came from the University of Tennessee's Maintenance and Reliability Center through an NSF grant. I want to thank Dr. T. W. Kerlin for the confidence in my abilities. Also, to Tom Byerley, present director of MRC, thank you for allowing me access to the MRC Teaching Laboratory. For the past two years the majority of my funding has been through the ENGAGE program. I would like to thank Dr. Parsons and Dr. Seat for the opportunity to teach, and learn, in the ENGAGE program.

For the past three years I have worked with Gary Henry and Craig Smith, both from the EPRI NDE Center, Charlotte, NC. EPRI furnished extremely valuable information about eddy current test (ECT) data as well as my ECT database. Without this insight and database, my research would not have been accomplished.

In addition, the technical assistance I received from Richard Bailey and Gary Graves was very much appreciated. I would like to extend my thanks to the students and visiting scholars of the NE Department of UTK, their input also contributed to my work. Also, there were so many close friends from UTK and home that have helped push me to finish, to them I would also like to say thank you.

Finally, I want to thank all my family whose support these past five years has meant the world to me. Specifically, I would like to thank my mother and father, Deana and James, my brother Chris and my grandparents Ken and Edith. I would never have made it to this point without you. I also would like to mention my other grandparents who are no longer with us, Monroe and Ollie.

ABSTRACT

Eddy Current Testing (ECT) is a Non-Destructive Examination (NDE) technique that is widely used in power generating plants (both nuclear and fossil) to test the integrity of heat exchanger (HX) and steam generator (SG) tubing. Specifically for this research, laboratory-generated, flawed tubing data were examined. The tubing data were acquired from the EPRI NDE Center, Charlotte, NC. The data are catalogued in the Performance Demonstration Database (PDD) which is used as a training manual for certification. The specific subset of the data used in this dissertation has an Examination Technique Specification Sheet (ETSS) and a blueprint of the flawed tube specimens.

The purpose of this dissertation is to develop and implement an automated method for the classification and an advanced characterization of defects in HX and SG tubing. These two improvements enhanced the robustness of characterization as compared to traditional bobbin-coil ECT data analysis methods. A more robust classification and characterization of the tube flaw in-situ (while the SG is on-line but not when the plant is operating), should provide valuable information to the power industry.

The following is a summary of the original contributions of this dissertation research.

1. Development of a feature extraction program acquiring relevant information from both the mixed, absolute and differential ECTD Flaw Signal (ECTDFS).
2. Application of the Continuous Wavelet Transformation (CWT) to extract more information from the mixed, complex differential ECTDFS.
3. Utilization of Image Processing (IP) techniques to extract the information contained in the generated CWT.
4. Classification of the ECTDFSs, using the compressed feature vector and a Bayes classification system.
5. Development of an upper bound for the probability of classification error, using the Bhattacharyya distance, for the Bayesian classification.
6. Tube defect characterization based on the classified flaw-type to enhance characterization
7. Development of a diagnostic software system EddyC and user's guide.

The important results of the application of the method are listed. The CWT contains at least enough information to correctly classify the flaws 64% of the time using the IP features. The Bayes classification system, using only the CWT generated features (after PCA compression), correctly identified 64% of the ECTD flaws. The Bayes classification system correctly identified 75% of the ECTD flaws using cross validation utilizing all the generated features after PCA compression. Initial template matching results (from the PDD database) yielded correct classification of 69%. The B-distances parallel and bound the percent misclassified cases. The calculated B-distance for 15 PCs were 0 and 14.22% bounding the 1.1% incorrectly classified. But, these Gaussian-based calculated B-distances may be inaccurate due to non-Gaussian features. The number of outliers seems to have an inverse relationship with the number of misclassifications. Characterization yielded an average error of 12.76 %. This excluded the results from flaw-type 1 (Thinning).

The following are the conclusions reached from this research. A feature extraction program acquiring relevant information from both the mixed, absolute and differential data was successfully implemented. The CWT was utilized to extract more information from the mixed, complex differential data. Image Processing techniques used to extract the information contained in the generated CWT, classified the data with a high success rate. The data were accurately classified, utilizing the compressed feature vector and using a Bayes classification system. An estimation of the upper bound for the probability of error, using the Bhattacharyya distance, was successfully applied to the Bayesian classification. The classified data were separated according to flaw-type (classification) to enhance characterization. The characterization routine used dedicated, flaw-type specific ANNs that made the characterization of the tube flaw more robust. The inclusion of outliers may help complete the feature space so that classification accuracy is increased.

Given that the eddy current test signals appear very similar, there may not be sufficient information to make an extremely accurate (> 95%) classification or an advanced characterization using this system. It is necessary to have a larger database fore more accurate system learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ANN :	Artificial Neural Network
AVB :	Anti-vibration Bar
B&W :	Babcock & Wilcox
CE :	Combustion Engineering
CWT :	Continuous Wavelet Transformation
DWT :	Discrete Wavelet Transformation
EC :	Eddy Current
ECT :	Eddy Current Testing
ECTD :	Eddy Current Test Data
ECTDFS :	Eddy Current Test Data Flaw Signal
EPRI :	Electric Power Research Institute
ETSS :	Examination Technique Specification Sheet
FF :	Fill Factor
ID :	Inner Diameter
IGA :	Intergranular Attack
IP :	Image Processing
NN :	Neural Network
OD :	Outer Diameter
ODSCC :	Outer-Diameter Stress Corrosion Cracking
PCA :	Principal Component Analysis
PDD :	Performance Demonstration Database
Pe :	Probability of Error
PFA :	Polynomial Function Approximation
PR:	Pattern Recognition
PSD :	Power Spectral Density
PWR :	Pressurized Water Reactor
PWSCC :	Primary-Water Stress Corrosion Cracking
SCC :	Stress Corrosion Cracking
SG :	Steam Generator

SOM :	Self-Organizing Map
STD :	Standard Deviation
STFT :	Short-Time Fourier Transformation
UPeBD :	Upper Bound of the Probability of Error based on the Bhattacharyya Distance
UPeBDZ :	UPeBD with Zeroed Off-diagonal Covariance Matrices
%TW :	Percent Through-wall

1. Introduction

The introduction is divided into five sections. The first section details the background and motivation for this research. The second section describes the problem statement, tasks accomplished, and the outline of the solution. The third section reviews previous work. The fourth section lists the contributions of this research, with the final section outlining the structure of this dissertation.

1.1. Background and Motivation

Pressurized water reactor (PWR) power plants contain either U-tube or once-through type steam generators (SG). These are complex structures with about 3,500 stainless steel tubes in a typical U-tube steam generator. Over a period of time these tubes degrade because of exposure to high temperature, pressure, and chemically corrosive environment. Typical tube degradations include stress corrosion cracking (SCC), intergranular attack (IGA), thinning, sludge pile, pitting, mechanical fretting, anti-vibration bar (AVB) wear, impingement, and denting. Often, the degraded tubes are either plugged or sleeved. As a result, about one-half of the PWR nuclear power plants in the world have been plugging or repairing steam generator tubes in any given year. This action reduces the efficiency of the steam generator.

In recent years, the average percentage of PWR tubes plugged per year has been about 0.3%. The number of steam generator tubes plugged per year during the last few years has ranged from 10,000 to 12,000 tubes. Although an average rate of 0.3% per year may seem acceptable, over a 40 year steam generator life, this amounts to 10 to 12% of the available tubes being plugged [1].

If a tube ruptures during operation, a complex plant transient will ensue. Usually the transient does not result in an enviromental release, but a plant shutdown and repairs will be needed. Spontaneous rupturing of tubes occurs about once every two years and incipient tube ruptures (tube failures usually identified with leak detection just before rupture) at the rate of one per year. This shutdown itself would cost the plant approximately $750,000 per day in lost revenues, not to mention the repair costs [1]. **The cost of replacing a steam generator is about $150 million in a 1,300 MWe four-loop plant** [2].

Eddy Current Testing (ECT) is performed periodically to check the integrity of these tubular structures within the HX or SGs. If more information can be obtained, specifically classification and advanced characterization of the flaw in-situ (while the SG is on-line but not when the plant is operating), ECT using a bobbin-coil probe would be more cost effective and would insure better overall operation.

In this research, laboratory-generated, flawed tubing data were examined. The tubing data were acquired from the Electric Power Research Institute (EPRI) Non-Destructive Examination (NDE) Center, Charlotte, NC. The data are catalogued in the Performance Demonstration Database (PDD) which is used as a training manual for certification. The specific subset of data used has an Examination Technique Specification Sheet (ETSS) and a blueprint of the flawed tubes.

1.2. Statement of the Problem, Tasks Accomplished, and Outline of the Approach

This section is divided into three parts. The first part details a statement of the problem. The second section specifies the two tasks accomplished by this research. Finally, an outline of the technical approach is given.

1.2.1. Statement of the Problem

The ECT technology has a proven track record at both detecting SG tubing defects and basic characterization of the defect (only defect sizing given in % through-wall or %TW) while the SG is on-line (but not when the plant is operating). The type of flaw is ususally narrowed down, but not determined, by the location of the flaw in the tube, whether the flaw occurs as an outer diameter (OD) or an inner diameter (ID) flaw, and the SG vendor. A profile of the physical degradation can be determined if there is information contained in the mixed absolute ECT signal. A degraded SG tube is plugged or sleeved after a certain %TW damage is determined by the ECT specialist. The type of degradation is usually determined after a tube was pulled out and inspected.

At this time, using basic bobbin-coil ECT, there is no method available to classify the type or volume (length, width, depth and volume) of degradation of a flaw while the tube is still in the steam generator.

1.2.2. Major Tasks Accomplished

The purpose of this dissertation was to develop and implement an automated method for the classification and advanced characterization of defects in HX and SG tubing.

Different degradation mechanisms cause the SG tube wall to physically deteriorate differently (classification of degradation). Therefore, two improvements were made in the basic bobbin-coil ECTD analysis.

1. In-situ classification of tube flaws as indicated by the ECTD signal.
2. In-situ characterization (flaw sizing using length, width, etc.) of the flaws.

These two improvements enhanced the robustness of characterization as compared to traditional methods. A more robust classification and characterization of the tube flaw should provide valuable information to the power industry.

1.2.3. Technical Approach and Definition of Tasks

The approach that was developed for the diagnosis of degradation (both classification and characterization) of SG tubes consists of several steps. For steps 3 through 7 new or modified analysis techniques were required. All the steps are enumerated below.

1. ECTD Pre-processing with EddyM.m

    a) Frequency mixing
    b) De-drifting
    c) De-noising.

2. Entering Known Information from the PDD

a) ECTD flaw identification
b) Location of flaw, if given
c) Differential impedance plane phase angle and magnitude
d) Classification (if known)
e) Characterizations (if known).

3. Transformation of the mixed, complex, differential ECTD flaw signal (ECTDFS) using the Continuous Wavelet Transformation (CWT).

4. Feature Extraction

    a) Polynomial function approximation (PFA) of the inductive reactance component of absolute mixed ECTDFS
    b) One-dimensional feature extraction for the inductive reactance component of the mixed differential ECTDFS
    c) Image processing (IP) characterization of the CWT of the complex, mixed differential ECTDFS.

5. Data compression of extracted features utilizing Principal Component Analysis (PCA.).

6. ECTD defect classification using compressed feature vector and CWT using a traditional pattern recognition (PR) technique.

7. ECTD defect characterization (or flaw sizing) using multiple artificial neural networks (ANNs), one for each flaw-type.

A flow diagram of these steps is given Figure 1. This diagram illustrates the interactions among the steps and the initial steps taken during the analysis. The solution, given in Figure 1, generated new information from the ECTDFS by Continuous Wavelet Transformation (CWT) processing, Polynomial Function Approximation (PFA) and a basic feature extraction. The CWT is a signal processing method that extracts time and frequency (scale) information from a signal.

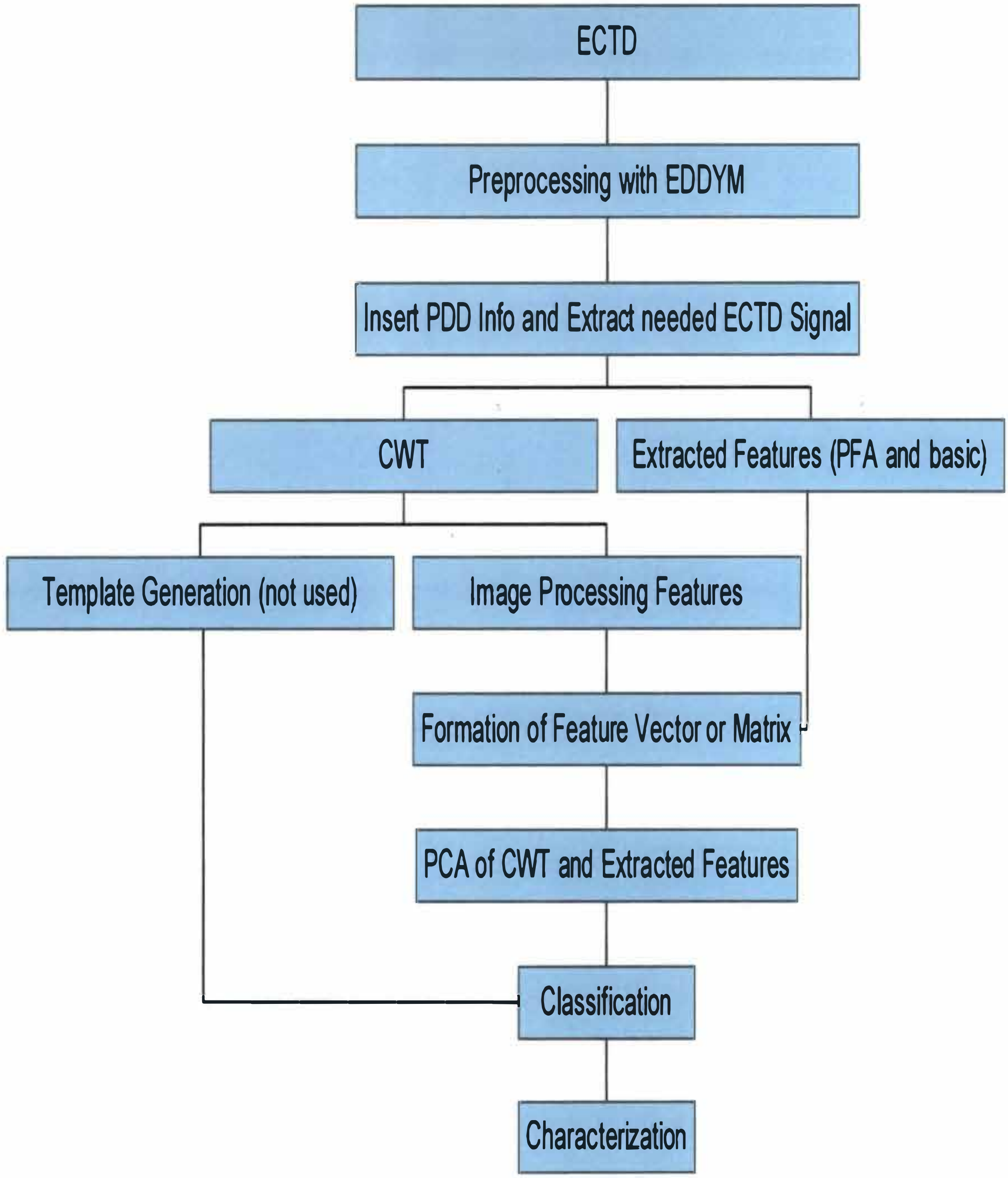

Figure 1. Flow diagram of ECTD analysis showing the various steps.

Then the new information, generated by the CWT, was compressed using image processing (IP) techniques. All the features were then included in a feature vector. This new feature vector was compressed using the PCA. The compressed feature vector was then used to classify the tubing flaw. Once the classification was complete, separate ANNs were used for flaw characterization.

1.3. Review of Previous Work

This section is divided into three parts: ECT and wavelet transforms, CWT and applications, and review of research at the University of Tennessee. The references provided here, were the most pertinent found during an extensive literature review.

1.3.1. Eddy Current Testing (ECT) and Wavelet Transformations

Only one ECT reference was located in this search that employs a CWT. This reference describes the implementation of the modulus of the CWT of the complex ECTDFS along with a Bayes strategy to determine the location of outer diameter notches along a tube. A signal-to-noise ratio was applied to the CWT to determine which scale (approximately the inverse of frequency) level has the highest signal to noise ratio. This scale level was then used with a Bayes strategy to determine if a flaw exists [3].

The next references use the discrete wavelet transform (DWT) for various applications in ECT. The first reference describes a DWT filtering technique to eliminate noise from the ECT signal [4]. An automated flaw detection algorithm for signals in the tube support plate (TSP) region was created using the affine transformation for pre-processing (ECT data frequency mixing), and wavelet transformation (DWT using Daubechies 2) for compression and feature extraction and regression for evaluation. The feature extraction consisted of thresholding a specified level of the DWT coefficients, then determining anti-polar peaks and a distance threshold [5]. Multi-frequency ECT (using a probe designed for flat surfaces) was used to generate EC flaw data. The material used to generate the ECT data had a cuboid geometry. The flaw data were then filtered and converted to a spectrogram. This process was done in both the X and Y directions in order to give a 3-D spectrogram of the flaw. Features were extracted from the 3-D spectrogram and used as input to a neural network (NN). The first NN determines a flaw location and the second NN determines shape characteristics [6]. ECT (using a probe designed for flat surfaces) was used to construct a 2-D image of the circuit board. The ECT data was first processed using the discrete wavelet transform (DWT) to filter and clean (extract relative DWT levels, then threshold) the signal. The DWT-processed signal was then used to construct a 2-D image [7].

All of the above described methods worked, but none used CWTs to obtain classification and advanced characterization of steam generator tubing utilizing bobbin-probe ECT.

1.3.2. Continuous Wavelet Transform (CWT) and Applications

Three references were located that use similar tools and algorithms as this research. The first reference was an application of CWT to speech signal, treating the resulting CWT as an image. The CWT "image" was characterized using global descriptors (geometric moments) and "blob" descriptors. The characterizing quantities are then used to classify the voice pattern [8]. The second reference details transforming a vibration signal from rotating equipment using continuous wavelets. The CWT was then converted to a binary image (image value of 0 or 1) using a coefficient threshold technique. The binary image was converted to a vector and used in a neural network to classify the condition of the equipment [9]. The third reference uses CWTs of ultrasonic signals to produce a fingerprint. The CWT fingerprints are then compressed using geometric moments. The geometric moments are used as input to a neural network to classify (or sort) different materials. The classification had a 100% success rate [10].

This use of geometric moments with CWTs was employed in this research. The technique used in this research also employs converting the CWT into a binary image for processing.

1.3.3. Research at The University of Tennessee

There have been two areas of investigation within the UTK-NE department directly related to this research, the first area was the analysis of ECT and the second area was the use of wavelet transforms. There have been five publications since 1996.

The first area of research focused on using various data descriptors (phase angle, magnitude, linear integral, radii from the center of gravity, and Fourier descriptors) derived from the ECT signal to determine if there was a flaw present (using fuzzy logic) and then determining %TW for the flaw (using neural networks). The results show that specific descriptors were effective for either defect identification or defect description. The Fourier descriptors were not very effective for either task [11]. Another area of research was to create a fuzzy logic system whose input was the phase angle of the flaw for three of the four channels of the ECT data. The problem was to determine if the signal was a flaw and to determine the %TW [12]. The third report defines a system based on the wavelet zero crossings. The wavelet zero crossing technique first performed a 2-level DWT on the signal, with the resulting DWT signal transformed using the zero crossing technique. A fuzzy logic system using the number of zero crossings for each level as input and defect sizing as an output was established. The accuracy of this system was fairly good [13]. The final area of research focused on extraction of features (signal segment, phase angle, linear predictive coding, and wavelet zero crossing) from the ECT flaw data with the features then used in a self-organizing map (SOM) neural network for classification. The results show that the SOM worked well with the real signal segment [14]. The final report was a general overview of the use of Power Spectral Density (PSD), Short Time Fourier Transformation (STFT) and wavelets (DWT) as research tools. Interestingly, the DWT was used to separate signals into 20 levels, and then the FFT was applied at each level. The resulting PSDs were grouped together generating a band-limited waterfall plot of PSDs [15].

1.4. Contributions of this Dissertation

This section is divided into two parts, original contributions and other contributions.

1.4.1. Original Contributions

The following is a summary of the original contributions of this dissertation research.

1. Development of a feature extraction program acquiring relevant information from both the mixed, absolute and differential ECTDFS. The features from the mixed, inductive reactance component of the differential ECTD flaw included, standard deviation (STD) normalized peak-to-peak magnitude and the number of data points between peaks. The PFA coefficients of the inductive reactance component of the mixed, absolute ECTDFS were also used as features.

2. The application of the CWT to extract more information from the mixed, complex differential ECTDFS. For the CWT to be useful, the information contained in the CWT must be extracted and utilized.
3. The use of IP techniques to extract the information contained in the generated CWT. The two IP features used were geometric moments and other basic IP parameters used for picture comparison.
4. Classification of tube defects, utilizing the compressed feature vector and using a Bayes classification system.
5. Development of a diagnostic software system EddyC and user's guide.

1.4.2. Other Contributions

The other contributions were:

1. Classification of the tube defects using dedicated ANNs. The characterization routine used separate, flaw-type specific ANN that resulted in robust characterization of the ECTD flaw.
2. Development of an upper bound for the probability of error, using the Bhattacharyya distance, for the Bayesian classification.

This research outlines the methods used to incorporate the new ECTDFS features for flaw classification and characterization. It also describes the methods used to incorporate the information contained in a CWT into pattern recognition algorithms.

1.5. Outline of the Dissertation

This dissertation is divided into seven sections. Section 1 is the introduction. Section 2 is a review of basic ECT and general information about SGs and tubing flaws. Section 3 describes data transformation utilizing the CWT. Section 4 describes the three types of features extracted from the ECTDFS and feature compression using the PCA. Section 5 describes the technique of flaw classification used in this research and the approach for flaw characterization (flaw sizing). Section 6 contains a discussion of the results. Section 7 includes a summary, conclusions, and recommendations for future work. Appendices A-G contain additional results and listings of computer codes.

2. Background Study of Eddy Current Testing (ECT) and Steam Generator Information

This section is divided into two parts. The first part gives a basic overview of ECT theory and application. The second part is a general review of steam generator information with an emphasis on the information contained in EPRI's Performance Demonstration Database (PDD).

2.1. Eddy Current Testing (ECT)

The ECT section is divided into five parts. The first section discusses basic ECT principles. The second section provides an overview of how ECT excitation frequencies are determined. The third section describes ECTD analysis. The final section lists the advantages and disadvantages of ECT.

2.1.1. Eddy Current Testing Basics

ECT is accomplished by using tubular-shaped coils (bobbin coils) that are excited by an alternating current. This alternating current produces a magnetic field that permiates the tubing. The permiating magnetic field produces circular electric currents (eddy currents) within the tube wall. These currents in turn generate a field that opposes the primary field. If there is a defect in the tube wall, the opposing field changes, thus changing the impedance (both resistance and inductive reactance) of the primary coil. This impedance is measured and processed to identify flaws in the tubing. A schematic of a differential bobbin coil probe is shown in Figure 2.

The properties of the eddy current are affected by and can detect changes in electrical conductivity and/or magnetic permeability of a specimen caused by changes in the following characteristics.

- Grain size
- Surface treatment, especially heat treatment
- Coating thickness

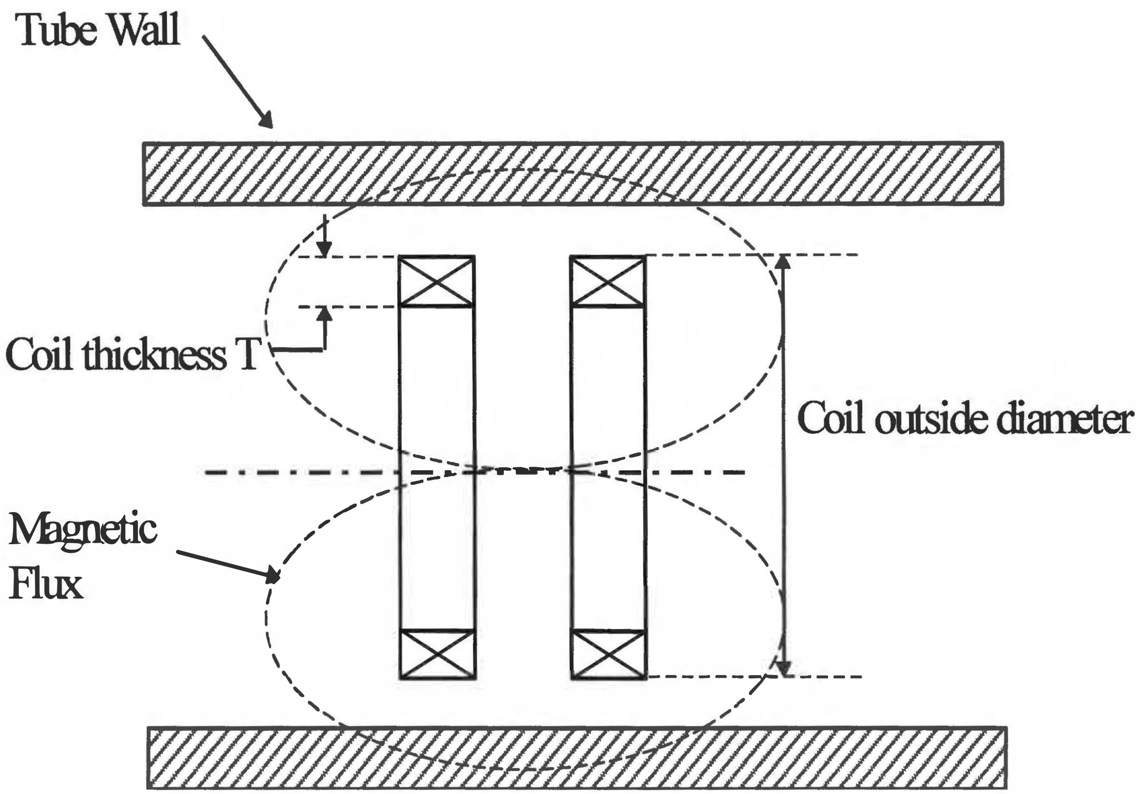

Figure 2. Schematic of a differential bobbin coil probe [12].

- Hardness
- Discontinuities such as cracks, inclusions, dents, and holes
- Dimensions such as thickness, eccentricity, diameter, or separation distance
- Alloy composition [16, 17].

There are many facets of ECT that could be detailed, but only issues relevant to this research will be discussed. The topics include ECT excitation frequencies and mixing, EC analysis and the application to steam generator tubing flaws.

2.1.2. ECT Excitation Frequencies

The frequency of the alternating current in the primary coil is extremely important. Most eddy current testing utilizes frequencies between 500 to 500,000 Hz. As the frequency increases, the depth of penetration of eddy currents decreases. This "skin effect" limits the depth of penetration or inspection. Extremely high frequencies are used to detect the position of the specimen (measure the distance between the specimen and the probe). Such detectors are also used as dynamic or vibration testing transducers.

Depth of penetration is also dependent upon conductivity and magnetic permeability of the specimen as is illustrated in Table 1. The "standard depth" Indicates the depth into or thickness of the specimen that decreases the signal to 1/e (37%) of the signal at the surface. Note that the depth of penetration varies in an orderly fashion with frequency (a straight line on a log-log plot) for non-magnetic materials but decreases more rapidly for iron and its alloys. The standard depth of penetration (S) can be calculated from the relationship:

$$S\ (inches) = 1980\ (\rho/\mu f)^{1/2} \tag{1}$$

with: ρ = resistivity (ohm-cm)

μ = magnetic permeability (constant, no dimensions)

f = frequency (Hz).

Table 1. Conductivity and Depth of Penetration for Various Metals [18].

Metal	Conductivity (% IACS)[1]	Depth of Penetration[2] (mils) 1 KHz	100 kHz	10MHz
Cu	100	80.0	8.00	0.80
Al	61.0	160	16.0	1.60
Ti	3.1	800	80.0	8.00
304 SS	2.5	550	55.0	5.50
Fe	10.7[3]	14.0	1.40	0.10

[1]International Annealed Copper Standard.

[2]Depth into specimen at which eddy current signal is l/e of the signal at the surface.

[3]Without saturation. At saturation, depth of penetration is approximately the same as that for stainless steel.

The ratios used to determine the needed depth of penetration and the primary frequency (phase difference between inner and outer wall defects of 90°) for a particular sample with a specified wall thickness and electrical resistivity are given as:

$$\frac{t}{\delta} = 1.1 \qquad f_{90} = \frac{3\rho}{t^2} \tag{2a, 2b}$$

where: δ = depth of penetration (or S in Eq. 1)

ρ = electrical resistivity

t = tube wall thickness

f_{90} = primary frequency phase difference between inner and outer wall defects (in kHz).

Use of a single frequency gives larger responses from the tubing supports than obtained from the tubing wall thickness. Recent applications of EC, especially to tubular goods, make use of multiple, simultaneous frequencies in the primary coil. There are usually four excitation frequency levels associated with ECT. These frequency levels are high, primary, half and quarter.

By proper selection of frequencies, unwanted information or interference from properties or structures in the specimen, of no interest, can be minimized or eliminated. For instance, the effect of support structures on measurement of wall thickness, pitting, and holes in thin wall tubing can essentially be eliminated by using a pair of frequencies. Wall thickness at the supports is often critical, as vibration of the tubes may have produced wear from rubbing of the tubes against the support. Bi-frequency analysis can adjust for the supports [18].

During ECT of SG tubing, the probe outputs a distorted signal. Usually, the signal distortions are caused by material either attached to or near the steam generator tubing. Other factors such as specimen conductivity, magnetic permeability, test specimen thickness and other geometrical parameters, coupling between the probe coil and specimen due to probe wobble, the presence of cracks and others result in unwanted contributions to the signal.

The above-mentioned interference affects the flaw signals generated at that site. Frequency mixing is a method to combine the lower and higher frequency signals to minimize the interference and maximize the flaw signal. The primary method of mixing is to utilize an affine transformation. The affine transformation includes rotation, scaling and translation.

Both the high frequency EC signal (*hf*) and the low frequency EC signal (*lf*) are first divided into their real and imaginary (resistance and inductive reactance) components.

$$hf = [hf_h \; hf_v]' \text{ and } lf = [lf_h \; lf_v]' \tag{3}$$

The Affine transformation is applied to the low frequency EC signal such that it matches the high frequency signal as close as possible. Then the transformed *lf* signal is subtracted from the *hf* signal. The resulting signal, *Z*, has minimum interference and maximum flaw signal. The procedure is detailed below.

The rotation part of the affine transform is given by the matrix *R*

$$R = \begin{bmatrix} \cos(\tau) & -\sin(\tau) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{4}$$

where: τ = independent (of θ) horizontal rotation.

θ = vertical rotation.

Next, the scaling matrix is given by

$$S = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \tag{5}$$

where α and β are real scalars.

The *lf* signal is then transformed and subtracted from the *hf* signal.

$$Z = hf - R \bullet S \bullet lf = [hf_h \; hf_v]' - \begin{bmatrix} \cos(\tau) & -\sin(\tau) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} [lf_h \; lf_v]' \quad (6)$$

where: $Z = [Z_h\ Z_v]'$.

The four variables are determined such that they minimize a cost function. The cost function is related to the hf and transformed lf signal. The cost function listed below is used for this purpose

$$J(\alpha, \beta, \tau, \theta) = \sum_{k=1}^{l} \sum_{i=h,v} | hf_i(k) - Z_i(k) |^2 \quad (7)$$

where: k = data points (1 through l)

i = vertical or horizontal.

To minimize this cost function, first order partial derivatives with respect to each variable are taken and set equal to 0.

$$\nabla J(\alpha, \beta, \tau, \theta) = 0 \quad (8)$$

or

$$\frac{\partial J}{\partial \alpha} = 0, \frac{\partial J}{\partial \beta} = 0, \frac{\partial J}{\partial \tau} = 0 \, and \, \frac{\partial J}{\partial \theta} = 0 . \quad (9)$$

A gradient descent program is then used to solve this minimization and determine the best values for τ, θ, α and β. The parameters are then used in Equation (4), resulting in the properly mixed ECT signal [19, 20]. A frequency mixing program was generated for the ECTD pre-processing step of the diagnostic approach outlined on pages 3 and 4.

Other specimen-to-probe effects of note include edge effect, fill factor, and lift-off. The edge effect results from the distortion of the magnetic field at the end or edge of the specimen. By decreasing the size of the probe coil or, better, by enclosing the coil in a magnetic shield such as a metal, the area of the specimen inspected by the probe can be decreased so that the edge can be approached closer. This is called focusing the probe. Even so, inspecting at less than 1/8-inch from the edge in non-magnetic materials or within 6 inches of the edge of magnetic materials is likely to produce distorted information. Likewise, the gap between a cylindrical specimen and an encircling coil can greatly affect readings. In general, the closer the specimen comes to filling the hole in the center of the coil, the better the sensitivity (fill factor = 1). Fill factor (FF) is defined as:

$$FF = (D_{SP} / ID_C)^2 \tag{10}$$

where: D_{sp} = diameter of the specimen

ID_C = inner diameter of the coil.

In a similar fashion, any gap between a probe and the surface of a specimen will reduce sensitivity. The lift-off effect can be used to measure the thickness of a non-conducting coating on a conductive material [18].

2.1.3. Analysis of ECT Flaw Signals

The resistance and inductive reactance, generated by one, or a mix, of the excitation frequencies, using a differential probe are plotted in a Lissajous-type plot (examples are shown in the top plots of Figure 3). In the Lassajous plot, the x-axis is the resistance and the y-axis is the the inductive reactance. To determine the phase angle and the magnitude (volt) of the flaw signal using the Lissajous plot, the following steps are used.

1. Identifying the end tips of the figure eight shaped curve.
2. Determine the first tip made.
3. An line is drawn from the first tip to the second tip.
4. The phase angle is the angle the line makes with the negative x-axis.
5. The voltage magnitude is the length of this line.

The phase angle of the mixed differential ECT flaw signal is the most utilized information. An example plot of a mixed differential ECTDFS for a tube flaw is shown in Figure 3.

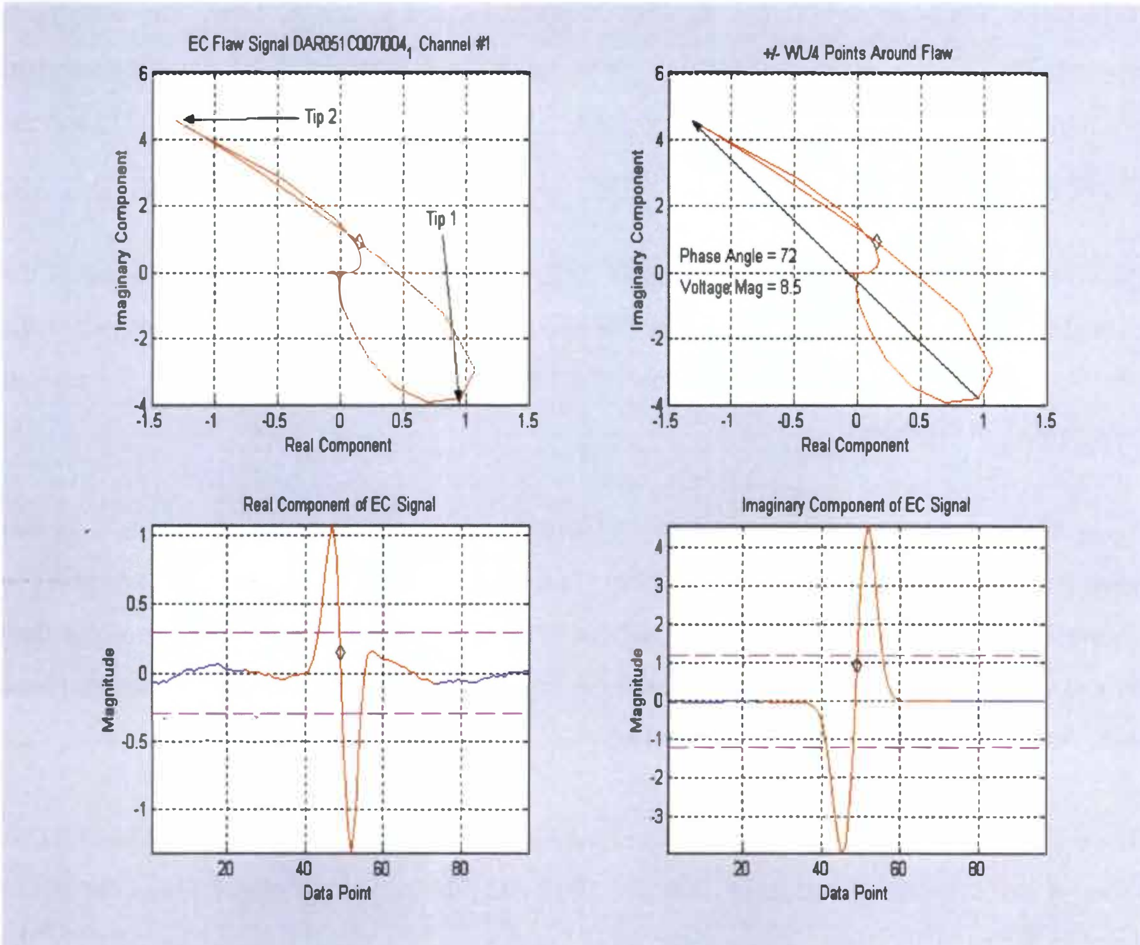

Figure 3. Example of a mixed differential ECTDFS.

Figure 3 contains a Lassajous plot (top left), a Lassajous plot of the data located around the flaw (top right), and the components of the impedance plotted separately (bottom). The middle portion of the signal, the upper right figure, signifies 20 points located around the flaw. The diamond is approximately the center point of the flaw. The dashed lines represent ± 2.5 times the standard deviation (STD) of the real and imaginary parts. The phase angle is approximately 72° and the voltage magnitude is approximately 8.5 Volt.

The voltage magnitude of the mixed complex differential signal is often not used because of the variability of this measurement caused by the relative location of the probe with respect to the tube wall. Thus, the phase angle becomes the major variable used to determine the percent through-wall of the defect.

Figure 4 is an example of the Lissajous-type plot of the complex impedance of a tubular standard specimen. The standard speciman has 5 flaws of varying depth. The depth of the flaw is given in percent through-wall. Percent through-wall (or %TW) is determined by the depth of the flaw divided by the thickness of the tube. Notice, in Figure 4, that as the %TW increases the phase angle decreases (or rotates counter-clockwise).

Figure 5 is a plot of the inductive reactance of the frequency-mixed absolute signal. The ECTD is a mix of two excitation frequencies, 200 and 100 kHz. The maximum magnitude of the mixed signal indicates %TW defect information, while the shape of the signal follows the profile of the flaw. The peaks in the signal were identified, along with their magnitudes (top right). The horizontal line represents 0.75 times the STD of the signal. This threshold is used to extract the information section of the signal. Data above the threshold is used as a profile of the EC flaw.

The usual information obtained from the ECTDFS is the location within the tube of the flaw and the %TW of the flaw. This information is obtained by either analysis of the mixed differential and/or the absolute signals.

A general rule of thumb, when the indications are shallow, the absolute ECT signal tends to perform better; once the indications hit 40% TW the differential signal tends to perform better. The best mix for absolute is half and quarter frequency, and for differential, prime and quarter frequency. You also have to look at the residual from the mix [21].

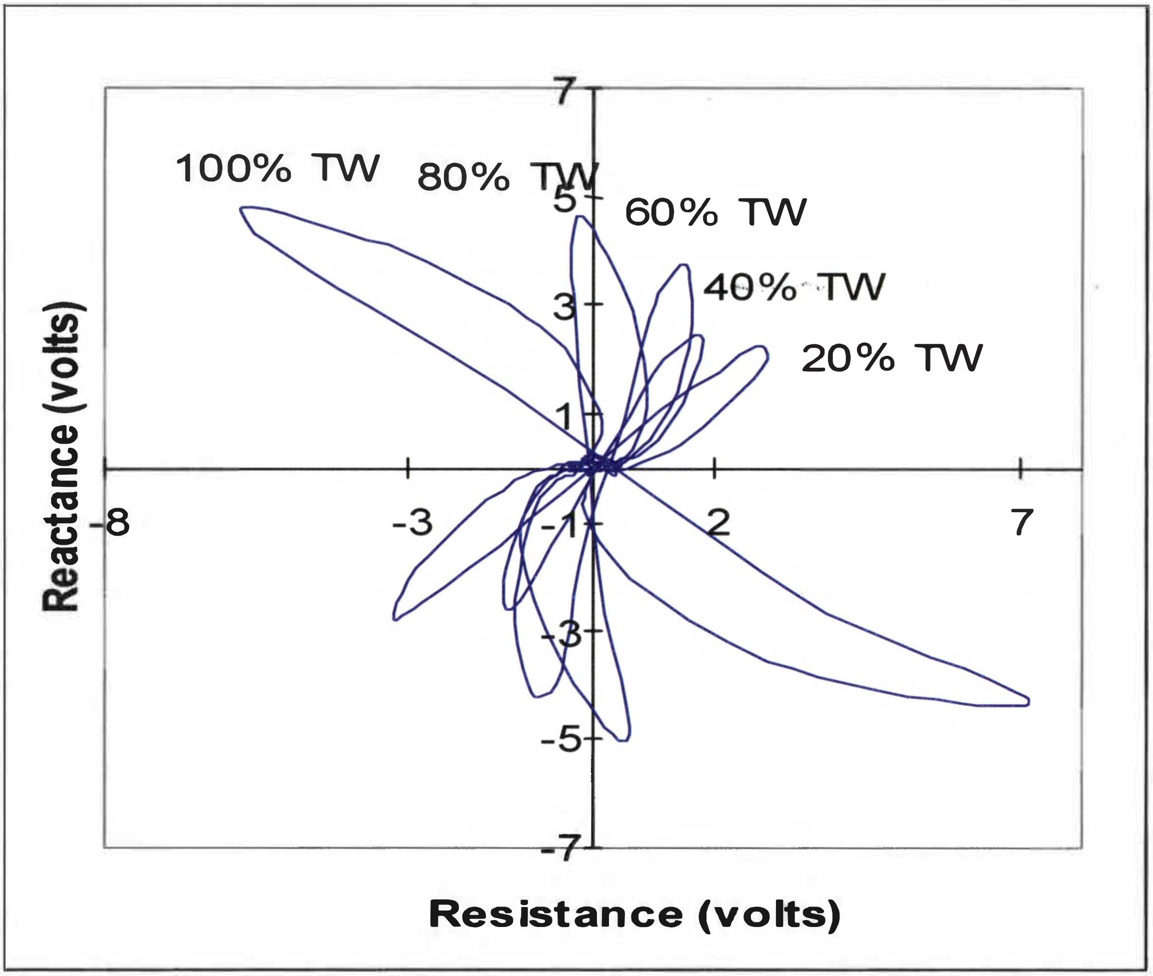

Figure 4. Differential Calibration ECT Data (100, 80, 60, 40 and 20 % Thru-hole) [11].

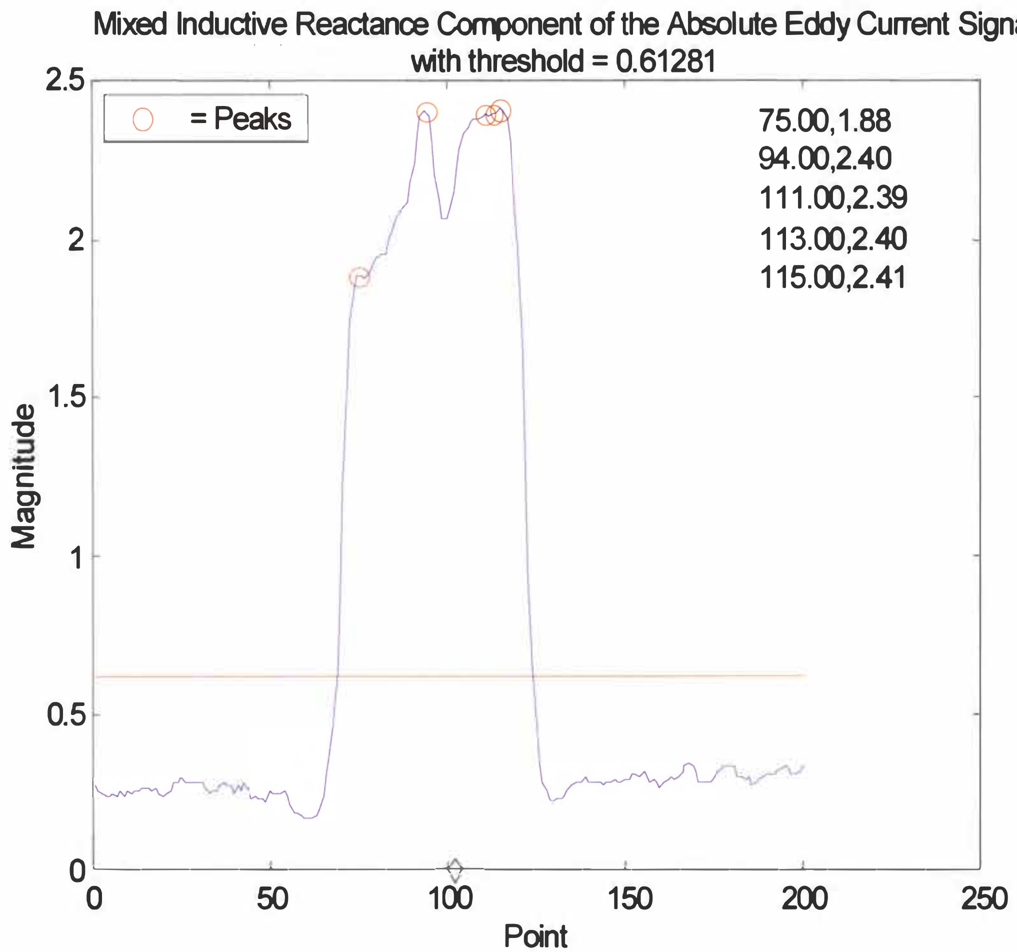

Figure 5. Inductive reactance component of the 200/100 kHz mixed absolute signal.

2.1.4. Advantages and Disadvantages of ECT

The following are the advantages of eddy current testing.

1. The eddy current testing technique can be extremely rapid. Most of these inspections are automated.

    - Tubing wall thickness and integrity can be inspected at 500 ft/min.
    - Ammunition cartridges can be inspected for wall thickness, eccentricity, and cracks on their entire circumference at 6000 per minute.
    - Heat exchanger tubes can be checked for dents, corrosion pitting, and wall thickness at several feet per minute.

2. Sorting of alloys can be accomplished in the field quite easily without great expense or much operator training and experience.
3. Very sensitive flaw detection, particularly for thin material, is possible.
4. The eddy current technique does not require contact with the specimen, which eliminates scratches, tears or other marring of the specimen and allows for rapid testing.
5. ECT can provide a permanent record.
6. Since a large variety of material properties affect eddy currents, many of the physical and metallurgical properties of the specimen can be determined.

The disadvantages of eddy current testing are:

1. Manual testing is very slow.
2. The material being tested must be electrically conductive.
3. Eddy current testing usually requires sophisticated electronic equipment except for very simple testing such as alloy identification. This sophistication translates into high cost, considerable operator training, and complex systems often suitable only for laboratory operation.
4. The technique is sensitive to geometry and shape of the specimen. Depth of penetration, and therefore the depth of discontinuity detection, is poor. A thickness of about ¼-inch is the maximum useful depth of penetration for most materials. The frequency of excitation

of the coil is important because it limits the useful depth of penetration. High frequencies give less depth of penetration than low frequencies.

5. Interpretation is sometimes difficult because specimen conductivity and magnetic permeability are responsive to so many material properties.

Since ECT is widely used, the advantages must outweigh the disadvantages [16, 17, 18].

2.2. Steam Generator Information

There are three Steam Generator (SG) manufacturers represented in EPRI's PDD [22]. The three manufacturers are Babcock & Wilcox (B&W), Combustion Engineering (CE) and Westinghouse. This section describes SG information from these three manufacturers.

The first section describes the types of tubing flaws that occur in the SG, with the second section providing locations where the flaws occur for specific steam generators.

2.2.1. Types of SG Tubing Flaws

There are nine specific SG flaw types

1. Cracking
2. Thinning
3. Wear
4. Impingement
5. Intergranular Attack (IGA)
6. Stress Corrosion Cracking (SCC), either Primary-Water (PWSCC) or Outer-Diameter (ODSCC)
7. IGA/SCC
8. Pitting
9. Denting [22].

Along with these flaw types, one must recognize that there is also the possibility of

1. No Defect
2. Multiple Defects (a combination of two or more of the above nine flaws at a specific point)
3. Undetermined.

Thus, there could be 12 different flaw types. The following flaw types will not be detailed in Section 2.2.3. These flaws were not available for processing.

1. IGA

   - Corrosion attack at grain boundaries, usually not stress related
   - Propagation (or fingers)
   - Function of Temperature

2. SCC (PWSCC or ODSCC)

   - Corrosion attack at grain boundaries, stress related
   - Propagation (or fingers)
   - Function of temperature

3. IGA/SCC

   - Combination of IGA and SCC
   - Fingers with loss of volume

4. Fatigue

   - Cracking caused by alternating stress cycles accelerated by corrosion

5. Denting – Self explanatory [22].

Figure 6 shows typical tubing flaws and their location in U-tube steam generators. Not all the different flaw types are shown in Figure 6.

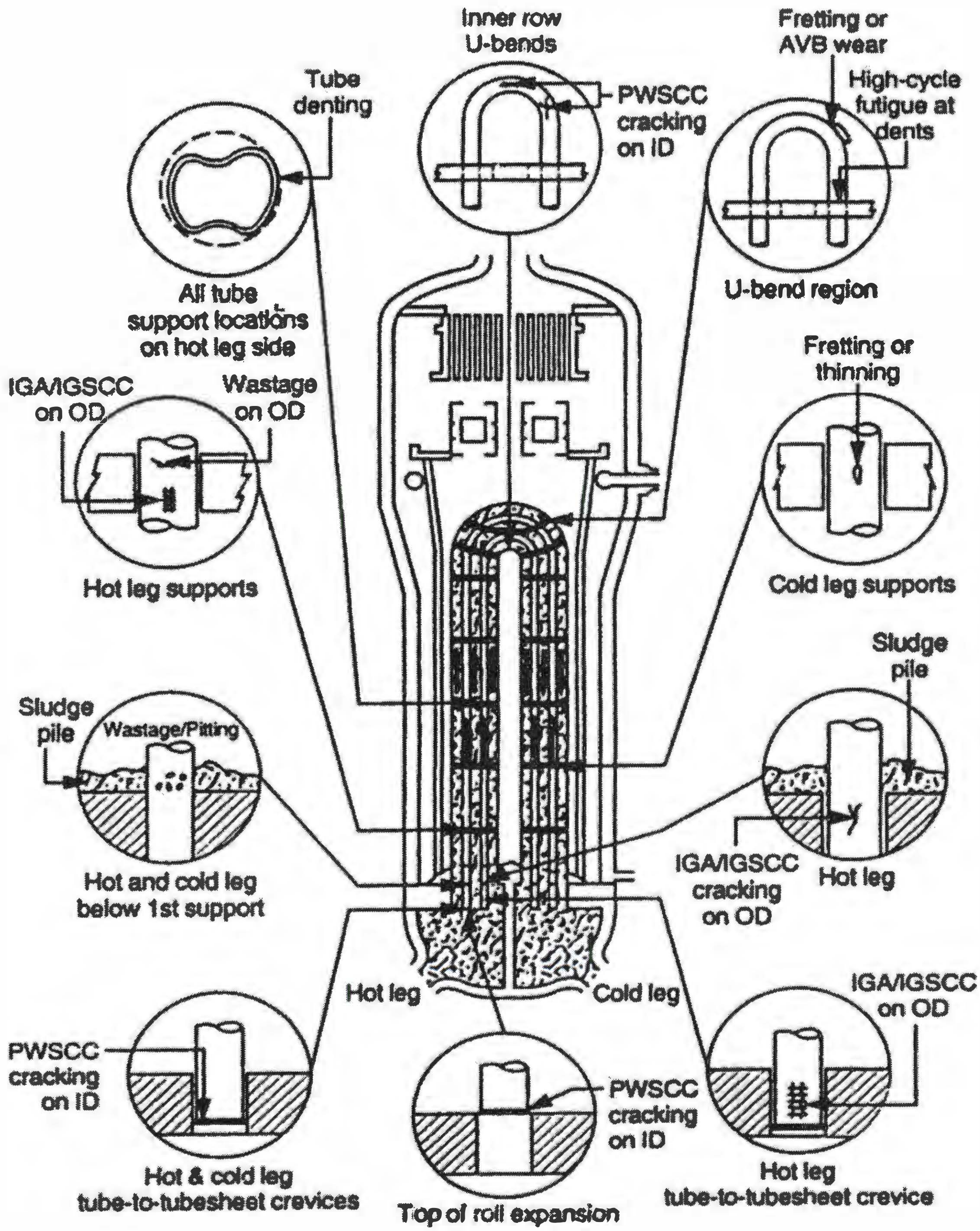

Figure 6. Location of Tubing Flaws in a U-tube steam generator [1].

2.2.2. Location and Flaw-type Information for Specific Manufacturers

Realizing that individual manufacturers do not exhibit all the different flaw-types and that the flaw-types sometime occur in a specific region, classification of unknown flaws according to flaw-type can be simplified. Table 2 illustrates the subsets of flaws for each manufacturer. B & W exhibits only four flaw-types, while CE exhibits five and Westinghouse exhibits six.

Table 3 summarizes position verse flaw-type for B&W Steam Generators as detailed in EPRI's PDD. The 10th location does not contain a flaw. This table clearly shows that specific flaw-types occur at specific regions within the B & W SG. This information may be used to classify the ECT flaw since the flaws location is known.

Again, as done previously in Table 3, Table 4 was organized to show the relationship between location and flaw-type for CE SGs. Depending on the location of the flaw, the flaw-type may be further narrowed from five to three at most. Some regions only exhibit one flaw-type.

The Westinghouse information is subdivided according to specific SG models. This information is given in Tables 5 – 8.

There is a strong relationship between flaw-type and the location in the SG manufactured by Westinghouse, similar to that indicated by B&W and CE steam generators. Therefore, if the ECTD are generated from SG tubing, the tables in this section would allow the narrowing of classification as a function of position.

2.2.3. EPRI's Performance Demonstration Database ETSS Subset

As outlined in Section 1.1, the ECT data used for this research were acquired from EPRI. The acquired database is part of the Performance Demonstration Database (PDD) [22] maintained by EPRI. The subgroup of data that was chosen for analysis was PDD data that included ETSS and blueprints. The blueprints were needed so that flaw characterization could be expanded from only a %TW to include other dimensions.

Table 2. Steam Generator Tube Degradation by Manufacturer [22].

		Corrosion				Mechanical		
		Thinning	**Pitting**	**IGA/SCC**	**PWSCC**	**Fatigue**	**Wear**	**Impinge-ment**
SG Manufacturer	**Babcock & Wilcox**			X		X	X	X
	Combustion Engineering	X	X	X	X		X	
	Westinghouse	X	X	X	X	X	X	

Table 3. Position vs. Flaw-type for B&W SG's [22].

<table>
<tr><th></th><th></th><th colspan="9">Location (Listed in Notes)</th></tr>
<tr><th></th><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th></tr>
<tr><td rowspan="6">Flaw-type</td><td>IGA/SCC</td><td></td><td>X</td><td>X</td><td rowspan="6">Not a Location</td><td rowspan="6">Not a Location</td><td></td><td></td><td>X</td><td></td></tr>
<tr><td>Wear</td><td>X</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Impingement</td><td></td><td></td><td></td><td>X</td><td></td><td></td><td></td></tr>
<tr><td>Fatigue</td><td></td><td></td><td></td><td></td><td>X</td><td></td><td></td></tr>
<tr><td>PWSCC</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Denting</td><td></td><td></td><td></td><td></td><td></td><td></td><td>X</td></tr>
</table>

Notes

(1) Upper span region of steam generator

(2) Predominately within the upper tubesheet crevice

(3) Minor IGA observed on a single pulled tube within the lower tubesheet crevice

(4) Diagnosed by eddy current

(5) Occurrence not related to operation

(6) Mostly confined to outer periphery tubes at the 9th support plate elevation

(7) Lane region

(8) Upper tubesheet crevice

(9) Diagnosed at broached or drilled tube support plates, or tubesheet

(10) Lower span.

Table 4. Position vs. Flaw-type for CE SG's [22].

		Location (Listed in Notes)									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Flaw-type	**Thinning**	X	X								
	Wear			X	X	X					
	IGA/SCC	X	X	X			X	X		X	X
	PWSCC						X				
	Pitting	X									
	Denting	Location not Designated									

Notes

(1) Sludge pile

(2) Eggcrates/support plate

(3) Vertical supports

(4) Batwings

(5) Cold-leg corner

(6) Top of tubesheet (expansion)

(7) Inner row U-bends

(8) Freespan manufacturing defects

(9) Associated with copper

(10) Freespan horizontal and vertical runs.

Table 5. Position vs. Flaw-type for Westinghouse SG's (24, 27, 33 & 44) [22].

		Location (Listed in Notes)					
		1	**2**	**3**	**4**	**5**	**6**
Flaw-type	**Thinning**	X					
	Wear		X				
	IGA/SCC	X		X	X	X	
	PWSCC					X	X
	Pitting	X					
	Fatigue	Location not Designated, only observed at 2 Units					
	Denting	Location not Designated					

<u>Notes</u>

(1) Sludgepile

(2) AVB's

(3) Tubesheet crevice

(4) Support plates

(5) Roll transition

(6) Inner row U-bends

Table 6. Position vs. Flaw-type for Westinghouse SG's (51 S/G) [22].

		Location (Listed in Notes)								
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Flaw-type	**Thinning**	X			X		X			
	Wear		X							
	IGA/SCC			X	X		X			
	PWSCC					X		X	X	X
	Pitting						X			
	Fatigue	Location not Designated, Only 1 Unit affected								
	Denting	Location not Designated								

Notes

(1) Cold-leg outer periphery support plates

(2) AVB's

(3) Tubesheet crevice

(4) Tube support plates

(5) Inner row U-bends

(6) Sludgepile

(7) Transition or expansion

(8) Row 1's plugged

(9) Dented supports

Table 7. Position vs. Flaw-type for Westinghouse F-Type Units [22].

		Location (Listed in Notes)			
		1	**2**	**3**	**4**
Flaw-type	**Wear**	X			
	IGA/SCC		X	X	X
	PWSCC			X	X
	Denting		X		

Notes

(1) AVB's

(2) Sludgepile

(3) Top of tubesheet

(4) Mill annealed tubing affected

Table 8. Position vs. Flaw-type for Westinghouse Framatone Units [22].

		Location (Listed in Notes)				
		1	**2**	**3**	**4**	**5**
Flaw-type	**Wear**	X				
	IGA/SCC		X	X	X	
	PWSCC				X	X
	Denting	Location not Designated (many units affected)				

Notes

(1) AVB's

(2) Support Plate

(3) Sludgepile

(4) Extension transition

(5) Inner row U-bends

The information listed in Table 9 is a summary of the PDD-subgroups with ETSS information and flaw descriptions for the data groups (flaw-types) used. The four data groups were 96001, 96002, 96004 and 96005.

In Figures 7 – 10, examples of flaw drawings (blueprints) are given for each of the four data groups. Notice that for Wear flaw-types, only two characteristics are given. The blueprints indicate the differences between flaw geometries. For various reasons, only 92 examples were used.

2.2.4. ECTD Preprocessing with EddyM and EddyC Start-up

The EddyM MATLAB program was generated by Hopper [12]. This program was designed to characterize EPRI's PDD ECTDFS within a MATLAB framework. The EddyM program generates two Graphical User Interfaces (GUIs). One GUI displays the full data file and the second displays a windowed section. Within the GUIs are two built-in preprocessing functions, mixing, and dedrifting and denoising.

Since the ETSS data subset was generated in the laboratory, dedrifting and denoising were not needed. But, there were two ECTD preprocessing tasks that were needed using the EddyM program:

1. Locating the flaw within the data file and
2. Mixing.

Once these two tasks were accomplished, the EddyC MATLAB program was initiated.

The EddyC MATLAB program initiates the EddyC system. The tasks accomplished by the EddyC system are detailed, using MATLAB command window inputs and outputs, in Appendix F. The MATLAB programs used to generate and operate the EddyC system are given in Appendix G.

Table 9. PDD Sub-groups with ETSS Information [22].

		PDD Sub-group with ETSS Information			
		96001 (Thinning)	96002 (Impingement)	96004 (Wear)	96005 (Pitting)
Description	**Shape**	Long Rectangular	Candle-flamed	Short Rectangular and Triangular	Oval
	Caused by	Water chemistry	Solids in coolant or liquid hitting solids	Mechanical action between two materials	Galvanic attack
	Affects	All tube material	All tube material	All tube material	All tube material
	Best Mix*	400/100 Diff 200/100 Abs	600/400 Diff 200/100 Abs	400/100 Diff 200/100 Abs	400/100 Diff 600/200 Abs
	Total Examples (Number Used)	26 (25)	29 (21)	92 (24)	61 (22)
	Flaw Dimensions (Characterization)	3	3	2	3

***Best:** The best is defined as the mix that yields the highest correlation (R^2) and the lowest RMSE (root mean squared error) as determined by the linear model developed using the ECT determined %TW and the measured (actual) %TW.

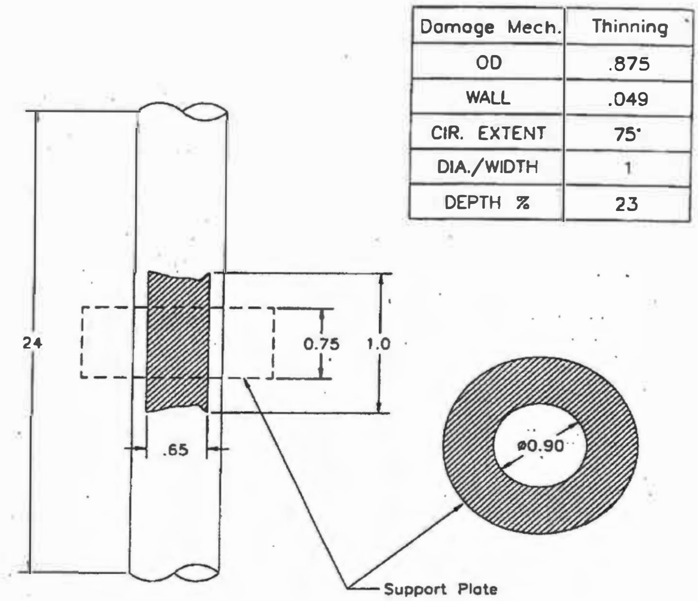

Figure 7. Example Blueprint of Thinning Flaw [22].

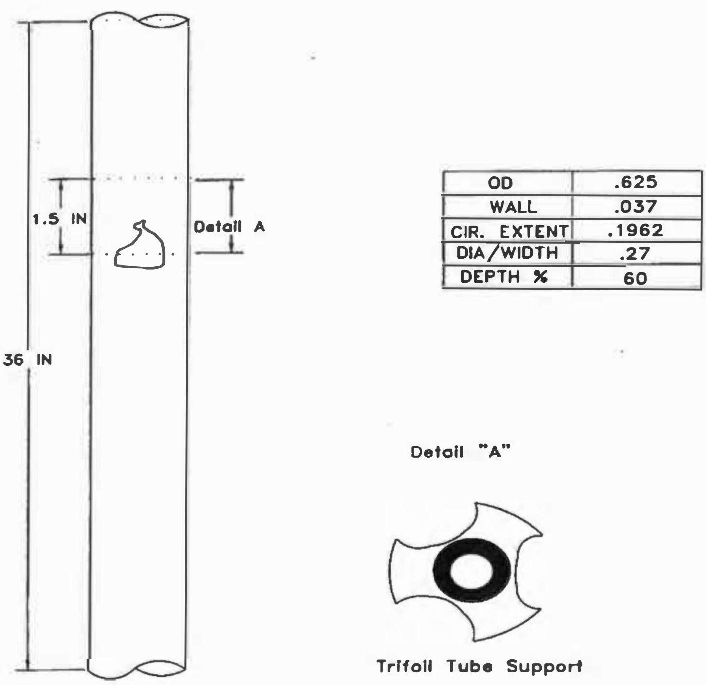

Figure 8. Example Blueprint of Impingement Flaw [22].

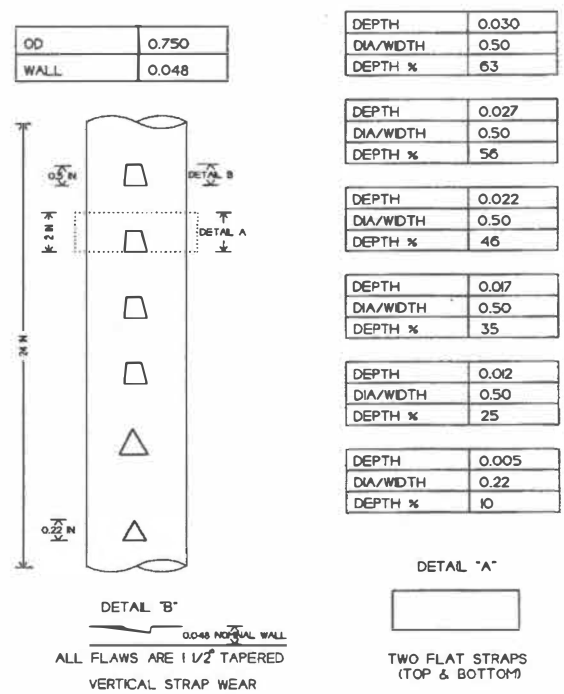

Figure 9. Example Blueprint of Wear Flaw [22].

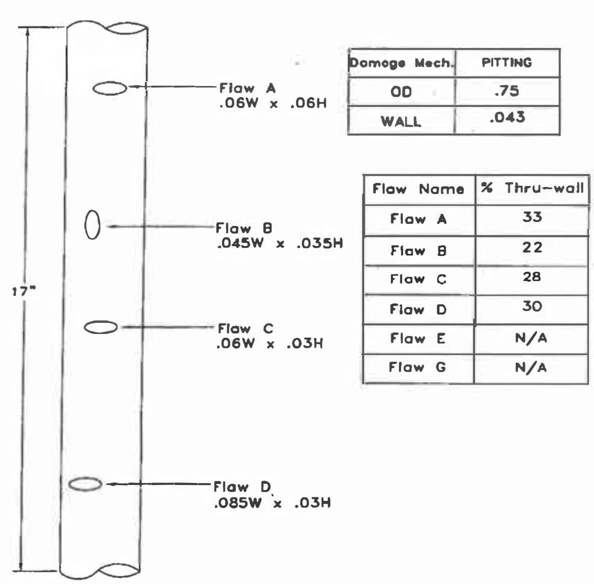

Figure 10. Example Blueprint of Pitting Flaw [22].

Throughout both processes, the EddyC system generates four ".mat" files with various types of information. Those four files are identified as

1. Basic Information (E_9600 …) Files
2. Stacked Basic Information (uTR_ …) Files
3. Compressed Processed Information (TR) Files
4. ANN Information (net_char_ …) Files

Each type of information file is detailed in the following chapters.

3.0. Eddy Current Test Data Transformation using the Continuous Wavelet Transformation (CWT)

As seen in the bottom two graphs of Figure 3, the ECTDFS is non-stationary or transient. The limited experience of previous investigations and the waveform property of the ECT signals indicated that CWT would yield better results than the traditional PSD or the STFT because the CWT is more effective in compressing short time samples (transient) and non-stationary waveforms.

This section is divided into three parts. The first part is an overview of CWT theory. The second part briefly describes the method of selecting a mother wavelet or transformation based on the ECTD flaws. The third section contains ECTD generated CWTs with an emphasis on determining usual and unusual ECTD flaw representations.

3.1. Signal Processing using the CWT

The CWT is a signal processing method that extracts time and frequency (scale) information from the ECT signal [22]. The CWT was formalized by A. Grossman and J. Morlet in 1984 [23]. The wavelet function $\psi(x) \in L^2(R)$ has two characteristic parameters, namely, dilation (a) and translation (b), which vary continuously. A set of wavelet basis function $\psi_{a,b}(x)$ is defined as

$$\psi_{a,b}(x) = \frac{1}{\sqrt{|a|}}\psi(\frac{x-b}{a}) \qquad a, b \in R\,;\, a \neq 0 \tag{11}$$

Here, the translation parameter, b, controls the position of the wavelet in time. The parameter, a, controls the dilation of the wavelet. A "narrow" wavelet can access high-frequency information, while a more dilated wavelet can access low-frequency information. This means that the parameter a varies for different frequencies. The CWT is defined as

$$W_{a,b}(f) = \langle f, \psi_{a,b} \rangle = \int_{-\infty}^{+\infty} f(x)\psi_{a,b}(x)dx \,. \tag{12}$$

The wavelet coefficients are given as the inner product of the function being transformed with each basis function [22, 23, 24, 25, 26, 27].

In order to plot the CWT of a complex signal, the absolute values of the coefficients are plotted as a function of time and scale a. An example plot is given in Figure 11. Clearly, the flaw in Figure 11 exhibits scale (frequency) and peak geometry characteristics that, if extracted properly, may provide valuable information relative to classification and characterization of the ECDT flaws.

3.2. Mother Wavelet Selection for the Eddy Current Test Data

To extract the most information from the ECTD using CWT, the best mother wavelet was selected. Two parameters were used to determine the best mother wavelet for CWT of the ECTDFS. The first parameter was entropy and the second was the residuals. Both parameters are determined by applying a discrete wavelet transform (DWT) to the ECT signal examples.

The entropy was calculated for each level of the DWT for each mother wavelet. The entropy values are compared between the mother wavelets at a specified level. The mother wavelet that produces the minimum entropy value, at a specified level, was selected. The entropy that was calculated was the first norm entropy. The first norm entropy was given as:

$$E_{norm1} = \sum_i | x_i | \tag{13}$$

where: x = signal

i = each signal value.

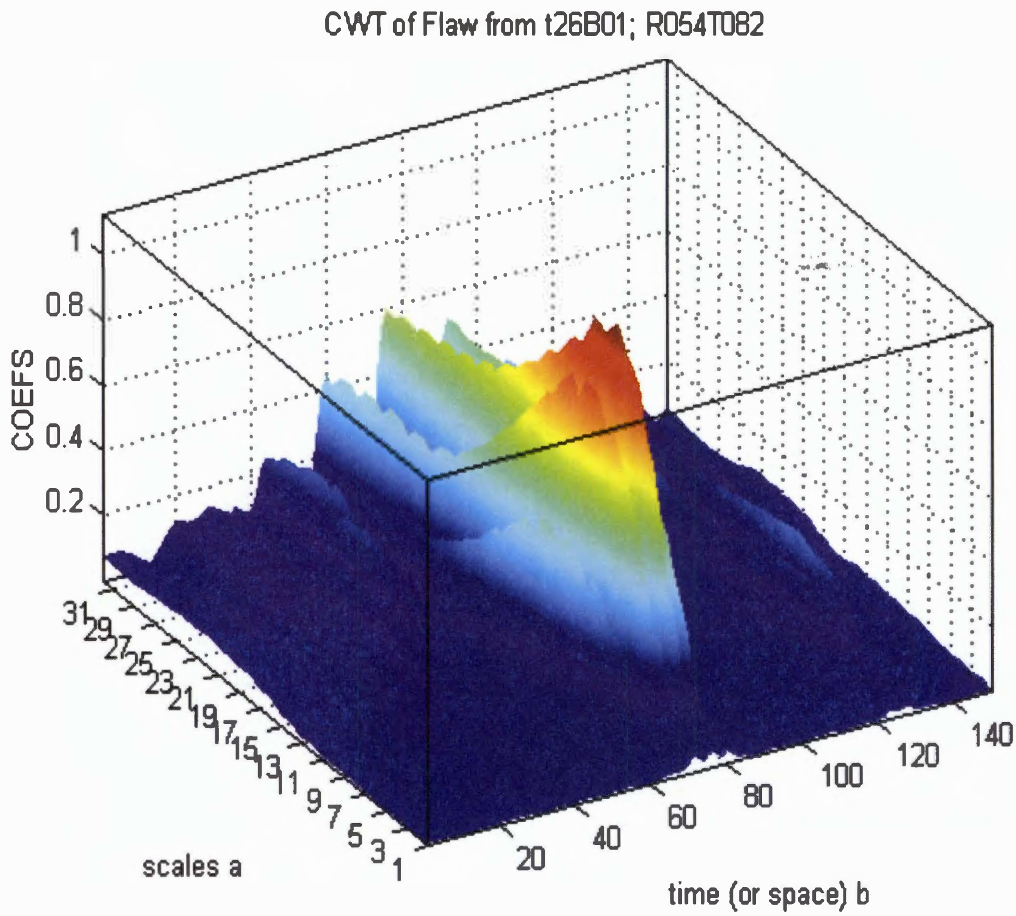

Figure 11. Absolute value of the CWT of the 400/100 kHz mixed complex differential signal of a flaw located near a support structure.

The residuals are calculated using the sum of the absolute value of the difference between one level and the next for each specified mother wavelet. The residual of L1 and L2 (Rs_{L1L2}) is mathematically given as:

$$Rs_{L1L2} = \sum_{i} | L1_i - L2_i | \tag{14}$$

where: $L1$ = higher level of decomposition

$L2$ = lower level of decomposition

i = each value of L1 and L2 [23].

The best mother wavelet, using the residuals, was one in which the residuals were minimum for that flaw-type, at the comparison levels. The results were tabulated in Table 10. The results were mixed. A bi-orthogonal level 3.5 ("bior3.5") was used.

3.3. Initial review of the CWT

An initial review of the generated CWTs was crucial to identify ECTD flaw examples that may be non-typical for that flaw-type. A non-typical CWT may cause the ECTD flaw example to be an outlier after the features were extracted and compressed. A list was generated identifying the non-typical CWTs for each flaw-type.

Section 3.3 is divided into two parts. The first part details typical ECTDFS generated CWTs for each flaw-type. The second section lists non-typical ECTDFS example CWTs generated for each flaw-type.

3.3.1. Typical CWTs of ECTD Flaws

The typical CWT for a group was selected visually by examining all the CWT examples for that group. A typical CWT is one which resembles many of the other CWTs in that group. The following five CWTs (Figures 12 through 16) seem to be typical for each flaw-type.

Table 10. Mother Wavelet Determination using Entropy and Residual Calculations.

		Entropy				Residuals		
		Signal	A2	D2	D1	S & A2	A2 & D2	D2 & D1
Flaw-type	Crack	1.18e04	9.67e03 db1	2.06e03 db3	0.57e03 db10	0.25e04 db3	1.14e04 bior2.2	0.23e04 db3
	Thinning	68.08	67.34 db1	2.53 bior3.5	0.81 db8	2.76 bior3.9	67.19 bior3.1	2.90 bior3.5
	Pitting	9.90e03	9.71e03 db1	0.36e03 db10	0.11e03 db4	0.40e03 db10	9.67e03 bior2.2	0.40e03 db10
	IGASCC	75.95	75.01 bior3.3	4.84 bior5.5	0.84 bior5.5	5.07 bior5.5	74.75 bior3.3	5.08 bior5.5

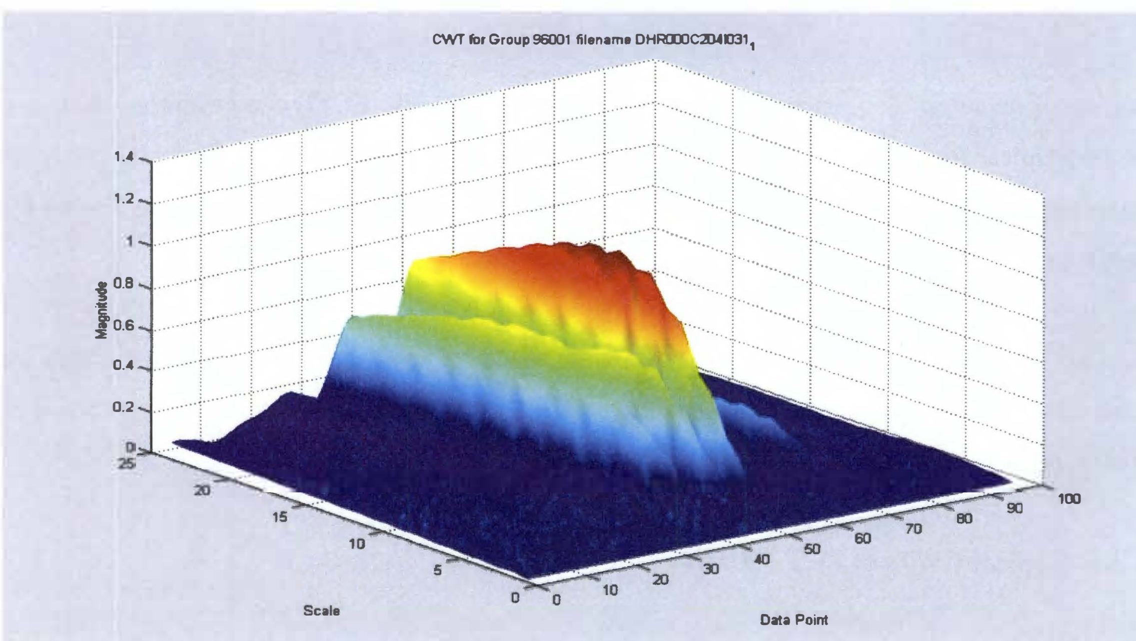

Figure 12. Typical CWT for Data Group 1 (Flaw-type Thinning).

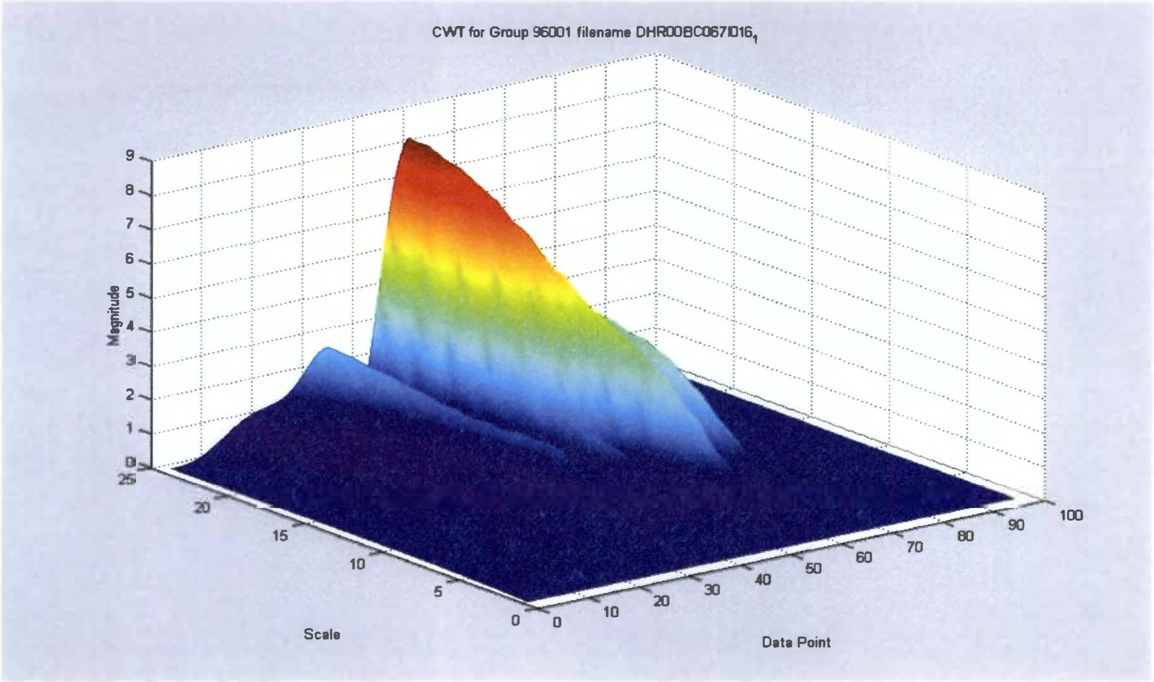

Figure 13. Another Typical CWT for Data Group 1 (Flaw-type Thinning).

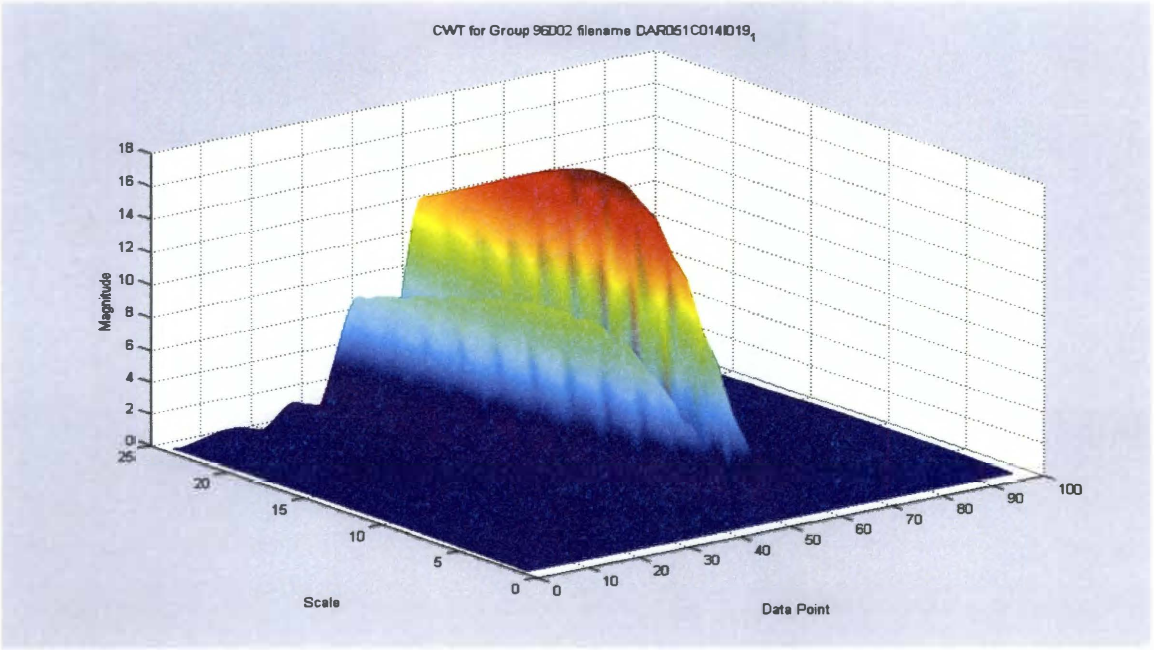

Figure 14. Typical CWT for Data Group 2 (Flaw-type Inpingement).

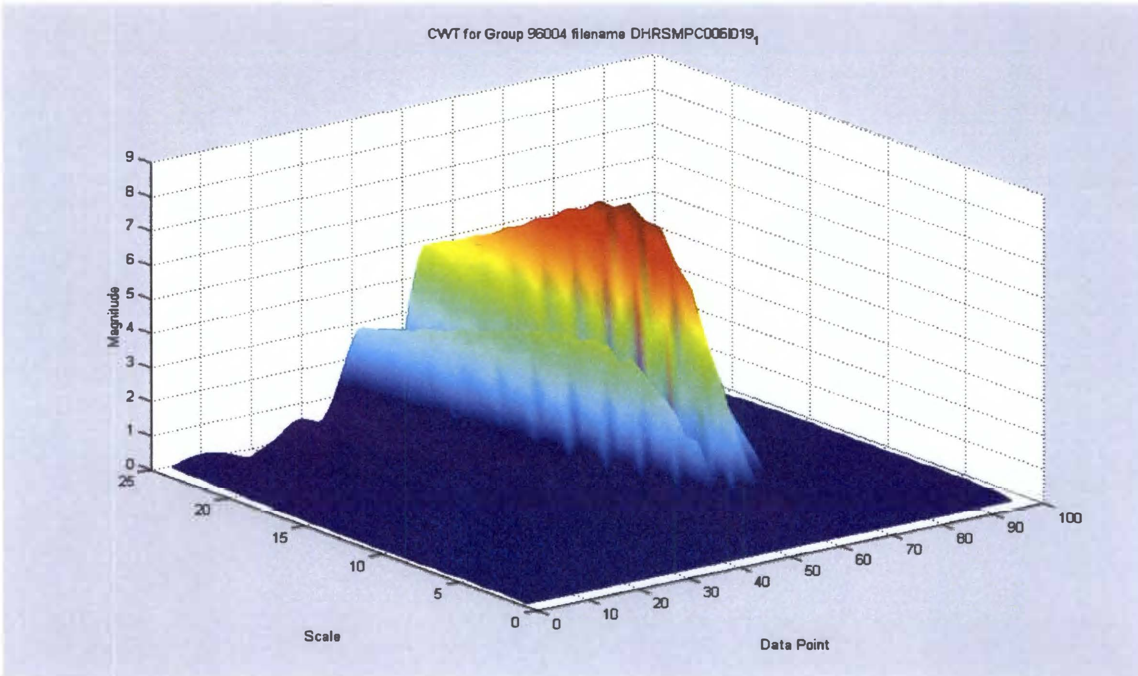

Figure 15. Typical CWT for Group 3 (Flaw-type Wear).

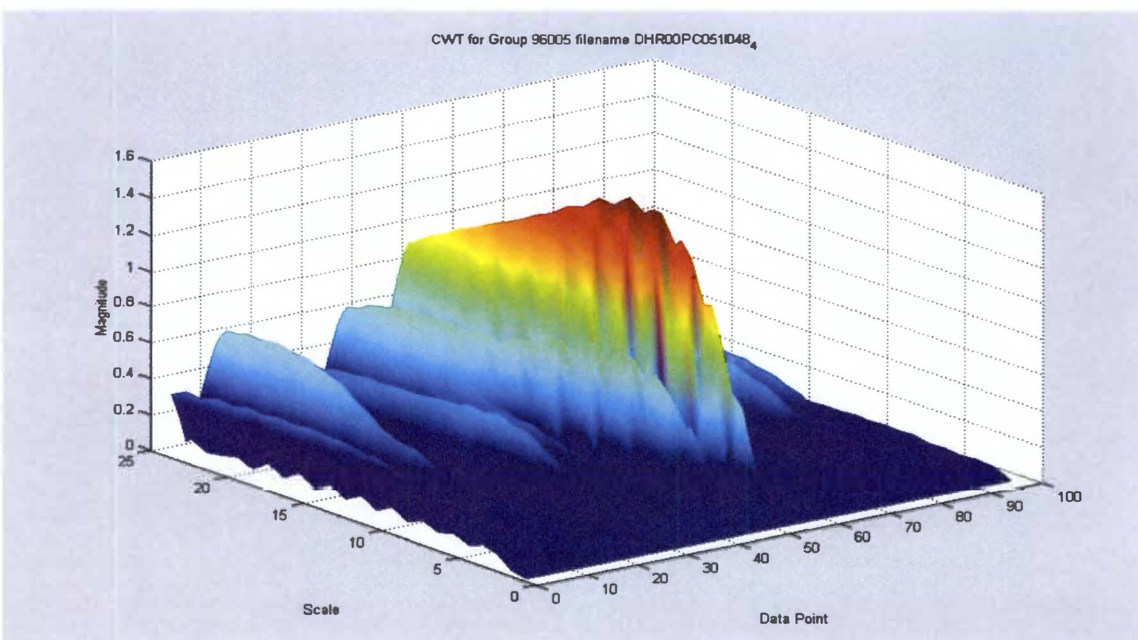

Figure 16. Typical CWT for Group 4 (Flaw-type Pitting).

Group 1 (or Thinning) seemed to have two typical CWTs. There were approximately equal numbers of each of these CWTs. Almost all of the CWTs for every flaw-types seemed visually similar. The one exception was the CWT for the second thinning flaw-type (Figure 13).

3.3.2. Non-typical CWTs of Eddy Current Test Data Flaw Signals

As stated previously, if a CWT was dissimilar from other CWTs within the group, the dissimilar CWT may be an outlier (non-typical) and may distort the results. By visually comparing the generated ECTDFS CWTs, unusual results were noted as follows in Table 11. Examples of dissimilar CWTs are given in Appendix B.

Even though the above CWTs were identified as non-typical, all the ECTD were used. The PDD ETSS subset did not have enough samples for non-typical example extraction. Also, by including the non-typical examples, the database simulates a real-world situation more closely.

Table 11. Unusual Results from Visual Comparison of the CWTs.

	Data Group (or Flaw-type)			
	1 (DHR000C … or DHR00BC …)	**2 (DAR0BWC …)**	**3 (DHRSMPC …)**	**4 (DHR00PC …)**
Non-Typical CWTs	009I023_1 202I032_1 062I021_1 063I009_1 066I006_1 075I011_1 077I015_1 078I004_1	080I018_1	001I004_1 001I004_3 005I016_3 008I025_1	048I063_3

4. Feature Extraction and Compression

The chapter is divided into two sections. The first section details the feature components extracted from the ECTD signal. The second section describes Principal Component Analysis and its usage to compress the feature components.

4.1. Feature Extraction

There were three types of features that were extracted from either the ECTDFS or the CWT generated from the ECTDFS. The first type of feature was extracted from both the inductive reactance component and the complex mixed differential ECTDFS. The second feature was extracted from the mixed absolute ECTDFS. The third type of feature was extracted from the CWT of the mixed differential ECTDFS.

4.1.1. Feature Extraction Technique for the Inductive Reactance and Resistance Components of the Differential ECTDFS

The following features were extracted using the mixed, differential ECTDFS.

1. Phase angle of the complex ECTDFS.
2. Magnitude of the complex ECTDFS.
3. Number of data points between the first and the last peaks of the inductive reactance component ECTDFS.
4. Magnitude between the peak values divided by STD for the inductive reactance component of ECTDFS.

The first two features are discussed in Section 2.1.3. The number of data points between the first and last peak contains information about the relative length of the flaw. The magnitude between the peak values divided by the STD may contain flaw volume information. Thus, two feature vectors were generated, one containing the first two elements and another one containing the last two.

Figure 17 shows an example plot obtained when using the EddyC.m program in the differential feature extraction section. For the example signal in Figure 17, feature #1 equals 50 (data points between the peaks, 74 – 24), and feature #2 = (1.69 - (-0.57)) / (RMS value of the signal).

4.1.2. Polynomial Function Approximation

PFA of the inductive reactance component of the mixed absolute ECTDFS was used to give a characterization of the profile of the signal. The polynomial function has the form

$$f(x) = p_1 x^n + p_2 x^{n-1} + \ldots + p_n x + p_{n+1} \tag{15}$$

The coefficients $\{p_1, p_2, \ldots$ and $p_{n+1}\}$ of the approximating polynomial were used as features [28].

Figure 18 is an example of a polynomial fit of an ECTDFS. As is seen in the figure, the polynomial fit matches the shape of the actual signal. The number of polynomial coefficients needed to fit the data was 18. The sum of squares of the residual was approximately 0.02.

4.1.3. Feature Generation Using Continuous Wavelet Transform (CWT)

Four techniques were used to characterize (or extract features from) the CWT. The first technique was to calculate the geometric moments. The other three IP techniques described in Sections 4.1.3.3-4.1.3.5) utilize features generated from a binary image, they are described in the MATLAB Image Processing Toolbox [29].

This section is subdivided into four parts. The first part details generating geometric moments. The second section describes converting a CWT into a binary CWT. The last two parts describe the IP features extracted from the binary CWT.

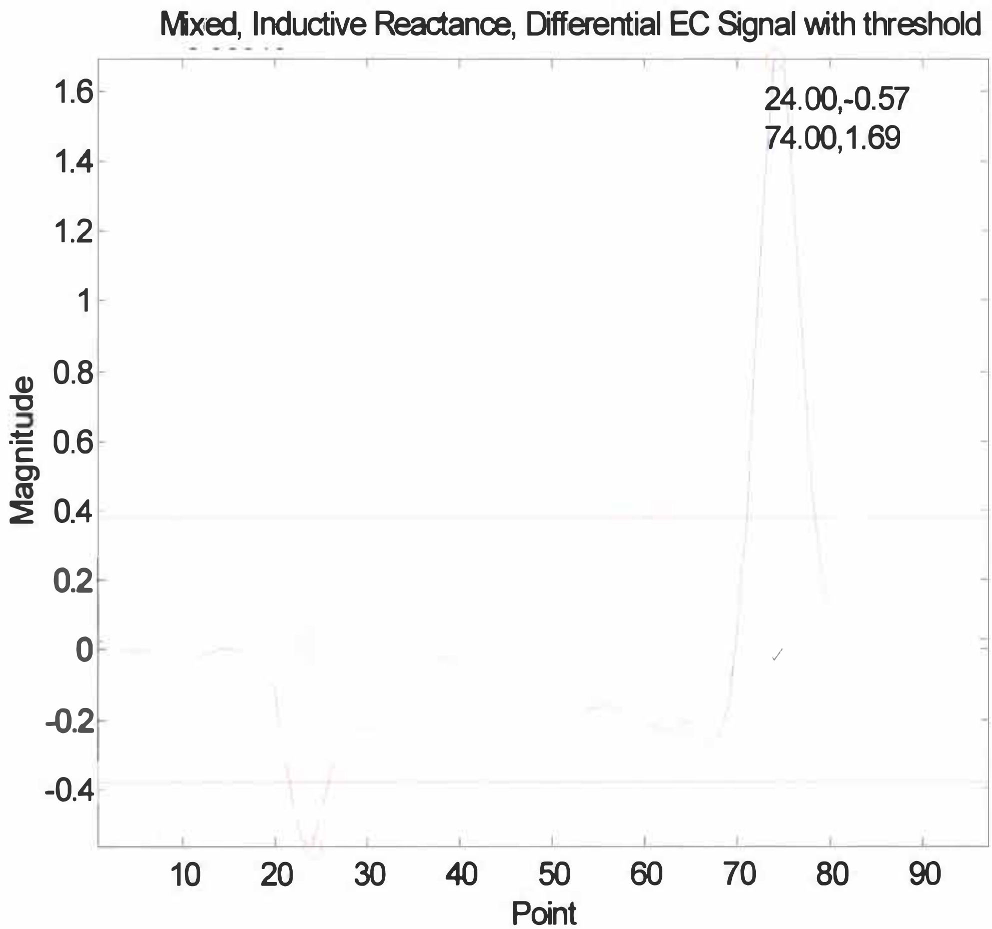

Figure 17. Mixed, inductive reactance component of the differential ECTDFS.

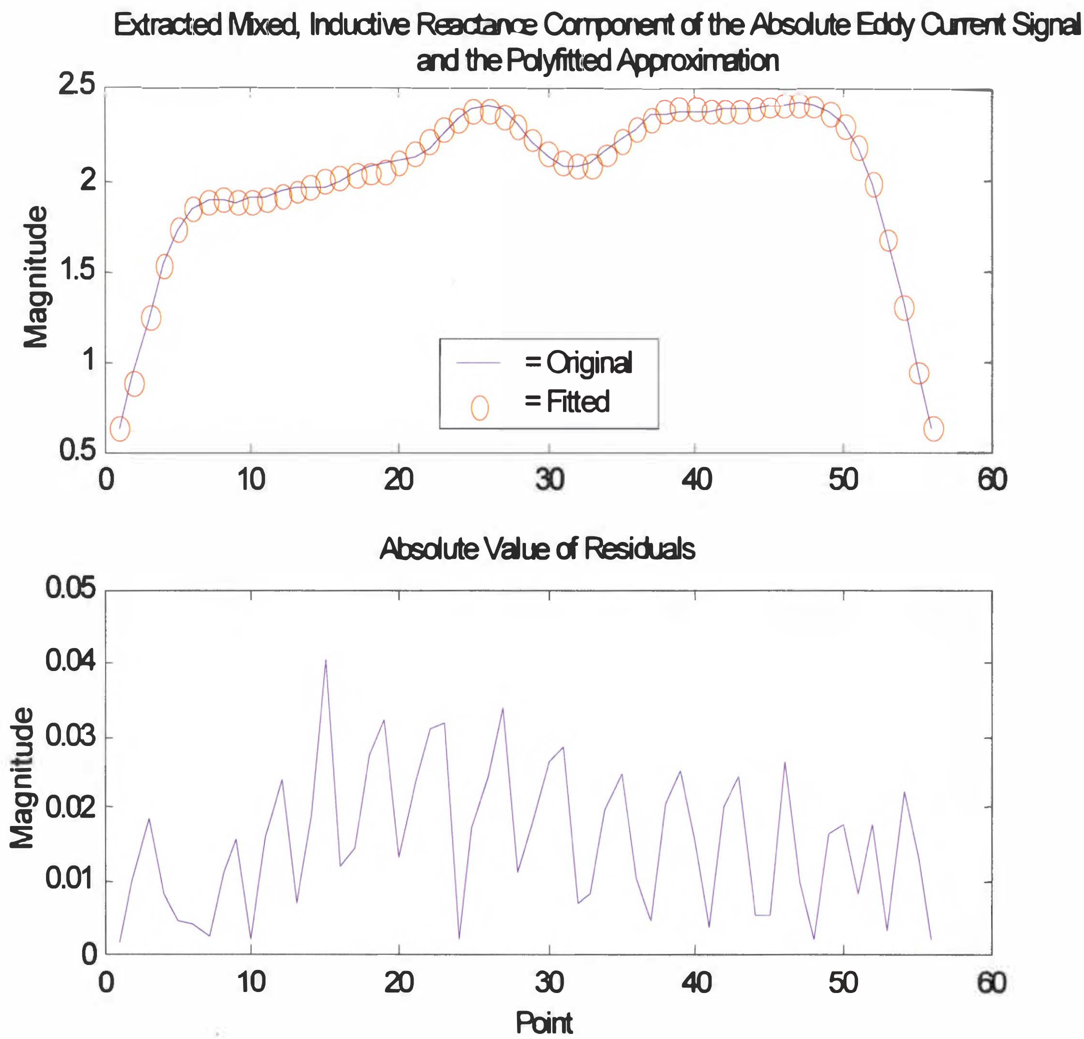

Figure 18. Comparison of the polyfitted signal to the original and the absolute value of the residuals.

4.1.3.1. Geometric Moments

Geometric moments provide rich information about the image and are popular features for pattern recognition [30]. Geometric moments are used for 2-D images whose intensities are a function of x and y. The geometric moment is defined as

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q I(x,y)\,dxdy \tag{16}$$

where: m_{pq} = the geometric moment of order $p + q$

$I(x,y)$ = continuous image function.

The moments depend on the coordinates of the object of interest within the image; thus, they lack the invariance property. The geometric moments may be transformed such that the moments will be translation invariant. The transformation was given by the central geometric moments:

$$\mu_{pq} = \iint I(x,y)(x-\bar{x})^p(y-\bar{y})^q\,dxdy \tag{17}$$

with: $$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}}. \tag{18}$$

The geometric moments are calculated for a discrete image as shown in Equations (19, 20).

$$\mu_{pq} = \sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} I(i,j)(i-\bar{i})^p(j-\bar{j})^q \tag{19}$$

where: $$\bar{i} = \frac{\mu_{10}}{\mu_{00}}, \bar{j} = \frac{\mu_{01}}{\mu_{00}}. \tag{20}$$

This may be well suited for use with transformed data if the position of the object with the image is not important [30]. For this application, the scale (or y) information was very important and was not be invariant. The location (or x) was invariant. Thus, the transformation used was

$$\mu_{pq} = \sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1} I(i,j)(i-\bar{i})^p j^q \tag{21}$$

This caused the μ_{10} value to be always equal 0. This was the only generated feature that was extracted from the feature matrix due to invariance. For this application, the 0 through 4th moments were generated [8]. This resulted in 25 geometric moments.

4.1.3.2. Conversion to a Binary CWT for Image Processing

The three other features (a weighted area, the Euler number and the Roundness Ratio) perform ridge characterization on the binary CWT. The binary CWT was constructed by thresholding the CWT and assigning a 0 value if the CWT value was less than the threshold and a 1 if the CWT value was greater than or equal to the threshold. Example CWTs were visually inspected as the threshold was changed. Information contained in the ridges of the binary CWT (shapes of the ridges and lack of noise) was used to determine the threshold. The threshold was determined, by this inspection, to be 1 times the 2D STD (std2.m) of the binary CWT. These three features are commonly used in image processing and are described in Sections 5.3.3.3-5.3.3.5.

4.1.3.3. Image Area (Weighted)

The area of the image was calculated by determining how many pixels are *on*. However, the pixels are weighted differently based on a 2 by 2 neighborhood around the pixel. There are six different patterns of weighting:

- Patterns with 0 'on' pixels (area = 0)
- Patterns with one 'on' pixel (area = ¼)
- Patterns with two adjacent 'on' pixels (area = ½)
- Patterns with two diagonal 'on' pixels (area = ¾)

- Pattern with three 'on' pixels (area = ¾)
- Pattern with all four pixels 'on' (area = 1)

Thus, the image area is the sum of the area values for each pixel of the specified image [29].

4.1.3.4. Euler Number

The Euler number is a measure of the topology of the image. It is defined as the total number of objects in the image minus the number of holes in the objects. Also specified is the connection type. The connection type refers to the neighborhood that is used. The neighborhood can be 4-connected or 8-connected. The connection is the pixels directly in contact with the center pixel. The threshold value must remain constant for all the images processed.

Determination of the threshold value was very important. A threshold level that was too low may introduce unwanted signal components of a low value and the shape of the ridges. A threshold value too high may lose valuable information (shape of the peaks and desired components) [29]. This is also discussed in Section 5.1.3.2. An 8-connected neighborhood was used.

4.1.3.5. Roundness Ratio

The Roundness Ratio (γ) is calculated using the perimeter and unweighted area. The relationship is given in the following equation:

$$\gamma = \frac{P^2}{4\pi A} \tag{22}$$

where: P^2 = perimeter squared
A = unweighted area.

The area and perimeter are calculated using the binary image (bwimage) as follows.

$$A = \sum_x \sum_y bwimage \tag{23}$$

This sums the total number of "On" pixels, which gives an **unweighted** area.

The perimeter is calculated using the same binary image. The perimeter was calculated using the MATLAB command "bwperim.m". The "bwperim.m" function determines the perimeter pixels of the objects in a binary image. One may use either a 4- or 8-connected neighborhood for perimeter determination. An 8-connected neighborhood was used. A pixel is considered a perimeter pixel if it satisfies both of these criteria:

1. It is an *on* pixel.
2. One (or more) of the pixels in its neighborhood is *off*.

Once the pixels are determined to be "on", the number of "on" pixels is summed and squared.

$$P^2 = \left(\sum \text{"}on\text{"}\, Pixels\right)^2 \tag{24}$$

The two values generated by Equations (23) and (24) are used in Equation (22) to determine the roundness ratio γ [29].

4.1.4. Scatter Plot Analysis of Initial Features for Grouping and Outlier Identification

Scatter plots are a useful general tool to determine outliers and groupings in data sets. For example, if different classes within the data set are visually discernable from one another, the feature type may be a good classifier. Another example is that if a feature value does not change then this feature may be deleted from the feature group.

The feature families are similarly generated features (described above) extracted from the raw ETSS PDD data and the computed CWT. The feature families are as follows:

1. PDD input data,
2. PFA coefficients generated using the inductive reactance component of the mixed absolute ECTDFS.

3. 2 features extracted from the mixed differential imaginary ECTDFS,
4. Geometric moments extracted from the complex CWT of the mixed differential signal
5. Image processed CWT of the mixed differential signal.

General observations were made and outliers were identified for each feature family. The following seven figures (Figures 19-21) are scatter plots of each feature group or family (with the geometric moments sub-grouped) with histograms of each feature.

Feature group 1 (Figure 19) contains the phase angle (feature 1) and magnitude (feature 2) derived from the mixed differential ECTD signal. A general observation was that minor grouping seems to be evident between flaw-types. There seems to be two outliers, DHR000C115I029_1 and DHR00PC006I006_1.

Feature group #2 scatter plot is given in Figure 20. Feature group 2 contains the distance between peaks (feature 1) and magnitude between peaks divided by the STD of the signal (feature 2), both derived from the mixed differential ECTD signal. A general observation was that separation between flaw-type groupings seems to be evident. No outliers were identified.

Figure 21 exhibits the relationship between the image processing features (or Feature Group 3). Feature group 3 contains area (feature 1), the Euler Number (feature 2) and Roundness Ratio generated using the CWT derived from the mixed differential ECTD signal. A general observation was that three was grouping between flaw-types. There seems to be no outliers.

A summary of this information is given in Table 12.

Obviously, there was information that would distinguish flaw-types contained in these features. Few outliers were detected. There was very little information that could be obtained from the scatter plots of the geometric moments or the PFA coefficients. These figures are shown in Appendix A.

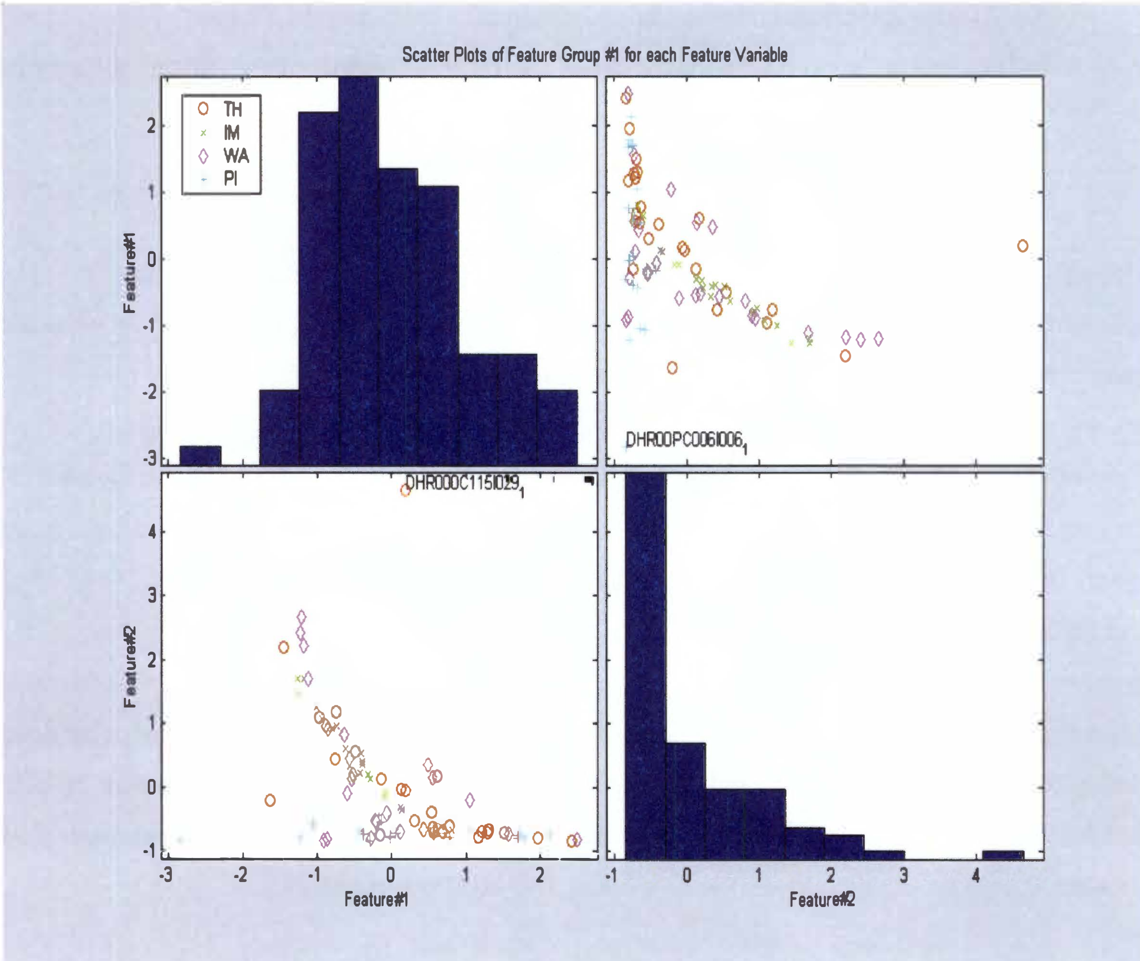

Figure 19. Scatter Plot of Phase Angle vs. Magnitude for Data Group #1.

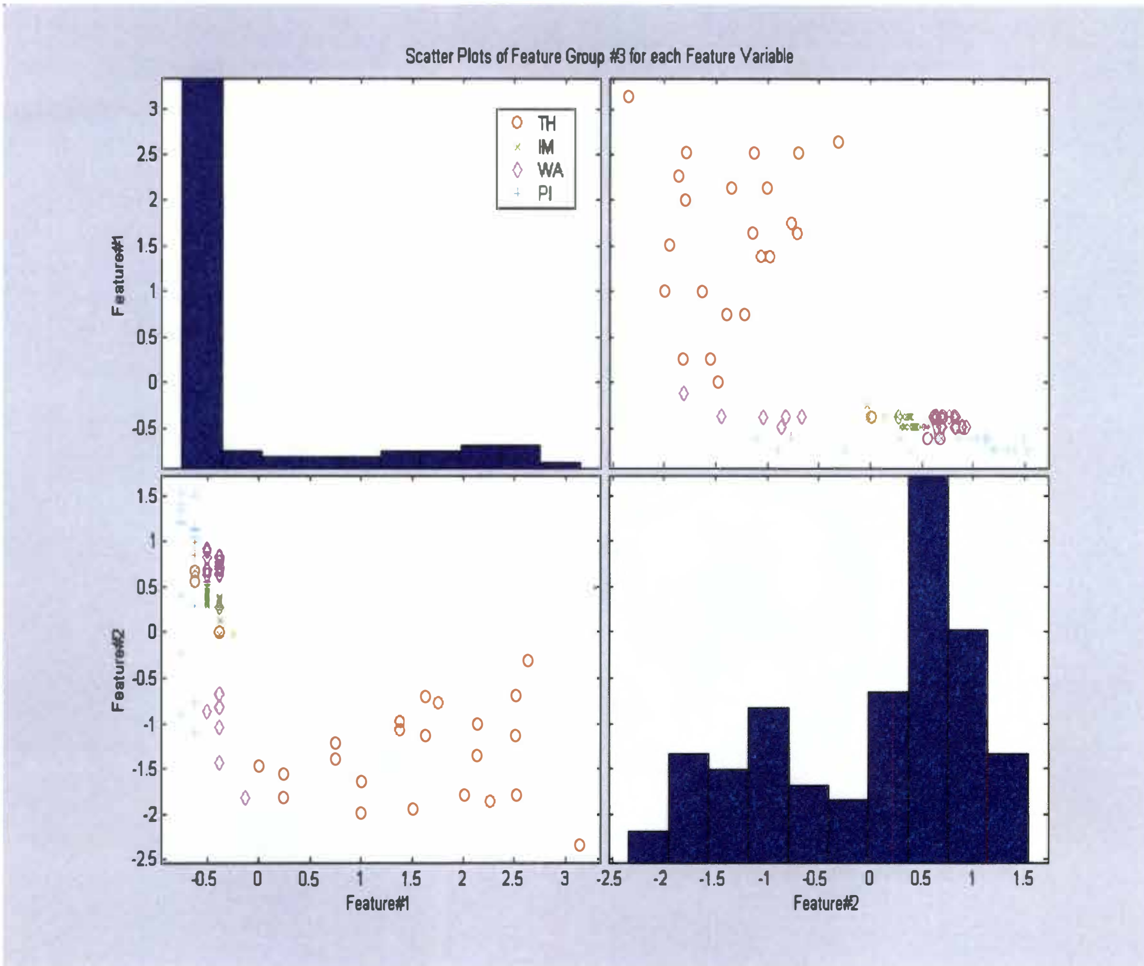

Figure 20. Scatter Plot of Feature Group #2. Distance Between Peaks verses Magnitude between Peaks / Std of Signal.

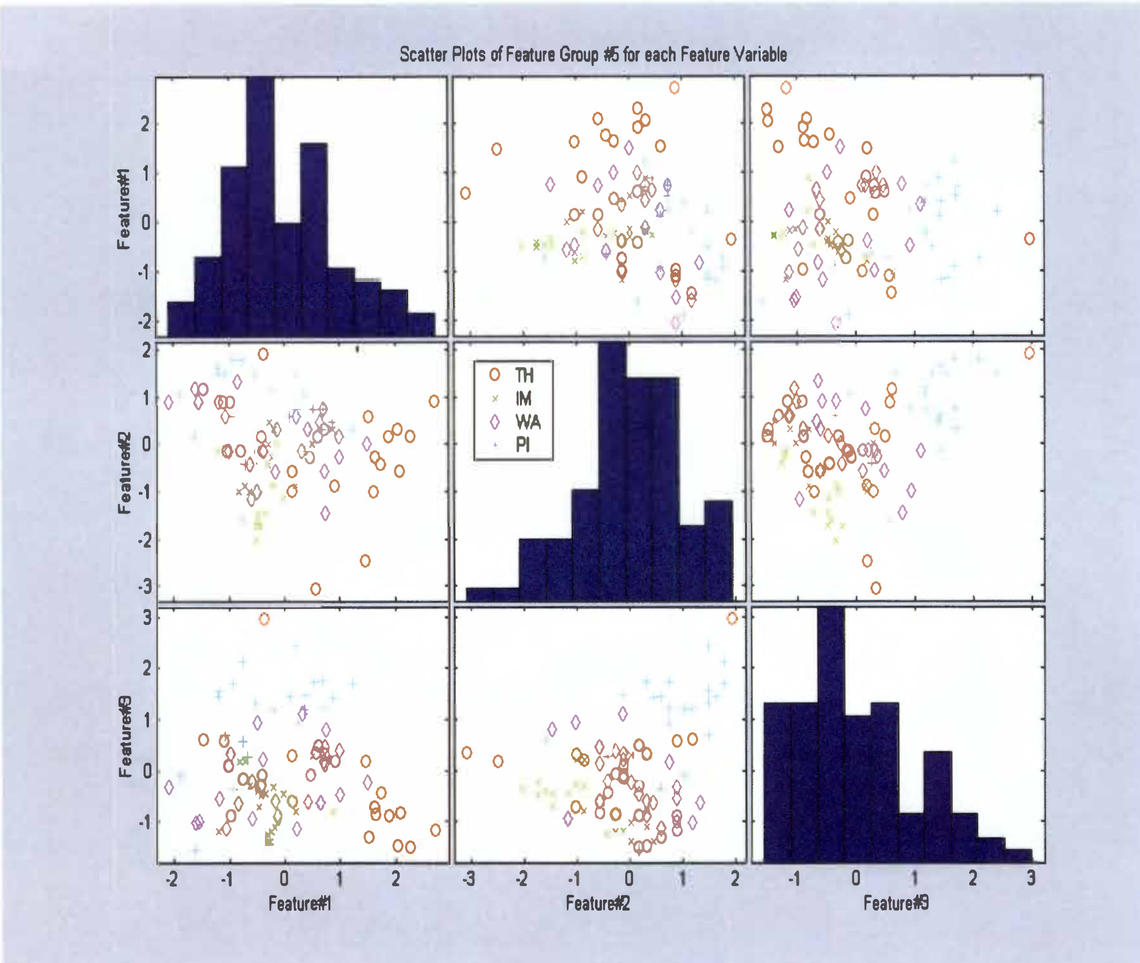

Figure 21. Scatter Plots of Image Processing Features.

Table 12. Summarizes the results for 3 of the feature groups.

Information	**Feature Group**		
	1	**2**	**3**
Grouping	little	yes	yes
Outliers	DHR000C115I029_1 DHR00PC006I006_1	no	no

All the features described in this section were included in a feature vector. The generated feature vector had 51 feature elements. The next step is to use the Principal Component Analysis (PCA) to compress the generated feature vector with as less information loss as possible.

4.1.5. Basic Information Files

The basic information files were created by the EddyC program after preprocessing, raw signal extraction, CWT processing, and feature extraction but before PCA compression of the feature vector (see Appendix F for the EddyC Users Guide and Appendix G for the actual MATLAB programs). These files contain three types of information.

1. Original information
2. Raw signal, flaw phase and magnitude, flaw location, CWT, and initial feature vector
3. Flaw classification and characterization information.

The basic information files were named utilizing four specifications:

1. Whether the data originated as regular PDD or the ETSS subset
2. The PDD or ETSS subset group
3. PDD or ETSS given filename (tube identification)
4. Flaw number (this is used to identify multiple flaws for a specific tube).

Thus, an example basic information filename is E_96001_DHR00BC066I006_1.

4.1.6. Stacked Basic Information (uTR) Files

As the name suggests, the stacked basic information files that have been assembled into a group. There is no additional processing of the basic information files that are contained within the stacked basic information files. The group could include all the example flaws or a subset of the full set. Again, this may be seen in Appendix F and G.

The stacked basic information files were named uTR files. The full name consisted of the uTR prefix, the data origin and an identifying number. An example uTR data file name is uTR_E_1.

4.2. Feature Compression using Principal Component Analysis (PCA)

This section is divided into 3 parts. The first section gives a brief overview of the Principal Component Analysis (PCA) theory. The second part details the determination of the number of PCs to be retained in the PC model. The final section is a scatter-plot analysis of the PCs retained for grouping and for determining outlier information.

4.2.1. Theory of Principal Component Analysis (PCA)

PCA is a quantitatively rigorous method for determining a linear transformation to maximize the variances of all the variables of a data group. PCA also calculates the variances of the transformed data, thus allowing the user to select a small set of variables that show the most significant contributions to the variance. This eliminates unneeded variables, and reduces the feature vector size.

To achieve this, PCA first determines the covariance matrix (Σ) using the original data matrix (X). The covariance matrix is given by:

$$\Sigma = \frac{X^T \cdot X}{N-1} \tag{25}$$

where: X = original data matrix with variable means subtracted.

N = number of samples.

Next, the eigenvalues of Σ are determined. The eigenvalues are determine by using the following:

$$\Sigma \psi = \lambda \psi \tag{26}$$

where: Σ = covariance matrix

ψ = eigenvector

λ = eigenvalue.

The eigenvectors are stacked in a matrix (Φ) that is used to transform the original data matrix X into the new data matrix. This new data matrix (Λ) is determined by

$$\Lambda = \Phi X \tag{27}$$

Λ has the properties that each variable is now orthogonal to any other variable and the new variables exhibit maximum variance. These variables (Principal Components or PCs) are stacked according to the amount of variance each one contains. Thus, the first PC contains more variance than any of the other PCs [31, 32, 33, 34, 35, 36].

One may include only certain PCs to form a compressed model. One method to determine how many PCs to retain is to plot the total amount of variance vs. the number of PCs retained. Usually, the amount of variance retained reaches a plateau at a certain number of retained PCs. This is a good indication of the number of PCs to retain to adequately model the original data.

The next step is to verify if the PCA model is actually a good fit to the data. Two criteria that are commonly used for this are the Hotelling's T^2 statistic and the Q statistic [34, 35, 36].

Hotelling's T^2 statistic is a measure of the variation within the PCA model. Hotelling's T^2 is given by

$$T_i^2 = t_i \lambda^{-1} t_i^T = x_i P_K \lambda^{-1} P_K^T x_i^T \tag{28}$$

where: t_i = i-th row vector of the matrix of k-score vectors from the PCA model.

λ^{-1} = diagonal matrix of inverse eigenvalues associated with the eigenvectors retained in the model.

x_i = i-th data sample.

P_K = transformation matrix (loading matrix with k-PCs retained).

If a data point has a value larger than the 95% confidence level, the data point may not be representative of the data in the PCA model [33, 34, 36].

The Q statistic is a measure of distance a data point falls outside the PCA model (indicating goodness of fit). This statistic relates how well the point fits the PCA model. Q is simply the sum of squares of each row of the error matrix. The Q statistic is then given by

$$Q_i = e_i e_i^T = x_i \left(I - P_K P_K^T \right) x_i \tag{29}$$

where: e_i = i-th row of the error matrix

I = identity matrix.

Again, if a data point has a value larger than the 95% confidence level, the data points are not modeled well using PCA [33, 34, 36].

The Confidence levels (CLs) for both the Q and T^2 statistic are calculated assuming normal distributions. The CL for Q is given by

$$Q_\alpha = \Theta_1 \left[\frac{c_\alpha \sqrt{2\Theta_2 h_0^2}}{\Theta_1} + 1 + \frac{\Theta_2 h_0 \left(h_0 - 1 \right)}{\Theta_1^2} \right]^{\frac{1}{h_0}} \tag{30}$$

where

$$\Theta_i = \sum_{j=k+1}^{n} \lambda_j^i \quad \text{for i = 1, 2, 3} \tag{31}$$

and

$$h_0 = 1 - \frac{2\Theta_1 \Theta_3}{3\Theta_2^2} \tag{32}$$

with c_α = standard normal deviate corresponding to the upper (1-α) percentile

k = number of principal components retained for the model

n = total number of principal components.

The residuals used to calculate the error Q are much more likely to have a normal distribution compared to the scores. This is because Q is a measure of the non-deterministic variation in the samples [36].

The statistical confidence limit for T^2 is calculated by using the F-distribution. The limits are calculated as follows

$$T^2_{k,m,\alpha} = \frac{k(m-1)}{m-k} F_{k,m-k,\alpha} \tag{33}$$

where m = number of samples used to develop the model

k = number of PCs retained in the model.

$F_{k,m-k,\alpha}$ = value of F-distribution at level α, with (k,k-m) degrees of freedom.

The assumption that the data are multivariate normal may not always hold true. If the data are clustered, the T^2 statistic may not accurately predict the outliers. However, the Q statistics are surprisingly well behaved in a wide variety of cases [36].

4.2.2. Determination of the Number of PCs to be Retained for the ECT Data Features

To determine the number of PCs that would accurately convey the information extracted using the ECTD feature parameters, the full data set was used. The full data set, uTR_E_1 contains the basic signatures for all 92 ECTD flaw examples. Two parameters were used for establishing the number of PCs to be retained.

1. The % variance retained by the model [33]
2. The % incorrect classification.

The notation TR_E_1a means that it was a processed data subgroup, extracted from uTR_E_1, with 3 PCs. TR_E_1b has 5 PCs and so on as listed in Table 13.

Table 13. PCs retained vs. % Variance of Model, and % Incorrect Classification.

	TR Run Number (or Subgroup)				
	3 PCs (TR_E_1a)	**5 PCs (TR_E_1b)**	**10 PCs (TR_E_1c)**	**15 PCs (TR_E_1d)**	**20 PCs (TR_E_1e)**
% Variance	69.7	84.5	97.0	99.7	99.9
% Incorrect	44.5	32.6	6.5	1.1	1.1

The results above indicate that 15 PCs would retain more than 99% of the variance and classify observed flaws correctly 98.9%. If 20 PCs were kept, a very little increase in % variance retained or decrease in the % incorrect classification would be obtained.

4.2.3. Basic PC Scatter Plots, Hotelling's T^2 Statistic, and Q Statistic Analysis of PCA Compressed Features

Figure 22 shows the first three PCs (from the compressed features) plotted for all 92 of the ECTD flaw examples. As given in the figure legend, the flaws and flaw centers are marked by a colored circle or square, respectively. The general observations from Figure 22 were that there was little grouping and some separation was evident. In addition, no outliers were identified.

Figure 23 shows the scatter plots of all combinations (in pairs) of the five PCs. The bar graphs located along the diagonal are histograms of each of the PCs. The general observations from Figure 23 were that there was little grouping and some separation was evident. The histograms (the diagonal sub-plots in Figure 23) of the first five PCs indicate non-Guassian distributions. Also, four outliers were identified. The outliers were DAR051C008I017_1, DHRSMPC008I025_2, DAR051C013I026_1 and DAR051C015I016_1.

Figure 24 is a plot of the T^2 statistic for the PCA model. For the full data set, a 95 % Confidence Level for the T^2 values is = 38.87. Thus, many data points may not be representative of the data set as a whole. There were 9 data points above 80.

Figure 25 is a plot of the Q statistic. For the full data set, a 95 % Confidence Level for the Q Statistic values is = 0.46. From the Figure 25, 8 data points were not modeled well. These data points are DHR00BC069I018_1, DHR00BC070I014_1, DHR00BC078I004_1, DHR00BC079I012_1, DHR00BC082I020_1, DHRSMPC001I004_2, DHRSMPC006I019_2 and DHRSMPC008I025_3.

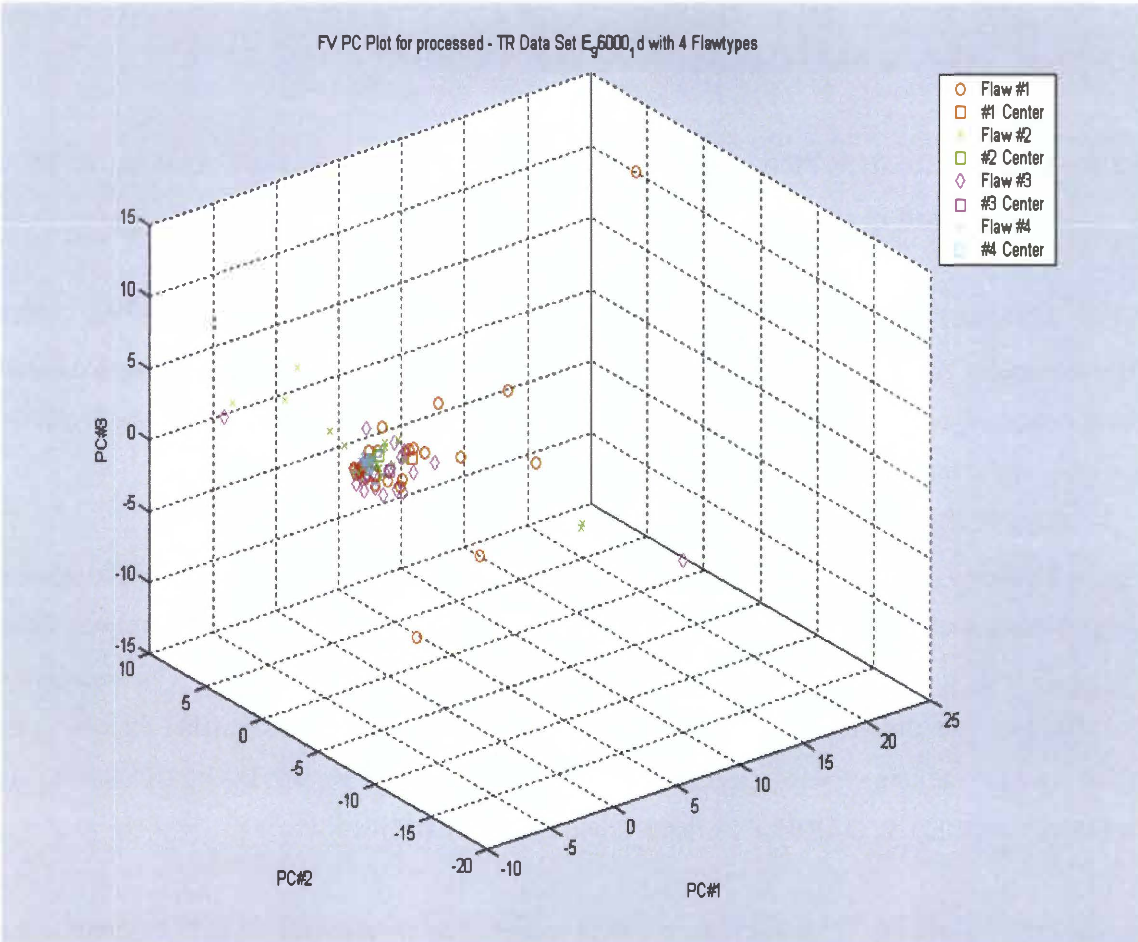

Figure 22. 3D Plot of P's #1, 2 and 3 for All 92 Examples.

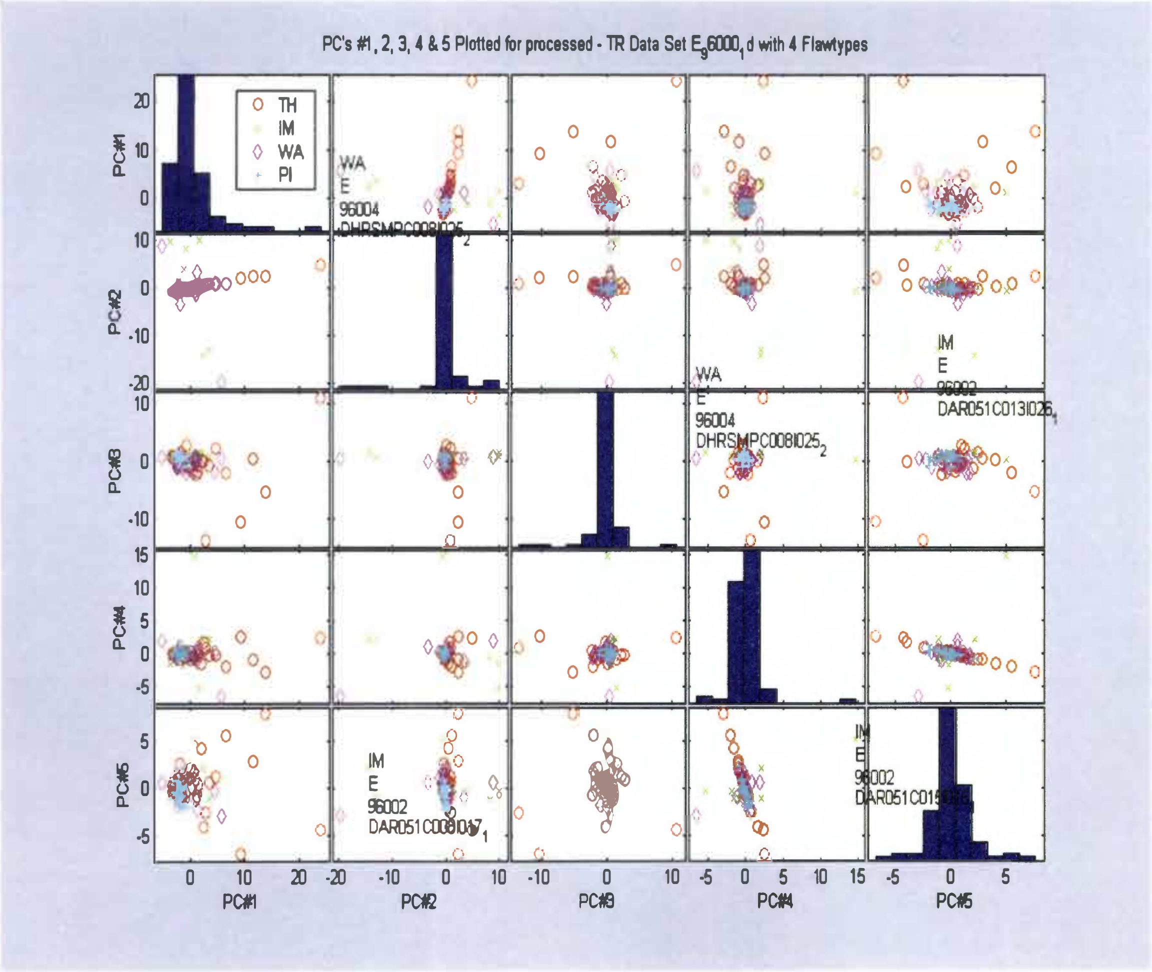

Figure 23. Scatter Plots and Histograms of the First 5 PCs.

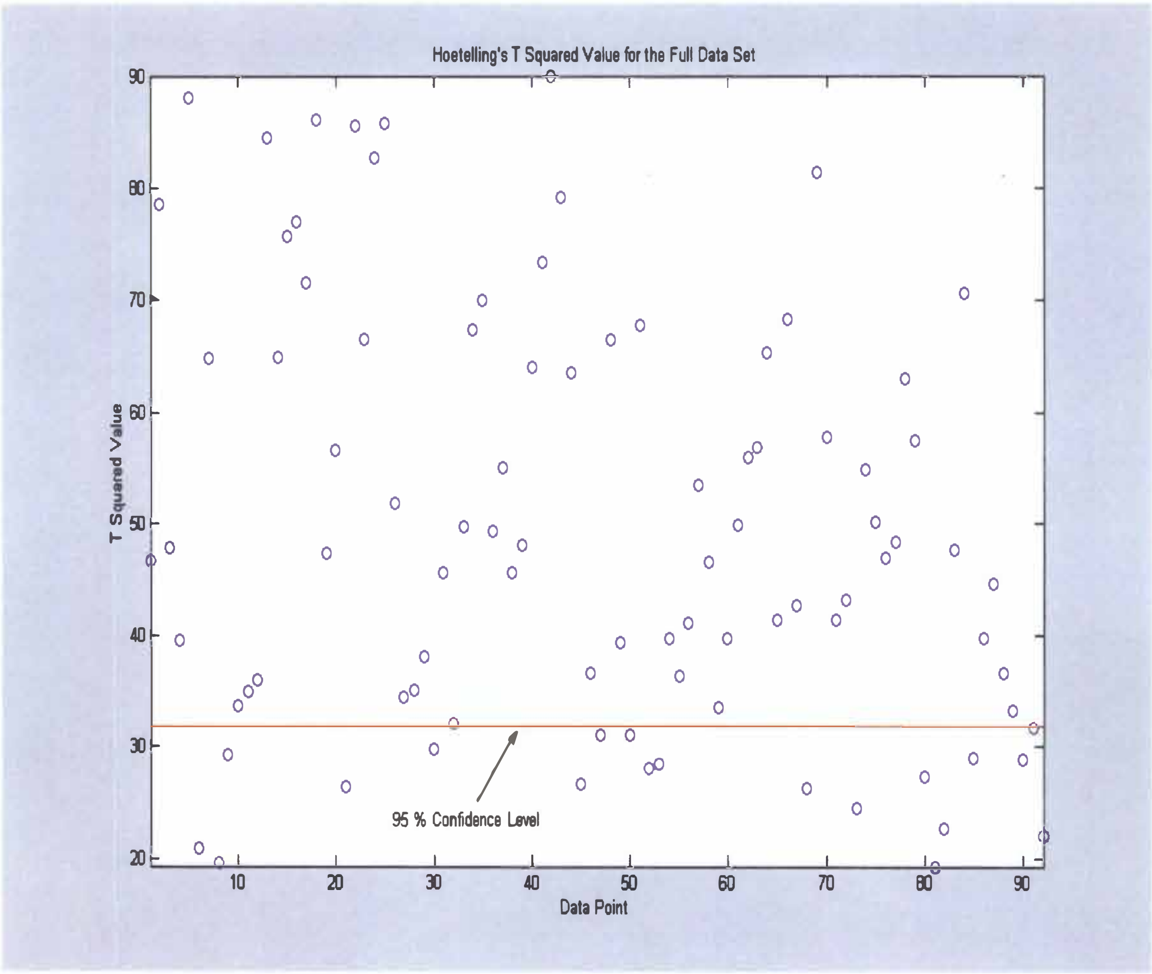

Figure 24. Hotelling's T^2 statistic for uTR_E_1 (Full data set).

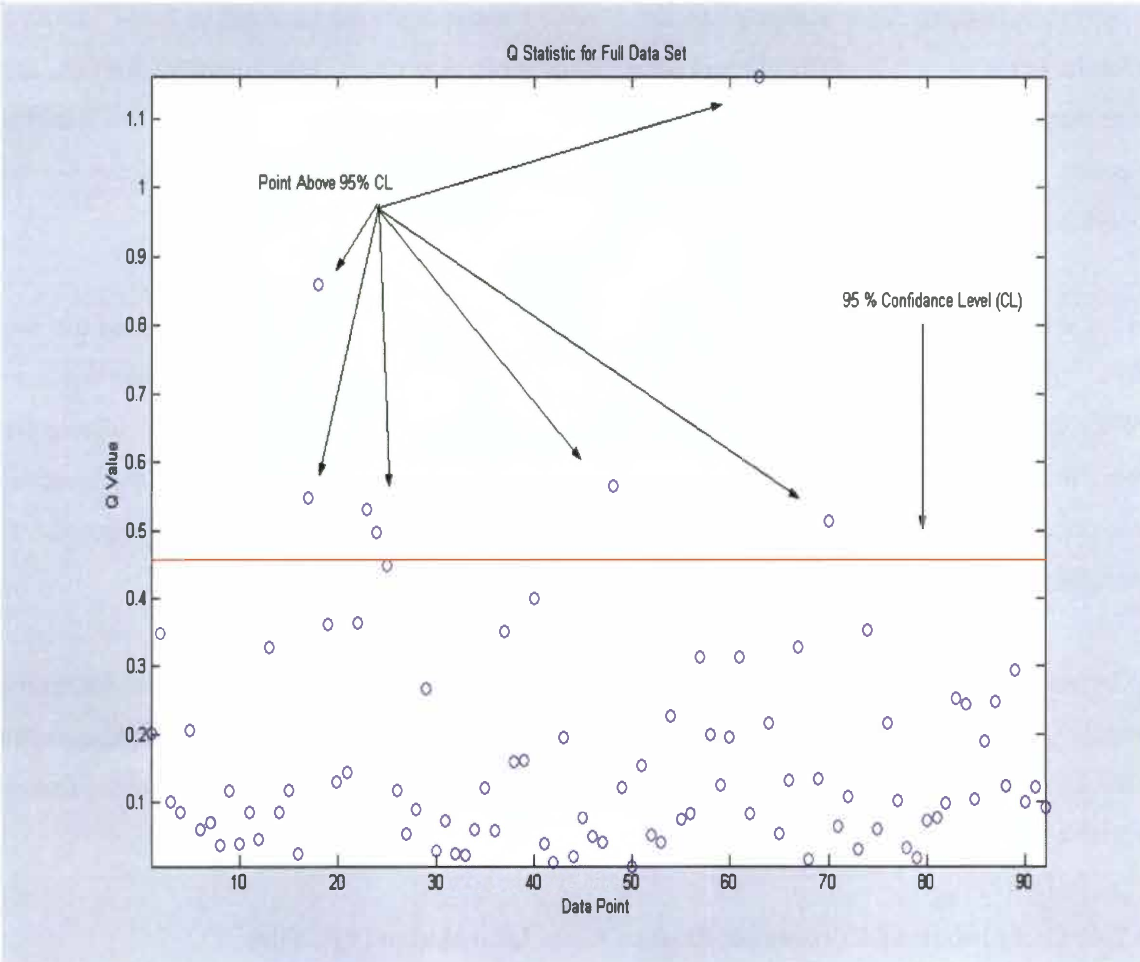

Figure 25. The Q statistic Magnitude vs. ECTD (example) Number for uTR_E_1.

The results from the basic scatter-plots, the T^2 and Q statistics are summarized in Table 14. As is seen in Table 14, a few of the outliers were named more than once. This coincides with the fact that the same example may be an outlier for more than one of the three analysis. As stated in Section 4.3.2, the identified non-typical flaw examples were retained in the database for the previously stated reason.

As is seen in Figure 25, the Q statistic identifies 7 possible outliers, but overall most of the data falls within the 95% CL. Thus, the PCA models the data well. Figure 24 clearly shows that most of the samples fall outside of the 95 % CL for T^2. This particular situation seems to indicate that there is large within model variation. But, as stated at the end of Section 5.2.1, if the resulting compressed data clusters, the Q statistic should show few points above the 95% CL but the T^2 statistic should show many. This explains the Q and T^2 results [36].

PCA was used as a linear transformation of the feature vectors, described in Section 4.2, into a smaller dimensional feature space and also to detect the outliers. The compressed feature vector retained almost all of the original feature vector's variance but has 1/3 the number of the feature variables.

4.2.4. Compressed and Processed, Stacked Basic Information (TR) Files

The compressed and processed information files are generated using a uTR file. The generated TR data file would contain multiple pages of data. Each page of data was specific for a classification of flaw. Each page of the TR data file was organized in the manner shown in Table 15.

The TR data files were named using the uTR data file information plus an identification corresponding to the number of PCs retained by the model. An example TR file name is TR_E_1a. This means uTR_E_1 is processed, retaining only 2 PCs. This is also explained in Section 5.2.2.

Table 14. Outliers for the Scatter-plots of the Principal Components.

	Flaw-type			
	1	2	3	4
Outliers	DHR000C115I029_1 DHR00BC065I017_1 DHR00BC070I014_1(2) DHR00BC077I015_1 DHR00BC079I012_1(2) DHR00BC082I020_1(2) DHR00BC069I018_1 DHR00BC078I004_1	DAR051C013I026_1 DAR051C015I016_1 DAR051C015I016_1 DAR051C008I017_1	DHRSMPC008I025_2 DHRSMPC008I025_2 DHRSMPC001I004_2 DHRSMPC006I019_2 DHRSMPC008I025_3	None

Table 15. Page Contents of a TR Data File.

		Column	
		1 (All Examples contained within the uTR used)	**2 (Specific for the Examples for that Flaw Classification)**
Row	1	Deleted Columns (0 variance)	Break Files
	2	STD and Mean (for each column)	CWT Compressed Matrix
	3	Feature Matrix (without deleted columns)	Feature Matrix (without deleted columns)
	4	PCA Transformation Matrix (using the specified number of PCs)	Flaw Characteristics
	5	PCA Transformed Data	PCA Transformed Data
	6	Tsquare	Tsquare
	7	QTR	QTR
	8	empty	Raw CWT for Flaw

5. Classification and Characterization of Flaws in Steam Generator (SG) Tubing

This chapter is divided into two parts. The first part details classification theory and its application. The second part describes the advanced characterization process and underlying theory.

5.1. Classification of Flaws in Steam Generator (SG) Tubing

The next step in this research was to classify the type of flaw in the SG tubing using the ECT data. Classification of the flaw was accomplished by using the compressed feature vector. In general, the classifications could include the following flaw types.

1. No Defect
2. Cracking
3. Intergranular Attack (IGA)
4. Thinning
5. Wear
6. Impingement
7. Stress Corrosion Cracking (PWSCC or ODSCC)
8. IGA/SCC
9. Pitting
10. Denting
11. Multiple Defects
12. Undetermined

Three factors help to narrow this list.

1. Not all the flaws listed above are exhibited by SGs by each vendor.
2. The location of the flaw within the SG.
3. The location of the flaw with respect to the tube itself (ID or OD).

As outlined in Section 2.2.3, the ECTD subset used included only 4 flaw-types. Thus, the classification was narrowed to thinning, impingement, wear, and pitting. In addition, since the data were lab-generated, the three factors listed above were not relevant. Classification of flaws was accomplished using a traditional (Bayes and distance-based) pattern recognition technique.

Section 5.1 is divided into five parts. The first part gives an overview of Bayesian pattern recognition. The second details the calculation of the upper bound for the total Probability of Error using the Bhattacharyya distance. The third describes cross validation techniques. The fourth combines cross validation and Bayes pattern recognition. The final part mentions template matching classification.

5.1.1. Bayesian Pattern Recognition Method

Bayesian pattern recognition is based on Bayes decision theory. Bayes decision theory uses the minimization of the probability of error and the a posteriori probability. The conditional probability is expressed as

$$P(B \mid A)P(A) = P(A \mid B)P(B) \tag{34}$$

where: $P(B|A)$ = probability of event B assuming A
$P(A|B)$ = probability of event A assuming B
$P(A)$ and P(B) = probability of A and the probability of B.

These can also be extended to random variables and probability density functions.

$$p(x \mid y)p(y) = p(y \mid x)p(x) \tag{35}$$

where: $p(x|y)$ = probability density function of x given y
$p(y|x)$ = probability density function of y given x
$p(x)$, $p(y)$ = probability density functions of x and y, respectfully.

Now, to adjust for classes (ω_i) and multi-dimensional variable x

$$P(\omega_i \mid x) = \frac{p(x \mid \omega_i)P(\omega_i)}{p(x)} \tag{36}$$

Where $p(x)$ is given by

$$p(x)=\sum_{i=1}^{N} p(x \mid \omega_i)P(\omega_i) \qquad (37)$$

The Bayes classification rule (or decision-making) states that among m-hypotheses, choose H_i such that $P(\omega_i|x)$ is maximized for ωi and is given by the following

Choose hypothesis H_i over H_j if:

$$p(x \mid \omega_i)P(\omega_i) \geq p(x \mid \omega_j)P(\omega_j) \qquad (38)$$

where: $P(\omega_i)$ = prior probability of class *i.*

$P(\omega_j)$ = prior probability of class *j.*

These probabilities are based on the number of examples for class *i* divided by the total number of examples (*N*) or

$$P(\omega_i)=\frac{\sum_j \omega_{ij}}{N} \qquad (39)$$

Assuming a multi-dimensional normal probability density function for the data under each hypothesis, the joint probability density function in Equation (38) has the form

$$p(x \mid \omega_i)=\frac{1}{(2\pi)^{l/2}|\Sigma_i|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right) \qquad (40)$$

where: x = vector of measurements.

μ_i = vector of mean value of the class ω_i

$|\Sigma_i|$ = determinant of the *l by l covariance matrix.*

The covariance matrix is defined as

$$\Sigma_i = E\left[(x-\mu_i)(x-\mu_i)^T\right] \tag{41}$$

For the minimum error probability case, the decision surface to classify between classes *i* and *j* is

$$p(x \mid \omega_i)P(\omega_i) - p(x \mid \omega_j)P(\omega_j) = 0 \tag{42}$$

These surfaces can be used to determine the class of a new test vector. Another approach is to transform each part of the decision surface using the natural log as

$$g_i(x) = \ln(p(x \mid \omega_i)P(\omega_i)) = \ln(p(x \mid \omega_i)) + \ln(P(\omega_i)) \tag{43}$$

Using Equation (40) this may be simplified as

$$g_i(x) = -\frac{1}{2}x^T \Sigma_i^{-1} x + \frac{1}{2}x^T \Sigma_i^{-1} \mu_i - \frac{1}{2}\mu^T \Sigma_i^{-1} \mu_i + \frac{1}{2}\mu^T \Sigma_i^{-1} x + \ln P(\omega_i) + c_i \tag{44}$$

where: $c_i = -(l/2)\ln 2\pi - (1/2)\ln|\Sigma_i|$.

Each class generates a decision function, $g_i(x)$. The decision function is used by substituting the unknown flaw vector values into each function $g_i(x)$, with the flaw being classified according to the largest value generated. That is

$$\max_i \; g_i(x) \tag{45a}$$

Decision surfaces are generated using all class combinations of the decision functions

$$g_{ij}(x) \equiv g_i(x) - g_j(x) = 0 \tag{45b}$$

These decision functions are hyper-quadratics when the number of classes is great than 2 [29, 30, 31].

The MATLAB classification routine "classify.m" uses the above strategy but makes the following assumption of Equiprobable Classes. This assumption lead to

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) \tag{46}$$

where constants have been neglected. If the covariance matrix is non-diagonal, maximizing $g_i(x)$ is equivalent to minimizing the Σ_i^{-1} norm, known as the Mahalanobis distance, or

$$d_m = \left((x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right)^{1/2} \tag{47}$$

Thus, if the smallest distance calculated using an unknown (x) is generated using the μ produced by data from group i, then the unknown is classified as group i [30, 31]. There are other distance functions that have a direct relationship to the probability of detection (or the total probability of errors).

5.1.2. Upper Bound on Probability of Error using Bhattacharyya Distance

Since this research employs Bayesian decisions, guaranteeing the lowest average error rate, calculation of the probability of error was important. The probability of error (Pe), for a two classification, system is given by

$$Pe = P(x \in R_2, \omega_1) + P(x \in R_1, \omega_2) \tag{48}$$

where: x = observation

R_1 = region 1 (Classification 1)

R_2 = region 2 (Classification 2)

ω_1 = true State (or Classification) 1

ω_2 = true State (or Classification) 2.

or in integral form

$$Pe = \int_{R_2} p(x \mid \omega_1) P(\omega_1) dx + \int_{R_1} p(x \mid \omega_2) P(\omega_2) dx \tag{49}$$

where: $p(x|\omega_1)$ = state-conditional probability density function for x given class 1

$p(x|\omega_2)$ = state-conditional probability density function for x given class 2

$P(\omega_1)$ = prior probability that nature was in state 1

$P(\omega_2)$ = prior probability that nature was in state 2.

The Chernoff Bound applies the inequality

$$\min[a, b] \le a^{\beta} b^{1-\beta} \qquad \text{for } a, b \ge 0 \text{ and } 1 \ge \beta \ge 0. \tag{50}$$

to simplify the integral form of Pe. Using this inequity, the upper bound for Pe can be estimated as

$$Pe \le P^{\beta}(\omega_1) P^{1-\beta}(\omega_2) \int P^{\beta}(x \mid \omega_1) P^{1-\beta}(x \mid \omega_2) dx \qquad \text{for } 1 \ge \beta \ge 0. \tag{51}$$

If the conditional probabilities are normal, the above integral can be evaluated analytically, yielding

$$\int P^{\beta}(x \mid \omega_1) P^{1-\beta}(x \mid \omega_2) dx = e^{-k(\beta)} \tag{52}$$

where

$$k(\beta) = \frac{\beta(1-\beta)}{2}(\mu_2 - \mu_1)^t [\beta\Sigma_1 + (1-\beta)\Sigma_2]^{-1}(\mu_2 - \mu_1) + \frac{1}{2}\ln\frac{|\beta\Sigma_1 + (1-\beta)\Sigma_2|}{|\Sigma_1|^{\beta}|\Sigma_2|^{1-\beta}} \tag{53}$$

β is varied until a minimum value of $e^{-k(\beta)}$ is determined. This β yields the Chernoff error bound. If $\beta = 0.5$, then the upper bound of the probability of error based on the Bhattacharyya distance (UPeBD) is determined. This simplifies as

$$Pe \leq \sqrt{P(\omega_1)P(\omega_2)} \int \sqrt{p(x \mid \omega_1)p(x \mid \omega_2)}dx \tag{54}$$

with the integral part equal to $e^{-k(\beta)}$. For the Gaussian case

$$k(1/2) = \frac{1}{8}(\mu_2 - \mu_1)^t \left[\frac{\Sigma_1 + \Sigma_2}{2}\right]^{-1} (\mu_2 - \mu_1) + \frac{1}{2}\ln\frac{\left|\frac{\Sigma_1 + \Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}} \tag{55}$$

If the distribution is not Gaussian, this estimation may not be accurate. The BPeUB provides an upper bound on the probability of error for the Bayesian decision method [31, 37, 38, 39]. Two UPeBDs are calculated, the regular UPeBD and a UPeBD calculated using zeroed off-diagonal covariance matrices, Σ_1 and Σ_2 (or as abbreviated UPeBDZ).

$P(\omega)$'s were assumed to be 0.25, corresponding to the case that there were 4 classes. A summary of the Bhattacharyya distances (Both the UPeBD and the UPeBDZ), including the % variance and % incorrect classification, is given below in Table 16. These calculations were made using the full data set of 92 examples (uTR_E_1 and TR_E_1).

A graph of the above information can be seen in Figure 26. The UPeBDZ (dashed line) seems to parallel and bound the % Incorrect Classification as the number of PCs were increased. Since the retained PCs were deemed non-Gaussian in Section 5.2.3, the calculated B-distances (based on a Gaussian assumption) may not be appropriate or very accurate.

Table 16. PCs retained vs. % Variance of Model, % Incorrect Classification, UPeBD and UPeBDZ.

	TR Run Number (or Subgroup)				
	3 PCs (TR_E_1a)	5 PCs (TR_E_1b)	10 PCs (TR_E_1c)	15 PCs (TR_E_1d)	20 PCs (TR_E_1e)
% Variance	69.7	84.5	97.0	99.7	99.9
% Incorrect	44.5	32.6	6.5	1.1	1.1
UPeBD	0.3249	0.1242	0.0024	1.734e-5	1.103e-8
UPeBDZ	0.4434	0.3273	0.1722	0.1422	0.0723

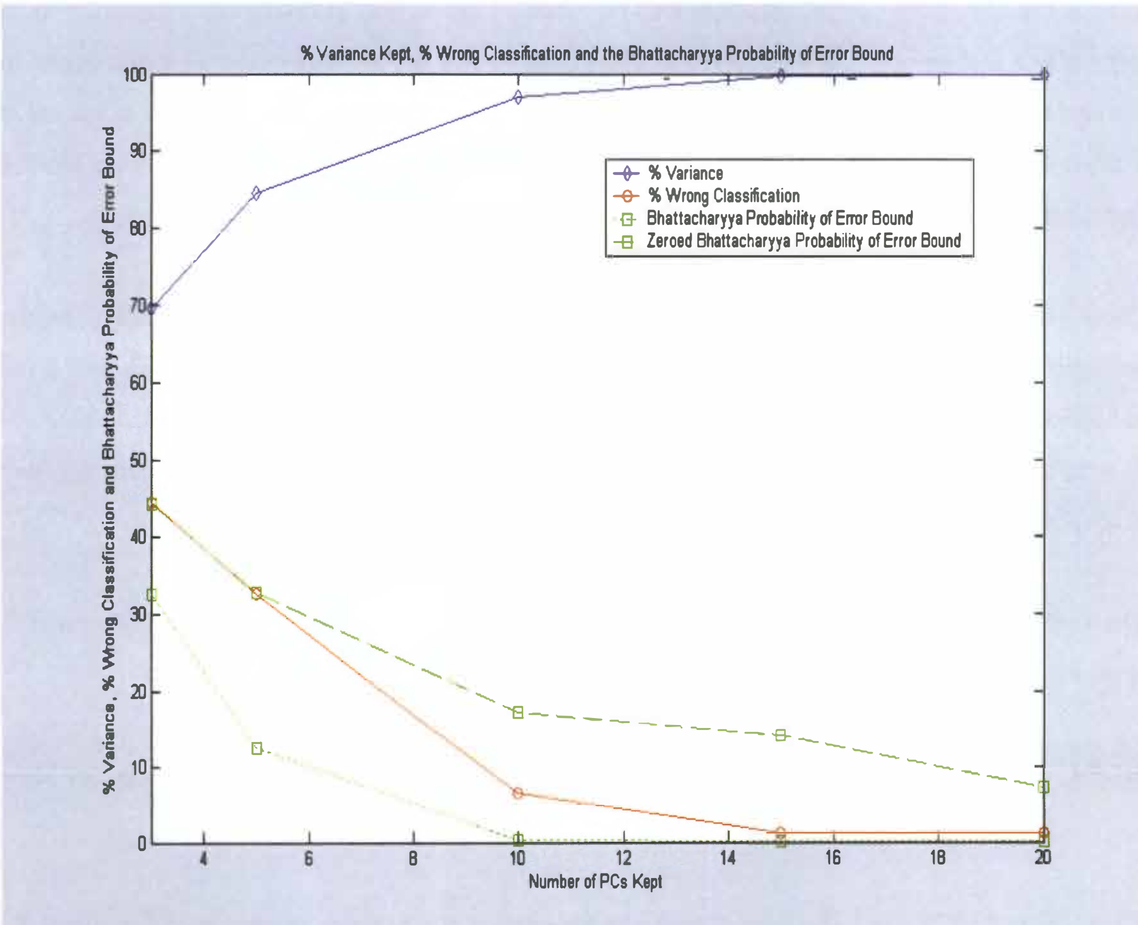

Figure 26. PCs retained vs % Variance, % Incorrect (Wrong) Classification, UPeBD and UPeBDZ.

5.1.3. Basic Cross Validation Theory and Applications

Cross validation is the procedure of randomly splitting a set of examples into a training set and a test set, training the classification system with the training set, then test the system with the test set. One cross validation method is to leave out m samples of n total examples, generating n/m subsets.

There were 92 total examples in the database. Two cross validation procedures were used. First, only four examples were extracted at a time, thus 23 subgroups were formed. Then the new training group was formed by recombining 22 of the subgroups, with the remaining subgroup used as the test group. This procedure was repeated 23 times, thus allowing each subgroup to be left out as a test group. The inaccuracy of classification was the average of the 23 validations. In the second approach, only one example was extracted at a time. Thus, one flaw was extracted at a time and the remaining 91 samples were used for train. This allows the training set the maximum information from the database without the system seeing the test example.

Cross validation was applied to three different scenarios with classification accuracy as the measuring stick. First, sensitivity of single feature groups was ascertained using the first cross validation system. Second, sensitivity of multiple feature groups was established, also using the first system. Finally, cross validation of the classification using all feature groups was determined using both the first and second system.

The first two applications correspond to the two parts of this section. The third application is briefly discussed in Section 5.1.4.

5.1.3.1. Cross Validation System Four used for Classification of Single Groups of Raw Features

When using MATLAB's Bayesian classifier (classify.m), the number of examples for a specific class must be greater than the number of features. With this in mind, the two groups of features, geometric moments and the polynomial coefficients, were not processed. The features were not compressed using PCA.

Table 17 lists the results using cross validation system four for the classification of single groups of raw features. The three individual feature families were highly inaccurate.

5.1.3.2. Cross Validation System Four used for Classification of Multiple Groups of Compressed Features

In order to use MATLAB's classify.m program, the multi-group feature families were compressed using the PCA. The first 15 PCs were kept. Feature family #2 was the polynomial coefficients derived from the Imaginary absolute ECTD signal. Feature family #4 was the geometric moments derived from the CWT of the complex differential ECTD signal. Classification inaccuracy was listed in Table 18.

To summarize, the average % incorrect classification when deleting a feature family was 23.91. The best case was when the absolute coefficients were deleted. The deletion of no one feature family had a significant effect on the average % incorrect classification.

5.1.4. Cross Validation with Application of Bayes Pattern Recognition

Both cross validation subset generation systems were outlined in the second paragraph of Section 5.1.3. The results listed in Section 6.1 were generated using these methods.

5.1.5. Classification using Template Matching

Classification using Template Matching utilized the ECTDFS generated CWT. Initial Template matching results (from the PDD database) yielded marginal results (correct classification 69%), and was not utilized. These results are summarized in Appendix E.

The results of template matching using the raw CWT signatures indicated that the CWT contained information about flaw types necessary for successful classification using image processing signatures.

Table 17. Average % Incorrect Classification vs. Feature Group

Feature Group	Average % Incorrect Classification
1 (Phase and Magnitude of ECTD Differential Signal)	60.87
3 (Parameters derived from the Imaginary Part of the Differential Signal)	53.26
5 (Image Processing Parameters derived from the Imaginary Differential CWT)	57.61

Table 18. Average % Incorrect Classification vs. Feature Family Deleted.

Feature Family Deleted	Average % Incorrect Classification
1	25.00
2	25.00
3	25.00
4	22.83
5	25.00
Average incorrect % (when deleting families)	23.91

5.2. Characterization of Flaws in SG Tubing

Characterization of the tube flaws was accomplished, after classification, by using trained flaw-type specific Artificial Neural Networks (ANNs). This section describes basic ANN theory and the various steps that were taken to accomplish this task.

This section is divided into four parts. The first part details basic theory of artificial neural networks, with the second part outlining the specifics of ANNs for this application. The third part briefly describes correlation analysis of the input and target output. The final part includes a description of training and storing the ANNs.

5.2.1. Artificial Neural Networks

ANNs are highly versatile modeling tools. ANNs can model almost any function (or system) with high accuracy [40-43]. Since there were multiple inputs (15 PCs and a classification) and outputs in this system, ANN seemed to be the logical choice for the task. This section contains a general overview of ANNs.

5.2.1.1. Basic Artificial Neural Network

A single neuron network consists of a weight, a bias, and a function. The weight and bias matrix transforms the input, the transformed input is expressed as

$$a = f(W * p + b) \tag{56}$$

where: a = Output

W = Weight

p = Input

B = Bias

f = Transformation Activation Function.

This is shown in Figure 27. Many inputs and neurons can be used to form a single layer-multiple neuron network. This is shown in Figure 28.

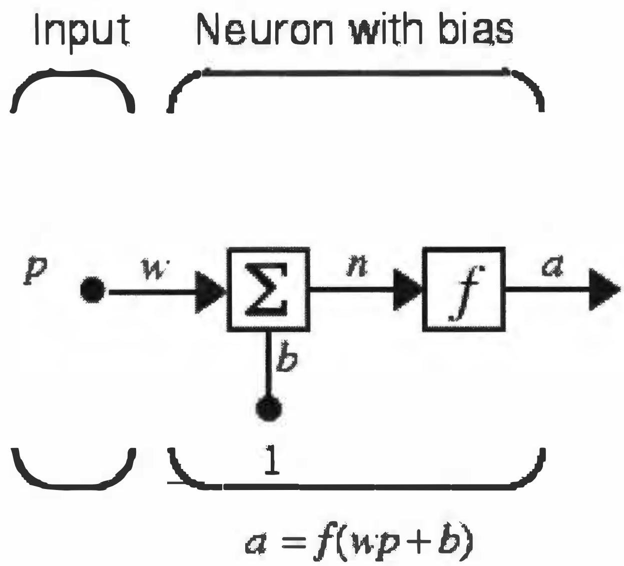

Figure 27. A Basic Input-Output Neuron [44].

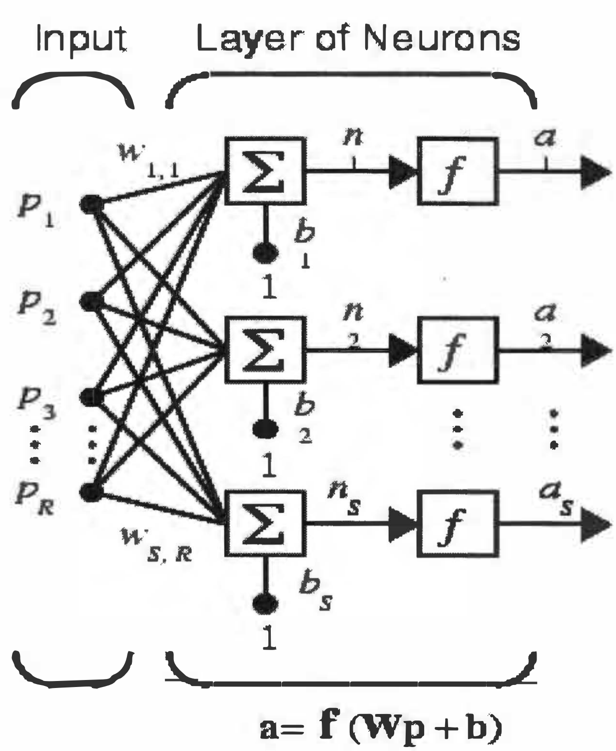

Figure 28. A simple layer ANN with multiple Inputs and Ouputs [44].

The next step is to form multiple layers of multiple neurons. This is the classic ANN. The formula for a three-layer multiple neuron ANN is given by:

$$a = f_3\left(W_3 f_2\left(W_2 f_1\left(W_1 * p + b_1\right) + b_2\right) + b_3\right) \tag{57}$$

where the *f*s are functions, the *W*s are weights and the *b*s are biases. This can be seen in Figure 29.

Each of the above individually performed functions is called a layer. Thus, the output of the *i*-th layer becomes the input for the (i+1)th layer. This transformation is executed for each individual layer of the ANN, noting that the ANN may have many layers. Each individual function group performs a transformation of the input data in an effort to obtain the target data as the output of the last layer.

An example of an activation function (also called the sigmoidal function), is given by the equation below:

$$f(X) = \frac{1}{1+e^{-\alpha X}} \tag{58}$$

where: X = input ($W*p+b$ in this case)

α = shape parameter.

One complete transformation (through all the ANN layers) is called an epoch. These epochs are repeated over and over until an error goal between the training data and the target data is accomplished, or the ANN cannot accomplish this goal in the allotted number of epochs [44].

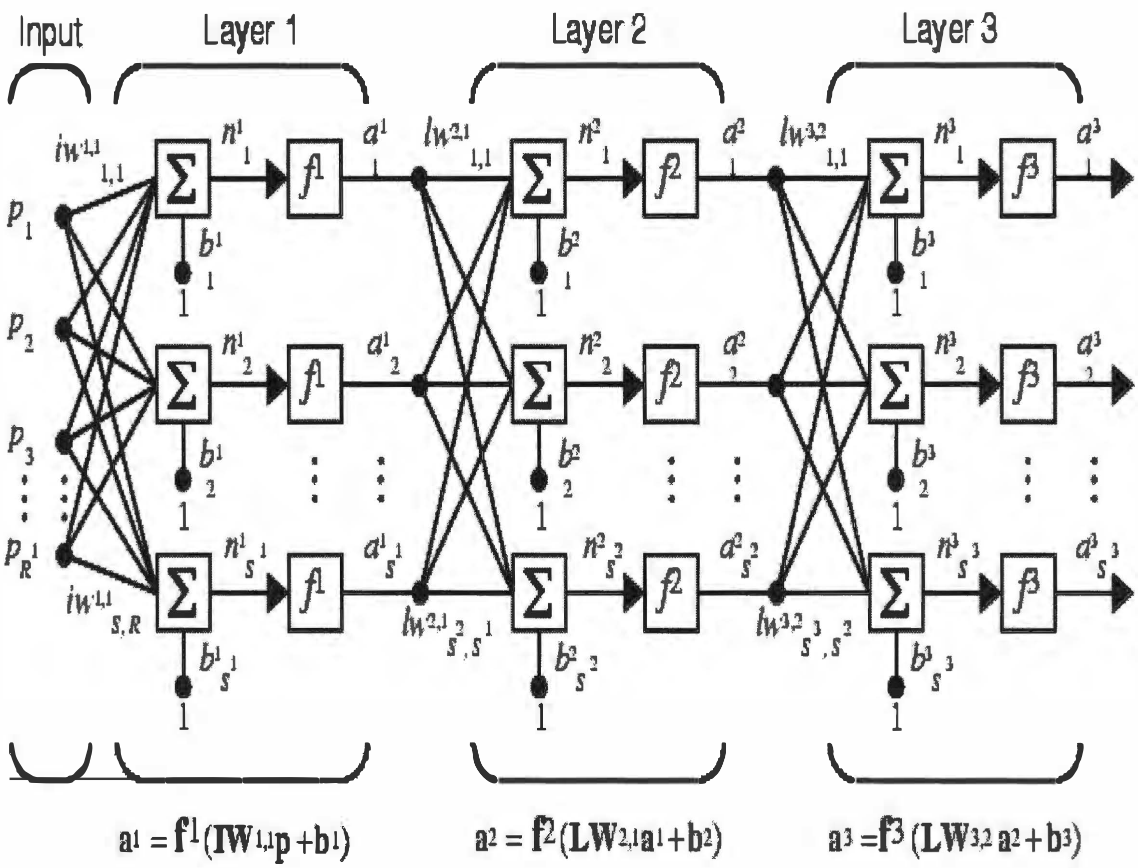

Figure 29. A multi-layer ANN with multiple Inputs, and multiple Outputs [44].

5.2.1.2. Error

The error is usually given by the sum of squared errors (*SSE*), which is defined as

$$SSE = \sum_{j=1}^{N}\sum_{i=1}^{M}\left(T_{ij} - a_{ij}\right)^2 \tag{59}$$

where: a_{ij} = ith training vector used to produce the hth output value of the ANN

T_{ij} = ith and hth Target Value.

N = total number of training vectors

M = total number of output variables [44].

This is only one of many performance errors that can be utilized. Other errors are detailed in Section 5.2.2.3. To train an ANN, the error of the system can be backpropagated through the ANN framework and used to adjust the weights in each layer. The backpropagation algorithm is one such method.

5.2.1.3. Backpropagation Algorithm

The backpropagation algorithm uses a steepest descent technique, which is very stable, when a small learning weight is used, but may be slow to converge. The error term used is given by

$$e = \left(T_{ij} - a_{ij}\right) \tag{60}$$

To backpropagate the error, the relationship between the error and the functions at each step must be analyzed. Therefore, to change the weights in the 1st layer, the error must be backpropagated through the hidden layer. For a 2 layer ANN, the change of error with respect to the change in weights at 1st layer is

$$\partial e/\partial w_1 = \partial e/\partial f * \partial f/\partial v_2 * \partial v_2/\partial w_2 * \partial w_2/\partial a_1 * \partial a_1/\partial f * \partial f/\partial v_1 * \partial v_1/\partial w_1 \tag{61}$$

where: f = transformation function

w = weights

$v = W^*p+b$ for each layer

a = Output of the layer.

In a two-layer ANN using backpropagation, the above term is equal to

$$\partial e/\partial w_1 = a_1{*}(1-a_1){*}\{W_2{*}[a_2{*}(1-a_2){*}e]\} \tag{62}$$

The derivative term is then used to update the weight in the 1st layer as

$$W_1^{new} = W_1^{old} + 2 * lr * \frac{\partial e}{\partial w_1} * TV \tag{63}$$

The weights in the second layer are updated in a similar fashion. The 3-layer case is slightly more complicated but the same principles apply as in the 2-layer case. Now, assume that the ANN has been trained. The next step is to check the adequacy of the ANN [44-46].

5.2.1.4. Over-fit and Under-fit of the ANN

Two problems can arise when modeling with a ANN. The first problem arises when the ANN over fits the data. The second problem occurs if the ANN under fits the data.

If the ANN over fits the data, the analyst has used too many degrees of freedom. The ANN would train to a very low RMSE, but the ANN would train to each individual data point and not the underlying function that describes the data. Thus, when the ANN is checked with a data sample that is in-between the points used, a high RMSE would be obtained, as shown in Figure 30.

If the ANN under fits the data, the ANN would yield a high RMSE result and not approximate the underlying function of the data. This means the ANN did not use enough degrees of freedom to identify the underlying function or the ANN had an appropriate number of degrees of freedom but was not trained properly [45].

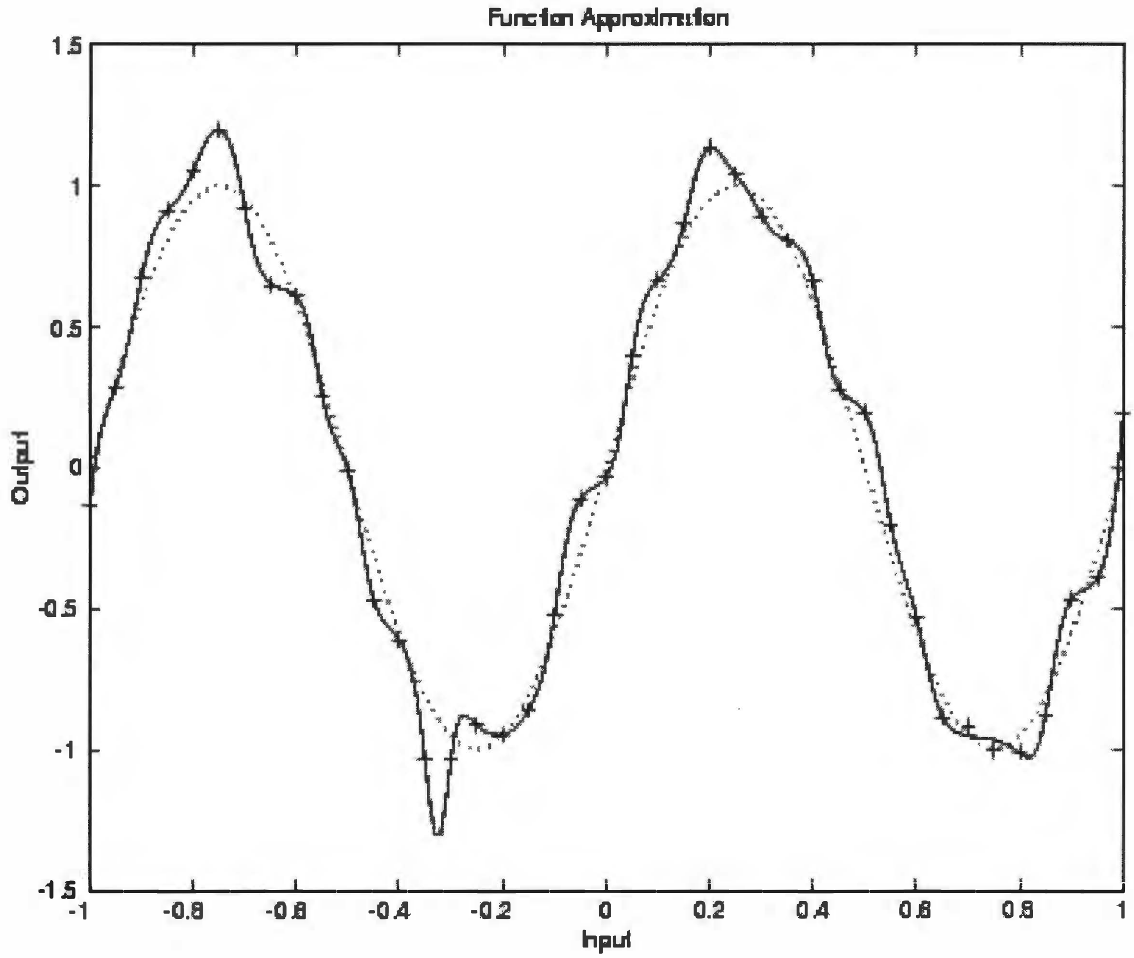

Figure 30. An Overfit ANN Approximation of a Sine Curve [44].

5.2.2. Application of ANNs for Characterization

Once the tube flaw was classified, a ANN was used for defect sizing (or characterization). But, instead of one ANN being trained to characterize all flaw types, individual flaw-specific ANNs were trained to characterize (the dimensions found in the blueprints) the flaw.

The first section defines what is meant by generalization subsets. The second section outlines a number of ANN parameters that were assigned rather than determined and details the justification for each. The third section details the logic in determining the appropriate training error level or goal. The fourth part establishes the number of hidden neurons needed to adequately characterize the training data given by the generalization subset 1 (uTR_E_2 and TR _E_2a).

5.2.2.1. Generalization Subsets

A generalization training subgroup was formed by randomly extracting four flaw examples (one from each flaw-type) from the full data set (uTR_E_1). Five generalization training subgroups were generated, uTR_E_2 through uTR_E_6. Each generalization training subgroup had four different flaws extracted from the full data set. No two test flaws were the same. As an example, TR_E_2a was the processed data subgroup, generated from uTR_E_2, using 15 PCs.

5.2.2.2. Fixed ANN Parameters

Many parameters associated with the neural networks were either assigned or determined. The assigned parameters were data preprocessing, layer functions, the number of hidden layer, type of training, maximum number of epochs, performance function, and a few others. The MATLAB code to assign these parameters is given below.

```
[Pn,minp,maxp,Tn,mint,maxt]=premnmx(P,T);
net = newff(minmax(Pn),[S1 S2],{'tansig' 'purelin'},'trainbr');
net.trainParam.goal = goal;
net.trainParam.mc = 0.95;
net.trainParam.show = 10;
net.trainParam.epochs = 200;
```

```
net = train(net,Pn,Tn);
Yn = sim(net,Pn);
Y = postmnmx(Yn,mint,maxt);
```

With Y being the flaw characteristic output of the ANN.

5.2.2.3. Error Performance and Goal

A good training error level is one that accurately generates results with both the training data and unseen test data. Sum of Squared Error (*SSE*) is the assigned error function for "trainbr". The *SSE* is defined in Equation (59).

The Mean Squared Error (*MSE*) performance indicator was chosen because it was not affected by the number of variables or the number of examples. The *MSE* is defined as

$$MSE = \frac{1}{MN}\sum_{j=1}^{N}\sum_{i=1}^{M}\left(T_{ij} - a_{ij}\right)^2 \tag{64}$$

where the values are defined Equation (59) [44-46].

An *SSE* = 0.1 yielded the results summarized in Table 19. An *MSE* goal of 0.01 was chosen. The *MSE* goal of 0.01 forced approximately a $\pm$ 1% error between the un-scaled target and output. Using the above data as a reference, an ***SSE* of 0.05** was selected. An *SSE* of 0.05 should yield the mean squared error (*MSE*) level discussed below.

Another performance indicator is the Root Mean Error (*ME*). The mean error is defined as

$$RME = \sqrt{MSE} = \sqrt{\frac{1}{MN}\sum_{j=1}^{N}\sum_{i=1}^{M}\left(T_{ij} - a_{ij}\right)^2} \tag{65}$$

Table 19. *SSE* and *MSE* for each Data Group (or Flaw-type).

Flaw-type (# of Examples, # of Outputs)	**Error Function**	
	SSE	**MSE**
1 (24,3)	0.1	0.0144
2 (20,3)	0.1	0.0130
3 (23,2)	0.1	0.0020
4 (21,3)	0.1	0.1723

Basically, the *RME* gives the physical error, while the *MSE* gives a squared error [44-46]. Another error measurement is the % Average Error (*%AE*). The *%AE* for a single target vector (T) is defined as

$$\%Average\ Error = 100 * \frac{1}{M}\sum_{i=1}^{M}\left|\frac{T_i - a_i}{T_i}\right| \qquad (66)$$

where: a = Predicted Value

T = Target Value

M = Number of Output Variables.

The *%AE* was utilized to determine the accuracy of the generalized characterization results.

5.2.2.4. Determination of the Number of Hidden Layer Neurons

Data sets uTR_E_2 and TR_E_2a were utilized to determine the appropriate number of hidden layer neurons for accurate and robust training. Note that each flaw-type has its own ANN. This was done for two reasons. The number of characterizations changes for different flaw-types, and yields more accurate characterization results.

The appropriate number of hidden layer neurons was determined by reducing the error to a minimum value before the ANN begins to over-fit. When the ANN over-fits, the error may increase, and generalization results would not be acceptable.

The number of neurons in the hidden layer was determined by trial and error. The determination of the number of hidden neurons in the hidden layer utilized data subgroup uTR_E_2 and TR_E_2a. An *MSE* goal of 0.01 was chosen. The *MSE* goal of 0.01 should force approximately a ± 1% error between the un-scaled target and the output.

As can be seen in Table 20, for each flaw-type, the *MSE* drops significantly from 3 to 5 hidden neurons, then very little. Flaw-type #4 could not be trained below an *MSE* = 0.17. The number of hidden neurons was determined to be 5.

Table 20. Number of Hidden Neurons vs. Flaw-type Listing the MSE (and ME) Values

# of Hidden Neurons	Flaw-type			
	1	2	3	4
3	0.0144 (0.12)	0.0130 (0.12)	0.0010 (0.03)	0.1723 (0.42)
5	0.0011 (0.03)	0.0007 (0.03)	0.0010 (0.03)	0.1710 (0.41)
7	0.0007 (0.03)	0.0007 (0.03)	0.0008 (0.03)	0.1708 (0.41)
10	0.0007 (0.03)	0.0008 (0.03)	0.0009 (0.03)	0.1706 (0.41)

5.2.3. Correlation Analysis of the Input Values and Target Output

A correlation analysis between the Input Values and the Target Output (Characterization Values) was prepared. The correlation coefficients for each flaw-type are presented in Appendix C.

5.2.4. Training and Storing the Neural Networks

Once all the aforementioned parameters were either set or determined, ANNs for each generalization group were trained. The neural network results were stored as a ".mat" data file. Net_char_E_2a5.mat contains the neural network structure and parameters generated by using uTR_E_2 and TR_E_2a subgroup with 5 hidden neurons. Thus, resulting in Net_char_E_2a5. The output of the training for net_char_E_2a5 can be found in Appendix D. These results were typical of results for net_char_E_3a5 through 6a5.

6. Results and Discussion

The main goal (specific to ECT) of this research, using the above outlined procedures, was to achieve an in-situ flaw classification and characterization technique. This section details the results using this process.

The results and discussion section is divided into three parts. The first section details important intermediate analysis. The second part details the Bayes classification of the ECTD test flaws. The third section presents an overview of the characterization of the ECTD test flaws. The final section is a discussion of the generated results.

6.1. Summary of Important Intermediate Analysis

This section summarizes important results of the intermediate analysis sections.

1. The resulting compressed data exhibit non-Guassian distributions and clustering.
2. The B-distances computed (both the UPeBD and UPeBDZ, in percent) seem to parallel and bound the % Incorrect Classification. The calculated upper bound for 15 PCs was 0 and 14.22% (UPeBD and UPeBDZ, in that order) bounding the 1.1% incorrectly classified. However, the Gaussian-based calculated B-distances may be inaccurate due to non-Gaussian data.
3. Using only the phase and magnitude, 39% of the flaws were accurately classified.
4. Using only the IP features generated from the CWTs, 42% of the flaws were accurately classified.
5. Using only the polynomial coefficients of the mixed, inductive reactance component of the absolute ECTDFS, 46% of the flaws were accurately classified.
6. Using four out of five of the feature families, 72% of the flaws were accurately classified.

These six results are analyzed in the following sections.

6.2. Bayes Classification of Test Flaws

The Bayes Classification of test flaws section is divided into four parts. The first section describes results generated using only the CWT generated features, after compression, with an extract one cross validation system. The second part contains the results generated with the cross validation system of extract four. The third section contains the results generated with the cross validation system of extract one. The fourth section looks at the relationship between outliers and misclassification.

6.2.1. Bayes Classification of Test Flaws using only the CWT Generated, Compressed Features Employing the Cross Validations System of Extract One

A Bayes classification was performed using only PCA compressed (15 PCs), CWT generated features. The CWT features included both the geometric moments and the IP features. The incorrect classification percentage using the extract one cross validation system was 35.87. The extract one cross validation system correctly classified 64.13 % of the flaws.

6.2.2. Bayes Classification of Test Flaws Using the Cross Validation System of Extract Four

As discussed in Section 5.1.3, the results, found in Table 21, were generated using the first cross validation system (extract four). The incorrect percentage using all feature families (4 extracted) was 25.00. Thus, the system correctly identified the flaw-type 75 % of the time. Table 22 reorganizes the above information to show the number of misclassified flaws by the flaw-type.

Flaw-type 3 was the least misclassified and flaw-type 4 was the most misclassified.

6.2.3. Bayes Classification of Test Flaws Using the Cross Validation System of Extract One

As discussed in Section 5.1.3, the following results, listed in Table 23, were generated using the second cross validation system (extract one). The average incorrect percentage (extracting one) using all feature families was 27.17.

Table 21. Cross Validation System One, Sub-Group # with Misclassified Flaw Example Names.

Sub-Group #	Flaw Example Names
1, 9, 11, 13, 15	none
2	'DHRSMPC001I004_2' (3)
3	'DHR00PC004I021_4' (4)
4	'DAR00BC100I022_1' (2) 'DHR00PC005I022_1' (4)
5	'DHR00PC005I022_4' (4)
6	'DAR051C002I013_1' (2) 'DHR000C204I031_1' (1)
7	'DHR00PC035I024_1' (4)
8	'DAR051C004I005_1' (2) 'DHR000C203I033_1' (1)
10	'DHR00PC044I059_1' (4)
12	'DHR00PC048I063_1' (4)
14	'DHR00PC049I046_2' (4)
16	'DHR00PC049I064_2' (4)
17	'DHR00PC049I064_4' (4)
18	'DHR00PC051I048_3' (4)
19	'DAR051C015I016_1' (2)
20	'DAR051C099I010_1' (2) 'DHR00PC051I048_5' (4)
21	'DHR00PC051I066_3' (4)
22	'DAR0BWC079I015_1' (2) 'DHR00PC051I066_4' (4)
23	'DAR0BWC080I018_1' (4)

Table 22. Number of Misclassified by Flaw-types for Cross-validation System One (Extract 4).

	Flaw-type			
	1	**2**	**3**	**4**
Number Misclassified	2	6	1	14

Table 23. Cross Validation System Two, Sub-Group # with Misclassified Flaw Example Names.

Sug-groups	Flaw Example Names (Flaw-type)
1-5,7, 9-26, 28, 30, 32-40, 44, 47, 50 – 71, 75, 77, 78, 80, 82, 84, 88, 92	none
6	'DHR000C204I031_1' (1)
8	'DHR000C203I033_1' (1)
27	'DAR00BC100I022_1' (2)
29	'DAR051C002I013_1' (2)
31	'DAR051C004I005_1' (2)
41-43	'DAR051C014I019_1' (2) 'DAR051C015I016_1' (2) 'DAR051C099I010_1' (2)
45, 46	'DAR0BWC079I015_1' (2) 'DAR0BWC080I018_1' (2)
48, 49	'DHRSMPC001I004_2' (3) 'DHRSMPC001I004_3' (3)
72 - 74	'DHR00PC004I021_4' (4) 'DHR00PC005I022_1' (4) 'DHR00PC005I022_4' (4)
76	'DHR00PC035I024_1' (4)
79	'DHR00PC044I059_1' (4)
81	'DHR00PC048I063_1' (4)
83	'DHR00PC049I046_2' (4)
85-87	'DHR00PC049I064_2' (4) 'DHR00PC049I064_4' (4) 'DHR00PC051I048_3' (4)
89-91	'DHR00PC051I048_5' (4) 'DHR00PC051I066_3' (4) 'DHR00PC051I066_4' (4)

Thus, the system correctly identified the flaw-type 73 % of the time. Table 24 reorganizes the above information to show the number of misclassified flaws by flaw-type.

Flaw-types 1 and 3 were the least misclassified and flaw-type 4 was the most.

Since there were many misclassified flaws, a look at the outliers as they relate to the misclassified flaws, is discussed in the next section.

6.2.4. Remarks on Outliers and Misclassification

Referring to Table 14, this lists all outliers by flaw-type. Now, reorganize Table 14 in the same manner as Tables 22 and 24. This yields Table 25.

If the results found in Tables 22, 24, and 25 are combined (Table 26), the number of outliers seems to have an inverse relationship with the number of misclassified. The outliers may help complete the feature space so that classification accuracy is increased.

6.3. Characterization of Test Flaws

Five trained ANNs (one each for the 5 generalization subsets, net_char_E_3a5 through 6a5) were generated to check for good generalization results. The MSE (and ME) calculated below did not use the scaled T and Y. Thus, the MSE values were much larger. The results can be seen in Table 27.

Results generated using the five subgroups (with 15 PCs and 5 hidden neurons) are given in Table 28. This table contains the % Average Error (%AE) given for each flaw-type. The %AE was determined using all the characteristics for that particular flaw-type.

As can be seen in Table 28, Flaw-type 1 error was very unstable. The errors range from 148 to 25%. Flaw-types 2-4 have moderately stable characteristic errors. The average error for all flaw-types (excluding flaw-type 1) = 12.8 %.

Table 24. Number of Misclassified by Flaw-types for Cross-validation System 2 (Extract 1).

	Flaw-type			
	1	**2**	**3**	**4**
Number Misclassified	2	8	2	13

Table 25. Number of Outliers by Flaw-types.

	Flaw-type			
	1	**2**	**3**	**4**
Number of Outliers	11	4	5	0

Table 26. Total Number of Misclassified and Outliers by Flaw-types.

	Flaw-type			
	1	**2**	**3**	**4**
Number Misclassified	4	14	3	27
Number of Outliers	11	4	5	0

Table 27. MSE and ME Values According to Flaw-type and Generalization Subset.

Flaw-type	**Generalization Subsets**				
	2	**3**	**4**	**5**	**6**
1	4383.00 (66.20)	27.88 (5.28)	137.40 (11.72)	277.31 (16.65)	205.47 (14.33)
2	28.82 (5.37)	32.54 (5.07)	1.35 (1.16)	32.85 (5.73)	4.75 (2.18)
3	0.19 (0.44)	0.57 (0.75)	0.83 (0.91)	5.75 (2.40)	0.79 (0.89)
4	20.72 (4.55)	21.68 (4.66)	20.14 (4.49)	29.05 (5.39)	6.52 (2.55)

Table 28. % Average Error of Flaw Characterizations divided into Neural Network Run Numbers (Subgroups) and Flaw-types.

Flaw-type	**Neural Network Run Number (corresponding to Subgroup)**				
	2a5	**3a5**	**4a5**	**5a5**	**6a5**
1	148.05	24.87	46.18	46.91	14.54
2	14.22	13.35	7.86	16.84	20.70
3	1.29	0.62	2.35	3.83	0.92
4	29.25	15.32	36.22	15.56	12.98

The EddyC ANN outputs containing the values for T (target) and Y (generated values) for each flaw-type group are assembled in Appendix D.

6.4. Discussion of Results

The results of this current research effort have shown the following.

1. The B-distance can be used to predict the % incorrect classification.
2. The information contained within the individual feature families compliment themselves when used together.
3. The CWT contains at least enough information to correctly classify the flaws 64% of the time using the IP features.
4. Initial Template matching results (from the PDD database) yielded correct classification of 69%.
5. The number of outliers seems to have an inverse relationship with the number of misclassifications.
6. The different SG tubing flaw-types may be classified using the ECTDFS features with very good accuracy as compared to traditional industry methods.
7. The characteristics can also be determined accurately for three of the four ECTD classifications. The characteristics are more robust than only a %TW as traditionally used in the industry.

7. Summary, Conclusions and Recommendations for Future Work

7.1. Summary

The ECT technology has a proven track record at both detecting SG tubing flaws and characterizing the flaws (flaw sizing given in % through-wall or %TW). The type of flaw is ususally narrowed down, but not determined, by the location of the flaw within the tube, whether the flaw occurs as an outer diameter (OD) or an inner diameter (ID) flaw, and the SG vendor. A profile of the physical degradation can be determined if there is information contained in the mixed absolute ECT signal. The decision about the plugging or pulling out a degraded SG tube after a certain %TW damage is determined by the ECT specialist. Different degradation mechanisms cause the SG tube wall to physically deterioate differently. The type of degradation is usually determined after a tube were pulled out and inspected.

The purpose of this dissertation is to develop and impliment an automated method for the classification and advanced characterization of defects in HX and SG tubing.

At this time, using basic bobbin-coil ECT, there was no method available to classify the type or volume of degradation of a flaw while the tube is still in (while the SG is on-line but not when the plant is operating) the SG. Therefore, two improvements were made in basic bobbin-coil ECTD analysis.

1. In-situ classification of tube flaws as indicated by the ECTD signal.
2. Expanded in-situ characterization (flaw sizing) of the flaws.

These two improvements enhanced the robustness of characterization as compared to traditional methods.

The approach that was developed for the diagnosis of degradation (both classification and advanced characterization) of SG tubes consists of several steps. For steps 3 through 6, new analysis techniques were required. All the steps are enumerated below.

1. ECTD pre-processing
2. Entering known information from the PDD
3. Transformation of the mixed, complex, differential ECTD flaw signal (ECTDFS) using the continuous wavelet transformation (CWT)
4. Feature extraction and compression of extracted features utilizing Principal Component Analysis (PCA)
5. Tube defect classification using compressed feature vector and CWT using a traditional pattern recognition (PR) technique
6. Tube defect characterization (or flaw sizing) using multiple neural networks (ANNs), one for each flaw-type.

The major results of this research support that the B-distance can be used to predict the % incorrect classification, that the CWT contains at least enough information to correctly classify the flaws, and that the different SG tubing flaw-types may be classified and characterized using the ECTDFS features with very good accuracy as compared to traditional industry methods.

7.2. Conclusions

The following are the conclusions reached from this research:

1. A feature extraction program acquiring relevant information from both the mixed, absolute and differential ECTDFS was successfully employed.
2. The CWT was utilized to extract more information from the mixed, complex differential ECTDFS.
3. IP techniques used to extract the information contained in the generated CWT, classified the ECTDFSs with success.
4. The ECTDFSs were accurately classified, utilizing the compressed feature vector, using a Bayes classification system.
5. An estimation of the upper bound for the probability of error, using Bhattacharyya distance, was successfully applied to the Bayesian classification.

6. The classified ECTDFSs were separated according to flaw-type (classification) to enhance characterization. The characterization routine used separate, flaw-type specific ANNs that made the characterization of the ECTD flaw more robust.
7. The inclusion of outliers may help complete the feature space so that classification accuracy is increased.

Given that the ECTD signals appear very similar, there may not be enough information to make a highly accurate (>95%) classification or characterization using this system. It is necessary to have a large database for more accurate system learning.

7.3. Recommendations for Future Work

There are four primary areas for future work. The first area would be to incorporate more flaw examples and variety into the database. After incorporating more flaw examples into the database, a more thorough sensitivity analysis for the geometric moments and the absolute polynomial coefficients as they pertain to classification should be done. The third area would be to examine other ECTD information extraction transformations, specifically using the Windowed Wigner-Ville Transformation and/or Short Time Fourier Transformation. Different classification technique, such as ANNs, could be utilized.

LIST OF REFERENCES

References

1. P. E. MacDonald, V. N. Shah, L. W. Ward, and P. G. Ellison, "Steam Generator Tube Failures," NUREG/CR-6365, INEL-95/0383, April 1996.

2. J. A. Jones; "Nondestructive Evaluation Sourcebook," EPRI NP-7466-SL Final Report, September 1991.

3. P. Y. Joubert, Y. L. Bihan, D. Placko, and F. Lepoutre; "A Wavelet/Bayes Approach for Small Notch Detection in Eddy Current Testing of SteamGenerator Tubes," Review of Progress in Quantitative Nondestructive Evaluation, CP509, pp. 823-830, American Institute of Physics, 2000.

4. A. Chen and K. Miya, "A Novel Signal Processing Technique for Edyy-Current Testing of Steam Generator Tubes," IEEE Transaction on Magnetics, Vol. 34, No.3, pp. 642-648, May 1998.

5. Y. Cho and J. P. Basart, "Automated Flaw Detection for Tube Support Plate Signals," Review of Progress in Quantitative Nondestructive Evaluation, CP509, pp. 839-845, American Institute of Physics, 2000.

6. T. Chady, M Enokizono, T. Todaka, Y. Tsuchia, R. Sikora, and M. Anan, "Eddy Current Tomography of Multi-layered Aluminum Structures," Review of Progress in Quantitative Nondestructive Evaluation, CP509, pp. 425-432, American Institute of Physics, 2000.

7. T. Taniguchi, D. Kacprzak, S. Yamada, and M. Iwahara, "Wavelet-Based Processing of ECT Images for Inspection of Printed Circuit Board," IEEE Transaction on Magnetics, Vol. 37, No. 4, pp. 2790-3, July 2001.

8. B. Damiano, S. W. Kercel, R. W. Tucker Jr., and S. A. Brown-VanHoozer, "Recognizing a Voice from its Model," 2000 IEEE International Conference on

Systems, Man and Cybernetics, Document Number 0-7803-6583-6/00, pp. 2216-2221, October 2000.

9. T. Kaewkonkga, Y. H. Joe Au, R. Rakowski, and B. E. Jones, "Continuous Wavelet Transform and Neural Network for Condition Monitoring of Roto-dynamic Machinery," IEEE Instruments and Measurement Technology Conference, pp. 1962-1966, May 2001.

10. J. Hou and M. K. Hinders, "Dynamic Wavelet Fingerprint Identification of Ultrasound Signals," Material Evaluation, pp.1089-1093, September 2002.

11. W. Yan and B.R. Upadhyaya, "Development of An Automated Diagnostics System for Eddy Current Analysis Using Applied Artificial Intelligence Techniques," EPRI/UTNE/93-09, February 1996.

12. W. C. Hooper and B. R. Upadhyaya, "An Automated Diagnostics System for Eddy Current Data Analysis of steam Generator Tubing," Research Report, EPRI/UTNE/93-17, January 1998.

13. S. Erbay, G. Hazi, and K. Y. Sung, "Eddy Current Test Data Analysis for Steam Generator Tubing diagnosis Using Artificial Intelligence methods," EPRI/UTNE/98-04: February 1998.

14. R. N. de Mesquita, D. K. S. Ting, E. L. L. Cabral, and B. R. Upadhyaya, "Feature Extraction Methods for Classification of Steam Generator Tube Defects Using Self-Organizing Maps," Proceedings of MARCON 2002, May 2002.

15. A. da Silva and B. R. Upadhyaya, "An Integrated Approach for Plant Monitoring and Diagnosis Using Multi-resolution Wavelet Analysis," Research Report, UTNE/AAS/97-01, December 1997.

16. H. L. Libby, *Introduction to Electromagnetic Nondestructive Test Methods*, John Wiley, New York, 1971.

17. J. Blitz, *Electrical and Magnetic Methods of Non-destructive Testing,* Chapman & Hall, 1997.

18. EPRI, "Survey of Nondestructive Examination (NDE) – An Overview for Junior or Senior Students", 1996.

19. C. K. Sword and M. Simaan, "Estimation of Mixing Parameters for Cancellation of Discretized Eddy Current Signals Uisng Time and Frequency Domain Techniques," Journal of Nondestructive Evaluation, Vol. 5, No. 1, pp. 37-35, 1985.

20. J. Stolte, L. Udpa and W. Lord, "Multifrequency Eddy Current Testing of Steam Generator Tubes Using Optimal Affine Transformation," Materials Science Research, Vol. 21, pp. 821-30.

21. Gary Henry, Private Communication, 2002.

22. EPRI, "Performance Demonstration Database," June 1994.

23. Wavelet Toolbox User's Guide, The MathWorks, Inc., 1999.

24. A. Grossman and J. Morlet, "Decomposition of Hary function into square integrable wavelets of constant shape," SAIM J. of Math. Anal., 15(4): 723-736, July 1984.

25. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1999.

26. O. Rioul and M. Vetterli, "Wavelets and Signal Processing," IEEE SP Magazine, pp. 14-38, October 1991.

27. G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, Massachusetts, 1996.

28. Using MATLAB, The MathWorks, Inc., pp. 12-7,9, 1999.

29. Image Processing Toolbox User's Guide, The MathWorks, Inc., 1999.

30. S. Theodoris and K. Koutroumbas, *Pattern Recognition,* Academic Press, 1999.

31. R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, Second Edition, John Wiley, New York, 2001.

32. Statistics Toolbox User's Guide; The MathWorks, Inc., 1999.

33. Notes from Nuclear Engineering 597, "Advanced Monitoring and Diagnostic Techniques," 2000.

34. E. L. Russell, L. H. Chiang, and R. D. Braatz, *Data-Driven Techniques for Fault Detection and Diagnosis in Chemical Processes*, Springer-Verlag, New York, 2000.

35. N. Kaistha and B. R. Upadhyaya, "Incipient Fault Detection and Isolation of Field Devices in Nuclear Power Systems Using Principal Component Analysis," Nuclear Technology, Vol. 136, pp. 221-230, November 2001.

36. B. M. Wise and N. B. Gallagher, PLS_Toolbox 2.0 for use with MATLAB, Eigenvector Research Inc., 1998.

37. B.R. Upadhyaya and H.W. Sorenson, "Bayesian Discriminant Approach to Input Signal Selection in Parameter Estimation Problems," Information Sciences, Vol. 12, pp. 61-91, Elsevier North Holland, 1977.

38. G. Saon and M. Padmannabhan, "Minimum Bayes Error Feature Selection for Continuous Speech Recognition," IBM T. J. Watson Research Center, Yorktown Heights, NY, 2000.

39. R. L. Kettig and D. A. Landgrebe, "Classification of Multispectral Image Data by Extraction and Classification of Homogeneous Objects," IEEE Transactions on Geoscience Electronics, Vol. GE-14, No. 1, pp. 19-26, January 1976.

40. G. Cybenko, "Approximation by Superpositions of a Sigmodial Function," Mathematics of Control, Signals and Systems, 2, pp. 303-314, 1989.

41. K. Funahashi, "On Approximate Realization of Continous Mappings by Neural Networks," Neural Networks, 2, pp. 183-192, 1989.

42. S. Haykin, *Neural Networks, A Comprehensive Foundation*, Macmillan, New York, 1994.

43. K. Hornik, M. Stinchcombe and H. White, "Multilayer Feedforward Networks are Universal Approximators," Neural Networks, 2, pp. 359-366, 1989.

44. MATLAB, Neural Networks Toolbox: Version 3, The Math Works, Inc., 1999.

45. L. H. Tsoukalas, R. E. Uhrig, *Fuzzy and Neural Approaches in Engineering*, Wiley-Interscience, pp. 191-405; 1997.

46. M. T. Hagan, H. B. Demuth, M. Beale, *Neural Network Design*, PWS Publishing Company, 1995.

BIBLIOGRAPHY

Bibliography

P. E. MacDonald, V. N. Shah, L. W. Ward and P. G. Ellison, "Steam Generator Tube Failures," NUREG/CR-6365, INEL-95/0383, 1995.

O. Rioul and M. Vetterli, "Wavelets and Signal Processing," IEEE SP Magazine, pp. 14-38, Oct. 1991.

A. Grossman and J. Morlet, "Decomposition of Hary function into square integrable wavelets of constant shape," SAIM J. of Math. Anal., 15(4): 723-736, July 1984.

N. Kaistha and B. R. Upadhyaya, "Incipient Fault Detection and Isolation of Field Devices in Nuclear Power Systems Using Principal Component Analysis," Nuclear Technology, Vol. 136, pp. 221-230, Nov. 2001.

T. Nguyen, D. Koilillia, "The Theory and Desing of Arbitrary-Length Cosine-Modulated Filter Banks and Wavelets, Satisfying Perfect Reconstruction", IEEE Transactions on Signal Processing, Vol. 44 No. 3, pp. 473-483, March 1996.

R. G. Stockwell, L. Manisiha, R. P. Lowe, "Localization of the Complex Spectrum: The S Transform", IEEE Transactions on Signal Processing, Vol. 44, No. 4, pp. 998-1001, April 1996.

Y. T. Chen, K. C. Ho, "Multiresolution Analysis, Its Link to the Discrete Parameter Wavelet Transform, and Its Initialization", IEEE Transactions on Signal Processing, Vol. 44, No. 4, pp. 1001-1006, April 1996.

L Cohen, "A General Approach for Obtaining Joint Representations in Signal Analysis-Part I: Characteristic Function Operator Method", IEEE Transactions on Signal Processing, Vol. 44, No. 5, pp. 1080-1090, May 1996.

L Cohen, “A General Approach for Obtaining Joint Representations in Signal Analysis-Part II: General Class and Local Values, and Bandwidth”, IEEE Transactions on Signal Processing, Vol. 44, No. 5, pp. 1091-1098, May 1996.

J. R. O’Hair, B. W. Suter, “The Zak Transform and Decimated Time-Frequency Distributions”, IEEE Transactions on Signal Processing, Vol. 44, No. 5, pp. 1099-1110, May 1996.

E. P. Serrano, M. A. Fabio, “Application of the Wavelet Transform to Acoustic Emission Signals Processing”, IEEE Transactions on Signal Processing, Vol. 44, No. 5, pp. 1270-1275, May 1996.

M. G. Strintzis, “Optimal Biorthogonal Wavelet Bases for Signal Decomposition”, IEEE Transactions on Signal Processing, Vol. 44, No. 6, pp. 1406-1417, June 1996.

H. M. Ozaktas, O. Arikan, M. A. Kutay, G. Bozdagi, “Digital Computation of the Fractional Fourier Transform”, IEEE Transactions on Signal Processing, Vol. 44, No. 9, pp. 2141-2150, Sept 1996.

R. Karakarala, J. A. Cadzow, “Estimation of Phase for Noisy Linear Phase Signals”, IEEE Transactions on Signal Processing, Vol. 44, No. 10, pp. 2483-2497, October 1996.

J. J. Rajan, P. J. W. Rayner, “Generalized Feature Extraction for Time-Varying Autoregressive Models”, IEEE Transactions on Signal Processing, Vol. 44, No. 10, pp. 2498-2507, Oct. 1996.

C. S. Detka, A. El-Jaroudi, “The Generalized Evolutionary Spectrum”, IEEE Transactions on Signal Processing, Vol. 44, No. 11, pp. 2877-2881, Nov. 1996.

G. Mandyam, N. Ahmed, “The Discrete Laguerre Transform: Derivation and Applications”, IEEE Transactions on Signal Processing, Vol. 44, No. 12, pp. 2925-2931, Dec. 1996.

Z. Xiong, K. Ramchandran, C. Herley and M. T. Orchard, “Flexible Tree-Structured Signal Expansions Using Time-Varying Wavelet Packets”, IEEE Transactions on Signal Processing, Vol. 45, No. 2, pp. 333-345, Feb. 1997.

B. Picinbono, P. Bondon, "Second-Order Statistics of Complex Signals", IEEE Transactions on Signal Processing, Vol. 45, No. 2, pp. 411-420, Feb. 1997.

S. Del Marco and J. Weiss, "Improved Transient Signal Detection Using a Wavepacket-Based Detector with an Extended Translation-Invariant Wavelet Transform", IEEE Transactions on Signal Processing, Vol. 45, No. 4, pp. 841-850, April 1997.

H. Bolcskei and F. Hlawatsch, "Disrete Zak Transforms, Polyphase Transforms, and Applications", IEEE Transactions on Signal Processing, Vol. 45, No. 4, pp. 851-866, April 1997.

D. Groutage, "A Fast Algorithm for Computing Minimum Cross-Entropy Positive Time-Frequency Distrabutions', IEEE Transactions on Signal Processing, Vol. 45, No. 8, pp. 1954-1970, Aug. 1997.

X. Xia, "System Identification Using Chirp Signals and Time-Variant filters in the Joint Time-Frequency Domain", IEEE Transactions on Signal Processing, Vol. 45, No. 8, pp. 2072-2084, Aug. 1997.

R. A. Carmona, W. L. Hwang and Brun Torresani, "Charactertization of Signals by the Ridges of their Wavelet Transform", IEEE Transactions on Signal Processing, Vol. 45, No. 10, pp. 2586-2590, Oct. 1997.

B. Scholkolf, K. Sung, C. J. C. Burges, F. Girosi, P. Nyiogi, T. Poggio and V. Vapnik, "Comparing Support Vector Machines with Gaussian Kernals to Radial Basis Fucntion Classifiers", IEEE Transactions on Signal Processing, Vol. 45, No. 11, pp. 2758-2765, Nov. 1997.

V. P. Kumar and E. S. Manolakos, "Unsupervised statistical Neural Networks for Model-Based Object Recognition", IEEE Transactions on Signal Processing, Vol. 45, No. 11, pp. 2709-2718, Nov. 1997.

L. M. D. Owsley, L. E. Atlas and G. D. Berard, "Self-Organizing Feature Maps and Hidden Markov Models for Machine-Tool Monitoring", IEEE Transactions on Signal Processing, Vol. 45, No. 11, pp. 2787-2798, Nov. 1997.

Murray and J. Penman, "Extracting Useful Higher Order Features for Condition Monitoring Using Artificial Neural Networks", IEEE Transactions on Signal Processing, Vol. 45, No. 11, pp. 2821-2828, Nov. 1997.

N. Polyak and W. A. Pearlman, "Filters and Filter Banks for Periodic Signals, the Zak Transform, and Fast Wavelet Decompression", IEEE Transactions on Signal Processing, Vol. 46, No. 4, pp. 857-873, April 1998.

M. S. Crouse, R. D. Nowark and R. G. Baraniuk, "Wavelet-Based Statistical Processing Uisng Hidden Markov Models", IEEE Transactions on Signal Processing, Vol. 46, No. 4, pp. 886-902, April 1998.

Q. Q. Huynh, L. N. Cooper, N. Intrator and H. Shouval, "Classification of Underwater Mammals Using Feature Extraction Based on Time-Frequency Analysis and BCM Theory", IEEE Transactions on Signal Processing, Vol. 46, No. 5, pp. 1202-1207, May 1998.

M. S. Richman, T. W. Parks and R. G. Shenoy, "Discrete-Time, Discrete-Frequency, Time-Frequency Analysis", IEEE Transactions on Signal Processing, Vol. 46, No. 6, pp. 1517-1527, June 1998.

E. Shusterman and M. Feder, "Analysis and Synthesis of 1/f Processes via Shannon Wavelets", IEEE Transactions on Signal Processing, Vol. 46, No. 6, pp. 1698-1702, June 1998.

G. G. Walter and J. Zhang, "Orthononrmal Wavelets with Simple Closed-Form Expressions", IEEE Transactions on Signal Processing, Vol. 46, No. 8, pp. 2248-2251, Aug. 1998.

T. R. Downie and B. W. Silverman, "The Discrete Multiple Wavelet Transform and Thresholding Methods", IEEE Transactions on Signal Processing, Vol. 46, No. 9, pp. 2558-2561, Sept. 1998.

W. Selenick, “Multiwavelet Bases with Extra Approximation Properties”, IEEE Transactions on Signal Processing, Vol. 46, No. 11, pp. 2898-2908, Nov. 1998.

F. Pedersen, “A Gabor Expansion-Based Positive Time-Dependent Power Spectrum”, IEEE Transactions on Signal Processing, Vol. 47, No. 2, pp. 587-590, Feb. 1999.

V. De4Brunner, M. Ozaydin and T. Przebinda, “Resolution in Time-Frequency”, IEEE Transactions on Signal Processing, Vol. 47, No. 3, pp. 783-788, March 1999.

J. C. O’Niell and W. J. Willliams, “A Function of Time, Frequency, Lag and Doppler”, IEEE Transactions on Signal Processing, Vol. 47, No. 3, pp. 789-799, March 1999.

L. Stankovic and V. Katkovnik, “The Wigner Distrabution of Noisy Signals with Adaptive Time-Frequency Varying Window”, IEEE Transactions on Signal Processing, Vol. 47, No. 4, pp. 1099-1108, April 1999.

W. Selenick, “The Slantlet Transform”, IEEE Transactions on Signal Processing, Vol. 47, No. 5, pp. 1304-1313, May 1999.

N. Keshava and J. M. F. Moura, “Matching Wavelet Packets to Gaussian Random Processes”, IEEE Transactions on Signal Processing, Vol. 47, No. 6, pp. 1604-1614, June 1999.

R. D. Nowark and R. G. Baraniuk, “Wavelet-Based Transformations for Nonlinear Signal Processing”, IEEE Transactions on Signal Processing, Vol. 47, No. 7, pp. 18521865, July 1999.

H. Watanabe, Y. Matsumoto, S. Tanaka and S. Katagiri, “A New Approach to Acoustic Monitoring Based on the Generalized Probabilistic Descent Method”, IEEE Transactions on Signal Processing, Vol. 47, No. 9, pp. 2615-2618, Sept. 1999.

M. Coulon, J. Tourneret and A. Swami, “Detection and Classification of Spectrally Equivilent Processes Using Higher Order Statistics”, IEEE Transactions on Signal Processing, Vol. 47, No. 12, pp. 3326-3335, Dec. 1999.

Q. Pan, L. Zhang, G. Dai and H. Zhang, "Two Denoising Methods by Wavelet Transform", IEEE Transactions on Signal Processing, Vol. 47, No. 12, pp. 3401-3406, Dec. 1999.

P. Maass, G. Teschke, W. Willmann and G. Wollmann, "Detection and Classification of Matterial Attributes-A Practicle Application of Wavelet Analysis", IEEE Transactions on Signal Processing, Vol. 48, No. 8, pp. 2432-2438, Aug. 2000.

D. N. Politis, "Computer-Intensive Methods in Statistical Analysis", IEEE Signal Processing Magazine, pp. 39-55, Jan. 1998.

H. D. I. Abarbabel, T. W. Frison and L. S. Tsimring, "Obtaining Order in a World of Chaos: Time Domain Analysis of Nonlinear and Chaotic Signals", IEEE Signal Processing Magazine, pp. 49-64, May 1998.

S. Pei and M. Yeh, "An Introduction to Discrete Finite Frames", IEEE Signal Processing Magazine, pp. 84-96, Nov. 1997.

L. G. Weiss, "Wavelets and Wideband Correlation Processing", IEEE Signal Processing Magazine, pp. 13-32, Jan. 1994.

J. R. Deller Jr., "Tom, Dick and Mary Discover the DFT", IEEE Signal Processing Magazine, pp. 36-50, April 1994.

P. Kraniauskas, "A Plain Man's Guide to the FFT", IEEE Signal Processing Magazine, pp. 24-35, April 1994.

F. Hlawatsch and G. F. Boudreaux-Bartels, "Linear and Quadratic Time-Frequency Signal Representations", IEEE Signal Processing Magazine, pp. 21-67, April 1992.

S. Haykin, "Neural Networks Expanded SP's Horizons", IEEE Signal Processing Magazine, pp. 24-49, March. 1996.

N. Morgan and H. Bourlard, "Continuous Speech Recognition", IEEE Signal Processing Magazine, pp. 25-42, May 1995.

D. R. Hush and B. G. Horne, "Progress in Supervised Neural Networks", IEEE Signal Processing Magazine, pp. 8-39, Jan. 1993.

D. C. Stanford and A. E. Rattery, "Finding Curvilinear Features in Spatial Point Patterns: Principal Curve Clustering with Noise", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 6, pp. 601-609, June 2000.

A. K. Jain, R. P. W. Duin and J. Mao, "Statistical Pattern Recognition: A Review", IEEE Transactions on Pattern Recognition and Machine Intelligence, Vol. 22, No. 1, pp. 4-37, Jan 2000.

S. Pittner and S. V. Kamarthi, "Feature Extraction from Wavelet Coefficients for Pattern Recognition Tasks", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 1, pp. 83-88, Jan. 1999.

T. Randen and J. H. Husoy, "Filtering for Texture Classification: A Comparative Study", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 4, pp. 291-310, April 1999.

H. Frigui and R. Krishaspuram, "A Robust Competitive Clustering Algorithm with Applications in Computer Vision", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 5, pp. 450-465, May 1999.

Q. Ji and R. M. Haralick, "Breakpoint Detection Using Covariance Propagation", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, pp. 845-851, Aug. 1998.

T. K. Ho, "The Random Subspace Method for Constructing Decision Forests", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, pp. 832-844, Aug. 1998.

K. I. Williams and D. Barber, "Baysian Classification with Gaussian Processes", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 12, pp. 1342-1351, Dec. 1998.

M. Gori and F. Scarselli, "Are Multilayer Perceptrons Adequate for Pattern Recognition and Verification?", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 11, pp. 1121-1132, Jan. 1999.

J. Kittler, M. Hatef, R. P. W. Duin and J. Matas, "On Combining Classifiers", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 3, pp. 226-239, March 1998.

J. Ben-Arie and Z. Wang, "Pictorial Recognition of Objects Employing Affine Invariance in the Frequency Domain", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 6, pp. 604-618, June 1998.

Q. M. Tieng and W. W. Boles, "Recognition of 2D Object Contours Using the Wavelet Transform Zero-Crossing Representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 8, pp. 910-916, Aug. 1997.

Y. Mallet, D. Coomans, J. Kautski and O. De Val, "Classification Using Adaptive Wavelets for Feature Extraction", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 10, pp. 1058-1066, Oct. 1997.

T. Hsu and S. Wang, "The K1-Map Reduction for Pattern Classification", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 6, pp. 616-622, June 1997.

J. P. Hoffbeck and D. A. Landgrebe, "Covariance Matrix Estimation and Classification With Limited Training Data", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 7, pp. 763-767, July 1996.

V. Anh, J. Y. Shi and H. T. Tsui, “Scaling Theorems for Zero Crossings of Bandlimited Signals”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 3, pp. 309-320, March 1996.

T. Hastie and R. Tibshirani, “Discriminat Adaptive Nearest Neighbor Classification”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 6, pp. 607-615, June 1996.

C. C. Holmes and B. K. Mallick, “Baysian Wavelet Networks for Nonparametric Regression”, IEEE Transactions on Neural Networks, Vol. 11, No. 1, pp. 27-35, Jan. 2000.

M. Meneganti, F. S. Saviello, R. Tagliaferri, “Fuzzy Neural Networks for Classification and Detection of Anomalities, IEEE Transactions on Neural Networks, Vol. 9, No. 5, pp. 848-861, Sept. 1998.

S. Behnke and N. B. Karayiannis, “Copetitive Neural Trees for Pattern Classification”, IEEE Transactions on Neural Networks, Vol. 9, No. 6, pp. 1352-1371, Nov. 1998.

C. S. Cruz and J. R. Dorronsoro, “A Nonlinear Discriminant Algorithm for Feature Extraction and Data Classification”, IEEE Transactions on Neural Networks, Vol. 9, No. 6, pp. 1370-1376, Nov. 1998.

H. Song and S. Lee, “A Self-Organizing Neural Tree for Large-Set Pattern Classification”, IEEE Transactions on Neural Networks, Vol. 9, No. 3, pp. 369-380, May 1998.

V. Chandrasekaran and Z. Liu, “Topology Constraint Free Fuzzy Gated Neural Networks for Pattern Recognition”, IEEE Transactions on Neural Networks, Vol. 9, No. 3, pp. 483-502, May 1998.

Zaknick, “Characterization of Aluminium Hydroxide Particles from Bayer Process Using Neural Network and Bayesian Classifiers”, IEEE Transactions on Neural Networks, Vol. 9, No. 4, pp. 919-931, July 1997.

J. N. K. Liu and K. Y. Sin, "Fuzzy Neural Networks for Machine Maintenance in Mass Transit Railway System", IEEE Transactions on Neural Networks, Vol. 8, No. 4, pp. 932-941, Sept. 1998.

J. Chang, G. Han, N. C. Griswold, J. F. Duque-Carrillo, E. Sanchez-Sinencio, "Cork Quality Classification System using a Unified Image Processing and Fuzzy-Neural Network Methology", IEEE Transactions on Neural Networks, Vol. 8, No. 4, pp. 964-973, July 1997.

S. Mitra, R. K. De and S. K. Pal, "Knowledge-Based Fuzzy MLP for Classification and Rule Generation", IEEE Transactions on Neural Networks, Vol. 8, No. 6, pp. 1338-1350, Nov. 1997.

C. Lee and D. A. Landgrebe, "Decision Boundary Feature Extraction for Neural Networks", IEEE Transactions on Neural Networks, Vol. 8, No. 1, pp. 75-83, July 1997.

Joshi, N. Ramakrishman, E. N. Houstis and J. R. Rice, "On Neurobiological, Neura-Fuzy, Machine Learning and Statistical Pattern Recognition Techniques", IEEE Transactions on Neural Networks, Vol. 8, No. 1, pp. 18-31, Jan. 1997.

S. Ridella, S. Rovetta and R. Zunino, "Circular Backpropagation Networks for Classification", IEEE Transactions on Neural Networks, Vol. 8, No. 1, pp. 84-97, Jan. 1997.

M. Prakash and M. N. Murty, "Growing Subspace Pattern Recognition Methods and Their Neural-Network Models", IEEE Transactions on Neural Networks, Vol. 8, No. 1, pp. 161-168, Jan. 1997.

E. Voudouri-Maniati, L. Kurz and J. M. Kowalski, "A Neural-Network Approach to Nonparametric and Robust Classification Procedures", IEEE Transactions on Neural Networks, Vol. 8, No. 2, pp. 288-298, March 1997.

S. Lee and H. Song, "A New Recurrent Neural-Network Architecture for Visual Pattern Recognition", IEEE Transactions on Neural Networks, Vol. 8, No. 2, pp. 331-340, March 1997.

Sperduti and A. Starita, "Supervised Neural Network for the Classification of Stuctures", IEEE Transactions on Neural Networks, Vol. 8, No. 3, pp. 714-735, May 1997.

C. Rodriguez, S. Rementeria, J. Ignacio, A. Lafuente, J. Muguerza and J. Perez, "A Modular Neural Network Approach to Fault Diagnosis", IEEE Transactions on Neural Networks, Vol. 7, No. 2, pp. 326-340, March 1996.

G. A. Carpenter, S. Grossberg and J. H. Reynolds, "A Fuzzy ARTMAP Nonparametric Probability Estmator for Nonstaionary Pattern Recognition Problems", IEEE Transactions on Neural Networks, Vol. 6, No. 6, pp. 1330-1336, Nov. 1995.

S. Y. Kung and J. S. Taur, "Decision-Based Neural Networks with Signal/Image Classification Applications", IEEE Transactions on Neural Networks, Vol. 6, No. 1, pp. 170-181, Jan. 1995.

R. Anand, K. Mehrota, C. K. Mohan and S. Ranka, "Efficient Classification for Multiclass Problems Using Modular Neural Networks", IEEE Transactions on Neural Networks, Vol. 6, No. 1, pp. 117-124, Jan. 1995.

J. Mao and A. K. Jain, "Artificail Neural Networks for Feature Extraction and Multivariate Data Projection", IEEE Transactions on Neural Networks, Vol. 6, No. 2, pp. 296-317, March 1995.

Z. H. Michalopoulou, L. W. Nolte and D. Alexandrou, "Performance Evaluation of Multilayer Perceptrons in Signal Detection and Classification", IEEE Transactions on Neural Networks, Vol. 6, No. 2, pp. 381386, March 1995.

Y. El-Sonbaty and M. A. Ismail, "Fuzzy Clustering for Symbolic Data", IEEE Transactions on Fuzzy Systems, Vol. 6, No. 2, pp. 195-204, May 1998.

J. Chiang and P. G. Gader, "Hybrid Fuzzy-Neural Systems in Handwritten Word Recognition", IEEE Transactions on Fuzzy Systems, Vol. 5, No. 4, pp. 497-510, Nov. 1997.

S. Abe and R. Thawonmas, "A Fuzzy Classifier with Ellipsoidal Regions", IEEE Transactions on Fuzzy Systems, Vol. 5, No. 3, pp. 358-368, Aug. 1997.

V. Petridis and A. Kehagias, "Predictive Modular Fuzzy Systems for Time-Series Classification", IEEE Transactions on Fuzzy Systems, Vol. 5, No. 3, pp. 381-397, Aug. 1997.

R. N. Dave and R. Krishnapuram, "Robust Clustering Methods: A Unified View", IEEE Transactions on Fuzzy Systems, Vol. 5, No. 2, pp. 270-293, May 1997.

K. Nozaki, H. Ishibuchi and H. Tanaka, "Adaptive Fuzzy Rule-Based Classification Systems", IEEE Transactions on Fuzzy Systems, Vol. 4, No. 3, pp. 238-250, Aug. 1996.

Z. Chi and H. Yan, "ID3-Derived Fuzzy Rules and Optimized Defuzzification for Handwritten Numerical Recognition", IEEE Transactions on Fuzzy Systems, Vol. 4, No. 1, pp. 24-31, Feb. 1996.

R. Krishnapuram, H. Frigui and O. Nasraoui, "Fuzzy and Possibilisitic Shell Clustering Algorithms and Their Application to Boundary Detection and Surface Approximation-Part I and II", IEEE Transactions on Fuzzy Systems, Vol. 3, No. 1, pp. 29-60, Feb. 1995.

S. Abe and M. Lan, "A Method for Fuzzy Rules Extraction Directly form Numerical Data and Its Application to Pattern Classification", IEEE Transactions on Fuzzy Systems, Vol. 3, No. 1, pp. 18-28, Feb. 1995.

H. K. Kwan and Y. Cai, "A Fuzzy Neural Network and its Application to Pattern Recognition", IEEE Transactions on Fuzzy Systems, Vol. 2, No. 3, pp. 185-193, Aug. 1994.

R. J. Hathaway and J. C. Bezdek, "Switching Regression Models and Fuzzy Clustering", IEEE Transactions on Fuzzy Systems, Vol. 1, No. 3, pp. 195-204, Aug. 1993.

R. Krishnapuram and J. M. Keller, "A Possibilistic Approach to Clustering", IEEE Transactions on Fuzzy Systems, Vol. 1, No. 2, pp. 98-110, May 1993.

P. K. Simpson, "Fuzzy Min-Max Neural Networks-Part 2: Clustering", IEEE Transactions on Fuzzy Systems, Vol. 1, No. 1, pp. 32-45, Feb. 1993.

S. G. Chang, B. Yu and M. Vetterli, “Adaptive Wavelet Thesholding for Image Denoising and Compression”, IEEE Transaction on Image Processing, Vol. 9 No. 9, pp. 1532-1546, Sept. 2000.

T. Chau and A. K. C. Wong, “Pattern Discovery by Residual Analysis and Recursive Partitioning”, IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 6, pp. 833-852, Nov/Dec. 1999.

S. P. Sullivan, S. P. Smith and F. L. Sharp, “Simultaneous Absolute and Differential Operation of Eddy Current Bobbin Probes for Heat Exchanger Tube Inspection”, Materials Evaluation, pp. 634-638, May 2000.

M. Blodgett, W. Haasan and P. B. Nagy, “Theoretical and Experimental Investigations of the Latteral Resolution of Eddy Current Imaging”, ”, Materials Evaluation, pp. 647-654, May 2000.

M. T. Shyamsunder, C. Rajagopalan, B. Raj, S. K. Dewngan, B. P. C. Rao and K. K. Ray, “Pattern Recognition Approaches for the Detection and Characterization of Discontinuities by Eddy Current Testing”, Materials Evaluation, pp. 93-101, Jan. 2000.

F. Morabito, “Independent Component Analysis and Feature Extraction Techniques for NDT Data”, Materials Evaluation, pp. 85-92, Jan. 2000.

L. W. Schmerr Jr. and S. Song, “The Use of Probabilistic Neural Networks for Discontinuity Classification Problems”, Materials Evaluation, pp. 80-84, Jan. 2000.

G. Katragadda, D. Lewis, J. Wallace and J. Si, “Swept Frequency Eddy Current Material Profiling Using Radial Basis Function Neural Networks for Inversion”, Materials Evaluation, pp. 70-73, Jan. 2000.

H. Chang, F. C. Schoenig Jr. and J. A. Soules, “Eddy Current Offers a Powerful Tool for Investigating Residual Stress and Other Metallurgical Properties”, Materials Evaluation, pp. 1257-1260, Dec. 1999.

H. Hoshikawa and K. Koyama, "Eddy Current Distrabution Using Parameters Normalized by Standard Penetration Depth", Materials Evaluation, pp. 587-593, June 1999.

R. Murthy, N. M. Bilgutay and O. K. Kaya, "Detection of Ultrasonic Anomoly Signals Using Wavelet Decomposition", Materials Evaluation, pp. 1274-1279, Nov. 1997.

D. J. Hagemaier and K. Nguyen, "Automated Eddy Current Scanning of Aircraft For Corrosion Detection", Materials Evaluation, pp. 91-95, Jan. 1994.

A. Sabbagh, J. C. Treece, R. K. Murphy and L. W. Woo, "Computer Modeling of Eddy Current Nondestructive Testing", Materials Evaluation, pp. 1252-1257, Nov. 1993.

T. Stepinski and N. Maszi, "Conjugate Spectrum Filters for Eddy Current Signal Processing", Materials Evaluation, pp. 839-844, July 1993.

R. Palanisamy, "Developments In Eddy Current Nondestructive Testing", Materials Evaluation, pp. 1158-1159, Sept. 1991.

Udpa and S. S. Udpa, "Eddy Current Defect Characterzation Using Neural Networks", Materials Evaluation, pp. 342-347, March 1990.

R. Valleau, "Eddy Current Nondestructive Testing of Graphite Composite Materials", Materials Evaluation, pp. 230-239, Feb. 1990.

R. A. Baker and R. S. Tombaugh, "Eddy Current Examination of Heat-Exchanger Tubing with Inside-Surface Pitting", Materials Evaluation, pp. 55-59, Jan. 1990.

D. A. Hagamaier and J. A. Register, "Mock Eddy Current Demonstration: Cracks versus Notches", Materials Evaluation, pp. 50-54, Jan. 1990.

C. K. Sword and M Simaan, "Estimation of Mixing Parameters for Cancellation of Discretized Eddy Current Signals Using Time Frequnecy Domain Techniques", Journal of Nondestructive Evaluation, Vol. 5, No. 1, pp. 27-35, 1985.

H. L. Libby, *Introduction to Electromagnetic Nondestructive Test Methods*, John Wiley and Sons, Inc., 1971.

J. Blitz, *Electrical and Magnetic Methods of Non-destructive Testing*, Chapman & Hall, 1997.

R. Carmona, W. Hwang and B. Torresani, *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with Implementation in S*, Academic Press, 1998.

S. V. Vaseghi, *Advanced Signal Processing and Digital Noise Reduction*, Wiley Teubner, 1996.

J. C. Goswami and A. K. Chan, *Fundamentals of Wavelets: Theory, Algorithms and Applications*, John Wiley and Sons, 1999.

A. D. Poularikas, *The Handbook of Formulas and Tables for Signal Processing*, CRC Press, 1999.

S. Theodoris and K. Koutroumbas, *Pattern Recognition*, Academic Press,1999.

L. H. Tsoukalas and R. E. Uhrig, *Fuzzy and Neural Approaches in Engineering*, John Wiley and Sons, 1997.

M. T. Hagan, H. B. Demuth and M. Beale, *Neural Network Design*, PWS Publishing Company, 1995.

E. L. Russell, L. H. Chiang, and R. D. Braatz, *Data-Driven Techniques for Fault Detection and Diagnosis in Chemical Processes, Springer-Vertag*, New York, 2000.

S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1999.

G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, Massachusetts, 1996.

MATLAB, Using MATLAB: Version 5, The Math Works, Inc.,1999.

MATLAB, Wavelet Toolbox User's Guide, The Math Works, Inc.,1999.

MATLAB, Statisitics Toolbox User's Guide, The Math Works, Inc.,1999.

MATLAB, Signal Processing Toolbox User's Guide, The Math Works, Inc.,1999.

MATLAB, Neural Networks Toolbox: Version 3, The Math Works, Inc.,1999.

MATLAB, Fuzzy Logic Toolbox User's Guide, The Math Works, Inc.,1999.

B. M. Wise and N. B. Gallagher, PLS_Toolbox 2.0 for use with MATLAB, Eigenvector Research Inc., 1998.

B. R. Upadhyaya, NE 521, 1996.

B. R. Upadhyaya, NE 522, 1997.

B. R. Upadhyaya, NE 653, 1998.

J. W. Hines and Uhrig, NE 577: "Introduction to Neural Networks", 1996.

J. W. Hines and Uhrig, NE 578: "Introduction to Fuzzy Logic", 1997.

APPENDICES

Appendix A. Scatter Plots of the CWT Geometric Moments and Polynomial Coefficients

Geometric Moments

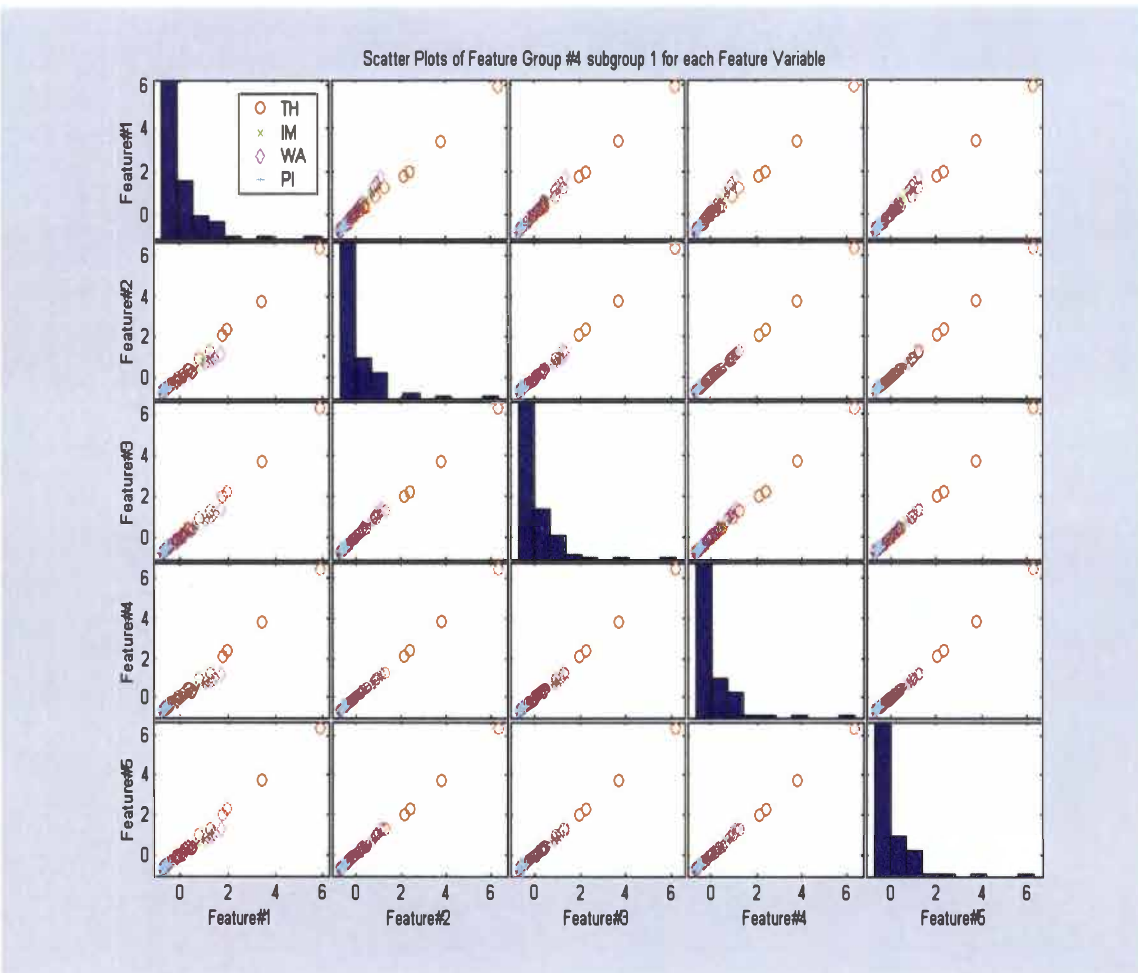

Figure A1. Scatter Plots of Geometric Moments, Subgroup 1 (G11, G12, G13 and G14, Note that G10 = 0).

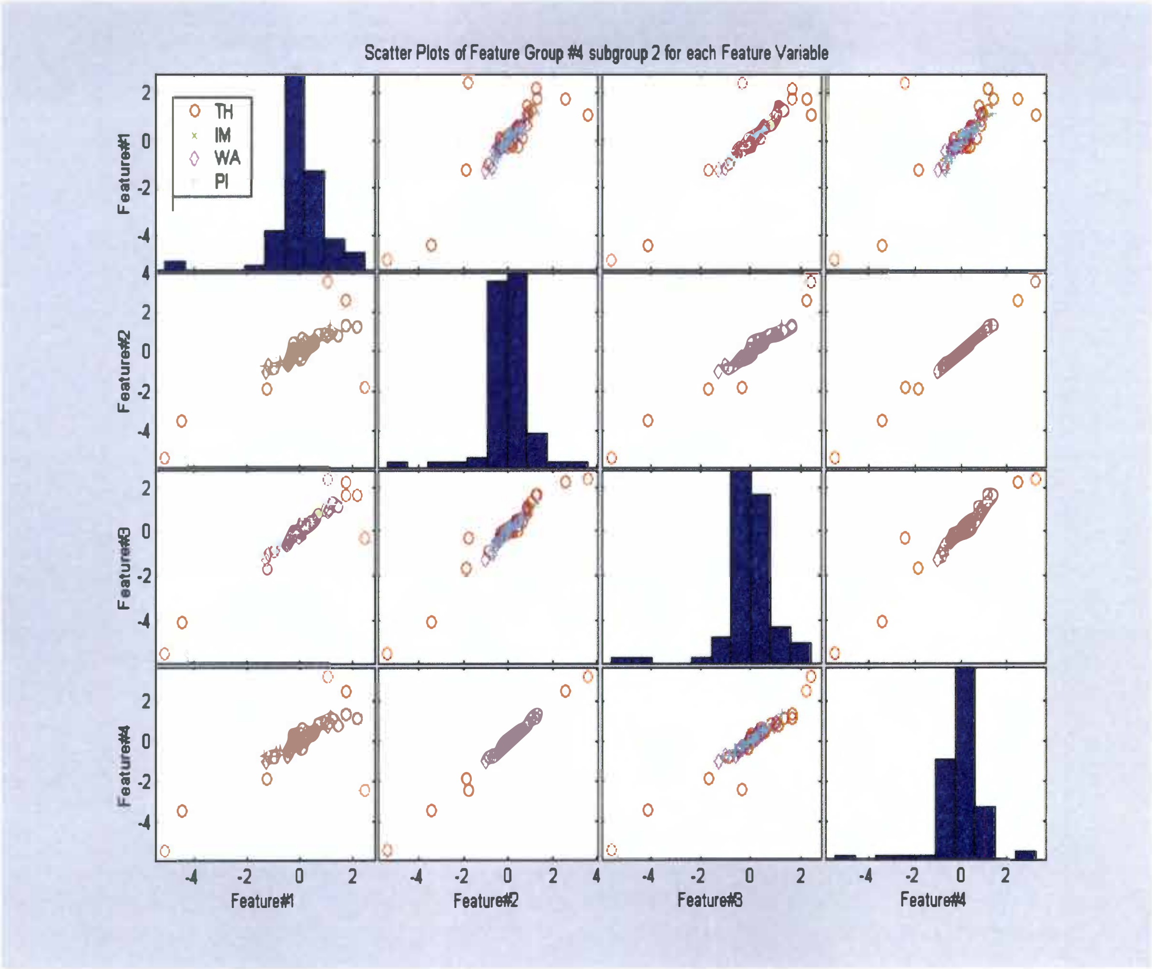

Figure A2. Scatter Plots of Geometric Moments, Subgroup 1 (G20, G21, G22, G23 and G24).

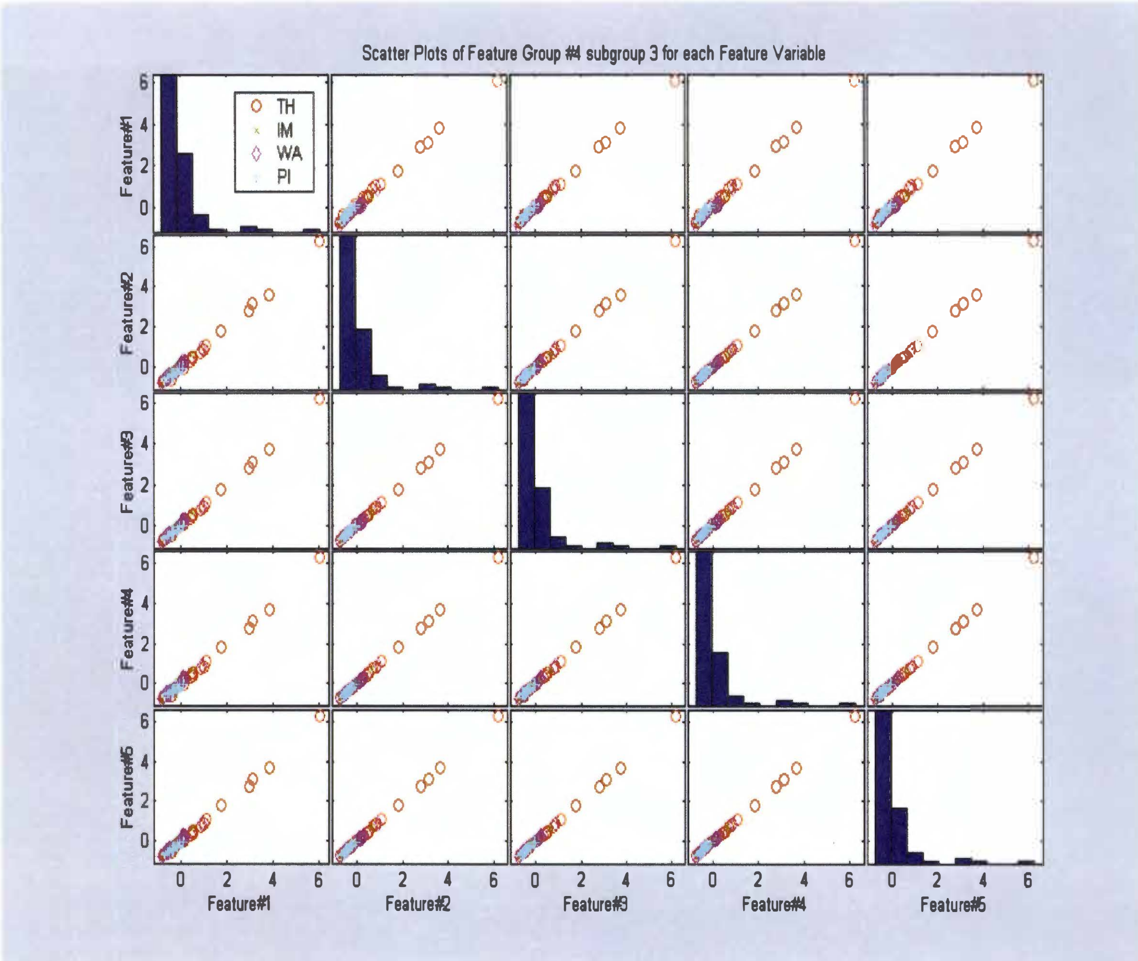

Figure A3. Scatter Plots of Geometric Moments, Subgroup 1 (G30, G31, G32, G33 and G34).

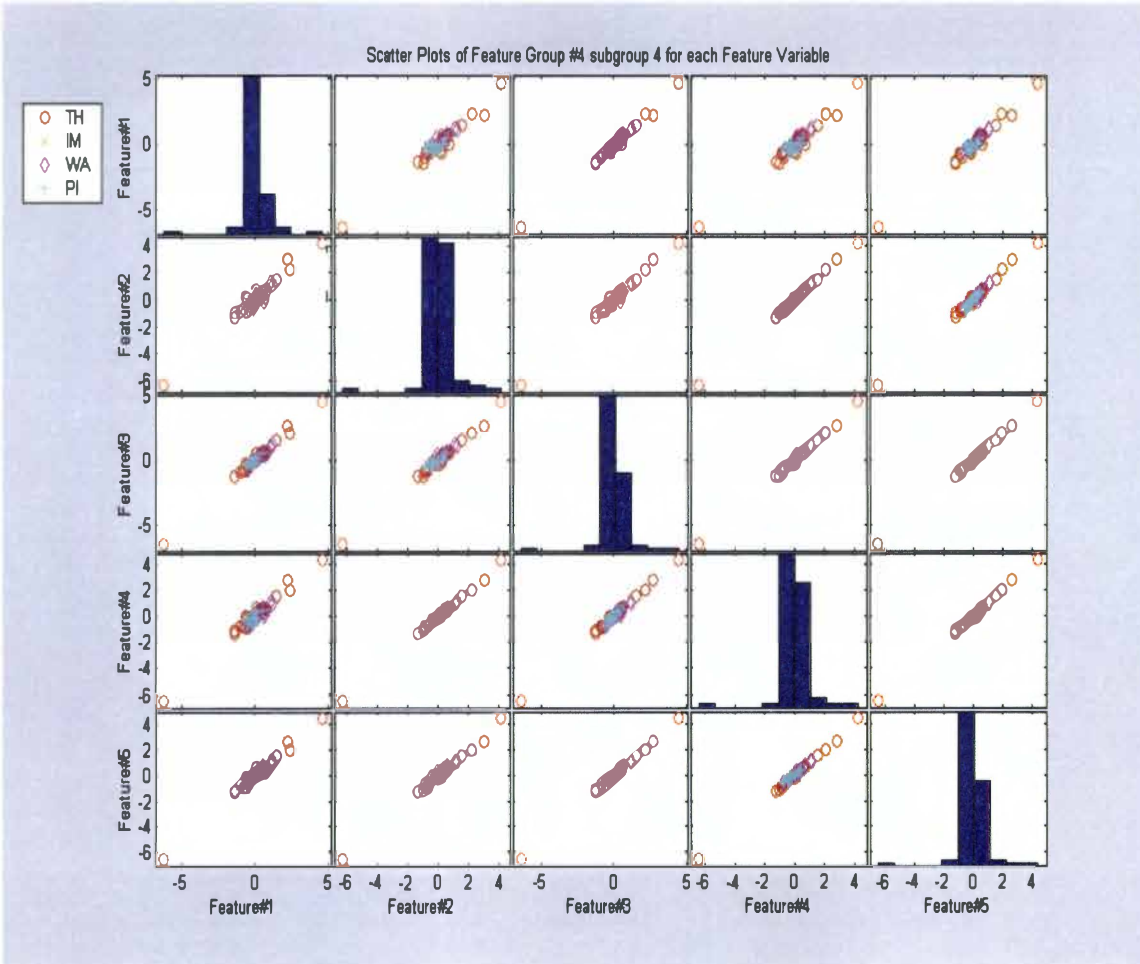

Figure A4. Scatter Plots of Geometric Moments, Subgroup 1 (G40, G41, G42, G43 and G44).

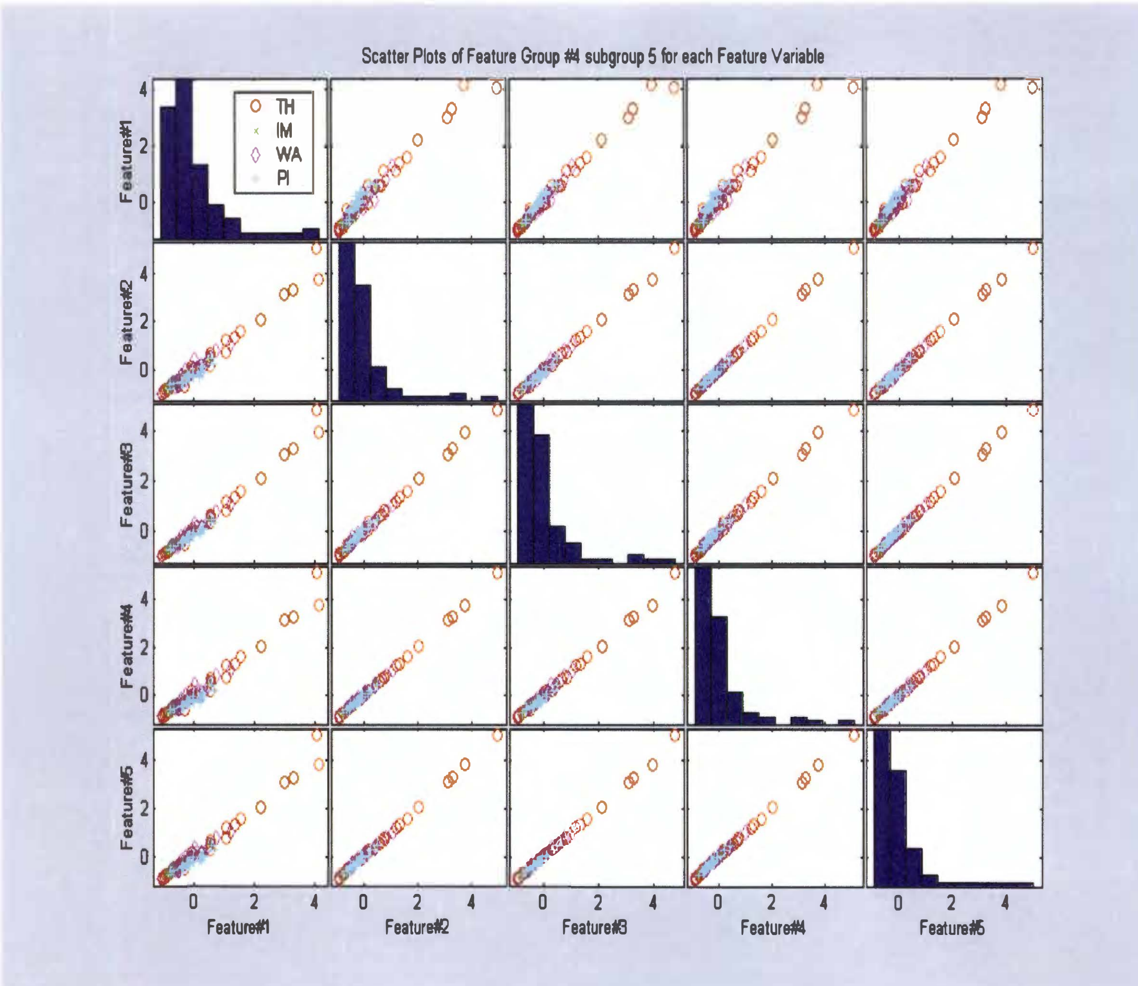

Figure A5. Scatter Plots of Geometric Moments, Subgroup 1 (G50, G51, G52, G53 and G54).

Polynomial Coefficients

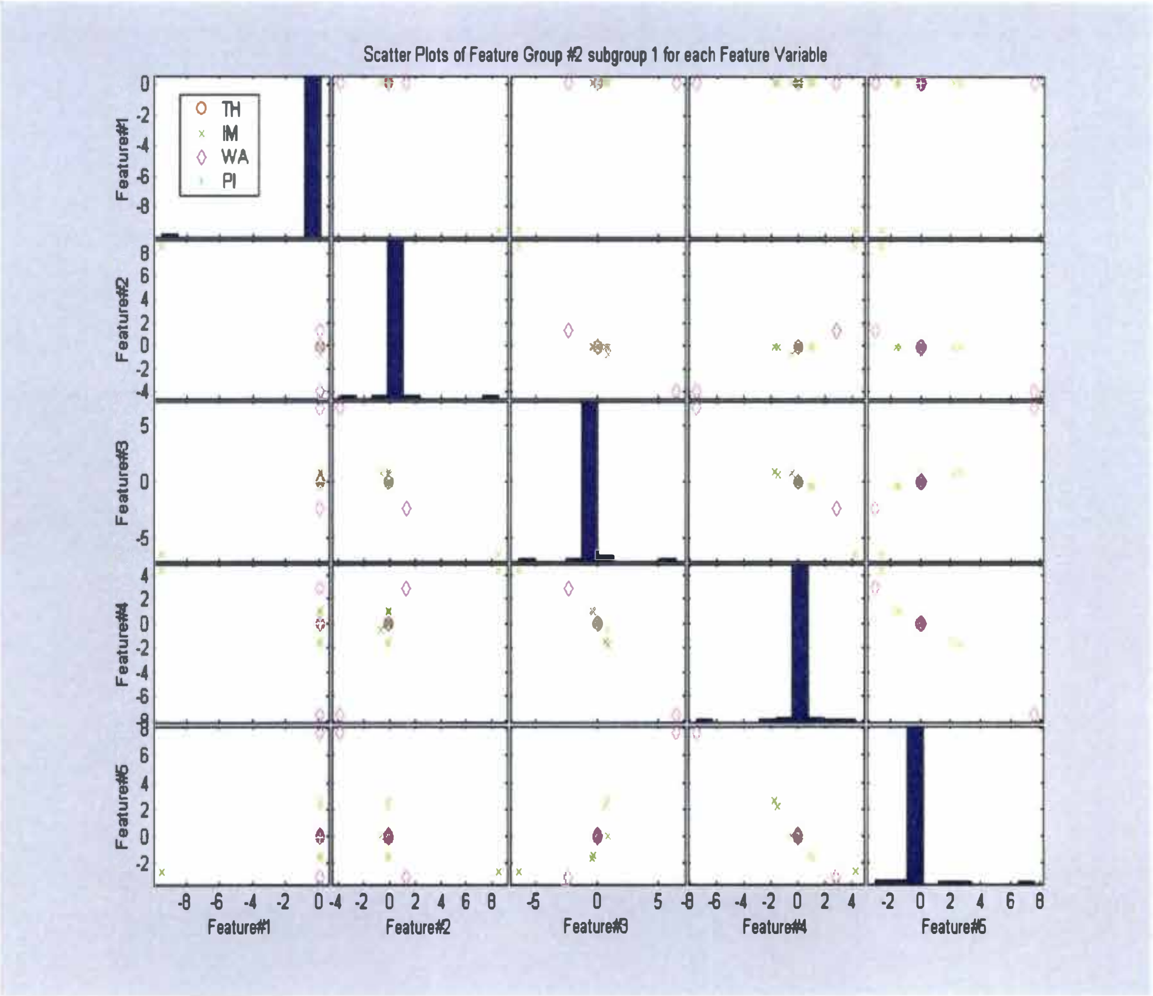

Figure A6. Scatter Plot of Polynomial Coefficients p_0, p_1, p_2, p_3 and p_4.

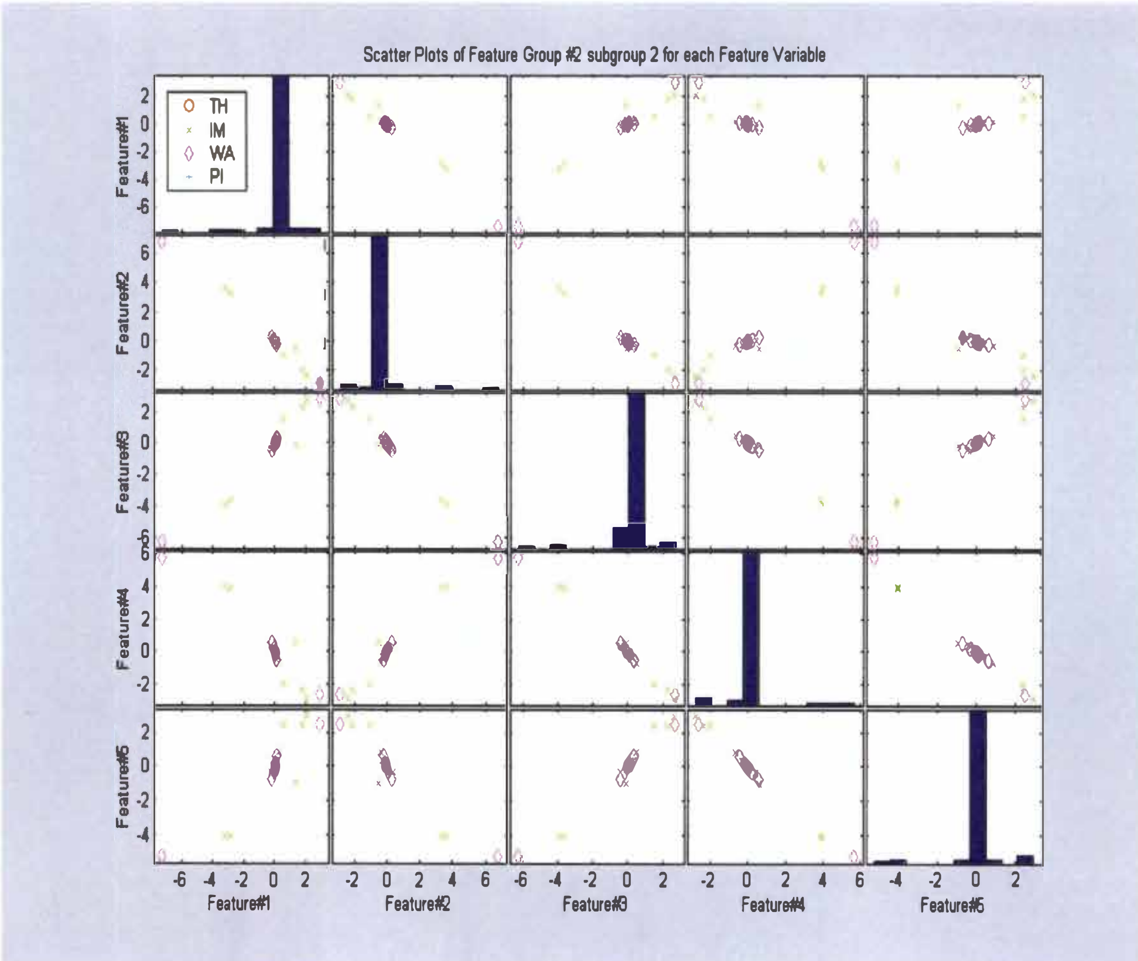

Figure A7. Scatter Plot of Polynomial Coefficients p_5, p_6, p_7, p_8 and p_9.

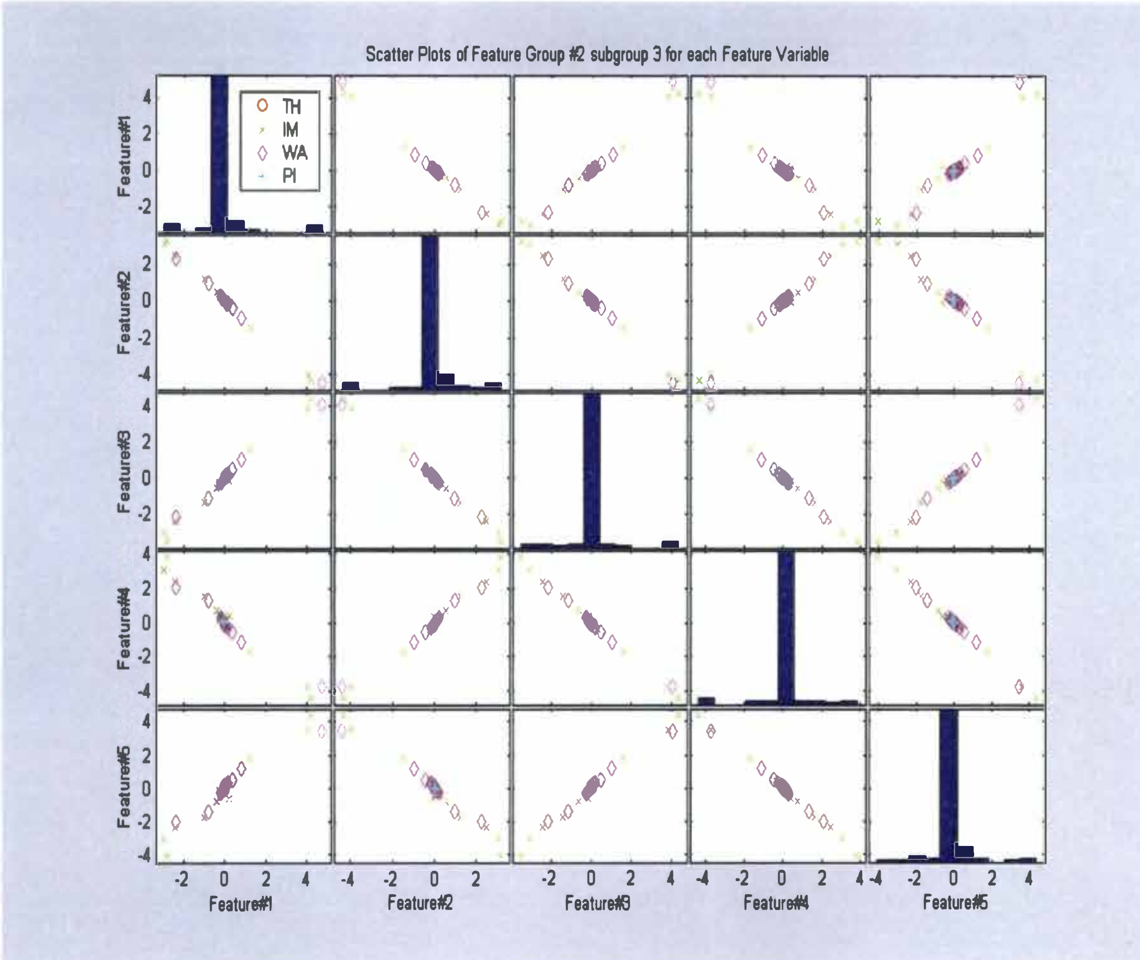

Figure A8. Scatter Plot of Polynomial Coefficients p_{10}, p_{11}, p_{12}, p_{13} and p_{14}.

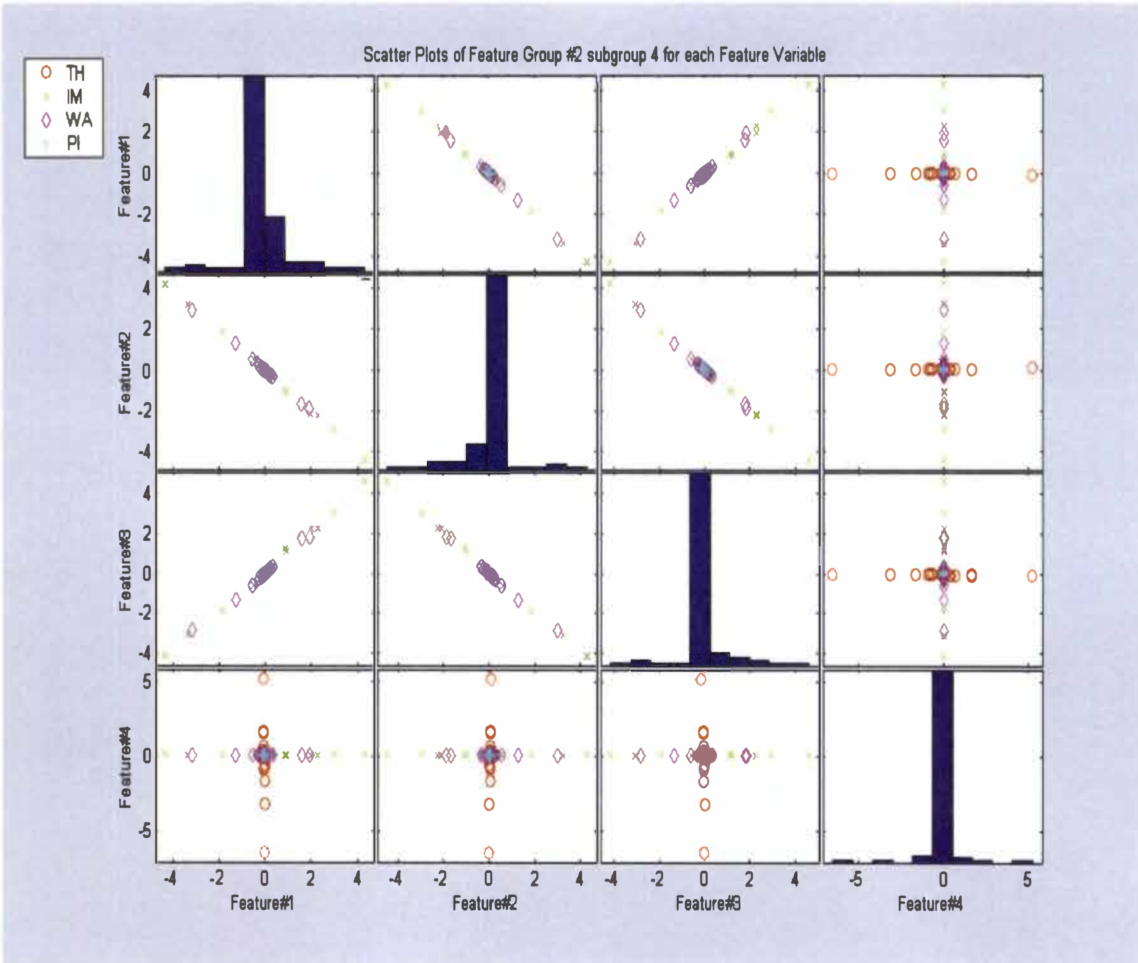

Figure A9. Scatter Plot of Polynomial Coefficients p_{15}, p_{16}, p_{17}, p_{18} and p_{19}.

Appendix B. Non-typical CWT Magnitude Examples for each Flaw type

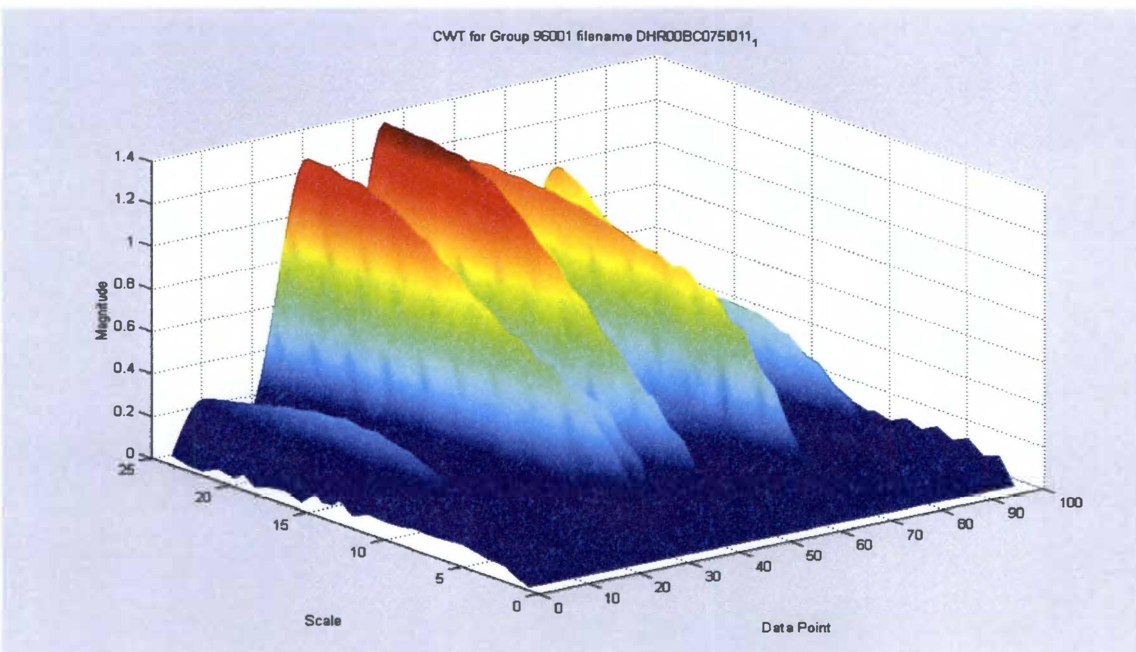

Figure B1. Non-Typical Flaw CWT for Data Group 1 (Thinning).

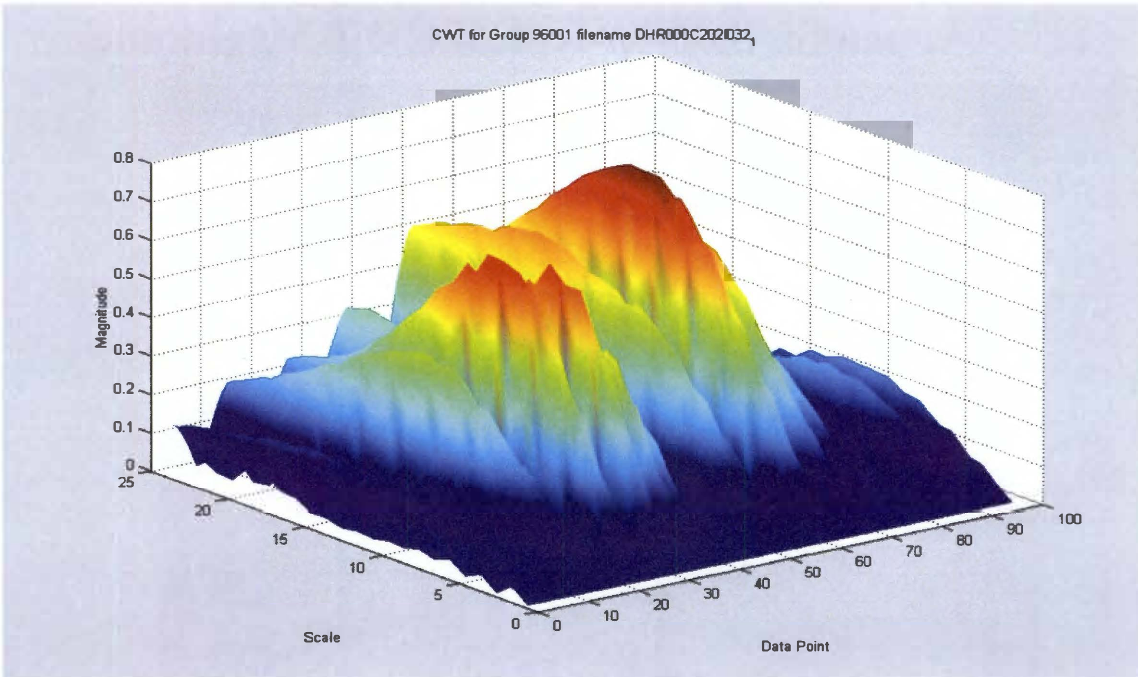

Figure B2. Non-Typical (multiple flaw ?) Flaw CWT for Data Group 1 (Thinning).

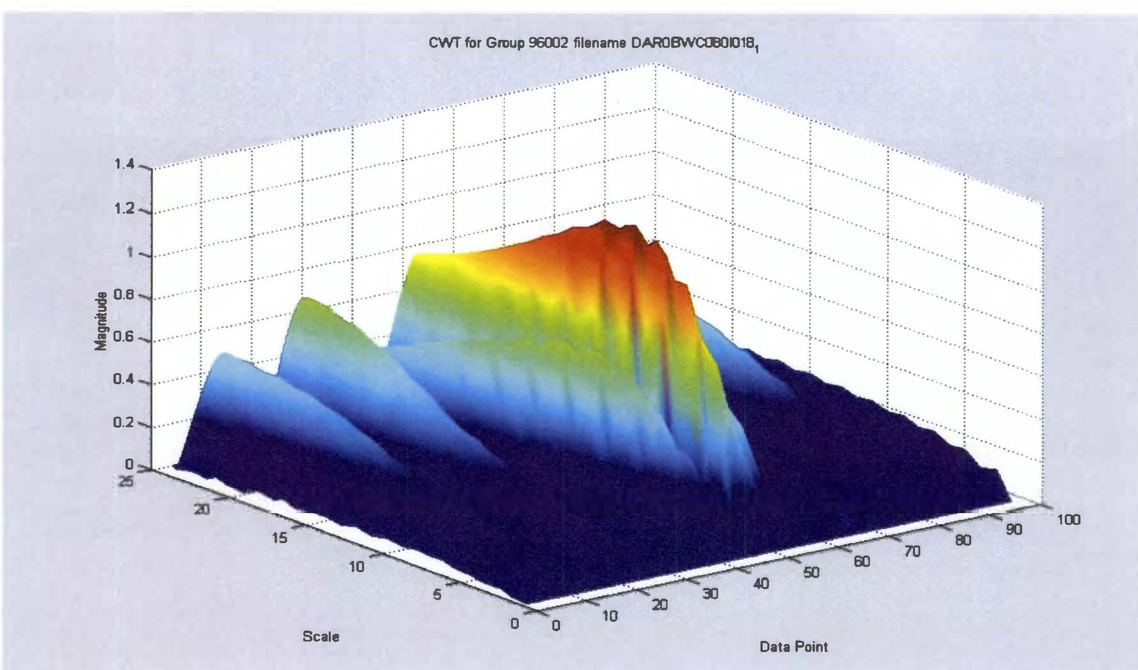

Figure B3. Non-typical Flaw CWT for Data Group 2 (Impingement).

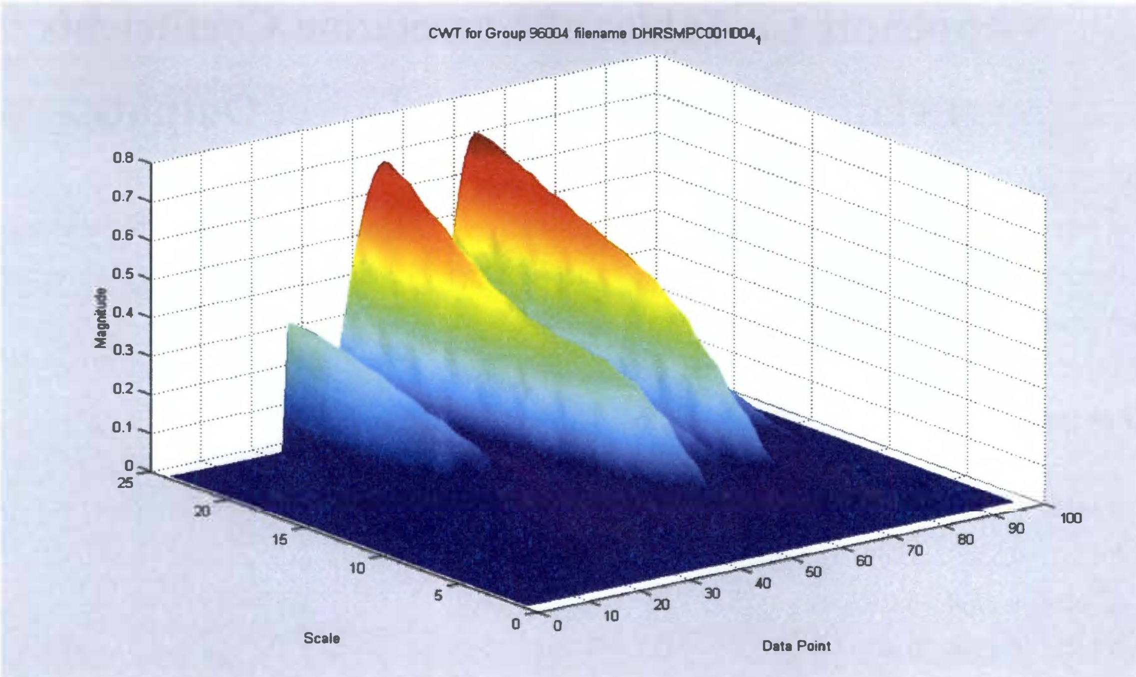

Figure B4. Non-typical Flaw CWT for Data Group 3 (Wear).

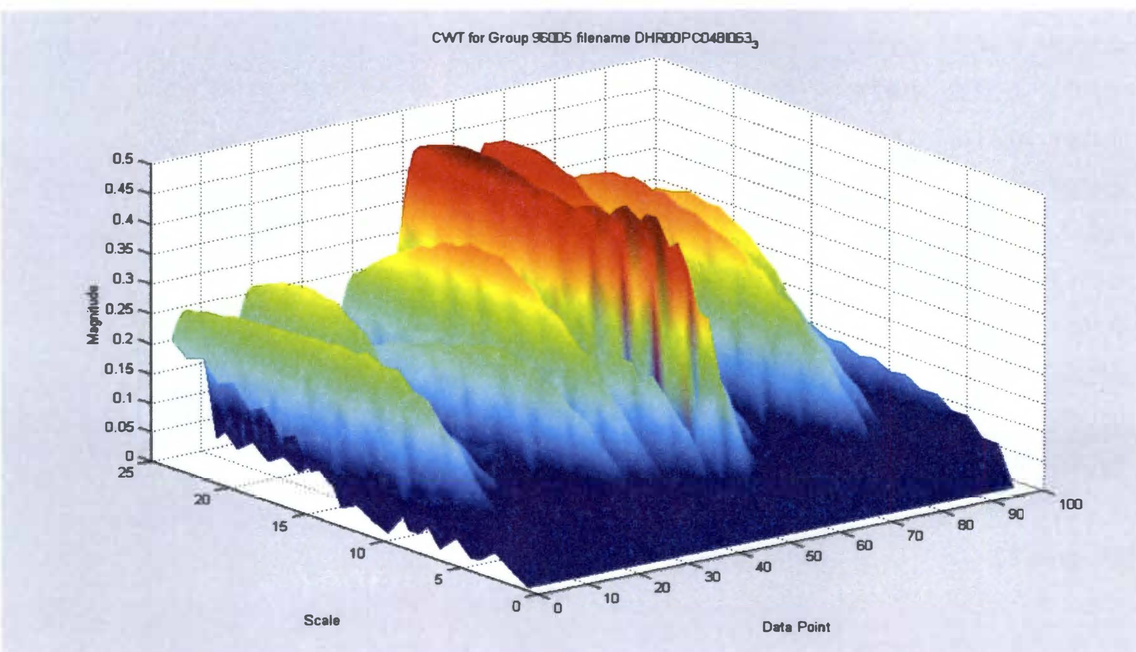

Figure B5. Non-typical Flaw CWT for Data Group 4 (Pitting).

Appendix C. Tables of Correlation Coefficients Relating Input Values and Target Outputs

The following correlation coefficients are structure with the rows 1 – 15 and columns 1 -3 as the correlation between the 15 PCs and 3 characteristic variable values. The final 3 rows (also columns 1 – 3) are the correlation coefficients between the characteristic variables.

Flaw-type #1

0.6943	0.6969	0.5107
0.5562	0.6345	0.4697
-0.3019	0.4968	-0.1756
0.1333	-0.4276	0.0497
0.1646	-0.2577	0.0884
0.3947	-0.2403	0.0359
0.5514	-0.0268	0.0195
0.0111	-0.0402	-0.2193
-0.1040	0.0121	-0.1329
-0.5024	-0.2438	-0.0773
0.4101	-0.0538	0.6605
-0.4024	-0.1350	-0.2857
-0.0242	-0.2791	0.1534
0.0155	0.0667	0.1545
-0.1053	0.0859	0.0460
1.0000	0.0893	0.6292
0.0893	1.0000	0.3147
0.6292	0.3147	1.0000

Flaw-type #2

0.6280	-0.4197	0.3179
-0.1383	0.0033	-0.0646
0.2735	-0.3560	0.3764
-0.1967	0.0553	-0.2354

0.2930	-0.1980	0.5560
0.6837	-0.6437	0.3801
0.0280	0.1291	-0.0764
0.1941	0.0165	0.4605
0.2572	-0.0546	0.3990
0.3073	-0.1974	0.3082
0.3734	-0.2004	0.5289
0.2443	-0.4663	0.1273
0.1330	0.0981	-0.0184
0.3189	-0.4772	-0.0004
0.0537	-0.0558	0.1516
1.0000	-0.7277	0.4736
-0.7277	1.0000	-0.4871
0.4736	-0.4871	1.0000

Flaw-type #3

0.8824	0.0000
-0.2399	0.0000
0.0046	0.0000
0.3635	-0.0000
-0.0358	-0.0000
0.8017	0.0000
0.6923	-0.0000
-0.4110	0.0000
-0.3560	0.0000
0.5686	-0.0000
-0.3580	-0.0000
0.2879	-0.0000
0.3372	-0.0000
-0.0046	-0.0000
-0.2707	0.0000
1.0000	-0.0000
-0.0000	1.0000

Flaw-type #4

0.5014	0.2363	0.3581
0.2654	0.1624	0.0023
-0.2409	-0.2404	-0.4107
0.2347	0.2248	-0.1495
0.2087	0.2192	-0.0824
0.1987	0.2801	0.3216
-0.0147	-0.1438	-0.0212
-0.6200	-0.3062	-0.3952
-0.7920	-0.4032	-0.1901
0.3948	0.0377	0.0056
-0.0845	-0.0209	0.4124
-0.3451	-0.2700	0.1630
0.1320	0.0300	-0.0506
0.0723	-0.1515	-0.0362
0.2681	0.2324	-0.1535
1.0000	0.3397	0.2462
0.3397	1.0000	0.2155
0.2462	0.2155	1.0000

To determine which PCs should have been retained, two restrictions could be applied. The first restriction could be to keep any PC with a correlation greater than 0.5. The second could be to retain any PC with a range of correlation greater than 0.1 but with at least one greater than 0.4 for the characteristic values.

PCs with at least one correlation greater than or equal to 0.5 – 1, 2, 5, 6, 7, 8, 9 and 10.
PCs with all correlation greater than 0.1 and at lest one greater than 0.4 – 1, 2, 3, 5, 6, 7, 8, 9, 10, 11 and 12.

If these two restriction are set to select higher correlated PCs (to be retained in the PC model), PCs 1-3, 5-12 would be kept.

Appendix D. Training Output for net_char_E_2a5.

==== Neural Network Characterization Results for ====

Data origin was E

Data Group was 96000

The Data run number was 2 a

==== Neural Network Analysis for Flawtype 1 ====

Number of neurons for the hidden layer. 5

Desire SSE goal. 0.05

TRAINBR, Epoch 0/200, SSE 272.31/0.05, SSW 25.0731, Grad 1.56e+002/1.00e-010, #Par 9.80e+001/98

TRAINBR, Epoch 200/200, SSE 0.0765691/0.05, SSW 25.0674, Grad 9.06e-002/1.00e-010, #Par 6.16e+001/98

TRAINBR, Maximum epoch reached.

Target Flaw characterization vector for flawtype # 1

T =

Columns 1 through 7

9.0000	40.0000	9.0000	23.0000	60.0000	38.0000	12.0000
195.0000	75.0000	75.0000	75.0000	360.0000	45.0000	45.0000
1.4000	1.0000	1.0000	1.0000	3.0000	0.4500	0.3300

Columns 8 through 14

22.0000	46.0000	30.0000	40.0000	50.0000	57.0000	66.0000
45.0000	45.0000	90.0000	90.0000	90.0000	90.0000	90.0000
0.3000	0.3550	3.0000	3.0000	3.0000	3.0000	3.0000

Columns 15 through 21

80.0000	90.0000	100.0000	30.0000	38.0000	44.0000	60.0000
90.0000	90.0000	90.0000	90.0000	90.0000	90.0000	90.0000
3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000

Columns 22 through 24

66.0000	88.0000	80.0000
90.0000	90.0000	90.0000
3.0000	3.0000	3.0000

ANN Flaw characterization vector for flawtype # = 1

Y =

Columns 1 through 7

9.7228	39.8231	11.1158	20.7447	60.1289	41.2281	11.5610
191.7737	78.6626	79.7349	70.3920	356.4926	45.8375	45.8267
1.4319	1.0229	0.9540	1.0761	2.9891	0.4635	0.3385

Columns 8 through 14

25.4834	39.9912	29.4587	38.5275	51.0159	56.9392	66.8311
40.8969	47.9546	101.5796	91.8237	89.0201	78.5705	94.5207
0.2865	0.3759	3.0147	2.9969	2.9880	2.9715	2.9992

Columns 15 through 21

79.2603	89.0128	101.1728	30.9231	37.0268	44.0948	60.7561
81.3758	91.0044	92.3122	92.0768	85.1997	86.7275	86.6746
3.0378	2.9909	3.0107	3.0028	2.9858	2.9222	2.9897

Columns 22 through 24

67.8055	86.0480	79.4594
91.6117	98.0453	92.1311
3.0005	2.9841	3.0146

The MSE between Tn and Yn for flawtype # 1 = 0.0011

Correlation Coeff between T and Y for flawtype # 1 variable 1 = 1.00
Correlation Coeff between T and Y for flawtype # 1 variable 2 = 1.00
Correlation Coeff between T and Y for flawtype # 1 variable 3 = 1.00

Does user want to save the generated NN and info ("y"es or "n"o)? y
NN char run number (usually 5a or 5b ... with 5 being general run number). 5

Figures D1 through D3 are generated.

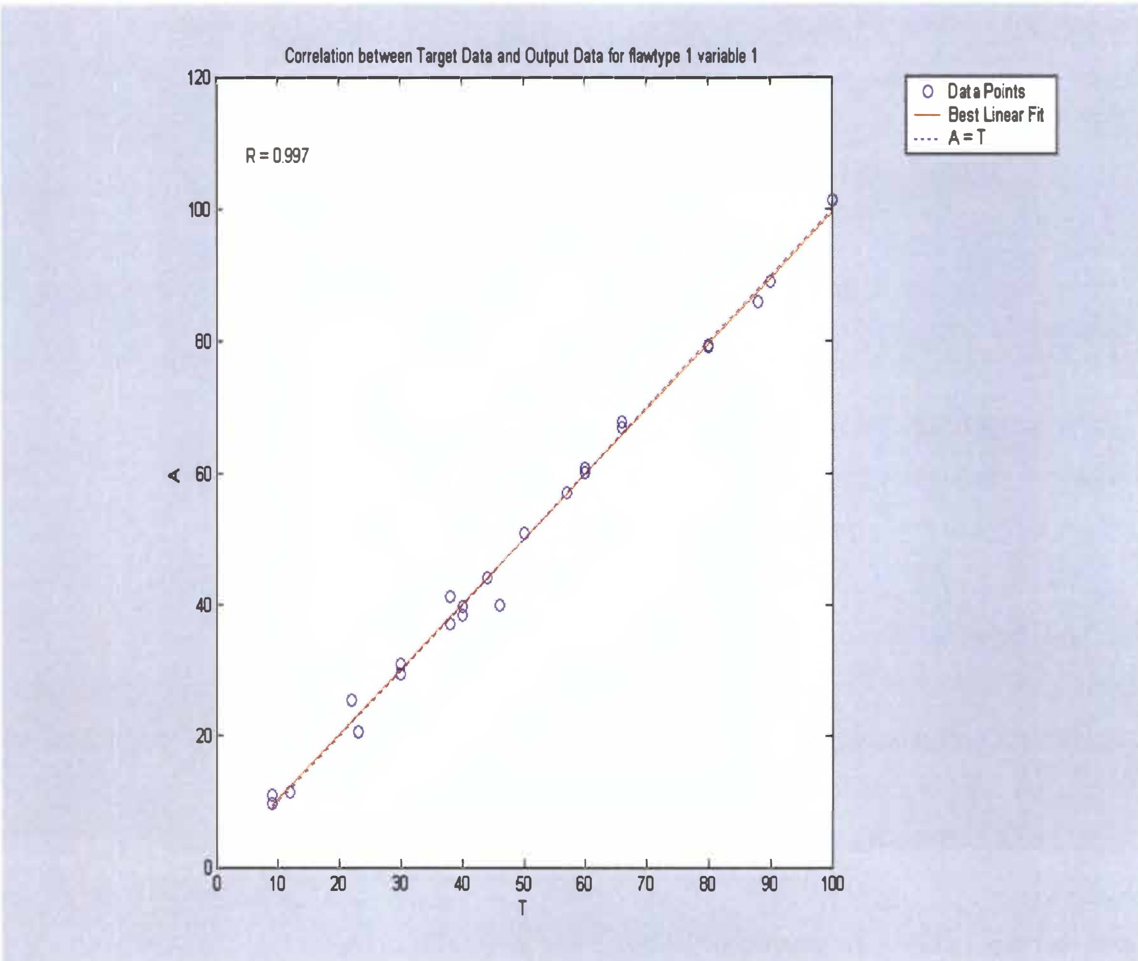

Figure D1. Correlation between Target Data and Output Data For Flaw-type 1 (Thinning), Characteristic 1.

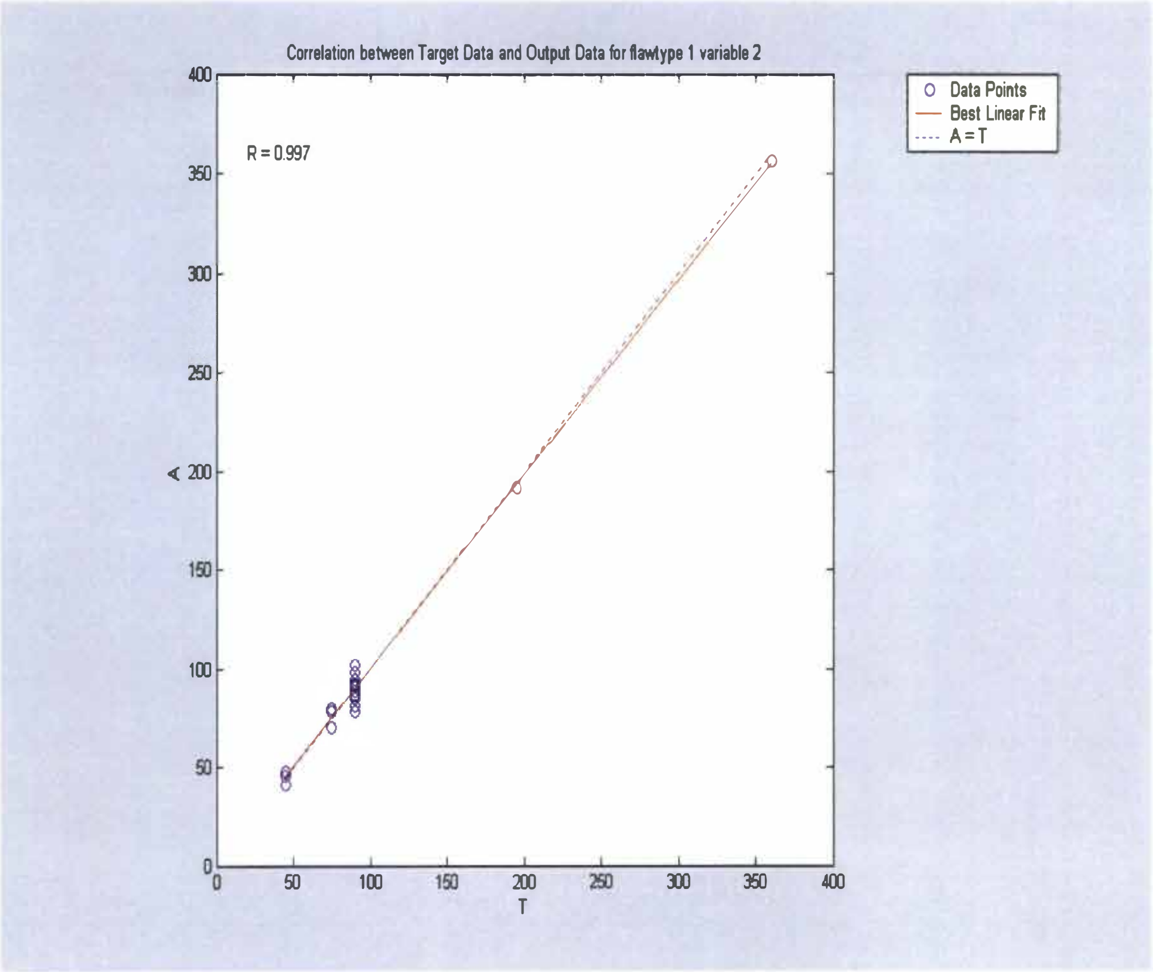

Figure D2. Correlation between Target Data and Output Data For Flaw-type 1 (Thinning), Characteristic 2.

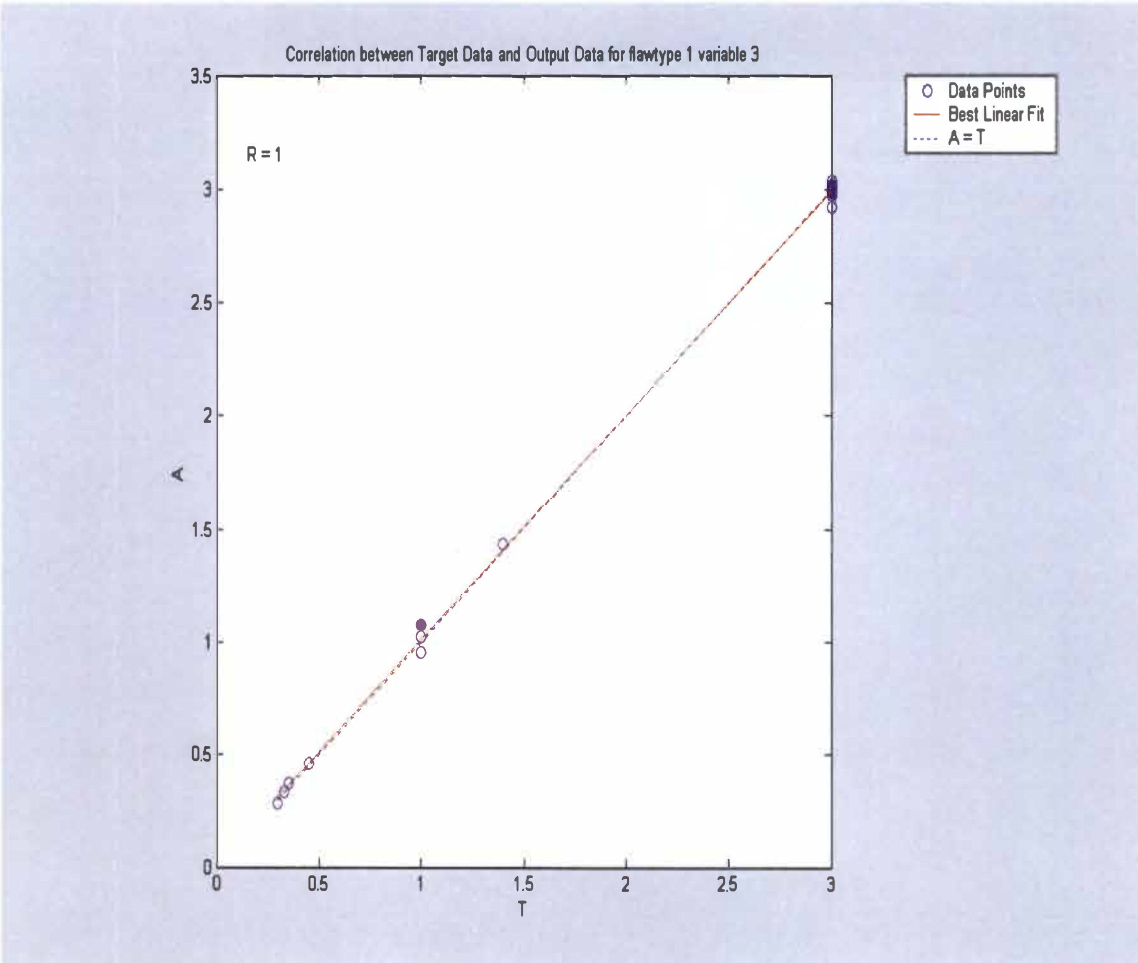

Figure D3. Correlation between Target Data and Output Data For Flaw-type 1 (Thinning), Characteristic 3.

═══ Neural Network Analysis for Flawtype 2 ═══

Number of neurons for the hidden layer. 5
Desire SSE goal. 0.05

TRAINBR, Epoch 0/200, SSE 72.3529/0.05, SSW 25.311, Grad 4.64e+001/1.00e-010, #Par 9.80e+001/98
TRAINBR, Epoch 41/200, SSE 0.0448119/0.05, SSW 15.3099, Grad 1.75e-001/1.00e-010, #Par 5.19e+001/98
TRAINBR, Performance goal met.

Target Flaw characterization vector for flawtype # 2

T =

Columns 1 through 7

58.0000	63.0000	65.0000	68.0000	73.0000	75.0000	76.0000
0.0850	0.0860	0.0950	0.0980	0.0930	0.0950	0.0900
0.3450	0.2000	0.2290	0.2160	0.2240	0.2230	0.2280

Columns 8 through 14

78.0000	79.0000	82.0000	84.0000	87.0000	87.0000	98.0000
0.0900	0.0820	0.0820	0.0730	0.0740	0.0700	0.0680
0.2290	0.3540	0.2270	0.2310	0.2390	0.2330	0.2660

Columns 15 through 20

95.0000	92.0000	60.0000	76.0000	60.0000	37.0000
0.0780	0.0780	0.0880	0.0910	0.1962	0.2110
0.3100	0.3320	0.2700	0.2150	0.2700	0.0754

NN Flaw characterization vector for flawtype # = 2

Y =

Columns 1 through 7

58.5564	61.9967	66.8606	68.0577	72.5460	74.7176	76.6117
0.0848	0.0909	0.0960	0.0955	0.0927	0.0981	0.0918
0.3405	0.2065	0.2263	0.2167	0.2236	0.2250	0.2292

Columns 8 through 14

77.6411	78.7649	80.8234	83.4476	86.6497	87.7425	97.7881
0.0905	0.0820	0.0760	0.0759	0.0745	0.0707	0.0690
0.2305	0.3513	0.2228	0.2352	0.2323	0.2402	0.2638

Columns 15 through 20

94.3642	91.9928	61.3398	75.8844	59.3983	37.9375
0.0782	0.0781	0.0887	0.0883	0.1947	0.2094
0.3098	0.3316	0.2690	0.2160	0.2682	0.0781

The MSE between Tn and Yn for flawtype # 2 = 0.0007

Correlation Coeff between T and Y for flawtype # 2 variable 1 = 1.00
Correlation Coeff between T and Y for flawtype # 2 variable 2 = 1.00
Correlation Coeff between T and Y for flawtype # 2 variable 3 = 1.00

Does user want to save the generated NN and info ("y"es or "n"o)? y
NN char run number (usually 5a or 5b ... with 5 being general run number). 5

Figures D4 through D6 are generated are generated.

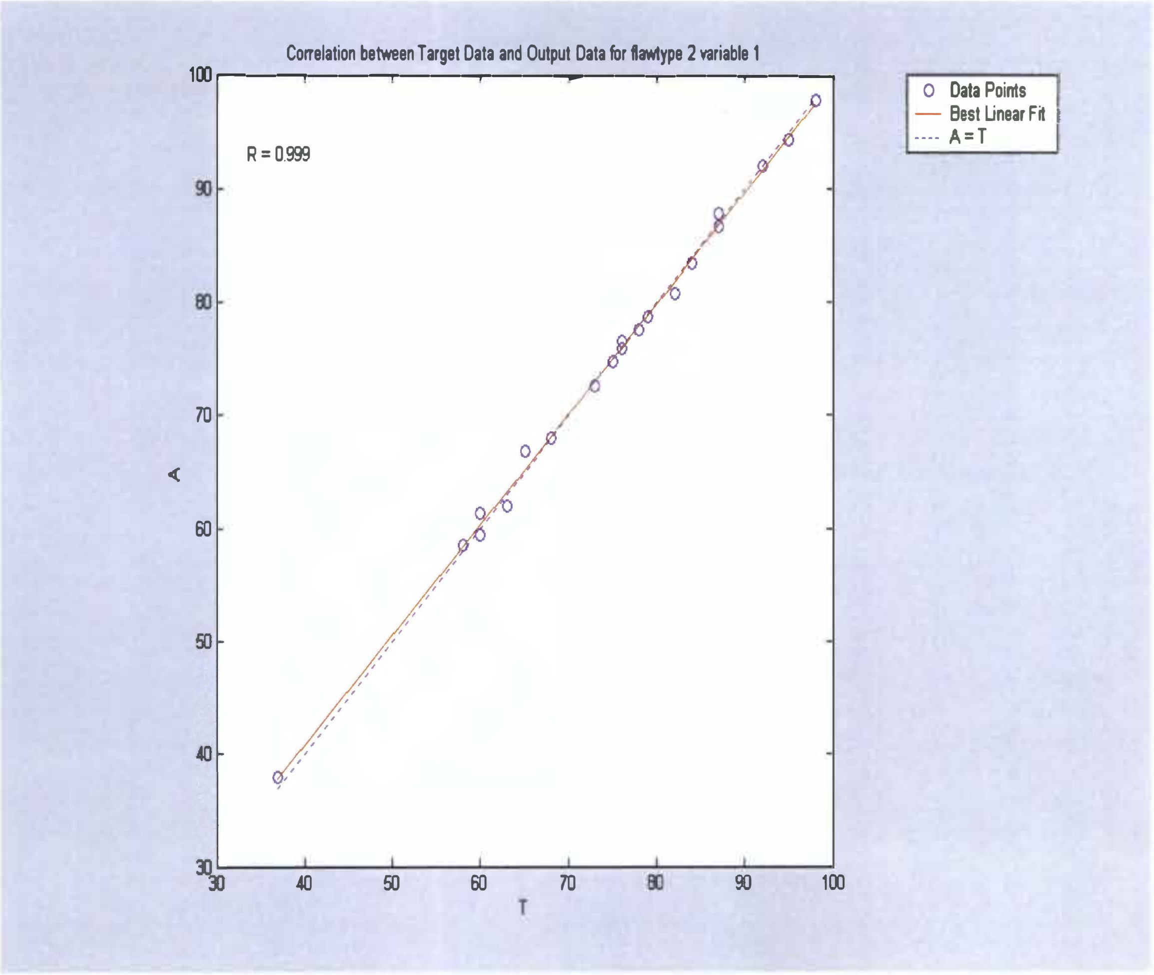

Figure D4. Correlation between Target Data and Output Data For Flaw-type 2 (Impingement), Characteristic 1.

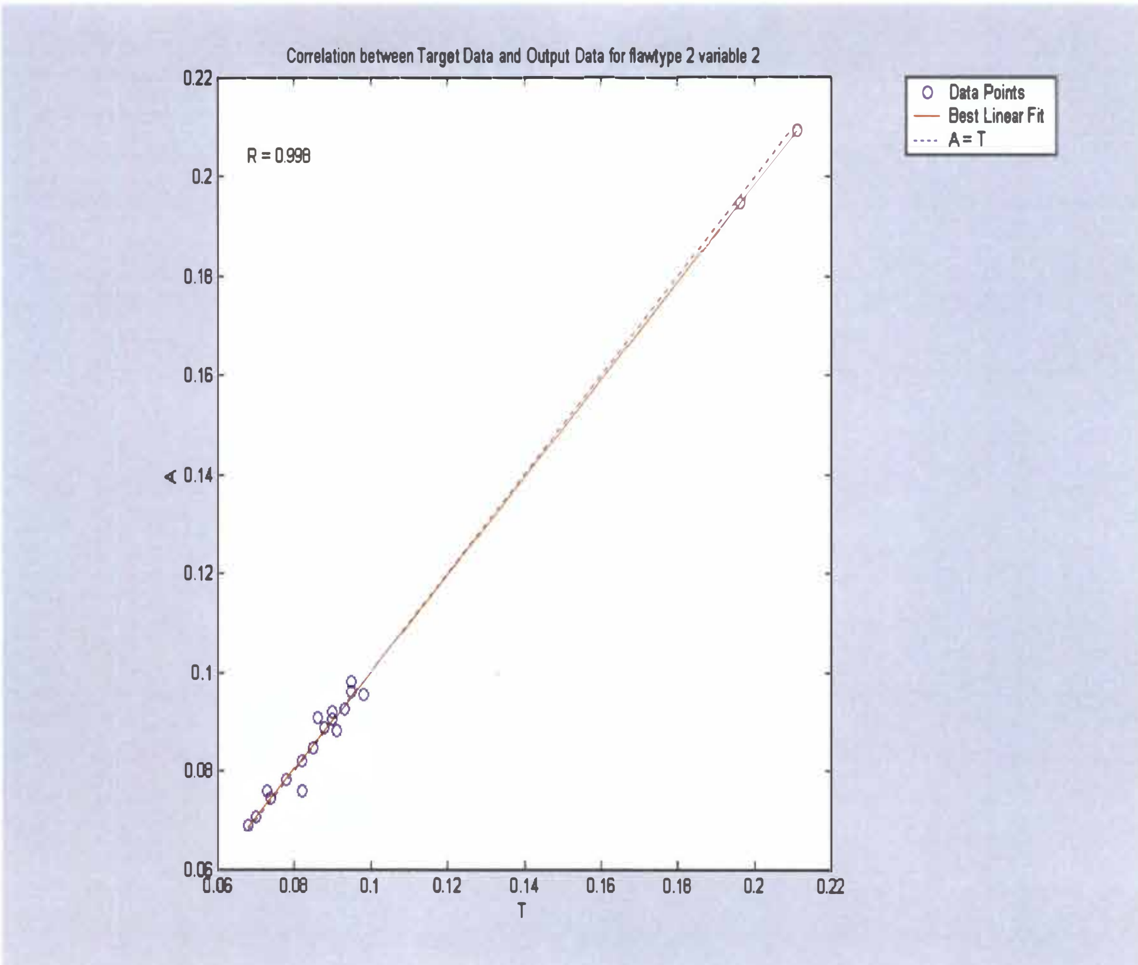

Figure D5. Correlation between Target Data and Output Data For Flaw-type 2 (Impingement), Characteristic 2.

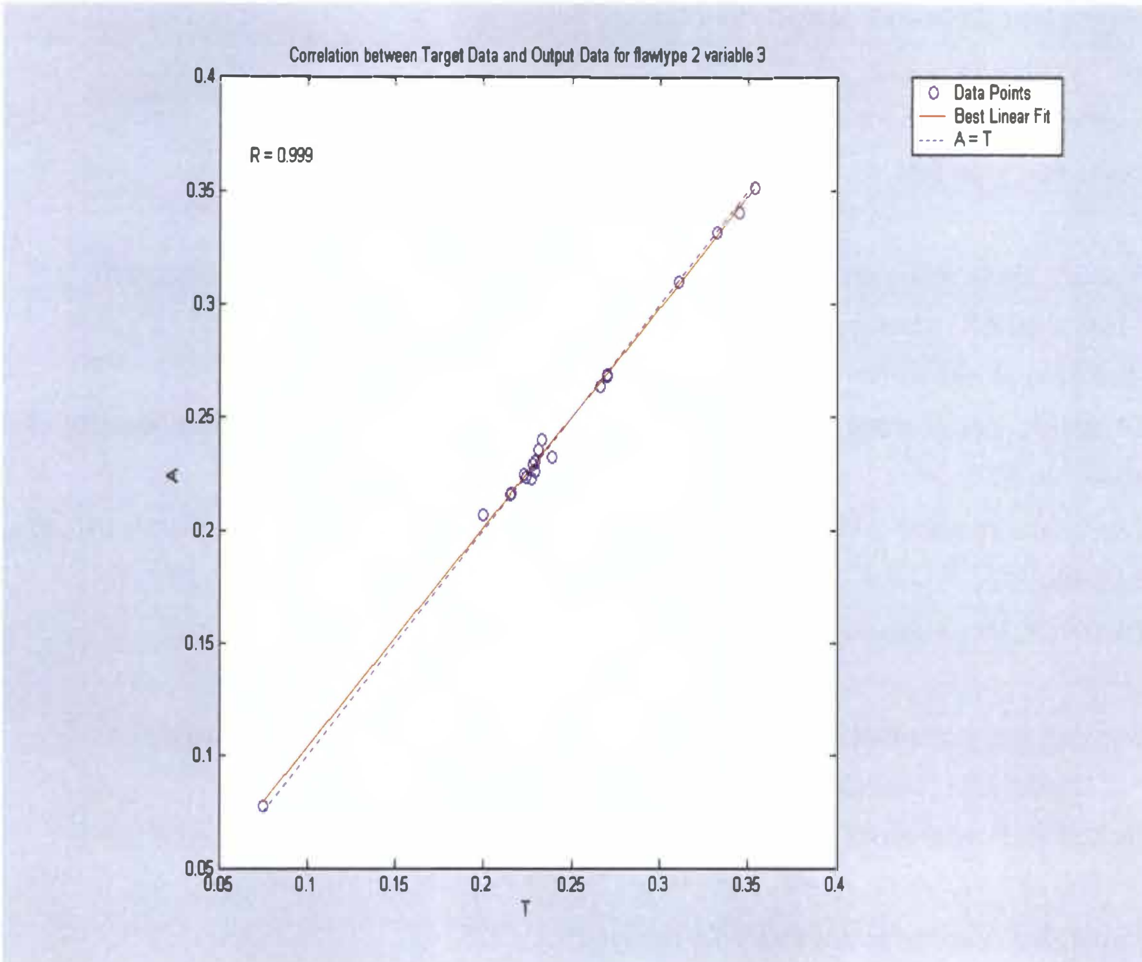

Figure D6. Correlation between Target Data and Output Data For Flaw-type 2 (Impingement), Characteristic 3.

═══ Neural Network Analysis for Flawtype 3 ═══

Number of neurons for the hidden layer. 5
Desire SSE goal. 0.05

```
Warning: Some maximums and minimums are equal. Those targets won't be transformed.
> In C:\matlabR12\toolbox\nnet\nnet\premnmx.m at line 77
  In C:\Patrick\eddym\NN_char.m at line 70
TRAINBR, Epoch 0/200, SSE 117.998/0.05, SSW 22.6067, Grad 1.02e+002/1.00e-010, #Par 9.20e+001/92
TRAINBR, Epoch 16/200, SSE 0.0446862/0.05, SSW 4.23562, Grad 1.89e-001/1.00e-010, #Par 2.33e+001/92
TRAINBR, Performance goal met.

Warning: Some maximums and minimums are equal. Those inputs won't be transformed.
> In C:\matlabR12\toolbox\nnet\nnet\postmnmx.m at line 59
  In C:\Patrick\eddym\NN_char.m at line 88
```

Target Flaw characterization vector for flawtype # 3

```
T =

  Columns 1 through 7

   17.0000   61.0000   10.0000   89.0000   37.0000   46.0000   79.0000
    0.2750    0.2750    0.2750    0.2750    0.2750    0.2750    0.2750

  Columns 8 through 14

   37.0000   50.0000   84.0000   55.0000   61.0000   67.0000   26.0000
    0.2750    0.2750    0.2750    0.2750    0.2750    0.2750    0.2750

  Columns 15 through 21
```

70.0000	90.0000	74.0000	46.0000	80.0000	50.0000	32.0000
0.2750	0.2750	0.2750	0.2750	0.2750	0.2750	0.2750

Columns 22 through 23

90.0000	70.0000
0.2750	0.2750

NN Flaw characterization vector for flawtype # = 3

Y =

Columns 1 through 7

18.4217	61.3516	11.9381	88.1088	34.0617	47.9653	79.3533
0.2741	0.2748	0.2736	0.2747	0.2746	0.2750	0.2748

Columns 8 through 14

35.9941	49.2409	84.6975	58.6848	60.1340	67.0242	27.7238
0.2746	0.2751	0.2750	0.2752	0.2750	0.2749	0.2746

Columns 15 through 21

71.9045	90.0559	72.9519	46.2938	77.0158	50.7998	31.1801
0.2750	0.2750	0.2752	0.2754	0.2751	0.2752	0.2749

Columns 22 through 23

89.1956	65.8629
0.2730	0.2750

The MSE between Tn and Yn for flawtype # 3 = 0.0010

Correlation Coeff between T and Y for flawtype # 3 variable 1 = 1.00

Warning: Rank deficient, rank = 1 tol =   2.4492e-014.

> In C:\matlabR12\toolbox\nnet\nnet\postreg.m at line 57

In C:\Patrick\eddym\NN_char.m at line 100

Warning: Divide by zero.

> In C:\matlabR12\toolbox\nnet\nnet\postreg.m at line 77

In C:\Patrick\eddym\NN_char.m at line 100

Correlation Coeff between T and Y for flawtype # 3 variable 2 = -Inf

Does user want to save the generated NN and info ("y"es or "n"o)? y

NN char run number (usually 5a or 5b ... with 5 being general run number). 5

Figures D7 and D8 are generated.

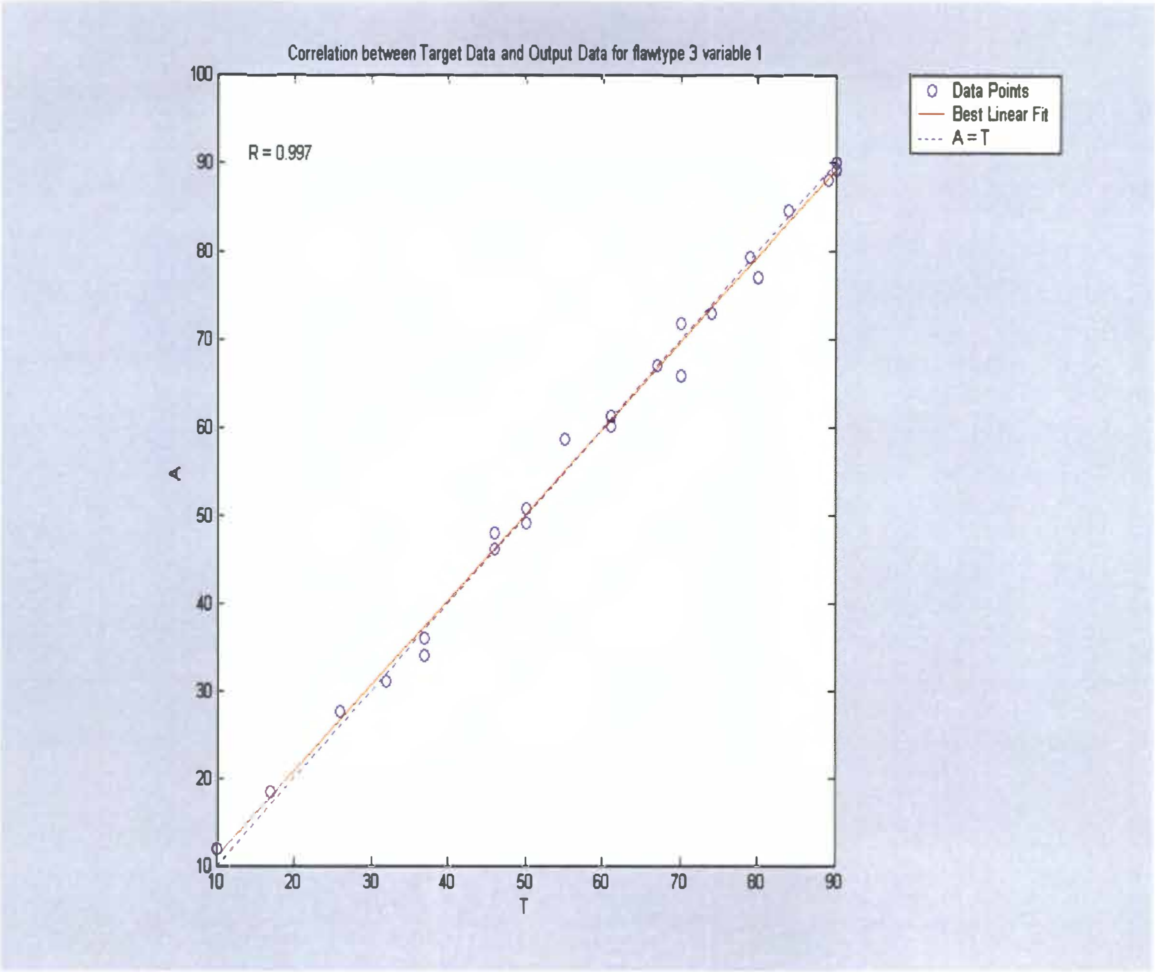

Figure D7. Correlation between Target Data and Output Data For Flaw-type 3 (Wear), Characteristic 1.

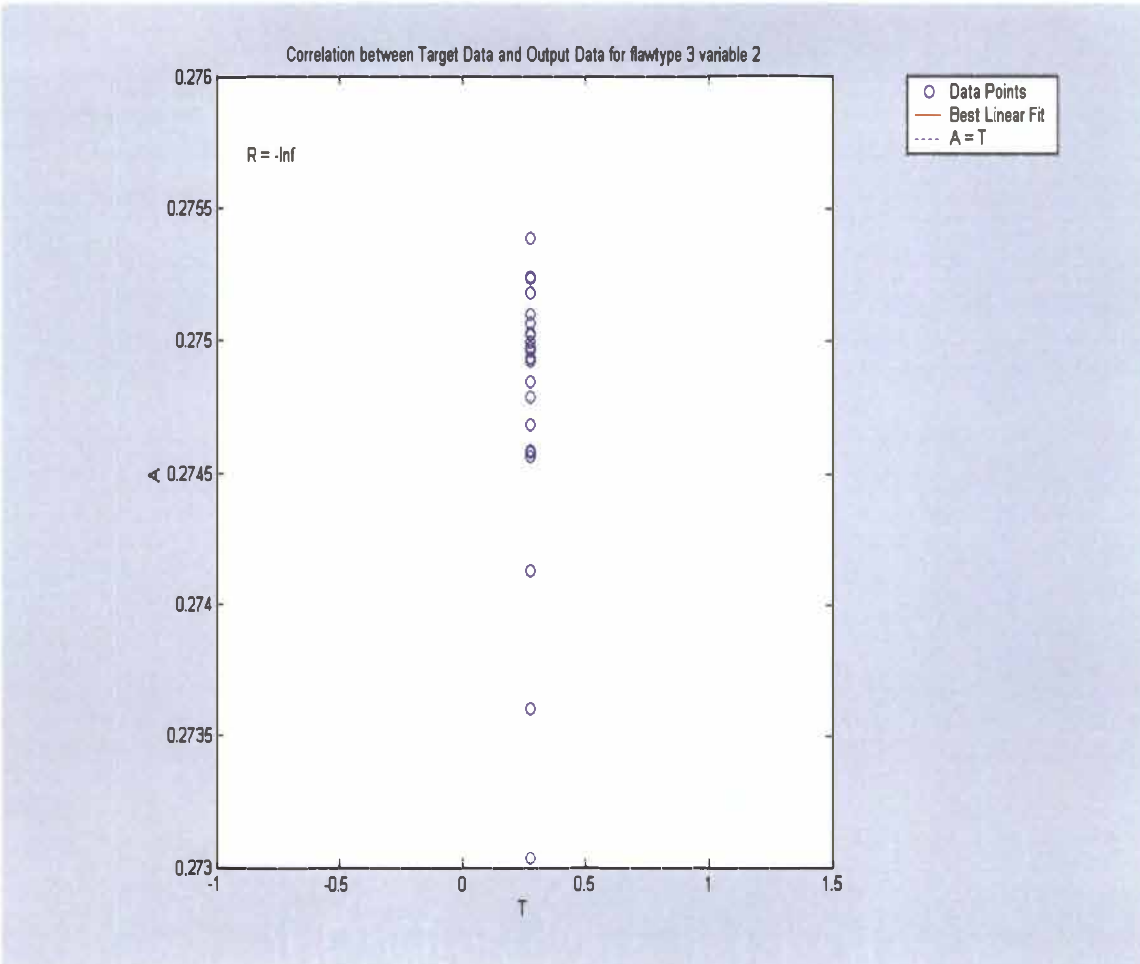

Figure D8. Correlation between Target Data and Output Data For Flaw-type 3 (Wear), Characteristic 2.

===== Neural Network Analysis for Flawtype 4 =====

Number of neurons for the hidden layer. 5

Desire SSE goal. 0.05

TRAINBR, Epoch 0/200, SSE 106.863/0.05, SSW 24.2804, Grad 6.40e+001/1.00e-010, #Par 9.80e+001/98

TRAINBR, Epoch 200/200, SSE 10.7702/0.05, SSW 2.12682, Grad 2.42e+000/1.00e-010, #Par 1.55e+001/98

TRAINBR, Maximum epoch reached.

Target Flaw characterization vector for flawtype # 4

T =

Columns 1 through 7

29.0000	30.0000	47.0000	46.0000	29.0000	42.0000	37.0000
0.0550	0.0850	0.0650	0.0650	0.0550	0.0800	0.0750
0.0500	0.0300	0.0650	0.0350	0.0500	0.0580	0.0720

Columns 8 through 14

47.0000	44.0000	47.0000	44.0000	62.0000	67.0000	62.0000
0.0800	0.1000	0.1150	0.1150	0.1000	0.1400	0.1000
0.0450	0.1650	0.0900	0.0450	0.1100	0.0900	0.1100

Columns 15 through 21

67.0000	44.0000	77.0000	53.0000	44.0000	77.0000	53.0000
0.1400	0.1450	0.0850	0.0850	0.1450	0.0850	0.0850
0.0900	0.0400	0.0500	0.0550	0.0400	0.0500	0.0550

NN Flaw characterization vector for flawtype # = 4

Y =

Columns 1 through 7

41.9825	37.2459	45.1249	44.4929	33.8579	52.5222	45.4426
0.0866	0.0795	0.0923	0.0861	0.0716	0.0976	0.0890
0.0798	0.0509	0.0726	0.0495	0.0671	0.0550	0.0496

Columns 8 through 14

38.9596	41.4228	46.0120	40.9487	58.9615	64.3786	61.7799
0.0805	0.0840	0.0932	0.0844	0.1056	0.1169	0.1118
0.0461	0.0768	0.0767	0.0538	0.0902	0.0752	0.0892

Columns 15 through 21

66.3965	54.2986	59.6348	49.6870	54.0965	62.3058	50.6391
0.1178	0.0999	0.1064	0.0920	0.1011	0.1107	0.0948
0.0863	0.0661	0.0654	0.0644	0.0700	0.0714	0.0699

The MSE between Tn and Yn for flawtype # 4 = 0.1710

Correlation Coeff between T and Y for flawtype # 4 variable 1 = 0.85
Correlation Coeff between T and Y for flawtype # 4 variable 2 = 0.58
Correlation Coeff between T and Y for flawtype # 4 variable 3 = 0.62

Does user want to save the generated NN and info ("y"es or "n"o)? y
NN char run number (usually 5a or 5b ... with 5 being general run number). 5

Figures D9 through D11 are generated.

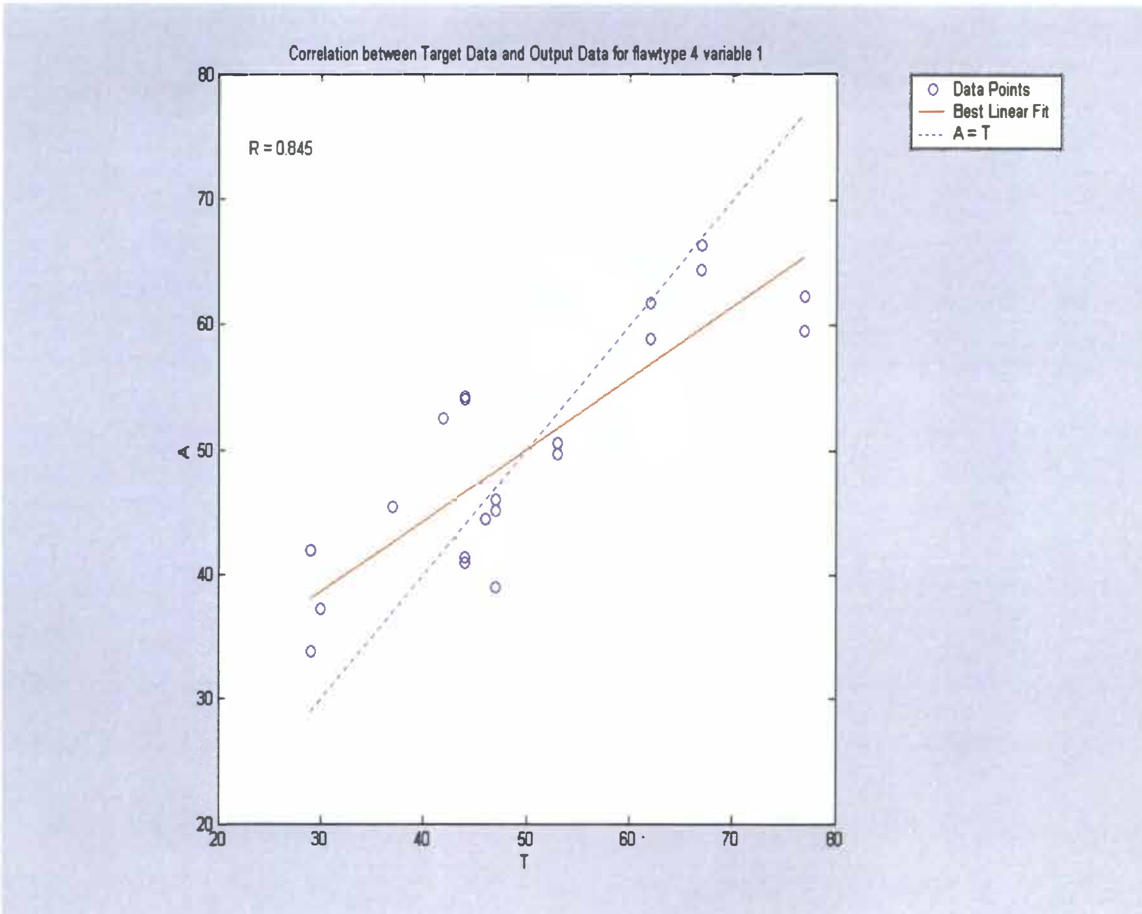

Figure D9. Correlation between Target Data and Output Data For Flaw-type 4 (Pitting), Characteristic 1.

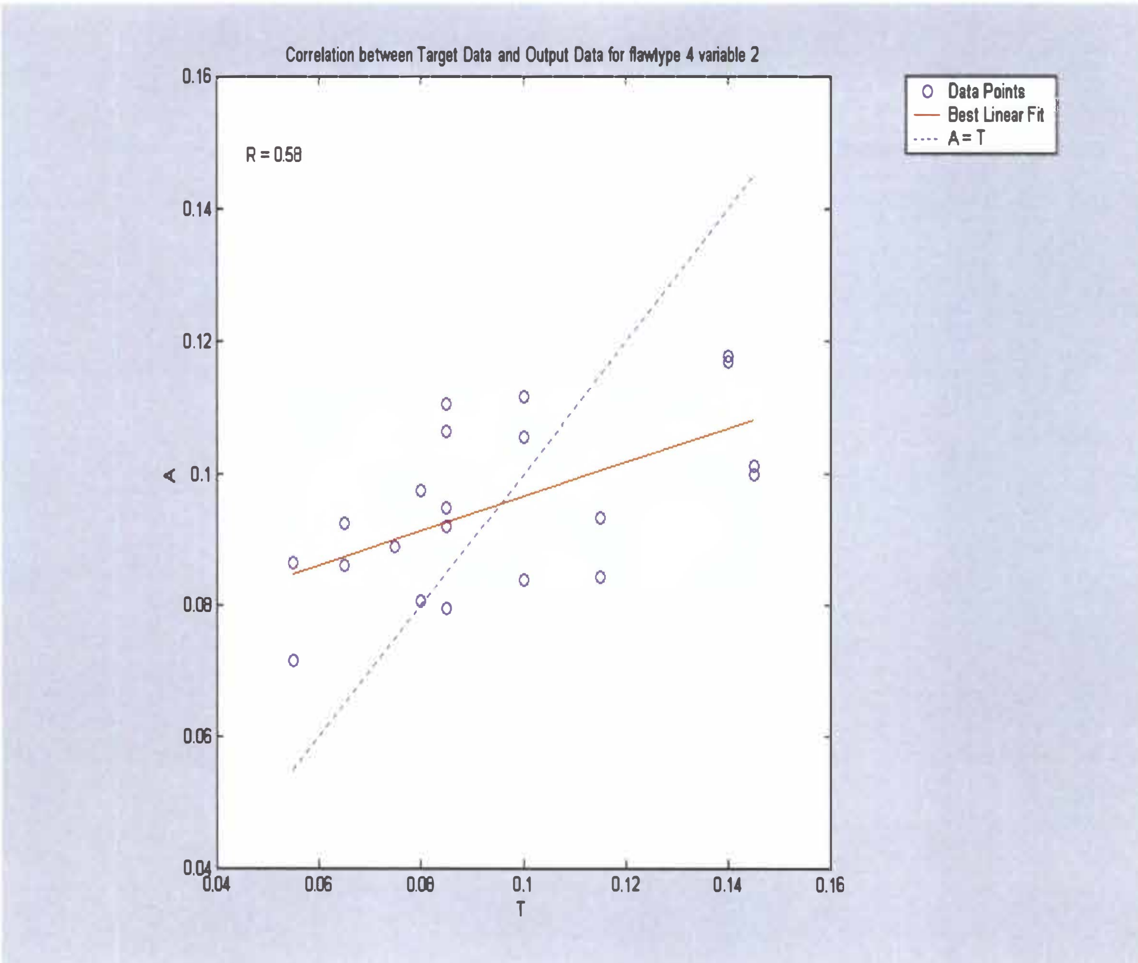

Figure D10. Correlation between Target Data and Output Data For Flaw-type 4 (Pitting), Characteristic 2.

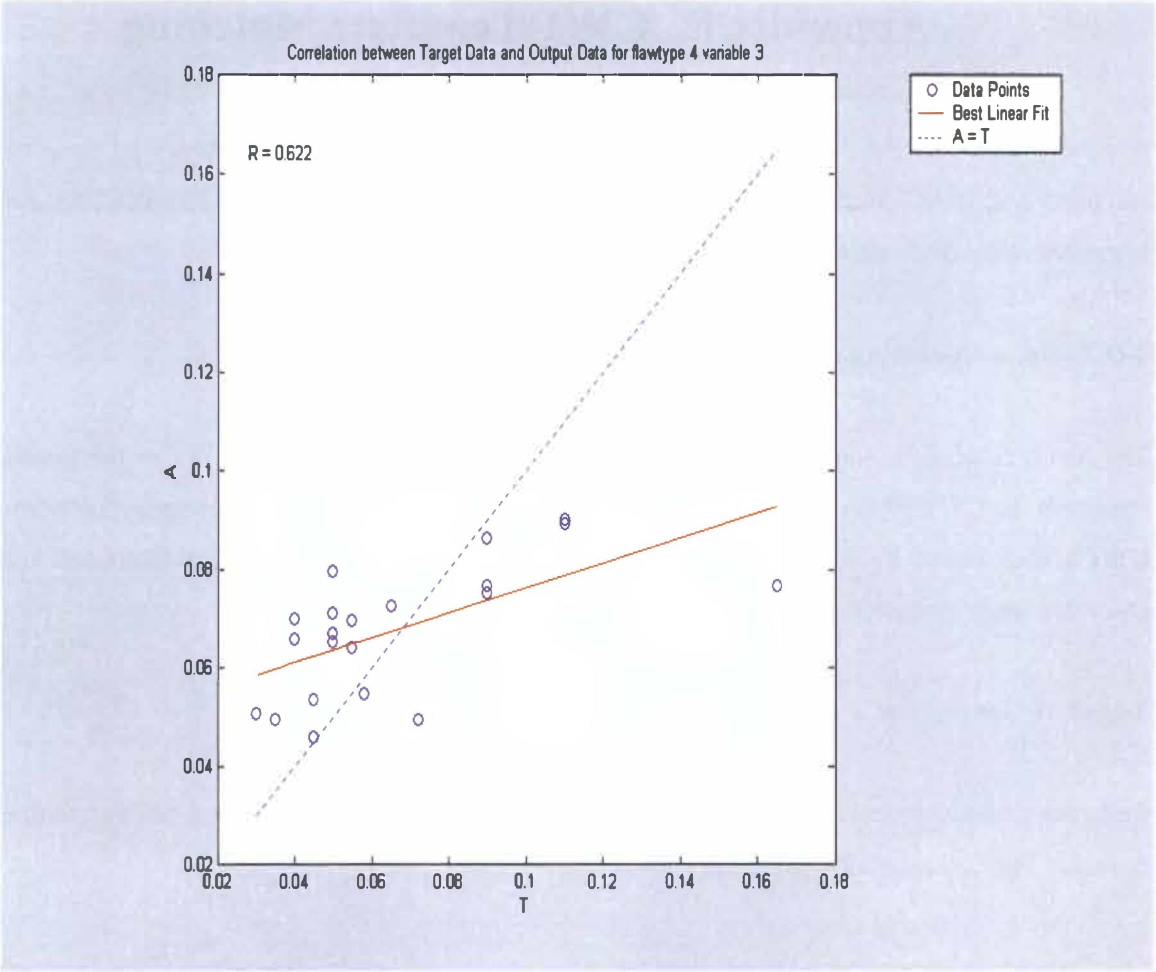

Figure D11. Correlation between Target Data and Output Data For Flaw-type 4 (Pitting), Characteristic 3.

Appendix E. CWT Template Matching

Appendix E was divided into two parts. The first section discusses the theory of template matching and its application to CWTs. The second section contains results generated for and presented in the dissertation.

2-D Template Matching (2DTM) Utilizing the CWT

The two-dimensional template matching technique may be use to compare CWT of the known flaw with the CWT of the unknown flaw. The use of this method requires no compression of the CWT and retains all the information in the transformation. Template matching was divided into two parts, template generation and template matching routine.

Template Generation

Template generation was comparable to temporal image blending. In this case, CWTs of similar flaws are blended together to form the known-flaw template.

The first step in generating a known-flaw template was to scale each known-flaw CWT between 0 and 1. This step was taken because the magnitude of the signal was not very important and was highly influenced by probe-wobble. Next, the known-flaw CWTs are averaged using MATLAB's mean2.m program. This program was specifically used to determine 2-D means. The result was the known-flaw template.

The known-flaw templates are then compared with the unknown-flaw CWT. The comparison methods are described next.

Template Matching Routine

Two-dimensional template matching was generally used in scene analysis to detect if a reference object image was present in a test image. If we are given an object image with dimension M x N and test image with dimensions I x J such that M ≤ I and N ≤ J, the sum of the squared-errors was given by

$$D(m,n) = \sum_{i=m}^{m+M-1} \sum_{j=n}^{n+N-1} | t(i,j) - r(i-m, j-n) |^2 \qquad \text{(E1)}$$

where: $t(i,j)$ was the test image

$r(i-m, j-n)$ was the object image.

Template matching was conducted by moving the object image within the test image for locations (m,n) and calculating D(m,n) at each position, then determining the location at which the error was minimum. **If there was little variation in the magnitude of the test image**, the minimum D(m,n) was achieved when

$$c(m,n) = \sum \sum t(i,j) r(i-m, j-n) \qquad \text{(E2)}$$

was maximum for all possible locations (m,n). The quantity c(m,n) was a cross-correlation between t(i,j) and r(i-m,j-n) computed at locations (m,n). **If the magnitude assumption was not valid, a normalized measurement**

$$c_N(m,n) = \frac{c(m,n)}{\sqrt{\sum_i \sum_j | t(i,j) |^2 \sum_i \sum_j | r(i,j) |^2}} \qquad \text{(E3)}$$

was a more appropriate measure [25]. Given that the CWT modulus fluctuates, employing the correlation coefficient may not be valid. Since the E^2 parameter does not have this restriction, it will be used.

The E^2 map may be used in a variety of ways, but for the purposes of this research, the best overall measurement of error may be the average value of E^2. The average E^2 value may be calculated using:

$$E_{avg}^2 = \frac{\sum_x \sum_y E^2}{R*C} \tag{E4}$$

where: R = number of rows of the E^2 map

C = number of columns of the E^2 map.

A similar parameter that may be calculated would be to apply the same procedure to the correlation coefficient map as was done in Equation (E2)

$$C_N^{AVG} = \frac{\sum_m \sum_n c_N(m,n)}{R*C} \tag{E5}$$

This parameter could yield a good estimation of how the flaw cwt matches with the generated flaw template. The best template matching parameter (either E^2, C_N, E^2_{avg} or C_N^{AVG}) will be used.

Template Matching Results

In this section, the template-matching results are generated using the first flaw (T23b01_T077R004_1) as the unknown. This CWT was then compared to the other six templates, generating an E^2 map. The E^2 maps are given (in figures E1-E6), along with the average E^2 values. A more thorough investigation was made using each flaw compared to all flaw-type templates. The results are tabulated in Table E1.

The results obtained using template matching required approximately 20 seconds to calculate and graph the above plots. This was accomplished on a 350 MHz system with 128 Mbyte RAM.

The best overall results (yielding the most correct classifications) were using the minimum E^2 value.

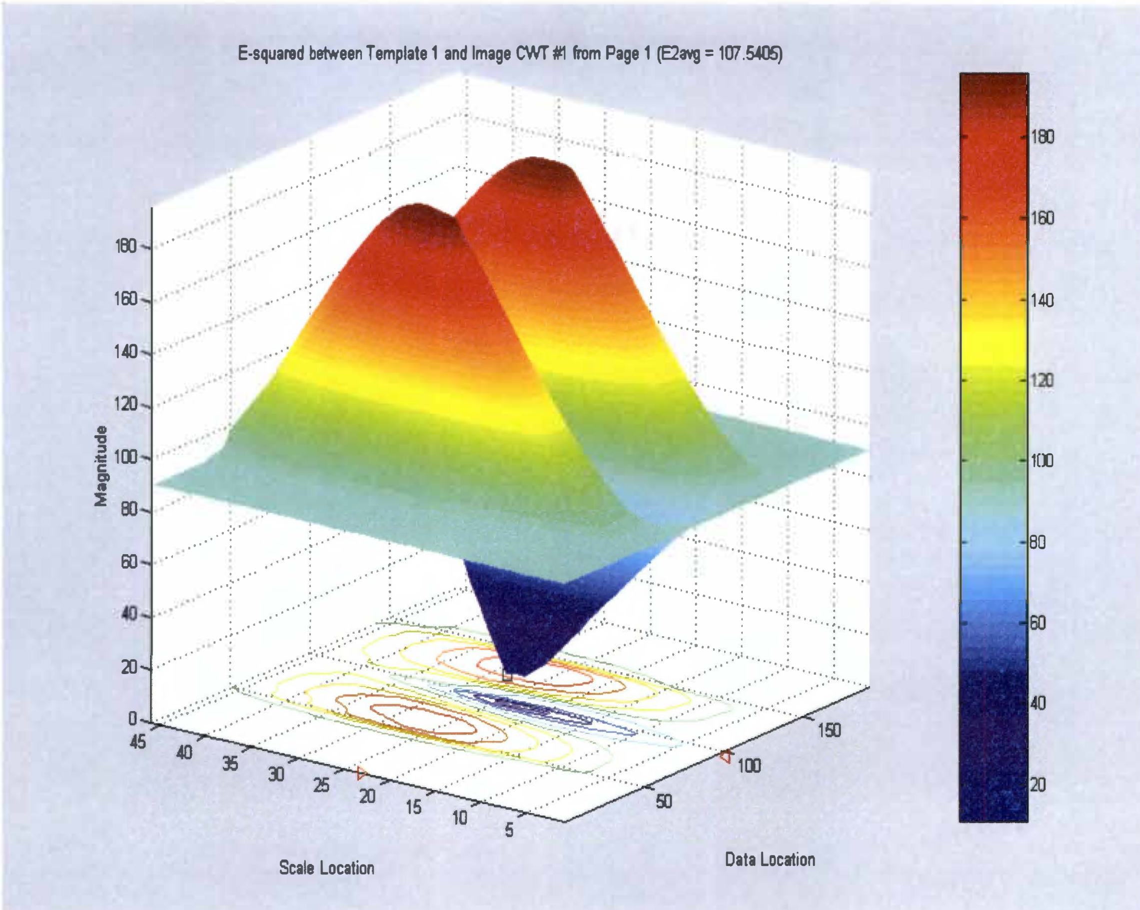

Figure E1. Template Matching Results (E^2) utilizing the first CWT template (Group T23b01 - WA) vs. CWT for T23b01_T077R004_1.

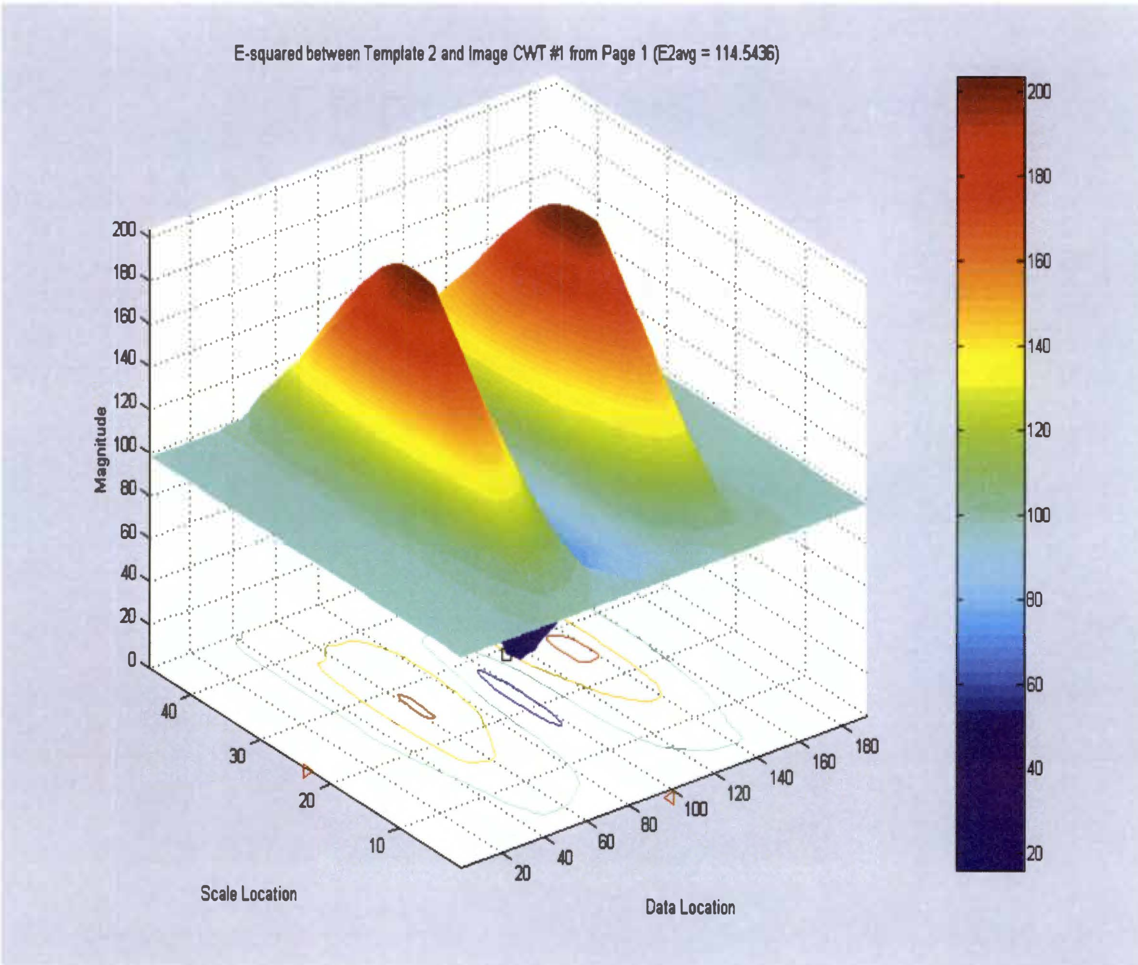

Figure E2. Template Matching Results (E^2) utilizing the first CWT template (Group T26b01) vs. CWT for T23b01_T077R004_1.

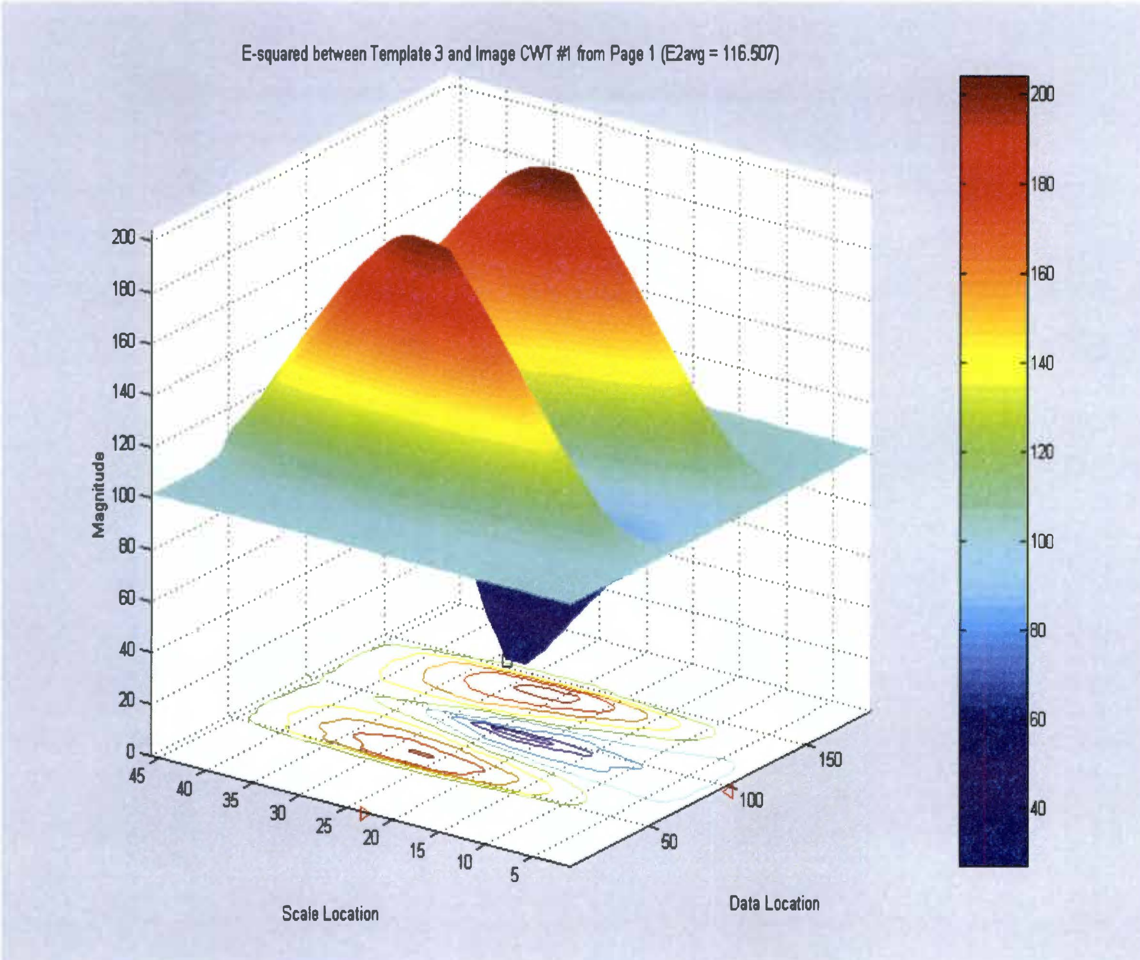

Figure E3. Template Matching Results (E^2) utilizing the first CWT template (Group T24b01) vs. CWT for T23b01_T077R004_1.

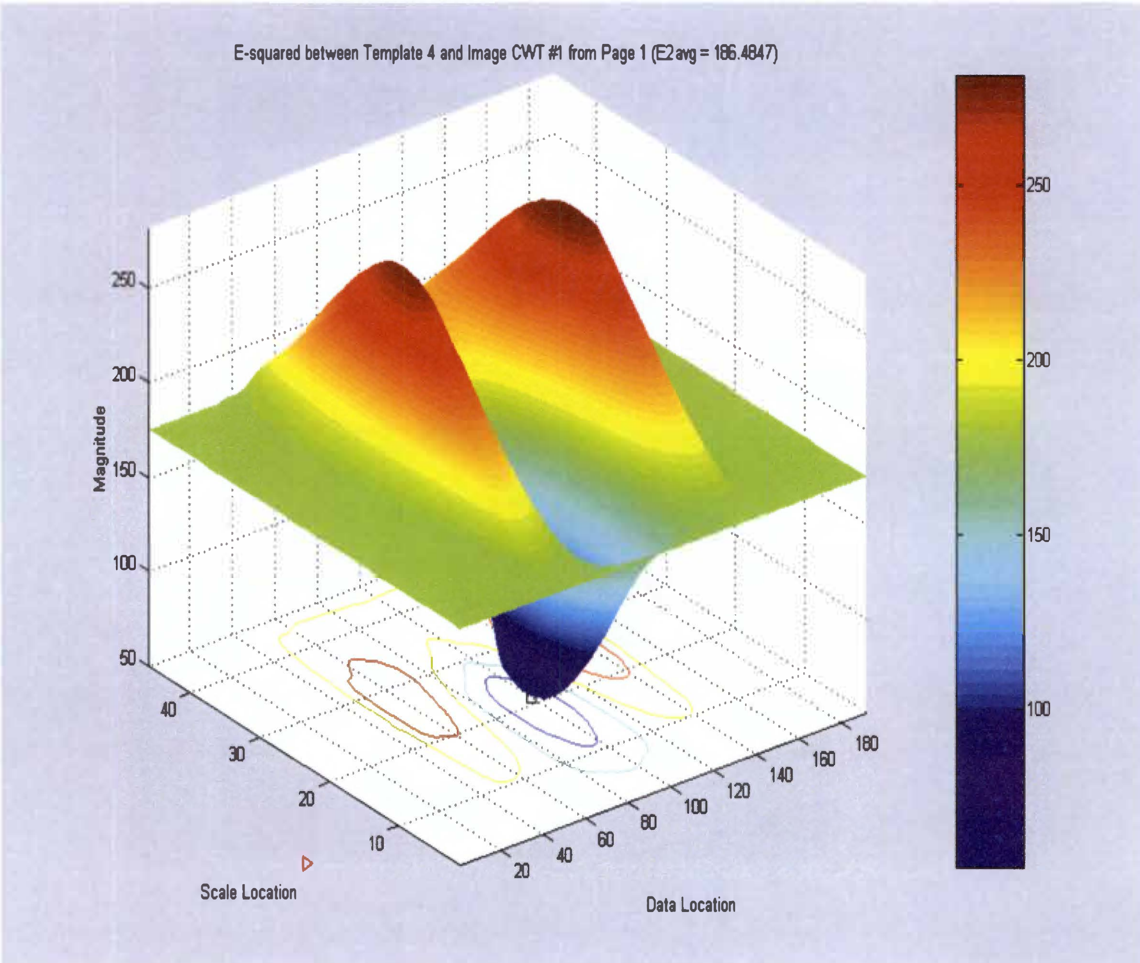

Figure E4. Template Matching Results (E^2) utilizing the first CWT template (Group T99b99) vs. CWT for T23b01_T077R004_1.

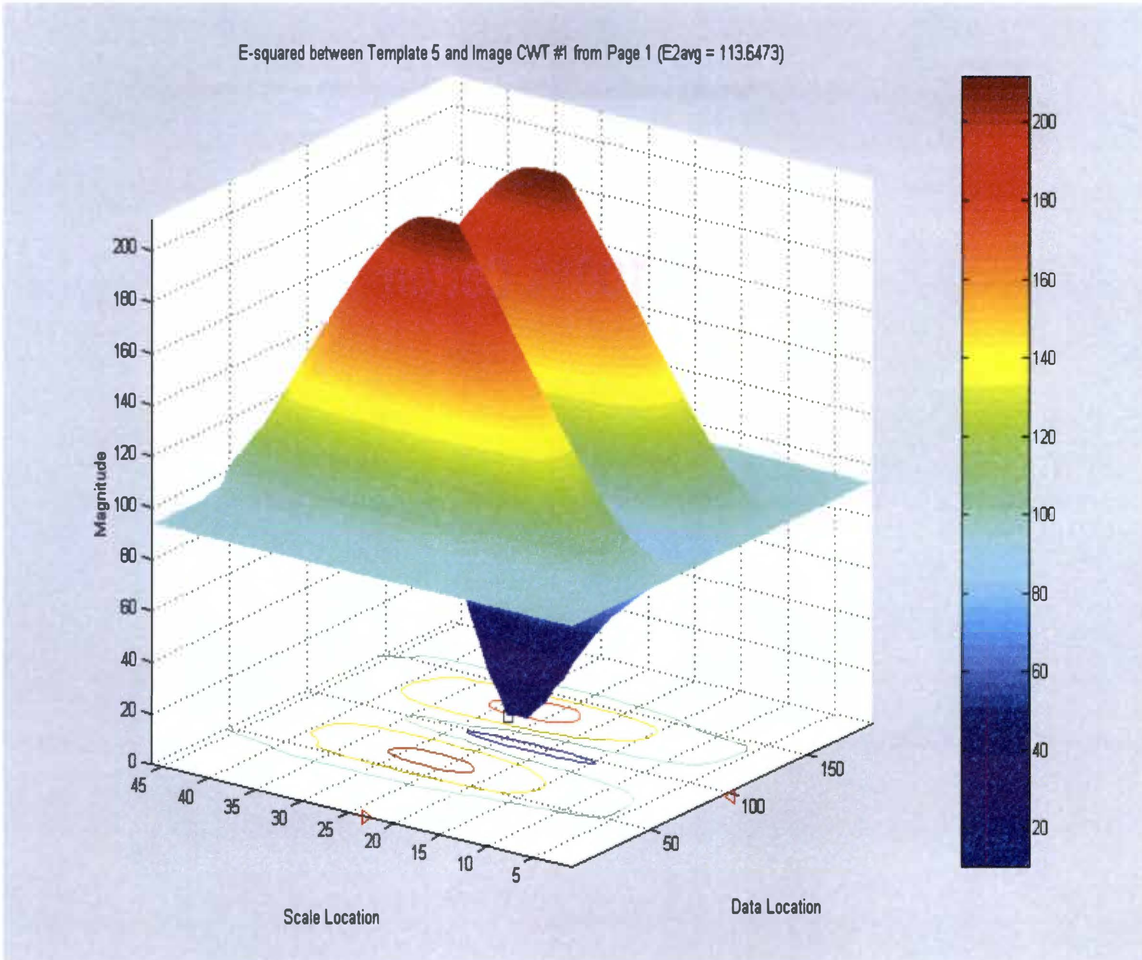

Figure E5. Template Matching Results (E^2) utilizing the first CWT template (Group T24b01 Cal) vs. CWT for T23b01_T077R004_1.

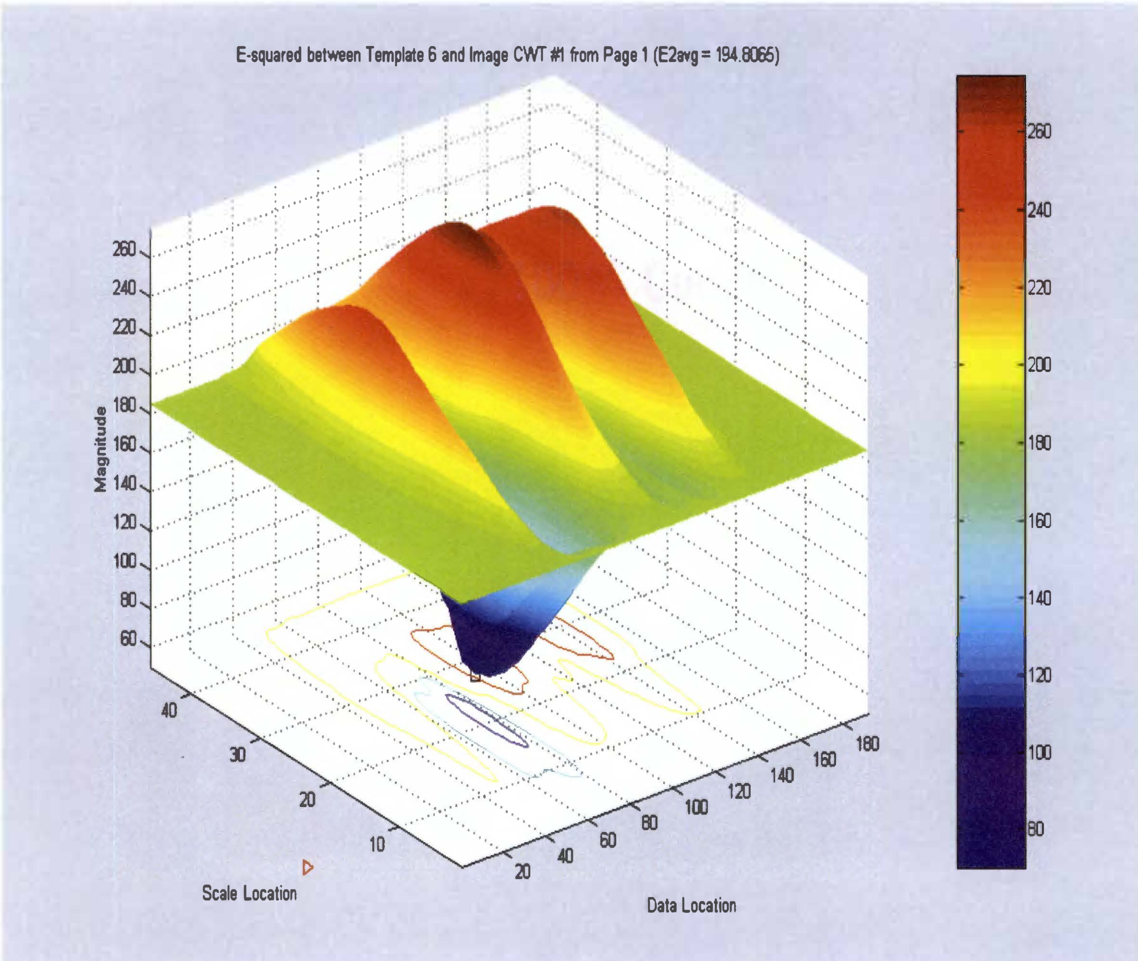

Figure E6. Template Matching Results (E^2) utilizing the first CWT template (Group T23b01-WB) vs. CWT for T23b01_T077R004_1.

Table E1. Initial Template Matching Results using Each Flaw as the Unknown.

Flaw	Actual Flaw-type	Average C_N Flaw-type	Maximum C_N Flaw-type	Average E^2 Flaw-type	Minimum E^2 Flaw-type
T23b01_T077R004_1	1	5	6	1	5
T23b01_T077R022_3	1	5	4	1	1
T23b01_T077R022_2	1	5	4	1	1
T23b01_T077R022_1	1	5	4	1	1
T23b01_T077R023_1	1	5	6	1	1
T23b01_T077R025_1	1	5	5	1	4
T23b01_T077R026_1	1	5	6	1	1
T23b01_T077R027_1	1	5	6	1	1
T23b01_T077R028_1	1	5	6	1	5
T26b01_T108R116_1	2	5	6	1	2
T26b01_T107R116_1	2	5	6	1	2
T26b01_T106R118_1	2	5	6	1	5
T26b01_T084R005_1	2	5	6	1	2
T26b01_T087R005_1	2	5	6	1	5
T26b01_T115R006_1	2	5	6	1	5
T26b01_T095R002_2	2	5	6	1	1
T26b01_T095R002_1	2	5	6	1	2
T26b01_T134R062_1	2	5	6	1	4
T26b01_T045R118_1	2	5	6	1	2
T26b01_T054R082_1	2	5	6	1	2
T24b01_T072R018_2	3	5	6	1	5
T24b01_T072R018_1	3	5	6	1	5
T24b01_T072R014_1	3	5	6	1	5
T24b01_T072R012_1	3	5	6	1	3
T24b01_T080R034_1	3	5	6	1	3
T24b01_T080R027_2	3	5	5	1	1
T24b01_T080R027_1	3	5	5	1	3

Table E1. Continued.

Flaw	Actual Flaw-type	Average C_N Flaw-type	Maximum C_N Flaw-type	Average E^2 Flaw-type	Minimum E^2 Flaw-type
T24b01_T080R026_2	3	5	6	1	3
T24b01_T080R026_1	3	5	6	1	2
T24b01_T080R025_2	3	5	6	1	5
T24b01_T080R025_1	3	5	6	1	4
T24b01_T080R023_1	3	5	6	1	5
T99b99_T999R999_2	4	5	5	1	4
T99b99_T999R999_1	4	5	6	1	4
T24b01_T999R100_3	5	5	6	1	5
T24b01_T999R100_2	5	5	6	1	5
T24b01_T999R100_1	5	5	6	1	5
T24b01_T999R080_3	5	5	6	1	5
T24b01_T999R080_2	5	5	6	1	5
T24b01_T999R080_1	5	5	6	1	5
T24b01_T999R060_3	5	5	6	1	5
T24b01_T999R060_2	5	5	6	1	5
T24b01_T999R060_1	5	5	6	1	5
T24b01_T999R020_3	5	5	6	1	5
T24b01_T999R020_2	5	5	6	1	5
T24b01_T999R020_1	5	5	6	1	5
T23b01_T075R006_1	6	5	2	1	6
T23b01_T078R006_1	6	5	4	1	6
Probability of Error (# wrong / total flaws)		**36 / 48 = 75%**	**48 / 48 = 100%**	**39 / 48 = 81%**	**15 / 48 = 31%**

Appendix F. EddyC Users Guide

The EddyC User's guide is divided into two sections. The first section details the procedure to load a training flaw into the uTR cell. The second section outlined the procedure to check an unknown flaw.

The uTR cell contains all the raw information about the flaw examples. The uTR cell is then processed into a TR cell. The TR cell extracts only the needed information from the uTR cell and processes the data into the needed formats. The uTR and TR data cells are then used in the classification and characterization processes. The characterization procedure generates ".mat" MATLAB data structures. These ".mat" structures contain all the neural network parameters needed to generate and operate the characterization ANNs. Once the uTR has been fully loaded with the training examples, processed into a TR and both have been used to train and generate a ".mat" NN structure, a flaw may be classified and characterized.

Both loading and checking procedure examples used a flaw that was saved. The EddyC program saves input data and basic flaw signals into a file that bears the flaws data-file name (example: E_96001_DHR00BC066I006_1). The difference between using a pre-saved flaw and one that is not is at the fifth step listed below. Instead of indicating a "S"aved data file (by typing "S"), the user indicates "W"indow data file (by typing "W"). The procedure to load or check a flaw that has not be pre-saved is exactly the same, except the information listed under the ETSS or PDD Input Information and ETSS or PDD Flaw Classification and Characterizations are input into the system.

An italicized sentence indicates that MATLAB is prompting the user for input information. The information typed after the period in the user input information. Bold type indicates comments about program input and/or output. All the figures are output.

Loading a Flaw (within the uTR) and Reprocessing the TR

>> eddyc

Is this "P"DD or "E"TSS data. E

Enter Manufacturer of Steam Generator (B, C or W) or ETSS #. 96005
Input ECT saved data filename (ex. T24b01_T080R025_1, no .mat needed). DHR00PC051I066_5
Is this a "S"aved data file or a command "W"indow data file? S

========== ETSS or PDD Input Information ==========

The origin (E = ETSS or P = PDD) of the data was E
The EC Data filename was DHR00PC051I066_5
The Steam Generator type or ETSS # was 96005
The PDD or ETSS location was 771 , doublecheck flaw location for the given filename!
The PDD or ETSS Flaw Magnitude was 0.72425
The PDD or ETSS Phase Angle was 85.113

======= ETSS or PDD Flaw Classification and Characterizations =======

The PDD or ETSS Flaw Type was PI
The PDD or ETSS Percent Thru-wall was 53
ETSS characteristics = 0.085 0.055

Does the data appear to be correct ("y"es or "n"o)? y

Figures F1 through F3 are generated.

Does user want to "l"oad the data cell into the uTR training cell or "c"heck flaw. l
Input the uTR run number. 99
Is this the first cell added to the uTR cell array? n
Does user want to input more data into uTR matrix, enter "y"es. n
Does user want to view statistical data for uTR Feature Matrix, enter "y"es. y
Input the number feature families in the feature vector (usually 5). 5
Enter the last position for each of the above feature families in MATLAB format ([2 21 23 48 51]). [2 21 23 48 51]

Figure F4 is generated.

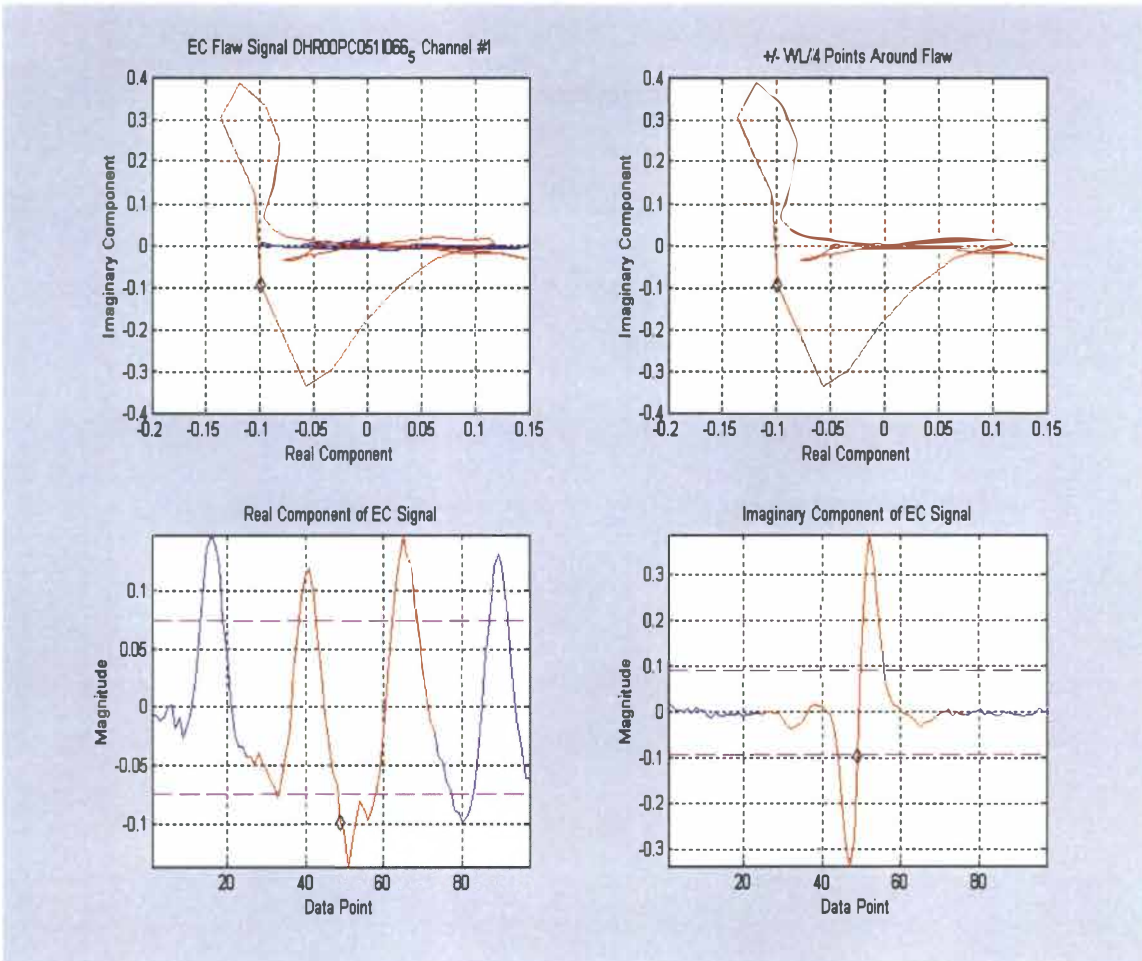

Figure F1. ECT Resistance Signal (Lissarious and Component Plots) of Flaw DHR00PC051I066_5.

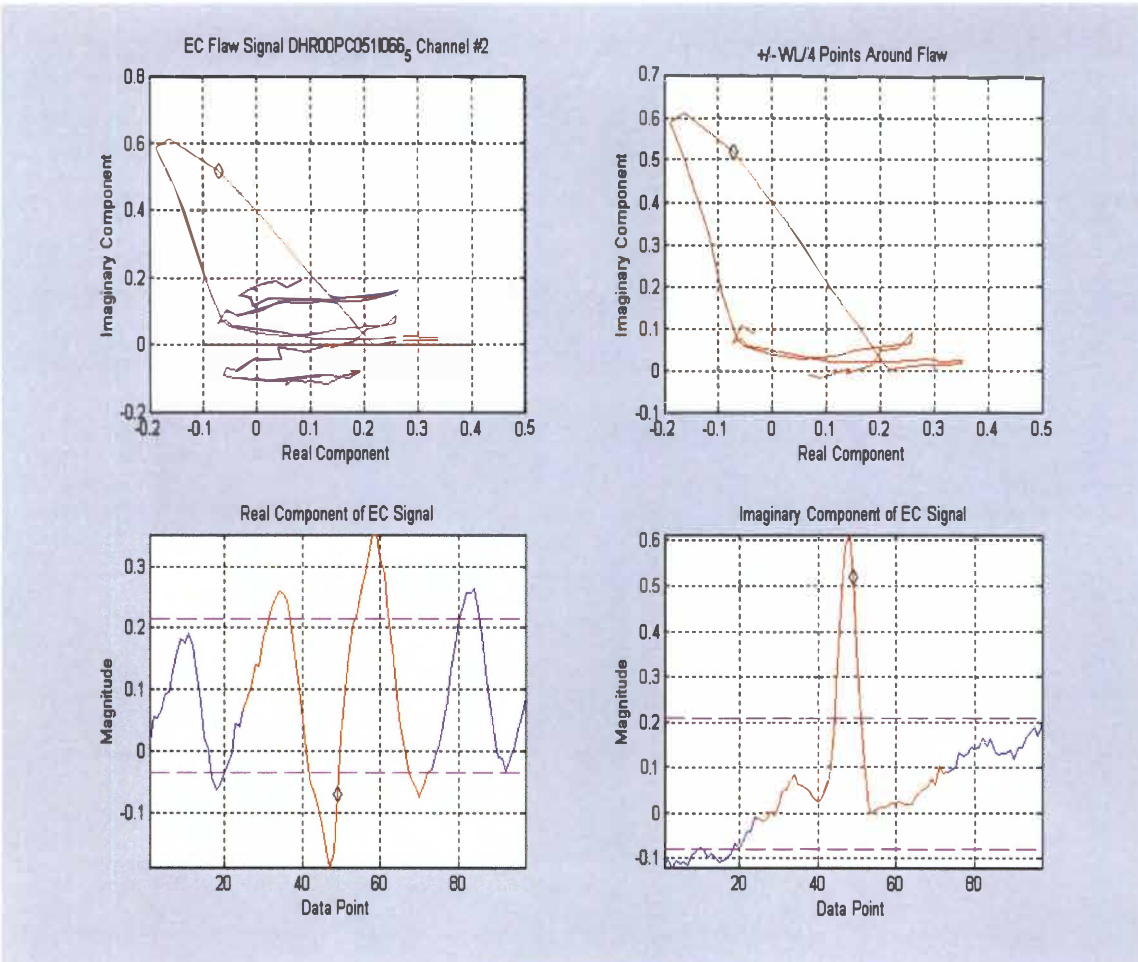

Figure F2. ECT Reactance Signal (Lissarious and Component Plots) of Flaw DHR00PC051I066_5.

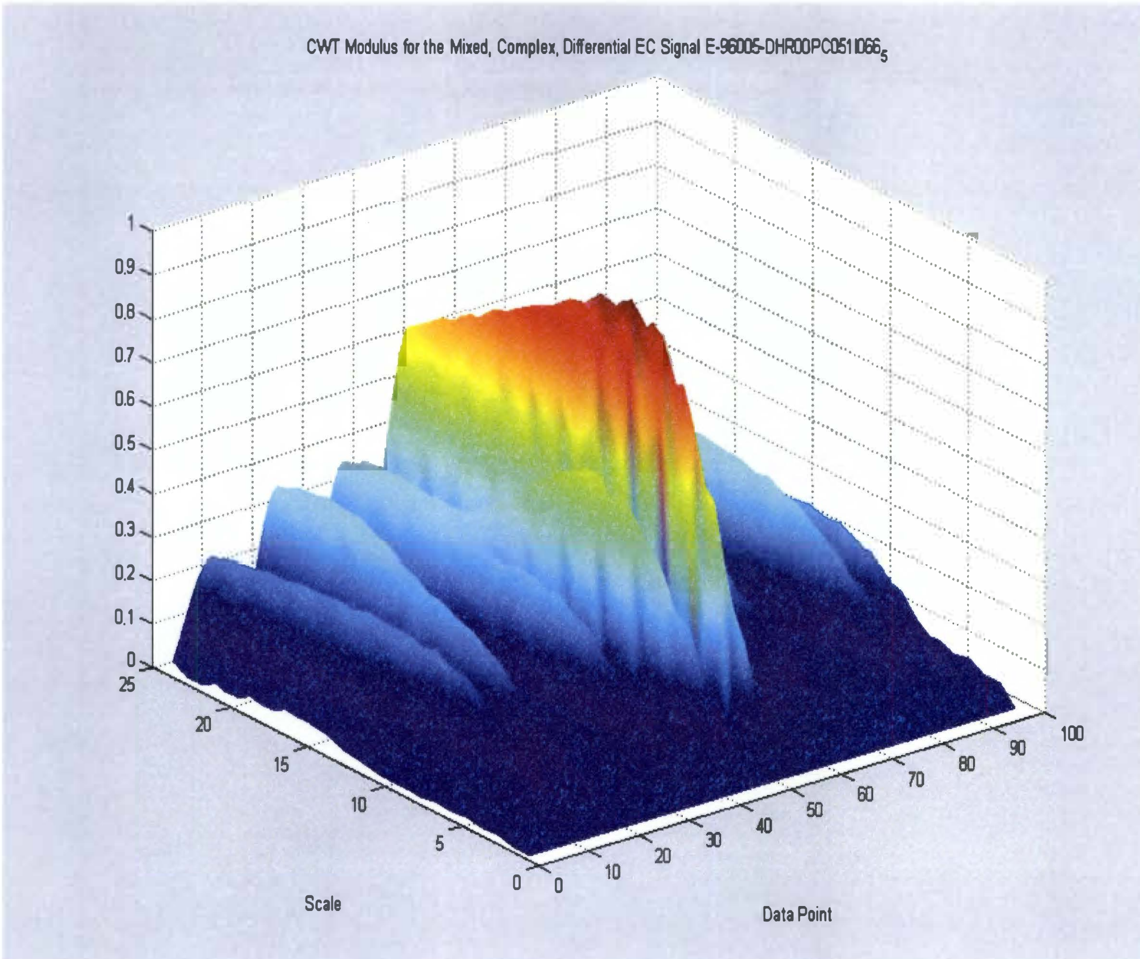

Figure F3. CWT of the ECT Resistance Signal for Flaw DHR00PC051I066_5.

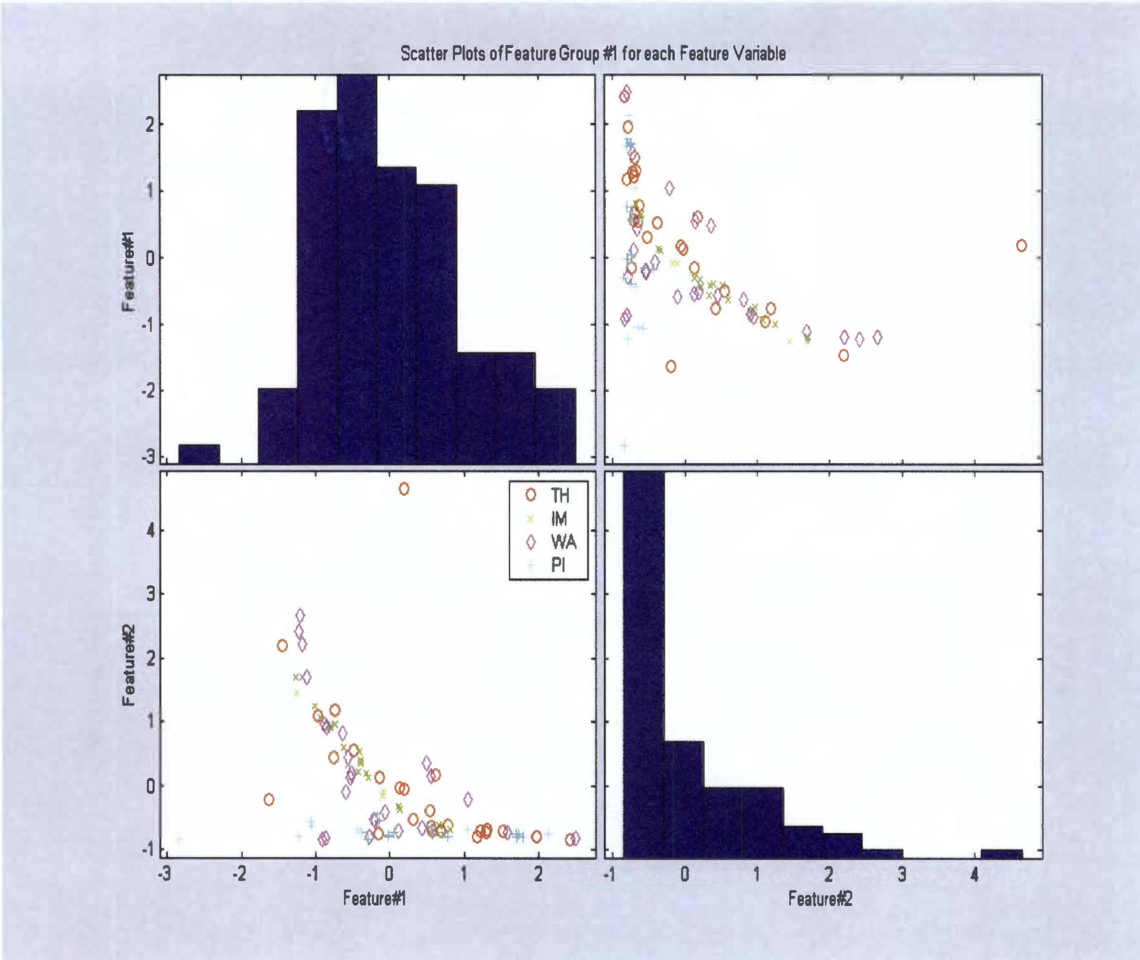

Figure F4. Scatter Plot of Raw Feature Family 1 of the ECT Flaw DHR00PC051I066_5.

Number of columns (variables) for feature group 2 = 19

Enter absolute coeff groupings in cell format {1:5 6:10 11:15 16:19}
or geometric groupings in cell format {1:5 6:9 10:14 15:19 20:24}. {1:5 6:10 11:15 16:19}

Figures F5 through F9 are generated.

Number of columns (variables) for feature group 3 = 2
Warning: Divide by zero.
> In C:\Patrick\eddym\uTR_statistics.m at line 79
In C:\Patrick\eddym\EddyC.m at line 198

The non-variance (defined as <= 0.010000) deleted columns for the Flaw-type # 4 Feature Matrix
was/are: 6

Number of columns (variables) for feature group 4 = 24

Enter absolute coeff groupings in cell format {1:5 6:10 11:15 16:19}
or geometric groupings in cell format {1:5 6:9 10:14 15:19 20:24}. {1:5 6:9 10:14 15:19 20:24}

Number of columns (variables) for feature group 5 = 3

Figures F10 through F16 are generated.

If uTR was fully loaded, user should "s"ave the statistical information .n
Does user want to process the uTR Feature Matrix, enter "y"es or "n"o. y

The non-variance (defined as == 0) deleted columns for the Feature Matrix
was/are: 29

Does user want to edit feature vector ("y"es or "n"o). n

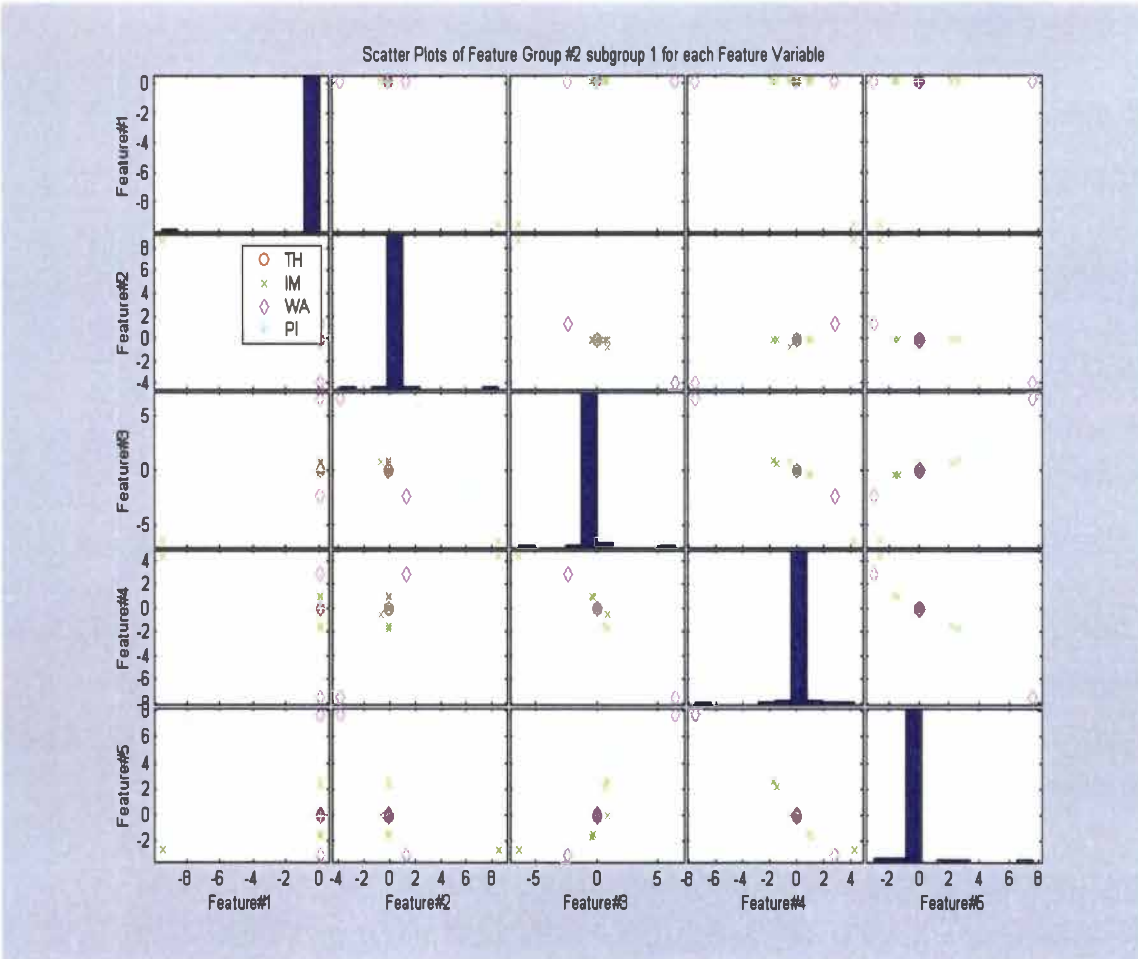

Figure F5. Scatter Plot of Raw Feature Family 2, Subgroup 1, of ECT Flaw DHR00PC051I066_5.

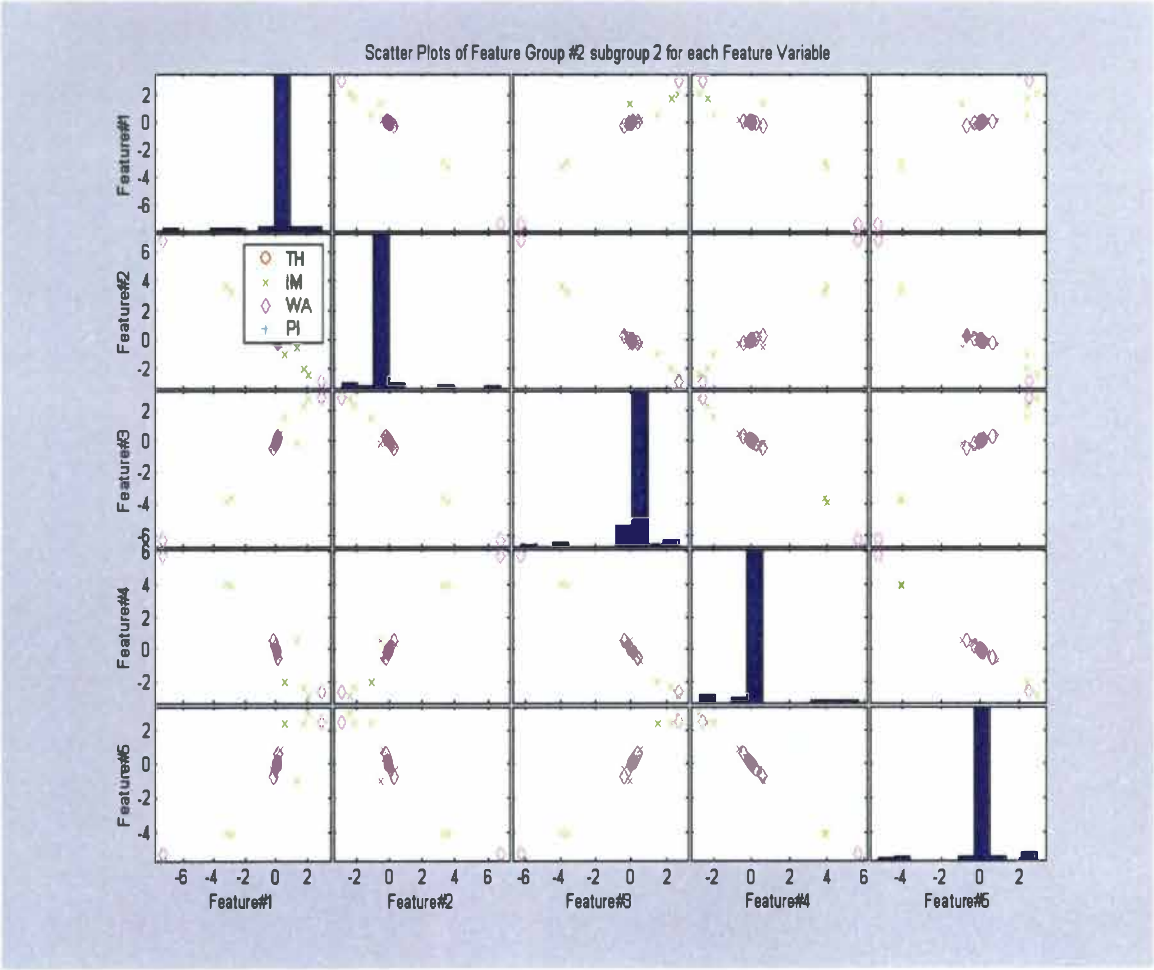

Figure F6. Scatter Plot of Raw Feature Family 2, Subgroup 2, of ECT Flaw DHR00PC051I066_5.

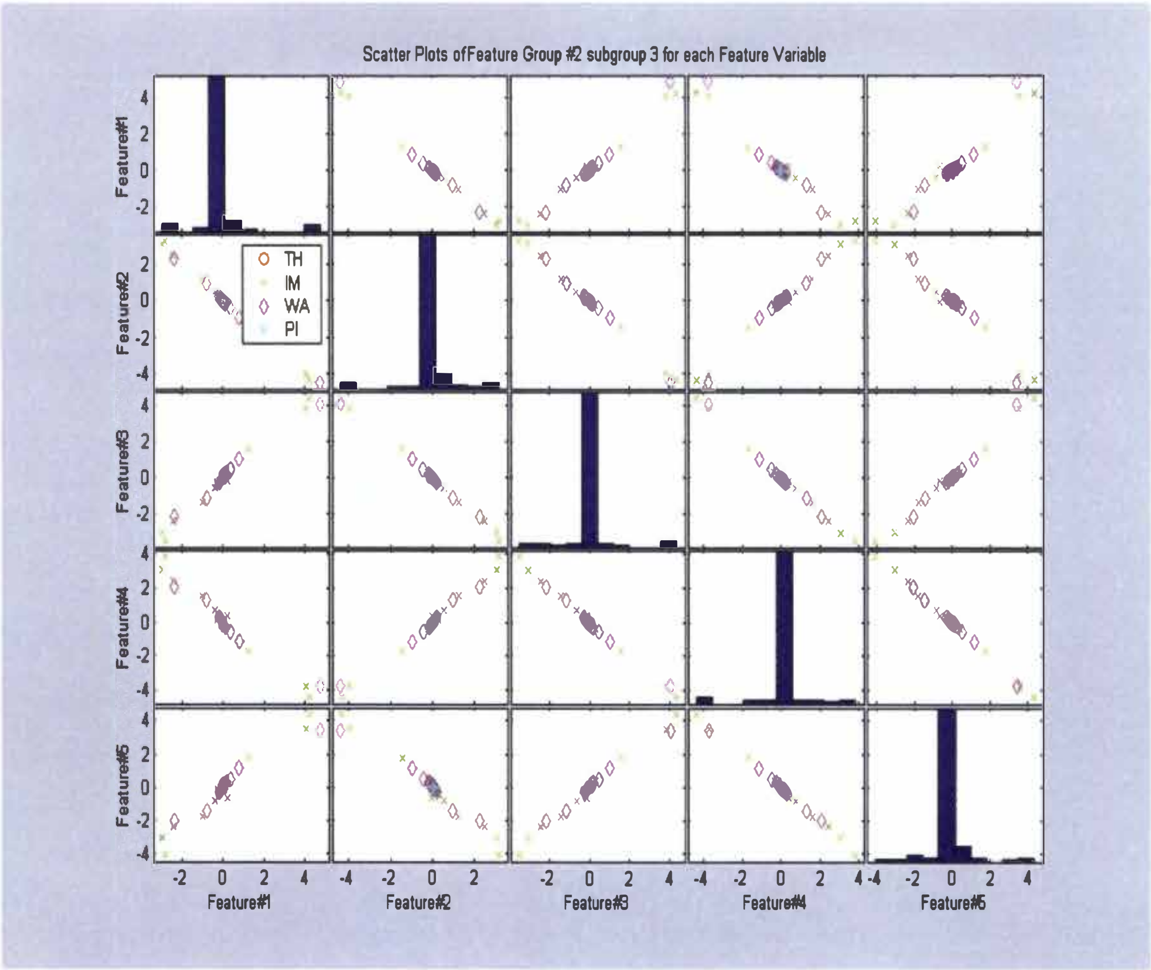

Figure F7. Scatter Plot of Raw Feature Family 2, Subgroup 3, of ECT Flaw DHR00PC051I066_5.

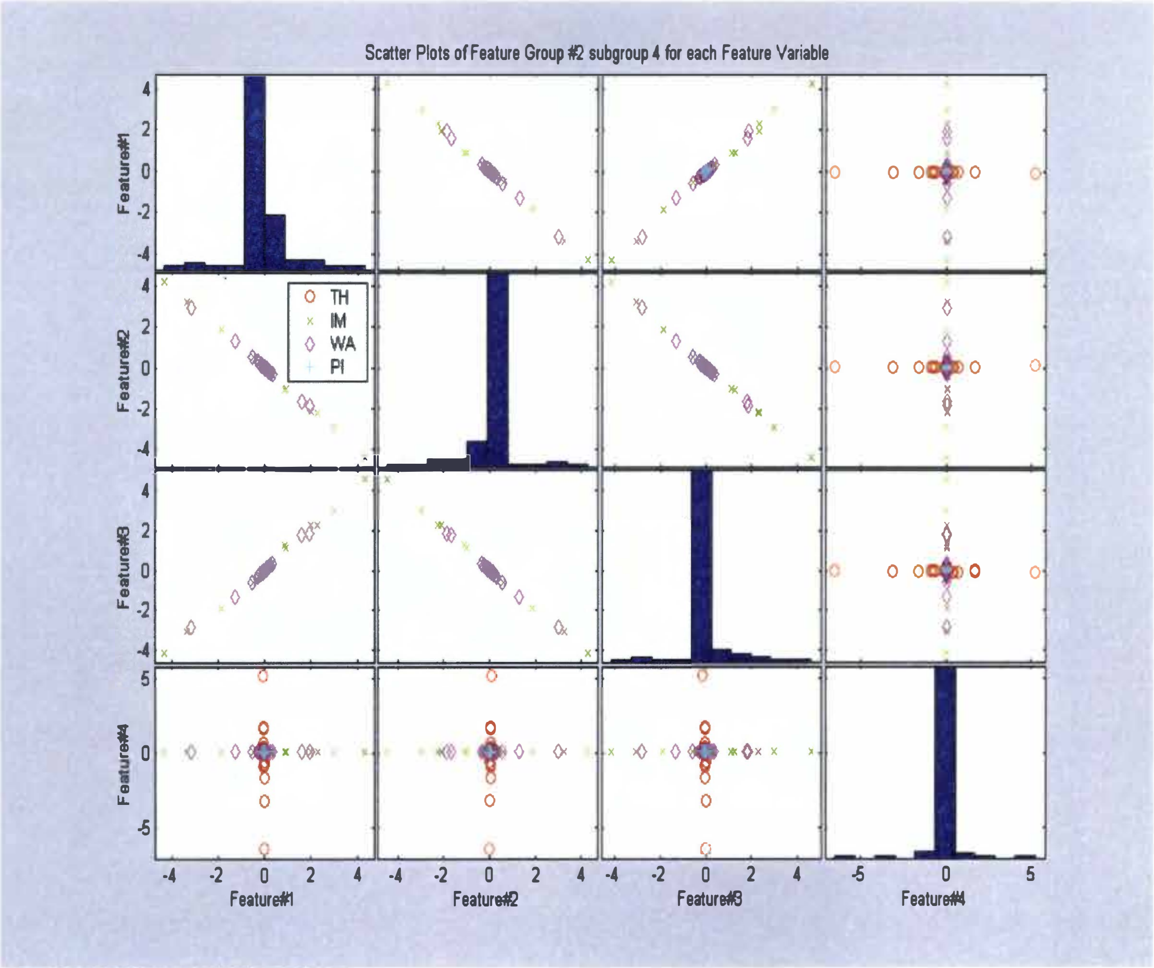

Figure F8. Scatter Plot of Raw Feature Family 2, Subgroup 4, of ECT Flaw DHR00PC051I066_5.

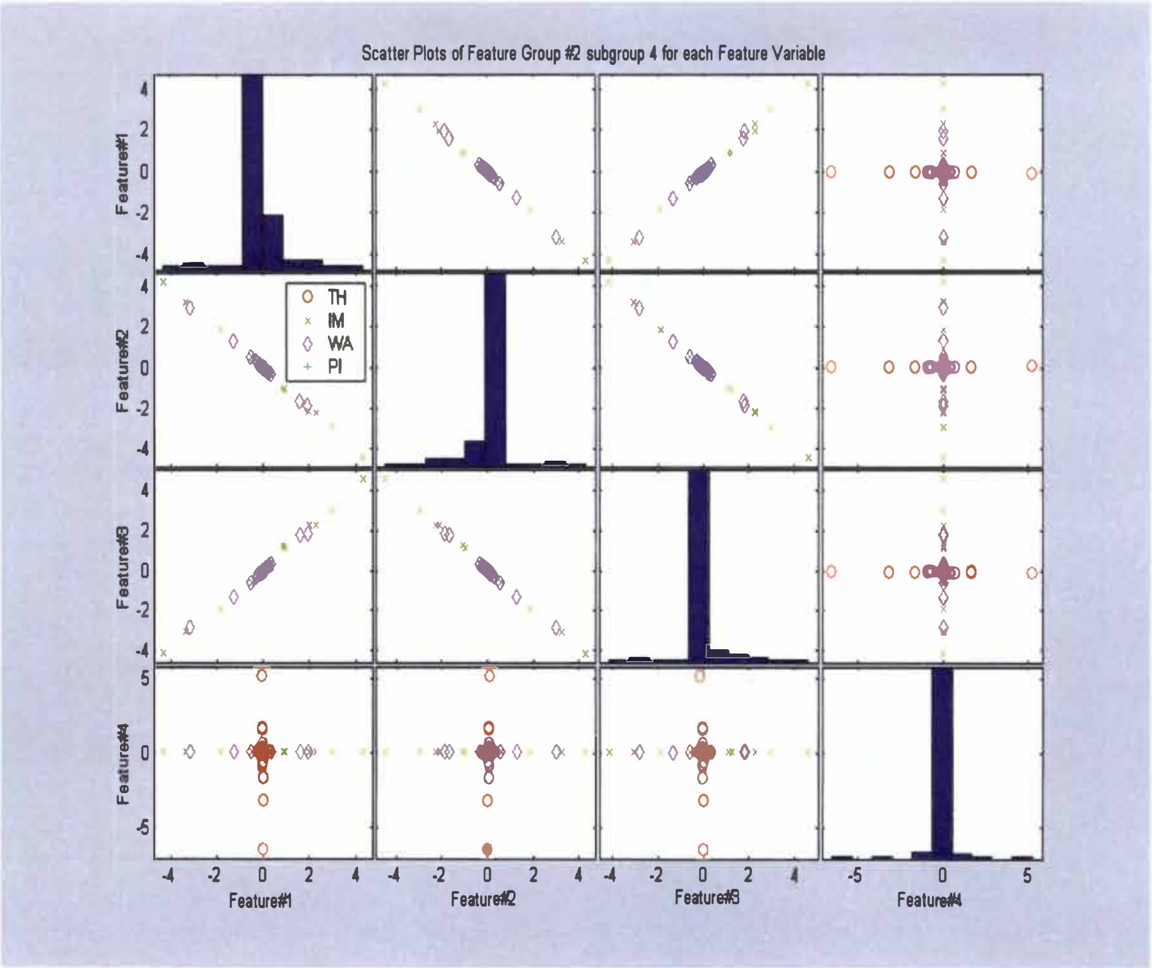

Figure F9. Scatter Plot of Raw Feature Family 2, Subgroup 5, of ECT Flaw DHR00PC0511066_5.

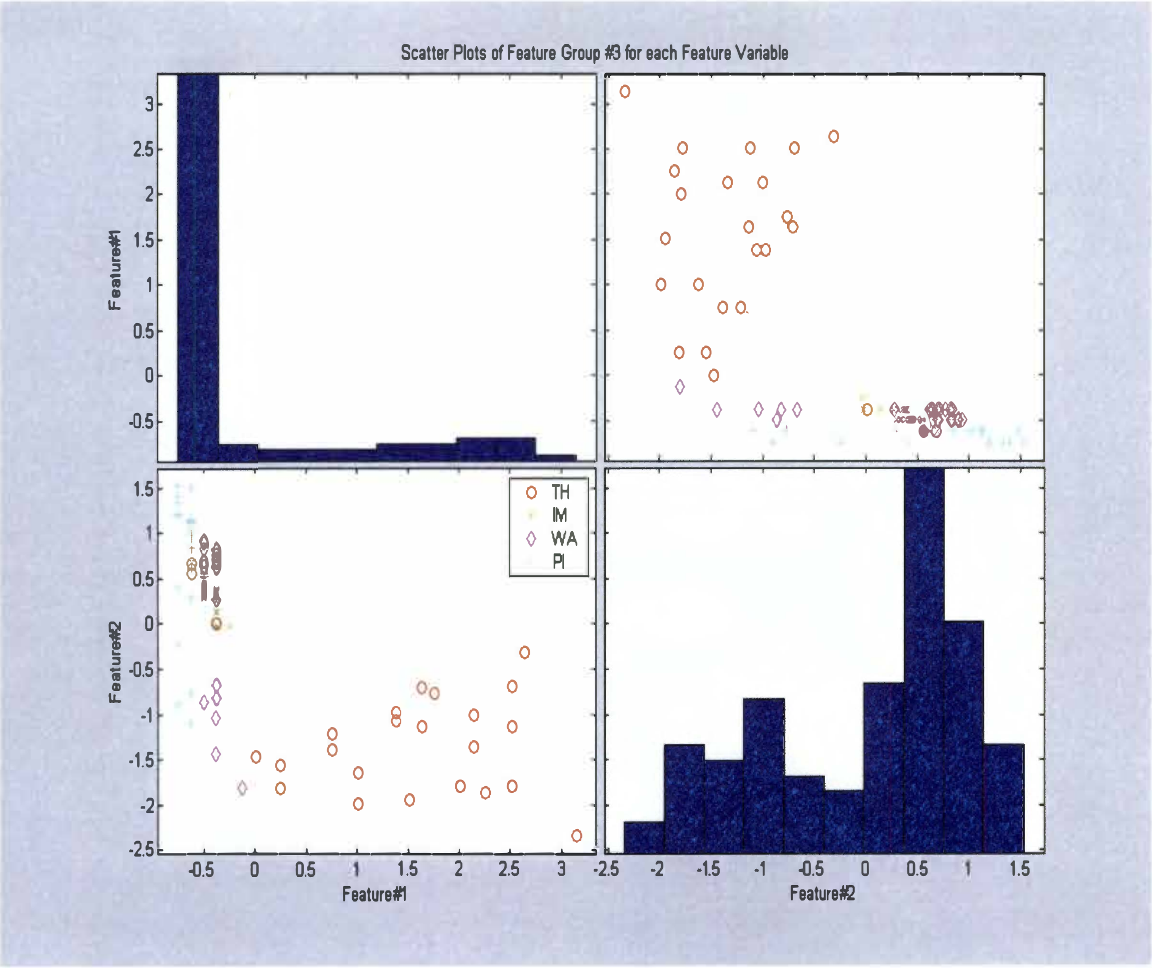

Figure F10. Scatter Plot of Raw Feature Family 3 of ECT Flaw DHR00PC051I066_5.

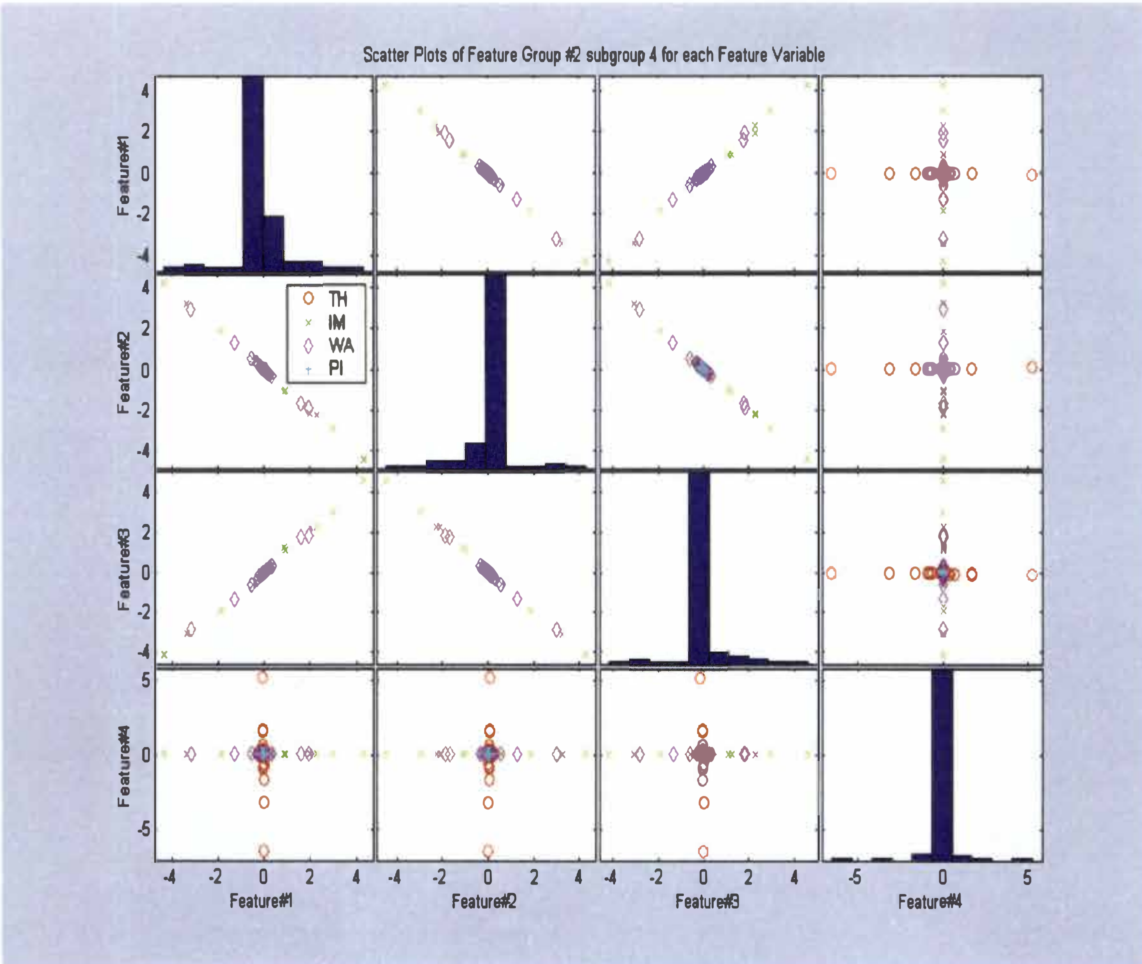

Figure F11. Scatter Plot of Raw Feature Family 4, Subgroup 1, of ECT Flaw DHR00PC051I066_5.

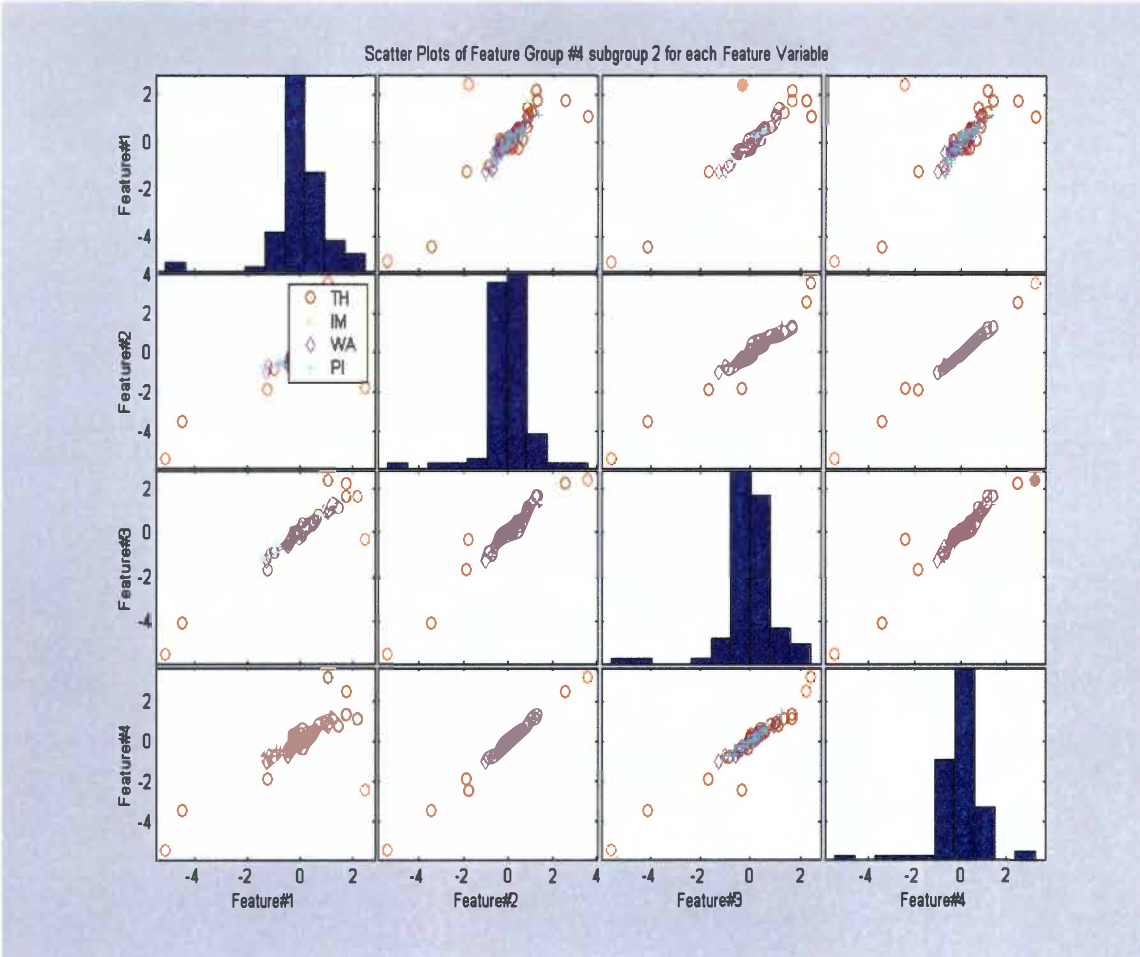

Figure F12. Scatter Plot of Raw Feature Family 4, Subgroup 2, of ECT Flaw DHR00PC051I066_5.

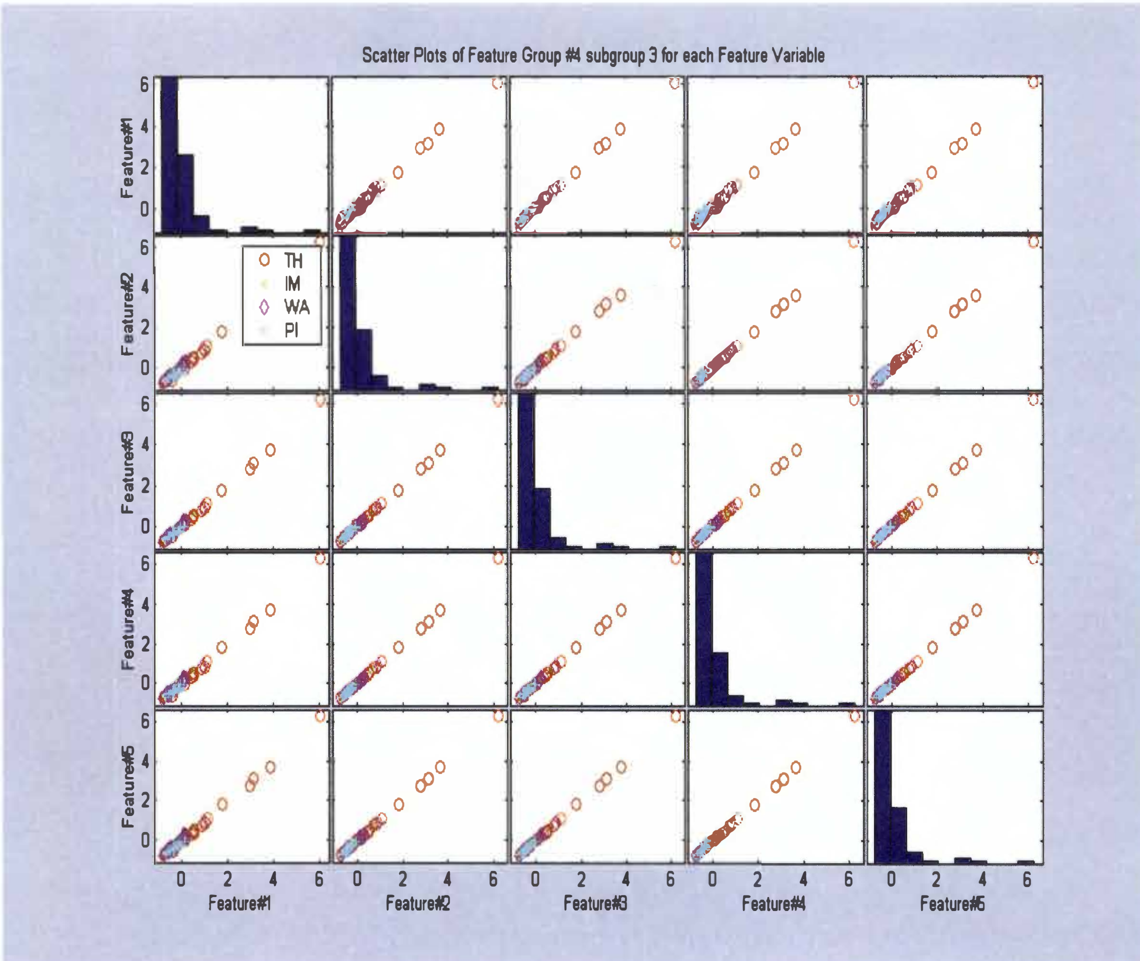

Figure F13. Scatter Plot of Raw Feature Family 4, Subgroup 3, of ECT Flaw DHR00PC051I066_5.

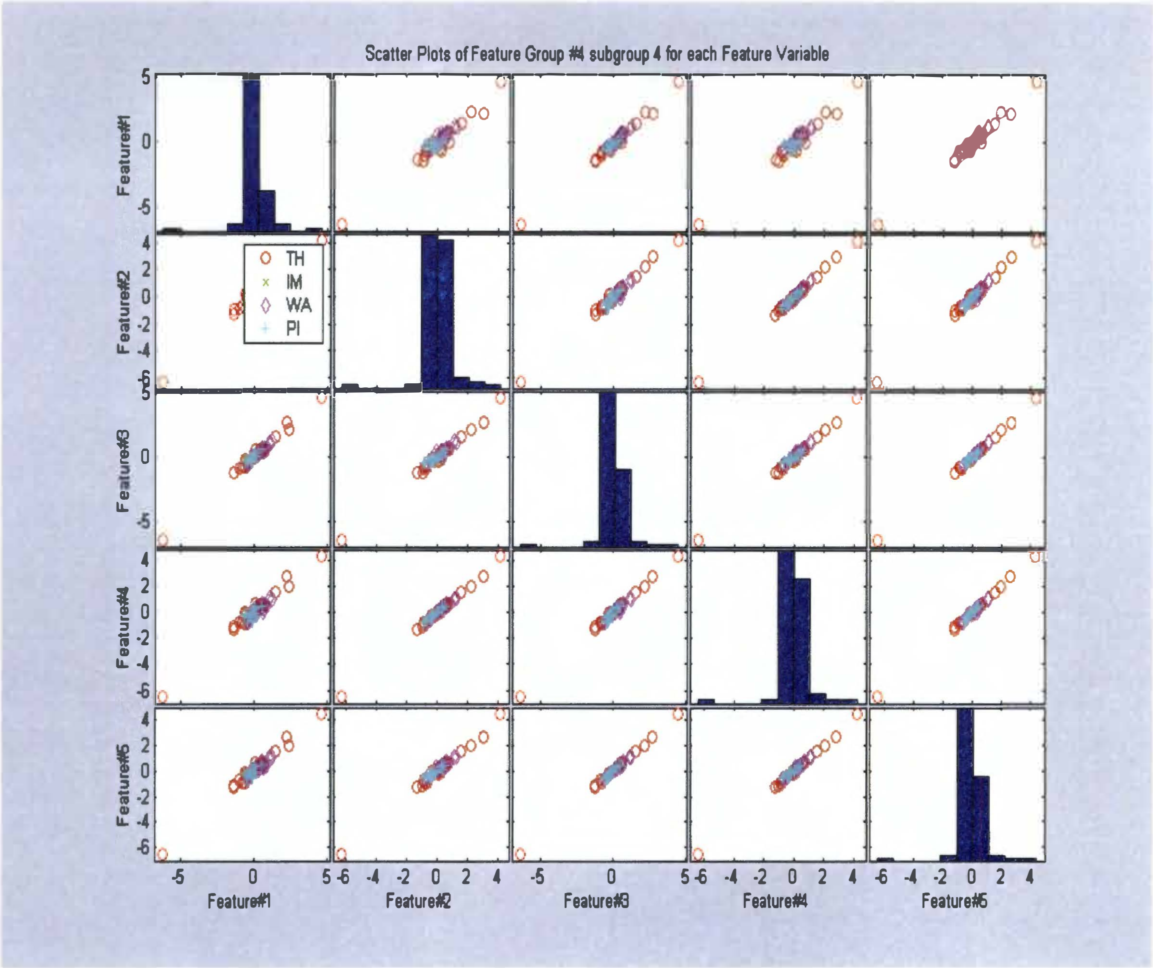

Figure F14. Scatter Plot of Raw Feature Family 4, Subgroup 4, of ECT Flaw DHR00PC051I066_5.

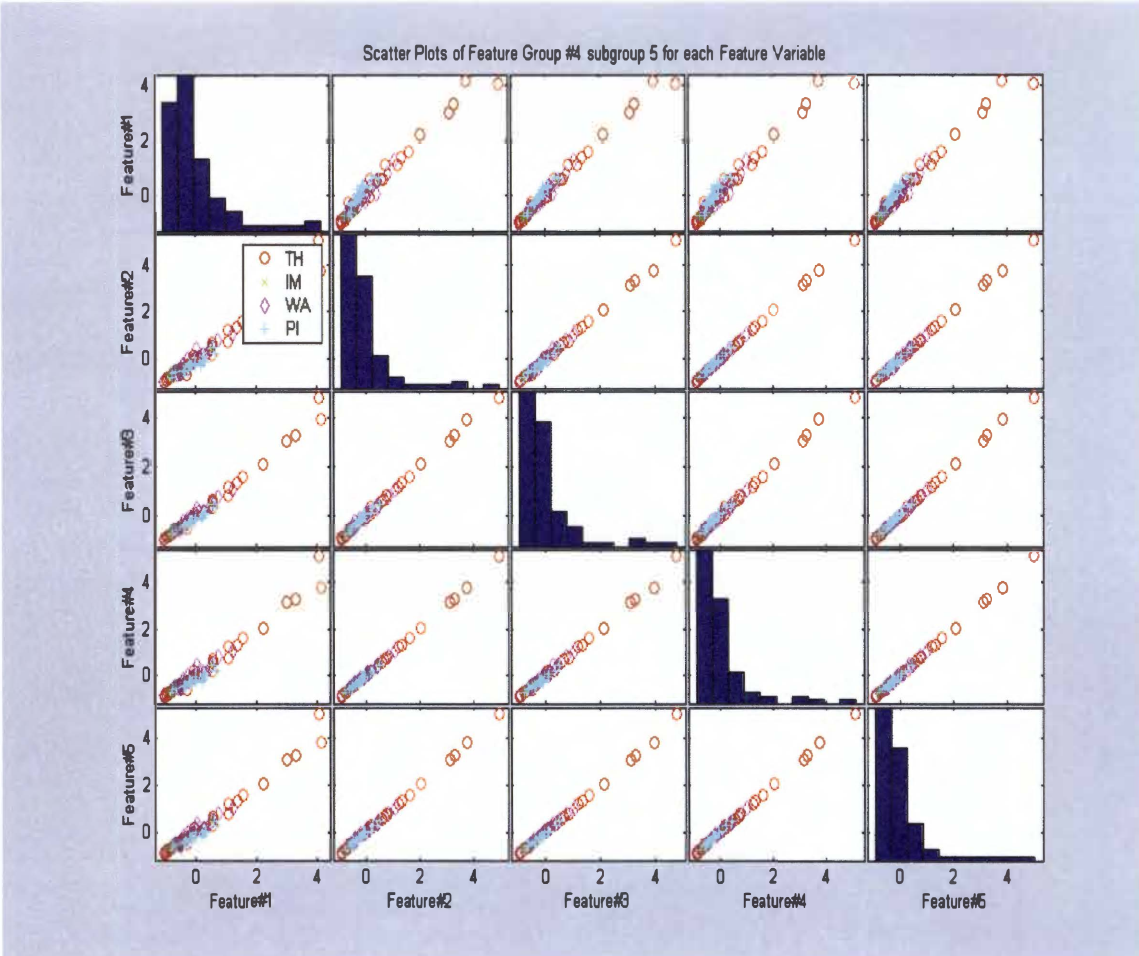

Figure F15. Scatter Plot of Raw Feature Family 4, Subgroup 5, of ECT Flaw DHR00PC051I066_5.

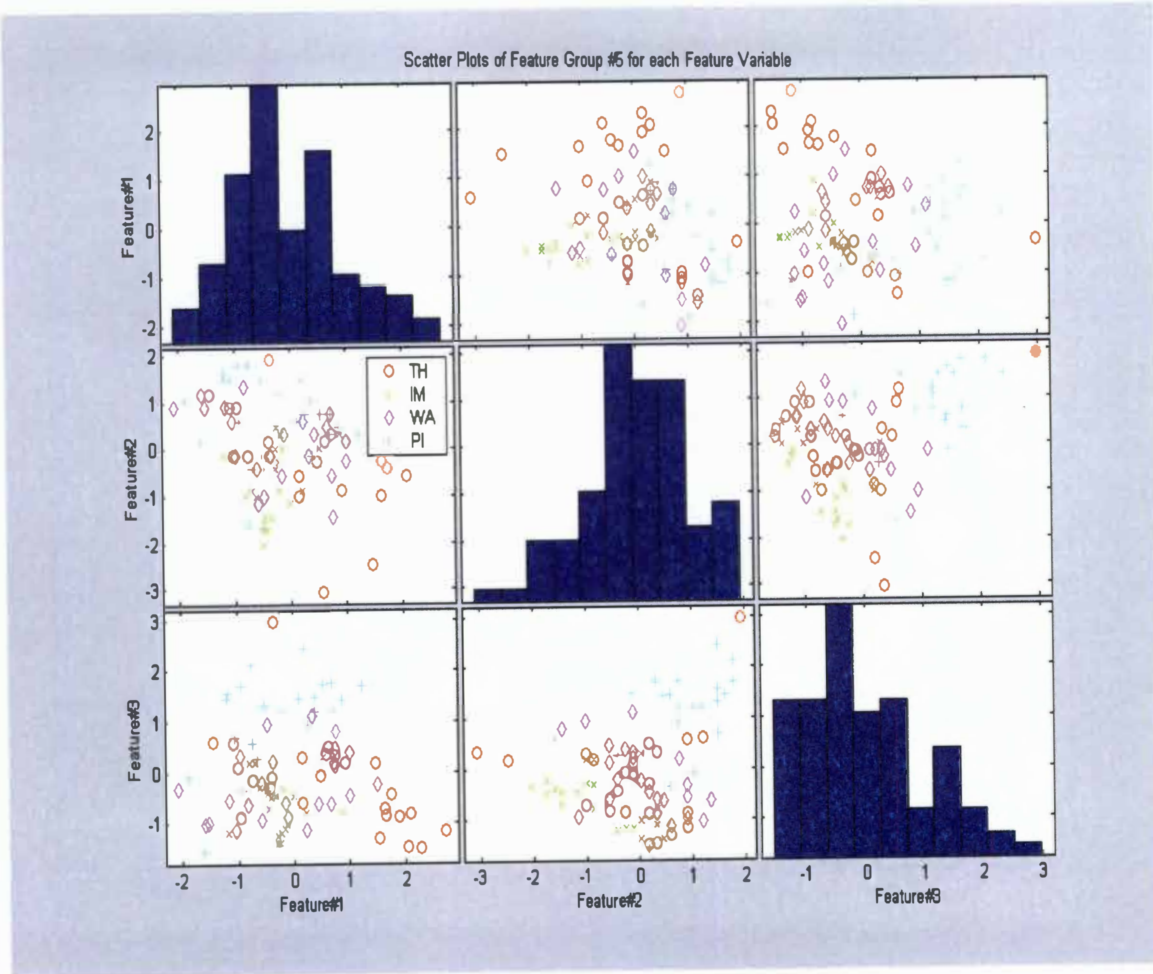

Figure F16. Scatter Plot of Raw Feature Family 5 of ECT Flaw DHR00PC051I066_5.

Percent Explained for TR Matrix =

31.721686

26.634286

11.331735

7.747438

7.094627

4.490173

2.888783

2.127487

1.645217

1.258826

1.046744

0.811940

0.446563

0.243495

0.169424

0.158936

0.094925

0.047941

0.025805

0.013967

Input the number of PC''s to retain. 15

Percent Explained for kept PCs = 99.658426

Input TR run number (actually a letter; a through z). z

Does user want to view PCA data for TR Feature Matrix, enter "y"es. y

Does user want a "2"D or "3"D plot for multiple D data? 3

Figures F17 through F20 are generated.

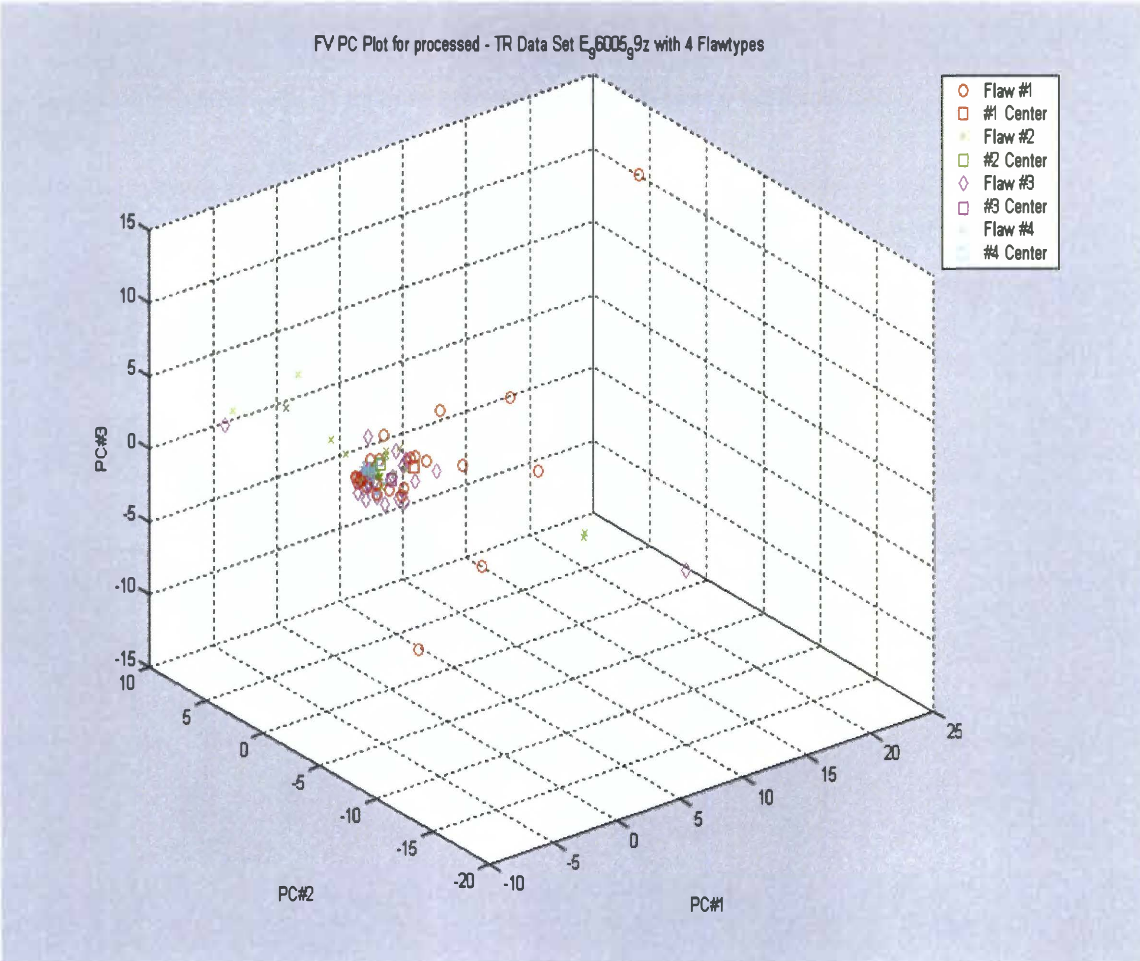

Figure F17. Plot of the First Three Major PCs of ECT Flaw DHR00PC051I066_5.

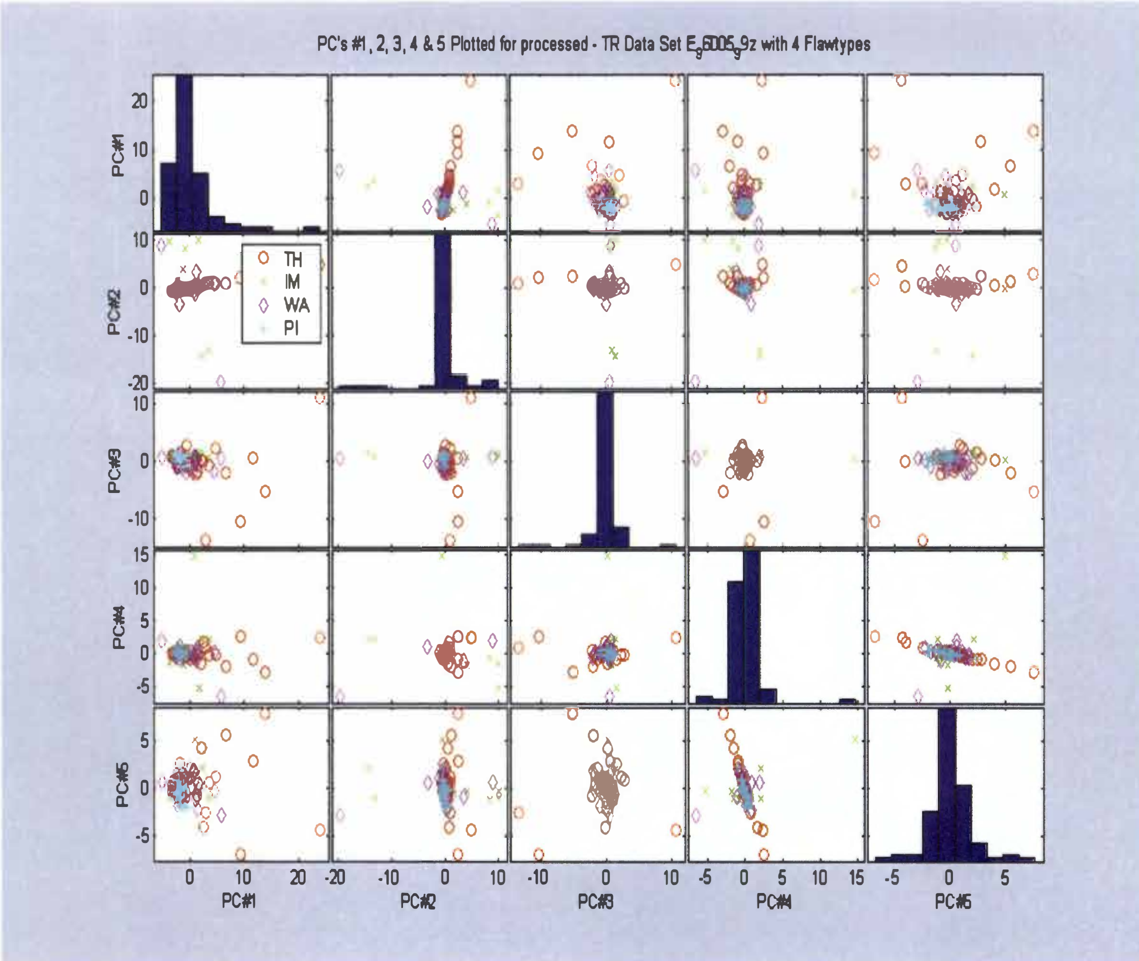

Figure F18. Plot of the First Five Major PCs of ECT Flaw DHR00PC051I066_5.

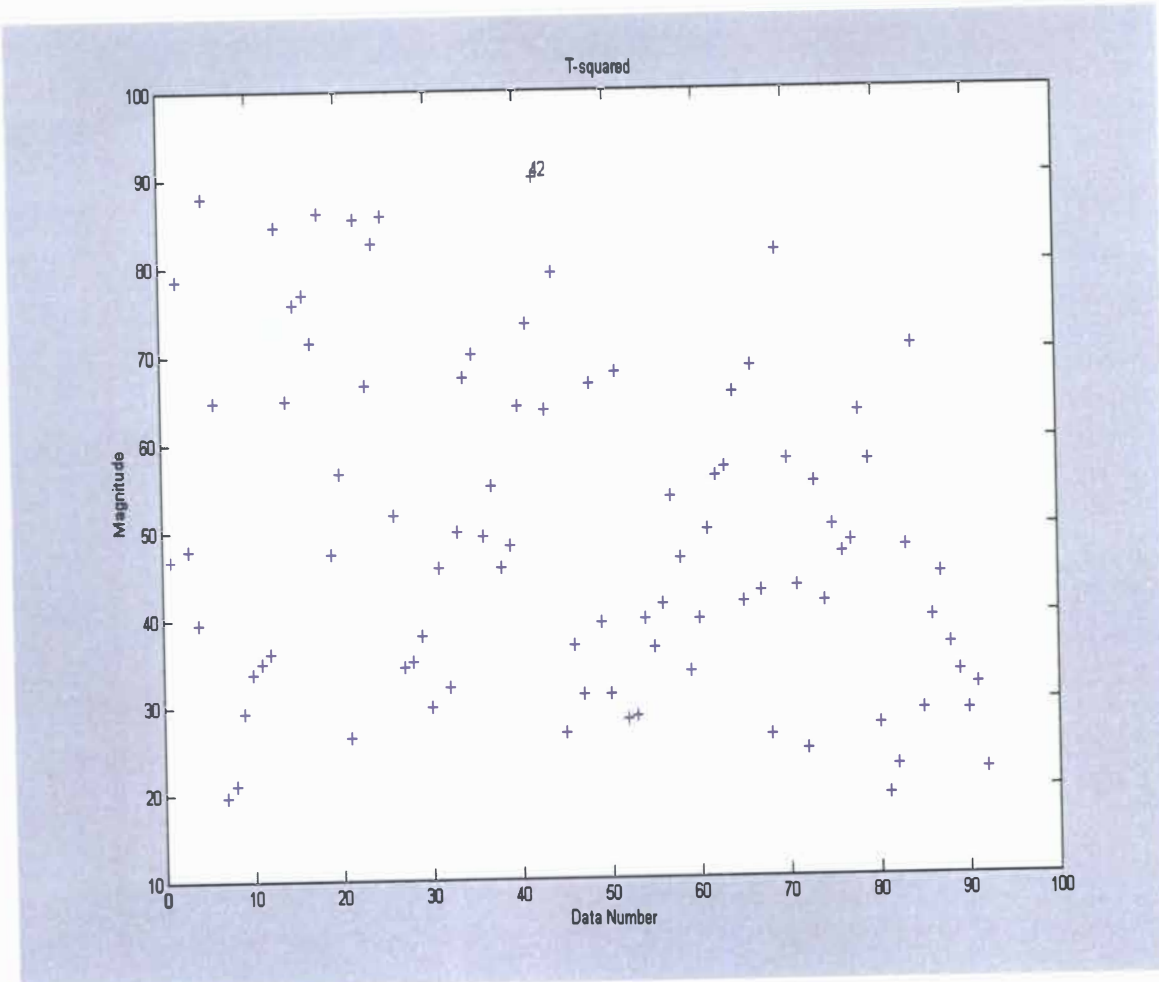

Figure F19. Plot of the T2 for All Data in TR_E_99a.

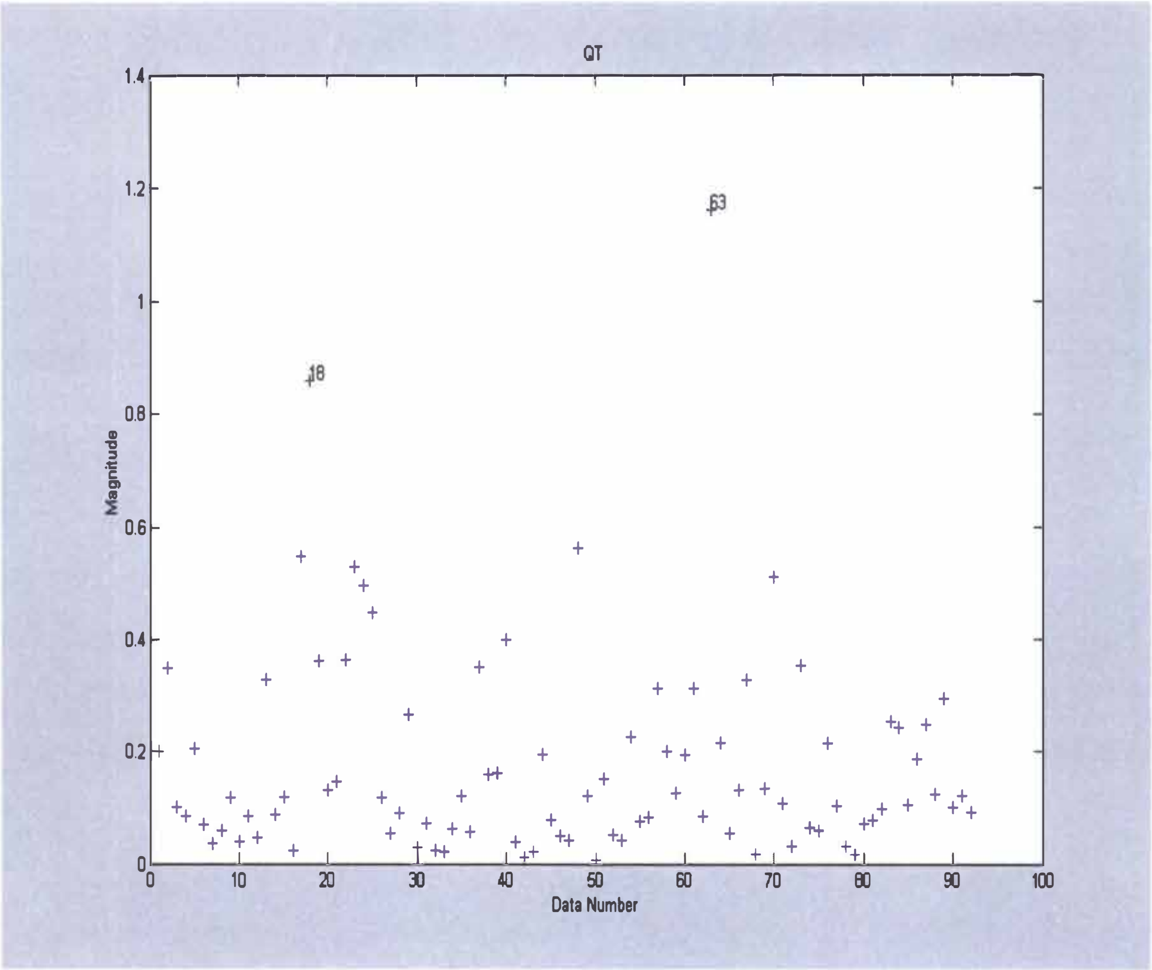

Figure F20. Plot of Q for All Data in TR_E_99a.

Procede with Classification of TR data ("y"es or "n"o). y

Does user want a "2"D or "3"D plot for multiple D data? 3

═══ Bayesian Classification Results for ═══

Data origin was E

Data Group was 96005

The uTR Data run number was 99

The TR Data run number was z

Does user want to classify using original features ("y"es or "n"o). n

Does user want to check a "s"ingle flaw from file or "a"ll? s

Enter which flaw (page #) to check against each FV data. 92

ClassPCA = 4

The Bhattacharyya Boundary (or maximum probablity of error percentage) = 0.14

The MATLAB output from this points was exactly as detailed in Appendix D.

═══ Neural Network Characterization Results for ═══

Data origin was E

Data Group was 96005

The Data run number was 99 z

═══ Correlation Analysis for Flawtype 1 ═══

CA1 =

0.7052	0.6900	0.4739
0.6696	0.7075	0.4654
-0.3036	0.4975	-0.1635

-0.0986	0.3694	-0.0445
0.1632	-0.2494	0.0962
-0.3982	0.2568	-0.0089
0.5827	-0.0321	-0.0108
-0.0023	-0.0384	-0.1748
0.1089	-0.0153	0.1041
0.6166	0.2352	0.2155
0.2680	-0.1114	0.6367
0.4193	0.0903	0.2114
-0.0389	-0.2948	0.1157
-0.0127	-0.0498	-0.0724
-0.2712	0.2401	0.0845
1.0000	0.0915	0.5775
0.0915	1.0000	0.3089
0.5775	0.3089	1.0000

═══ Neural Network Analysis for Flawtype 1 ═══

Number of neurons for the hidden layer (5). 7
Desire SSE goal (0.05). 0.05

TRAINBR, Epoch 0/200, SSE 367.926/0.05, SSW 33.3812, Grad 1.85e+002/1.00e-010, #Par 1.36e+002/136
TRAINBR, Epoch 10/200, SSE 3.59395/0.05, SSW 7.48435, Grad 2.47e+000/1.00e-010, #Par 3.47e+001/136
TRAINBR, Epoch 20/200, SSE 2.59448/0.05, SSW 9.10421, Grad 1.23e+000/1.00e-010, #Par 3.84e+001/136
TRAINBR, Epoch 30/200, SSE 2.30931/0.05, SSW 9.8587, Grad 9.06e-001/1.00e-010, #Par 3.95e+001/136
TRAINBR, Epoch 40/200, SSE 2.01141/0.05, SSW 10.898, Grad 8.20e-001/1.00e-010, #Par 4.13e+001/136
TRAINBR, Epoch 50/200, SSE 1.47584/0.05, SSW 13.2834, Grad 7.03e-001/1.00e-010, #Par 4.49e+001/136

TRAINBR, Epoch 60/200, SSE 0.821089/0.05, SSW 17.3803, Grad 5.25e-001/1.00e-010, #Par 5.03e+001/136

TRAINBR, Epoch 70/200, SSE 0.344223/0.05, SSW 21.7771, Grad 3.15e-001/1.00e-010, #Par 5.85e+001/136

TRAINBR, Epoch 80/200, SSE 0.179035/0.05, SSW 23.5168, Grad 3.45e-001/1.00e-010, #Par 6.23e+001/136

TRAINBR, Epoch 90/200, SSE 0.0577469/0.05, SSW 26.4792, Grad 2.03e-001/1.00e-010, #Par 6.66e+001/136

TRAINBR, Epoch 91/200, SSE 0.0494963/0.05, SSW 26.7537, Grad 1.61e-001/1.00e-010, #Par 6.75e+001/136

TRAINBR, Performance goal met.

Target Flaw characterization vector for flawtype # 1

T =

Columns 1 through 7

9.0000	40.0000	9.0000	23.0000	60.0000	12.0000	22.0000
195.0000	75.0000	75.0000	75.0000	360.0000	45.0000	45.0000
1.4000	1.0000	1.0000	1.0000	3.0000	0.3300	0.3000

Columns 8 through 14

38.0000	46.0000	20.0000	30.0000	40.0000	50.0000	57.0000
45.0000	45.0000	90.0000	90.0000	90.0000	90.0000	90.0000
0.4500	0.3550	3.0000	3.0000	3.0000	3.0000	3.0000

Columns 15 through 21

66.0000	80.0000	90.0000	100.0000	30.0000	38.0000	44.0000
90.0000	90.0000	90.0000	90.0000	90.0000	90.0000	90.0000
3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000

Columns 22 through 25

60.0000	66.0000	88.0000	80.0000
90.0000	90.0000	90.0000	90.0000
3.0000	3.0000	3.0000	3.0000

NN Flaw characterization vector for flawtype # = 1

Y =

Columns 1 through 7

8.7877	39.4014	8.7552	24.2352	60.2877	11.8228	27.3298
191.9851	76.1551	75.4915	72.7176	358.0414	46.4620	42.3095
1.4078	1.0147	0.9771	1.0997	2.9936	0.3468	0.3005

Columns 8 through 14

38.1854	40.8894	19.7705	29.9228	39.7991	50.3060	55.6477
47.7518	47.2349	88.6879	95.4072	91.1491	91.0565	87.4596
0.4710	0.3318	2.9359	3.0109	3.0054	2.9823	2.9787

Columns 15 through 21

67.3235	78.7898	89.4428	100.5608	30.2425	37.8442	43.8455
94.0945	88.1485	89.1663	90.0727	90.0044	91.4789	88.2982
2.9932	3.0160	2.9995	3.0133	2.9981	2.9873	2.9589

Columns 22 through 25

60.6071	66.8091	87.8160	79.1022
89.9614	83.3287	95.1711	89.5176

2.9922   2.9737   3.0102   3.0044

The MSE between Tn and Yn for flawtype # 1 = 0.0007

Correlation Coeff between T and Y for flawtype # 1 variable 1 = 0.9981

Correlation Coeff between T and Y for flawtype # 1 variable 2 = 0.9991

Correlation Coeff between T and Y for flawtype # 1 variable 3 = 0.9997

Does user want to save the generated NN and info ("y"es or "n"o)? y

NN char run number (usually 1, 2 ... with 5a1 being full run ID). 7

Figures F21 through F23 are generated.

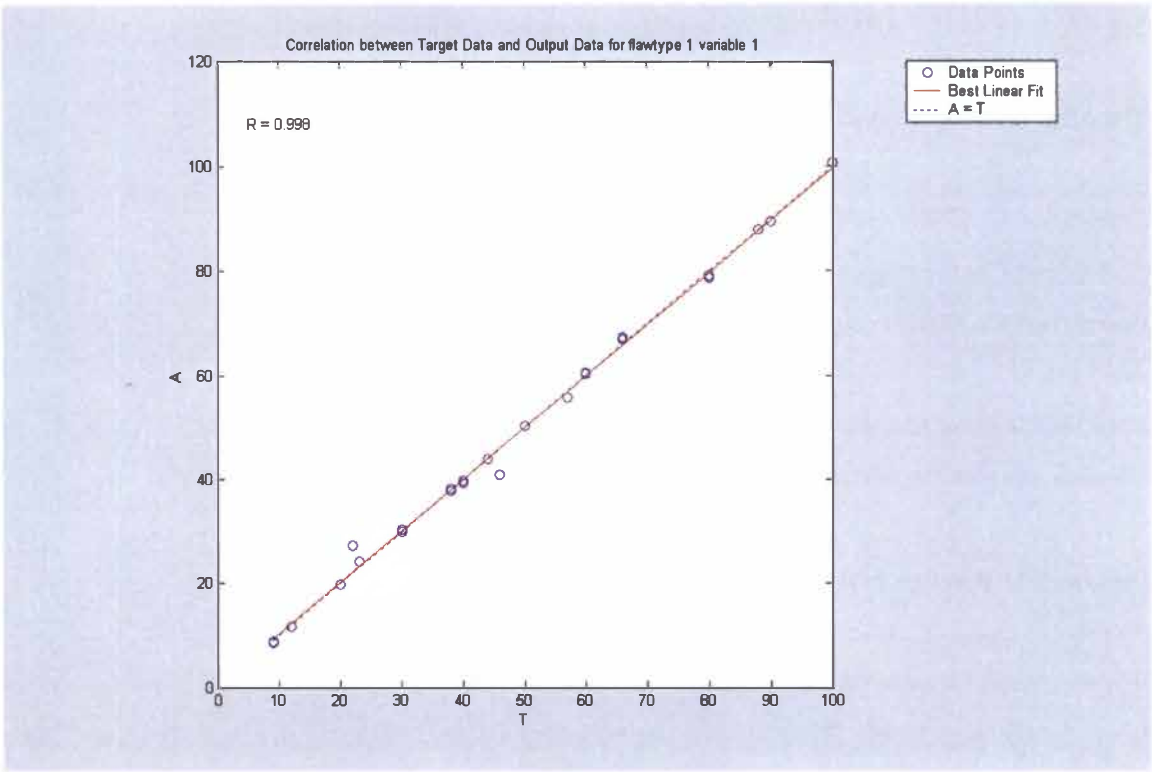

Figure F21. Plot of the Tn vs Yn for Characteristic1 for Flaw-type 1 (with Regression Information) for All Data in TR_E_99a.

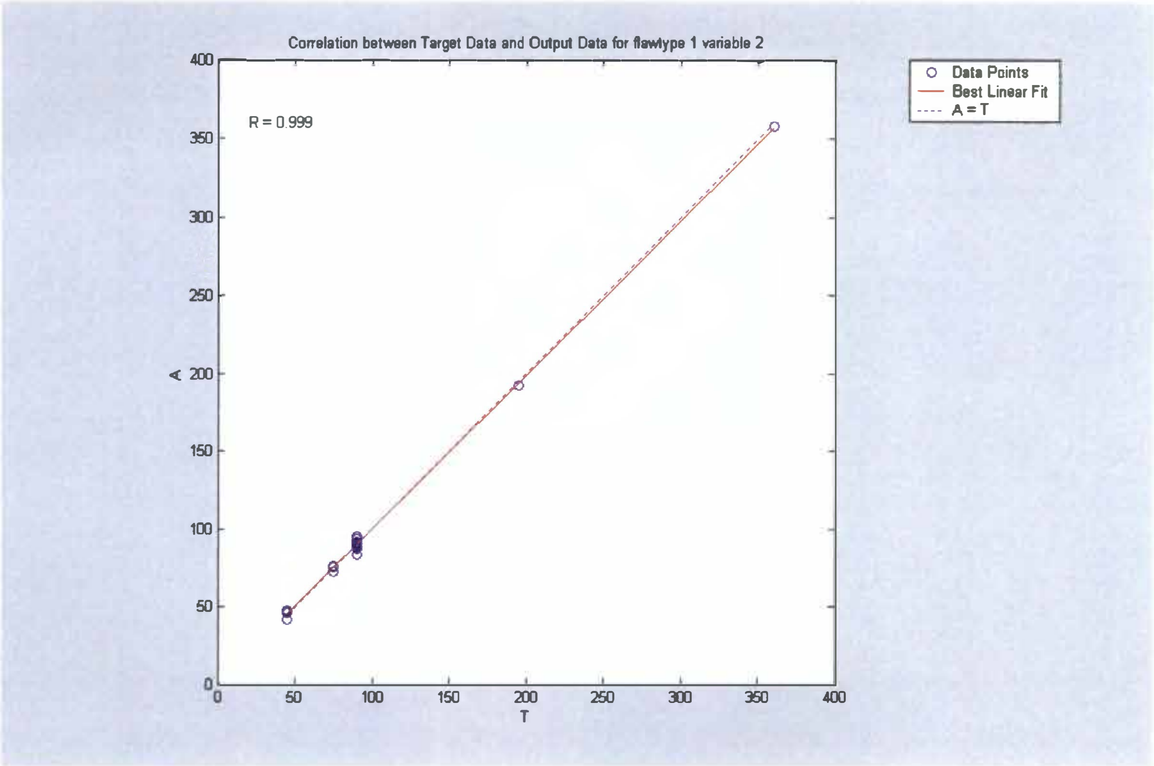

Figure F22. Plot of the Tn vs Yn for Characteristic2 for Flaw-type 1 (with Regression Information) for All Data in TR_E_99a.

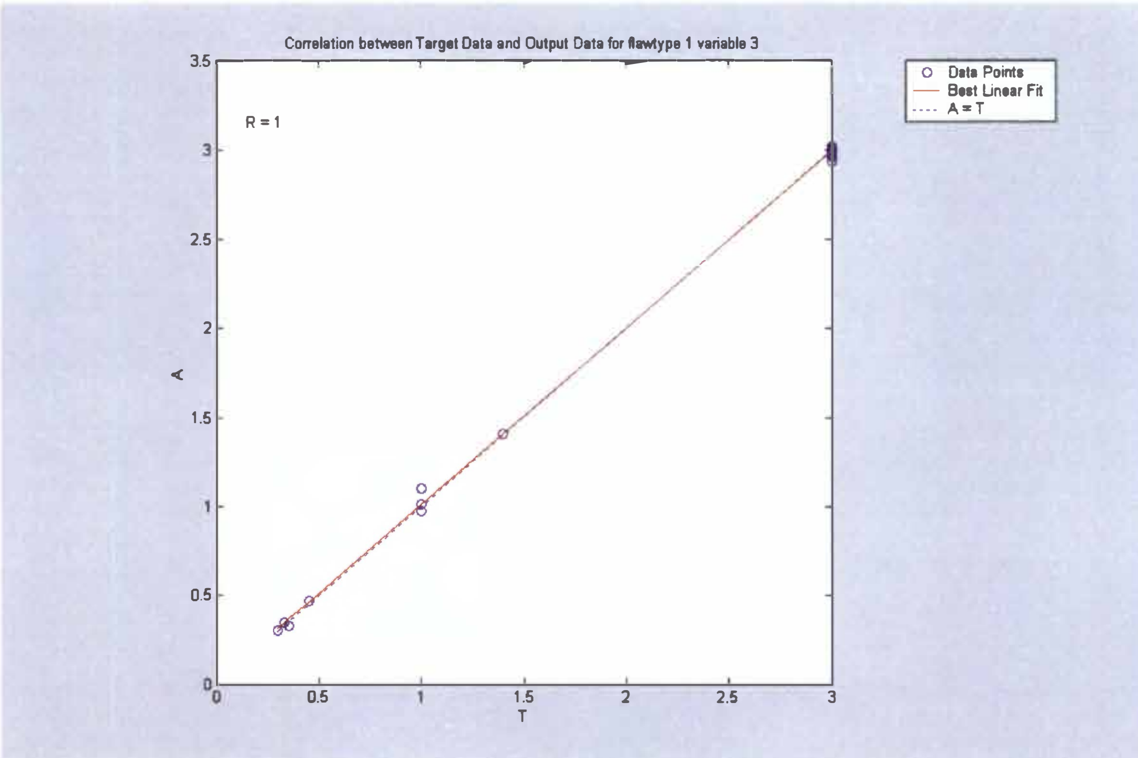

Figure F23. Plot of the Tn vs Yn for Characteristic3 for Flaw-type 1 (with Regression Information) for All Data in TR_E_99a.

═══ Correlation Analysis for Flawtype 2 ═══

CA2 =

0.5881	-0.3491	0.2973
-0.1108	-0.0145	-0.0473
0.3212	-0.3714	0.4313
0.2019	-0.0563	0.2273
0.2863	-0.1671	0.5037
-0.6862	0.6603	-0.3678
0.0050	0.1402	-0.0987
0.2532	-0.0417	0.4722
-0.3000	0.0909	-0.4188
-0.2661	0.1728	-0.2676
0.4565	-0.2999	0.5258
-0.2139	0.4482	-0.0378
0.1444	0.0834	-0.0024
-0.3258	0.4884	-0.0169
0.0768	-0.0750	0.2026
1.0000	-0.7271	0.4758
-0.7271	1.0000	-0.4864
0.4758	-0.4864	1.0000

═══ Neural Network Analysis for Flawtype 2 ═══

Number of neurons for the hidden layer (5). 7
Desire SSE goal (0.05). 0.05

TRAINBR, Epoch 0/200, SSE 173.156/0.05, SSW 33.1112, Grad 1.35e+002/1.00e-010, #Par 1.36e+002/136
TRAINBR, Epoch 10/200, SSE 8.85904/0.05, SSW 2.03044, Grad 7.60e+000/1.00e-010, #Par 1.47e+001/136

TRAINBR, Epoch 20/200, SSE 1.78626/0.05, SSW 5.89654, Grad 1.13e+000/1.00e-010, #Par 3.03e+001/136
TRAINBR, Epoch 30/200, SSE 0.548795/0.05, SSW 10.5305, Grad 5.18e-001/1.00e-010, #Par 4.12e+001/136
TRAINBR, Epoch 40/200, SSE 0.069924/0.05, SSW 15.1878, Grad 8.65e-001/1.00e-010, #Par 5.26e+001/136
TRAINBR, Epoch 41/200, SSE 0.0424465/0.05, SSW 15.0677, Grad 1.59e-001/1.00e-010, #Par 5.33e+001/136
TRAINBR, Performance goal met.

Target Flaw characterization vector for flawtype # 2

T =

Columns 1 through 7

58.0000	63.0000	65.0000	68.0000	71.0000	73.0000	75.0000
0.0850	0.0860	0.0950	0.0980	0.0980	0.0930	0.0950
0.3450	0.2000	0.2290	0.2160	0.2240	0.2240	0.2230

Columns 8 through 14

76.0000	78.0000	79.0000	82.0000	84.0000	87.0000	87.0000
0.0900	0.0900	0.0820	0.0820	0.0730	0.0740	0.0700
0.2280	0.2290	0.3540	0.2270	0.2310	0.2390	0.2330

Columns 15 through 21

98.0000	95.0000	92.0000	76.0000	60.0000	60.0000	37.0000
0.0680	0.0780	0.0780	0.0910	0.0880	0.1962	0.2110
0.2660	0.3100	0.3320	0.2150	0.2700	0.2700	0.0754

NN Flaw characterization vector for flawtype # = 2

Y =

Columns 1 through 7

58.9398	61.7120	66.6127	68.4143	71.6642	72.5653	75.4228
0.0846	0.0906	0.0949	0.0955	0.0967	0.0924	0.0981
0.3405	0.2035	0.2295	0.2190	0.2180	0.2243	0.2273

Columns 8 through 14

75.4972	77.8533	79.0286	80.7119	82.9806	86.3088	87.4536
0.0930	0.0917	0.0822	0.0756	0.0760	0.0747	0.0703
0.2287	0.2324	0.3506	0.2245	0.2313	0.2364	0.2339

Columns 15 through 21

97.8593	94.7830	92.0590	76.2217	60.9373	59.3602	37.7817
0.0676	0.0775	0.0782	0.0899	0.0880	0.1950	0.2077
0.2650	0.3100	0.3322	0.2154	0.2685	0.2685	0.0756

The MSE between Tn and Yn for flawtype # 2 = 0.0007

Correlation Coeff between T and Y for flawtype # 2 variable 1 = 0.9987
Correlation Coeff between T and Y for flawtype # 2 variable 2 = 0.9980
Correlation Coeff between T and Y for flawtype # 2 variable 3 = 0.9991

Does user want to save the generated NN and info ("y"es or "n"o)? y
NN char run number (usually 1, 2 ... with 5a1 being full run ID). 7

Figures F24 through F26 are generated.

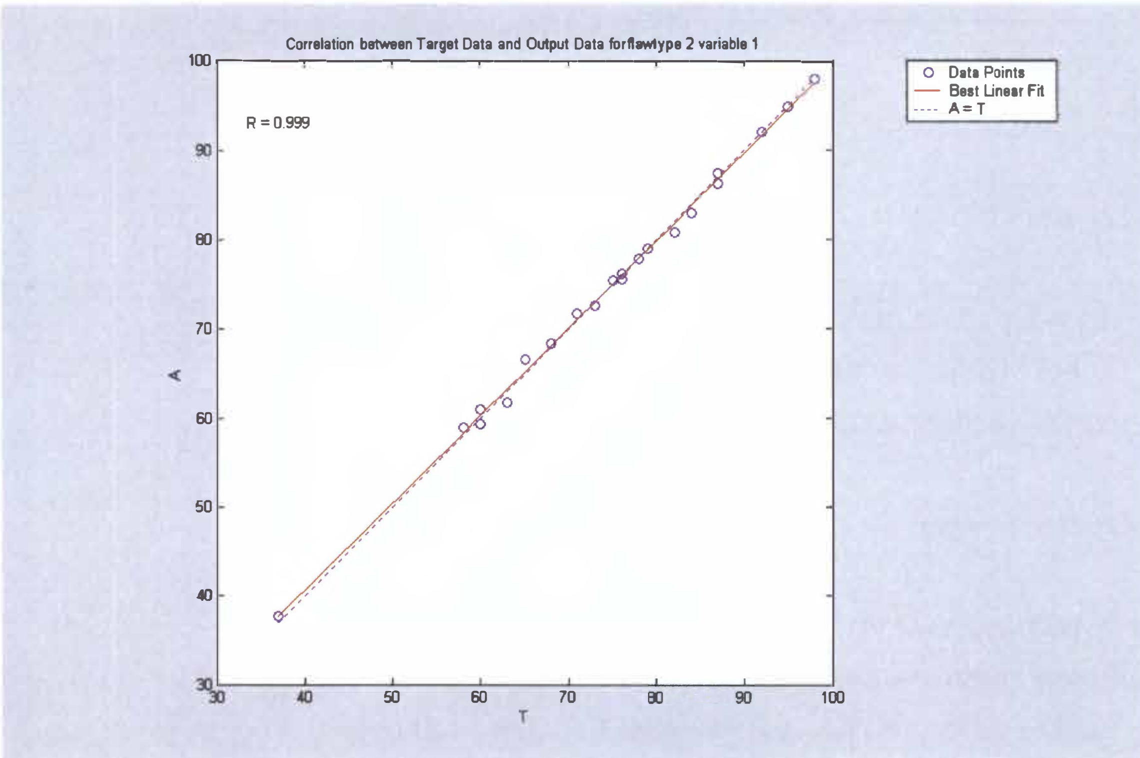

Figure F24. Plot of the Tn vs Yn for Characteristic 1 for Flaw-type 2 (with Regression Information) for All Data in TR_E_99a.

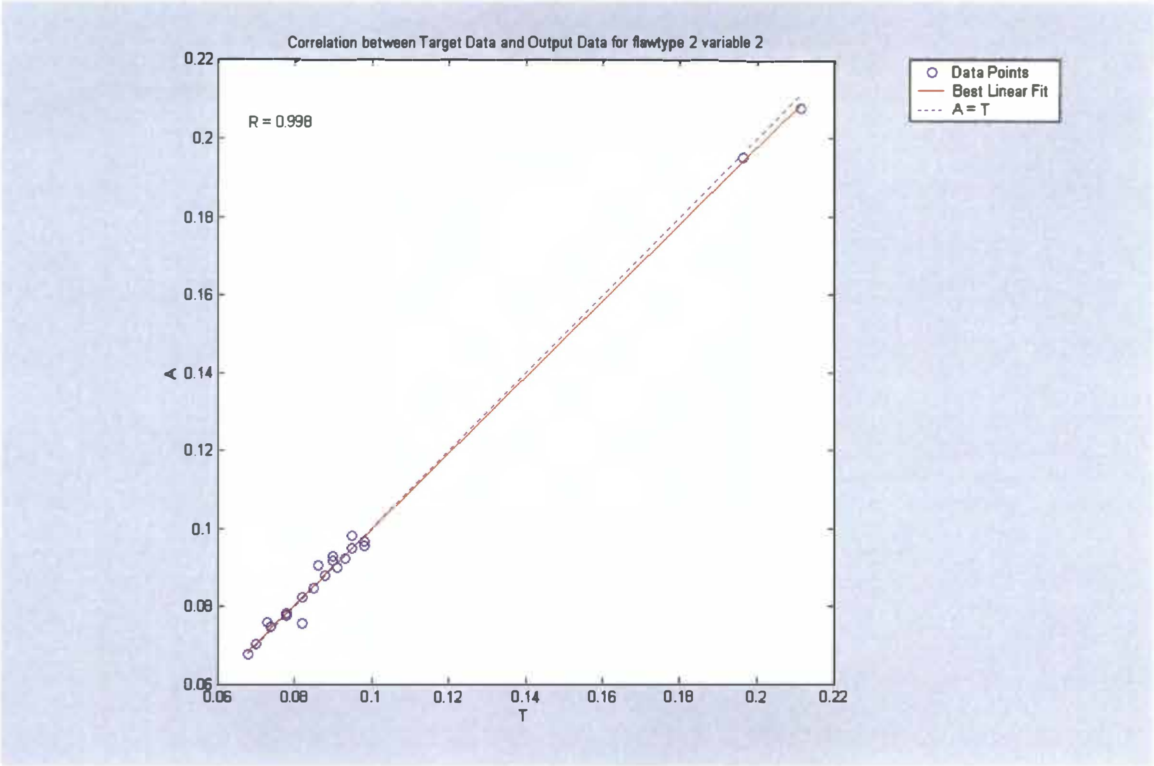

Figure F25. Plot of the Tn vs Yn for Characteristic 2 for Flaw-type 2 (with Regression Information) for All Data in TR_E_99a.

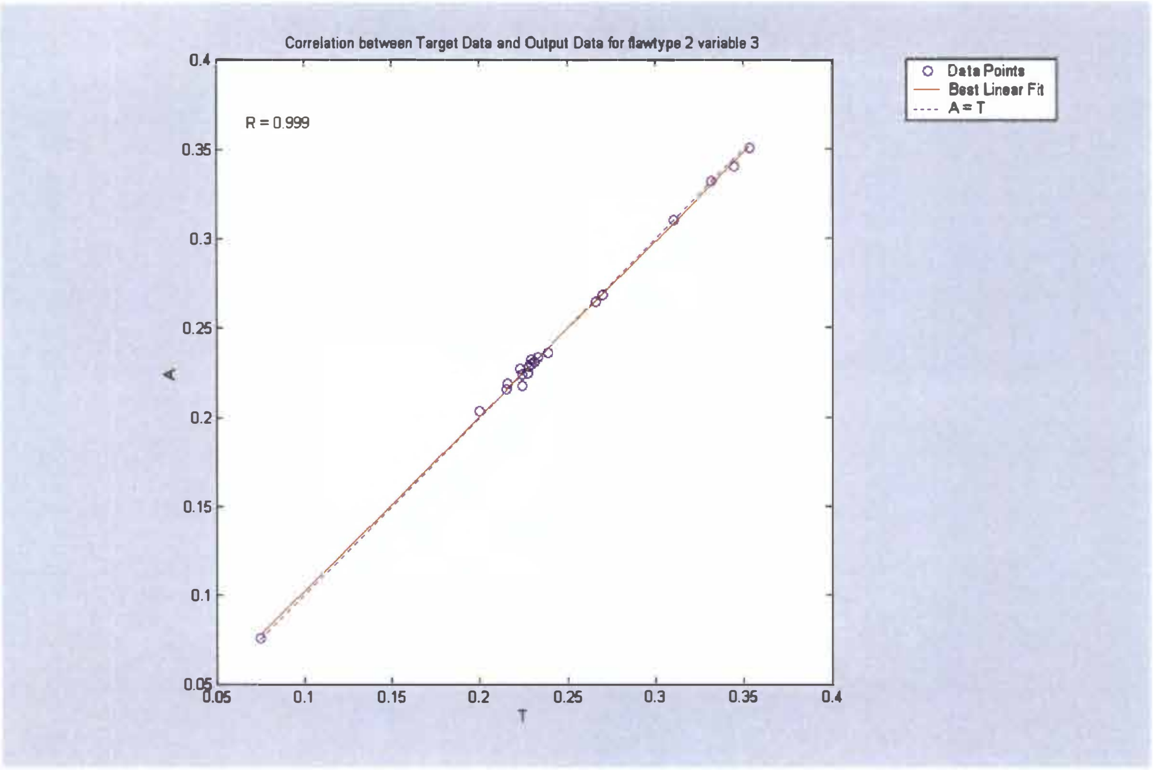

Figure F26. Plot of the Tn vs Yn for Characteristic 3 for Flaw-type 2 (with Regression Information) for All Data in TR_E_99a.

===== Correlation Analysis for Flawtype 3 =====

CA3 =

```
  0.8588   0.0000
 -0.2768   0.0000
 -0.0593        0
 -0.4158   0.0000
 -0.1101   0.0000
 -0.7902  -0.0000
  0.6935  -0.0000
 -0.3207   0.0000
  0.3296  -0.0000
 -0.6418   0.0000
 -0.1975   0.0000
 -0.3533   0.0000
  0.3366   0.0000
  0.0425  -0.0000
 -0.1497  -0.0000
  1.0000        0
       0   1.0000
```

===== Neural Network Analysis for Flawtype 3 =====

Number of neurons for the hidden layer (5). 7
Desire SSE goal (0.05). 0.05

Warning: Some maximums and minimums are equal. Those targets won't be transformed.
> In C:\matlabR12\toolbox\nnet\nnet\premnmx.m at line 77
  In C:\Patrick\eddym\NN_char.m at line 71
TRAINBR, Epoch 0/200, SSE 70.9952/0.05, SSW 31.1861, Grad 7.04e+001/1.00e-010, #Par 1.28e+002/128

TRAINBR, Epoch 10/200, SSE 1.05778/0.05, SSW 1.66063, Grad 4.18e+000/1.00e-010, #Par 1.32e+001/128

TRAINBR, Epoch 20/200, SSE 0.176181/0.05, SSW 2.7885, Grad 1.09e-001/1.00e-010, #Par 2.03e+001/128

TRAINBR, Epoch 29/200, SSE 0.0296906/0.05, SSW 4.87417, Grad 4.01e-001/1.00e-010, #Par 2.64e+001/128

TRAINBR, Performance goal met.

Warning: Some maximums and minimums are equal. Those inputs won't be transformed.

> In C:\matlabR12\toolbox\nnet\nnet\postmnmx.m at line 59

In C:\Patrick\eddym\NN_char.m at line 89

Target Flaw characterization vector for flawtype # 3

T =

Columns 1 through 7

```
17.0000   61.0000   10.0000   26.0000   89.0000   37.0000   46.0000
 0.2750    0.2750    0.2750    0.2750    0.2750    0.2750    0.2750
```

Columns 8 through 14

```
79.0000   37.0000   50.0000   84.0000   55.0000   61.0000   67.0000
 0.2750    0.2750    0.2750    0.2750    0.2750    0.2750    0.2750
```

Columns 15 through 21

```
26.0000   70.0000   90.0000   74.0000   46.0000   80.0000   50.0000
 0.2750    0.2750    0.2750    0.2750    0.2750    0.2750    0.2750
```

Columns 22 through 24

32.0000	90.0000	70.0000
0.2750	0.2750	0.2750

NN Flaw characterization vector for flawtype # = 3

Y =

Columns 1 through 7

20.2465	61.1859	12.2306	26.0512	88.1133	34.4329	47.1371
0.2748	0.2749	0.2746	0.2750	0.2753	0.2746	0.2746

Columns 8 through 14

78.3384	36.2568	49.9571	84.1740	57.4455	62.6462	66.1788
0.2751	0.2748	0.2748	0.2751	0.2751	0.2748	0.2749

Columns 15 through 21

27.4545	70.4676	90.4883	74.4101	44.6013	80.1254	51.2731
0.2745	0.2749	0.2751	0.2751	0.2752	0.2755	0.2749

Columns 22 through 24

32.5809	90.0424	67.5066
0.2744	0.2736	0.2748

The MSE between Tn and Yn for flawtype # 3 = 0.0006

Correlation Coeff between T and Y for flawtype # 3 variable 1 = 0.9984
Warning: Rank deficient, rank = 1 tol = 2.6107e-014.
> In C:\matlabR12\toolbox\nnet\nnet\postreg.m at line 57
In C:\Patrick\eddym\NN_char.m at line 101

Warning: Divide by zero.

> In C:\matlabR12\toolbox\nnet\nnet\postreg.m at line 77

In C:\Patrick\eddym\NN_char.m at line 101

Correlation Coeff between T and Y for flawtype # 3 variable 2 = -Inf

Does user want to save the generated NN and info ("y"es or "n"o)? y

NN char run number (usually 1, 2 ... with 5a1 being full run ID). 7

Figures F27 and F28 are generated.

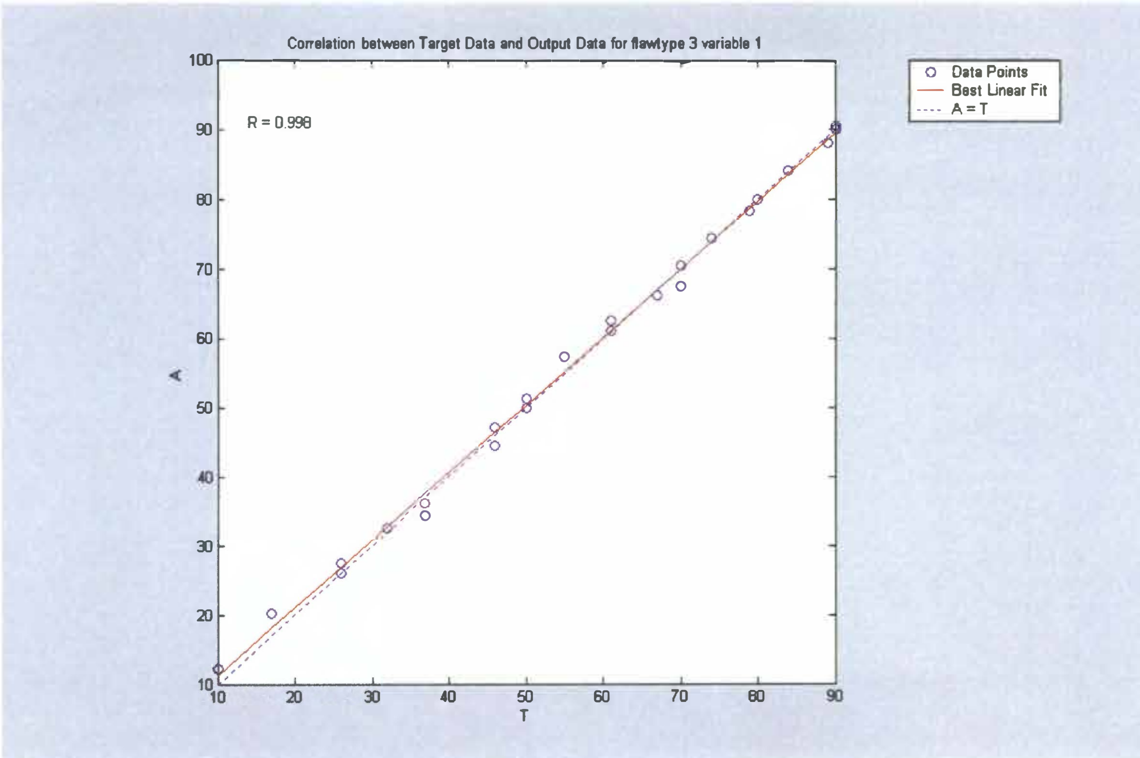

Figure F27. Plot of the Tn vs Yn for Characteristic 1 for Flaw-type 3 (with Regression Information) for All Data in TR_E_99a.

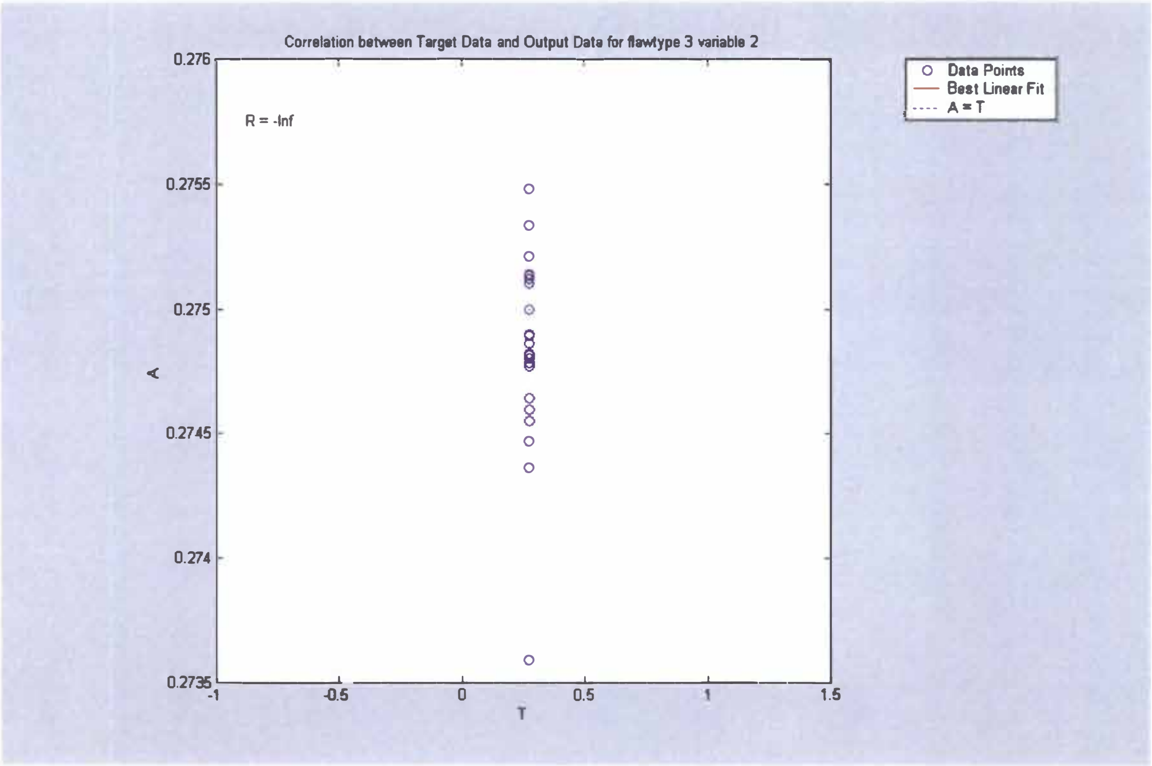

Figure F28. Plot of the Tn vs Yn for Characteristic 2 for Flaw-type 3 (with Regression Information) for All Data in TR_E_99a.

═══ Correlation Analysis for Flawtype 4 ═══

CA4 =

0.5167	0.2671	0.3746
0.3657	0.2021	0.1875
-0.2555	-0.2702	-0.4200
-0.1643	-0.0969	0.1937
0.1572	0.1360	-0.1060
-0.2524	-0.3377	-0.3395
-0.0075	-0.1642	-0.0339
-0.6654	-0.3910	-0.4161
0.8033	0.4386	0.1723
-0.3731	-0.0549	0.1457
-0.0153	0.0057	0.4262
0.2793	0.2733	-0.1246
0.1412	0.0736	-0.0308
-0.0824	0.0820	0.0152
0.1557	0.0484	-0.2163
1.0000	0.3808	0.2667
0.3808	1.0000	0.2501
0.2667	0.2501	1.0000

═══ Neural Network Analysis for Flawtype 4 ═══

Number of neurons for the hidden layer (5). 7
Desire SSE goal (0.05). 0.05

TRAINBR, Epoch 0/200, SSE 226.608/0.05, SSW 37.0335, Grad 1.29e+002/1.00e-010, #Par 1.36e+002/136
TRAINBR, Epoch 10/200, SSE 11.0186/0.05, SSW 1.82827, Grad 3.83e+000/1.00e-010, #Par 1.38e+001/136

TRAINBR, Epoch 20/200, SSE 10.3251/0.05, SSW 2.02656, Grad 2.92e+000/1.00e-010, #Par 1.51e+001/136

TRAINBR, Epoch 30/200, SSE 10.0814/0.05, SSW 2.10868, Grad 2.52e+000/1.00e-010, #Par 1.56e+001/136

TRAINBR, Epoch 40/200, SSE 9.97086/0.05, SSW 2.15227, Grad 2.35e+000/1.00e-010, #Par 1.60e+001/136

TRAINBR, Epoch 50/200, SSE 9.90652/0.05, SSW 2.18122, Grad 2.27e+000/1.00e-010, #Par 1.62e+001/136

TRAINBR, Epoch 60/200, SSE 9.87129/0.05, SSW 2.19831, Grad 2.23e+000/1.00e-010, #Par 1.63e+001/136

TRAINBR, Epoch 70/200, SSE 9.85417/0.05, SSW 2.2069, Grad 2.21e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 80/200, SSE 9.84603/0.05, SSW 2.21104, Grad 2.20e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 90/200, SSE 9.8421/0.05, SSW 2.21306, Grad 2.20e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 100/200, SSE 9.84016/0.05, SSW 2.21407, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 110/200, SSE 9.83918/0.05, SSW 2.21459, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 120/200, SSE 9.83867/0.05, SSW 2.21486, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 130/200, SSE 9.83841/0.05, SSW 2.215, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 140/200, SSE 9.83827/0.05, SSW 2.21508, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 150/200, SSE 9.83819/0.05, SSW 2.21512, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 160/200, SSE 9.83815/0.05, SSW 2.21514, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 170/200, SSE 9.83813/0.05, SSW 2.21516, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136

TRAINBR, Epoch 180/200, SSE 9.83812/0.05, SSW 2.21516, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136
TRAINBR, Epoch 190/200, SSE 9.83811/0.05, SSW 2.21517, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136
TRAINBR, Epoch 200/200, SSE 9.83811/0.05, SSW 2.21517, Grad 2.19e+000/1.00e-010, #Par 1.64e+001/136
TRAINBR, Maximum epoch reached.

Target Flaw characterization vector for flawtype # 4

T =

Columns 1 through 7

30.0000	47.0000	46.0000	29.0000	29.0000	42.0000	37.0000
0.0850	0.0650	0.0650	0.0550	0.0550	0.0800	0.0750
0.0300	0.0650	0.0350	0.0500	0.0500	0.0580	0.0720

Columns 8 through 14

47.0000	44.0000	37.0000	47.0000	44.0000	62.0000	67.0000
0.0800	0.1000	0.0450	0.1150	0.1150	0.1000	0.1400
0.0450	0.1650	0.0450	0.0900	0.0450	0.1100	0.0900

Columns 15 through 21

62.0000	67.0000	44.0000	77.0000	53.0000	44.0000	77.0000
0.1000	0.1400	0.1450	0.0850	0.0850	0.1450	0.0850
0.1100	0.0900	0.0400	0.0500	0.0550	0.0400	0.0500

Column 22

53.0000

0.0850
0.0550

NN Flaw characterization vector for flawtype # = 4

Y =

Columns 1 through 7

36.0383	45.5713	43.9003	40.2896	33.2587	52.9112	45.1787
0.0750	0.0940	0.0801	0.0832	0.0683	0.0949	0.0845
0.0477	0.0743	0.0457	0.0820	0.0660	0.0554	0.0471

Columns 8 through 14

38.0308	41.2692	38.9532	46.1087	39.2946	59.6988	64.0552
0.0753	0.0832	0.0761	0.0937	0.0780	0.1080	0.1156
0.0440	0.0784	0.0483	0.0774	0.0539	0.0930	0.0756

Columns 15 through 21

60.8648	67.7177	53.8559	60.7949	50.3789	54.7772	63.1460
0.1103	0.1203	0.0980	0.1061	0.0914	0.1009	0.1104
0.0904	0.0871	0.0675	0.0643	0.0638	0.0681	0.0697

Column 22

51.7842
0.0950
0.0679

The MSE between Tn and Yn for flawtype # 4 = 0.1491

Correlation Coeff between T and Y for flawtype # 4 variable 1 = 0.8594

Correlation Coeff between T and Y for flawtype # 4 variable 2 = 0.6047

Correlation Coeff between T and Y for flawtype # 4 variable 3 = 0.6434

Does user want to save the generated NN and info ("y"es or "n"o)? y

NN char run number (usually 1, 2 ... with 5a1 being full run ID). 7

>>

Figures F29 through F31 are generated.

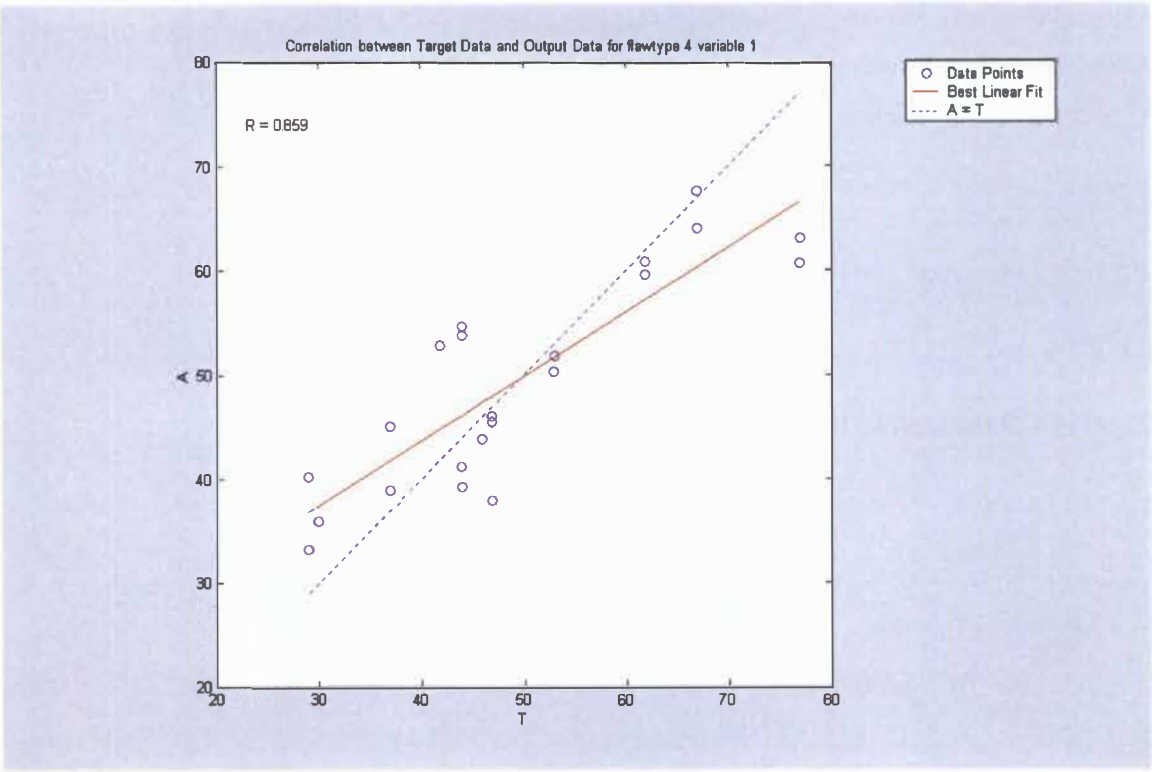

Figure F29. Plot of the Tn vs Yn for Characteristic 1 for Flaw-type 4 (with Regression Information) for All Data in TR_E_99a.

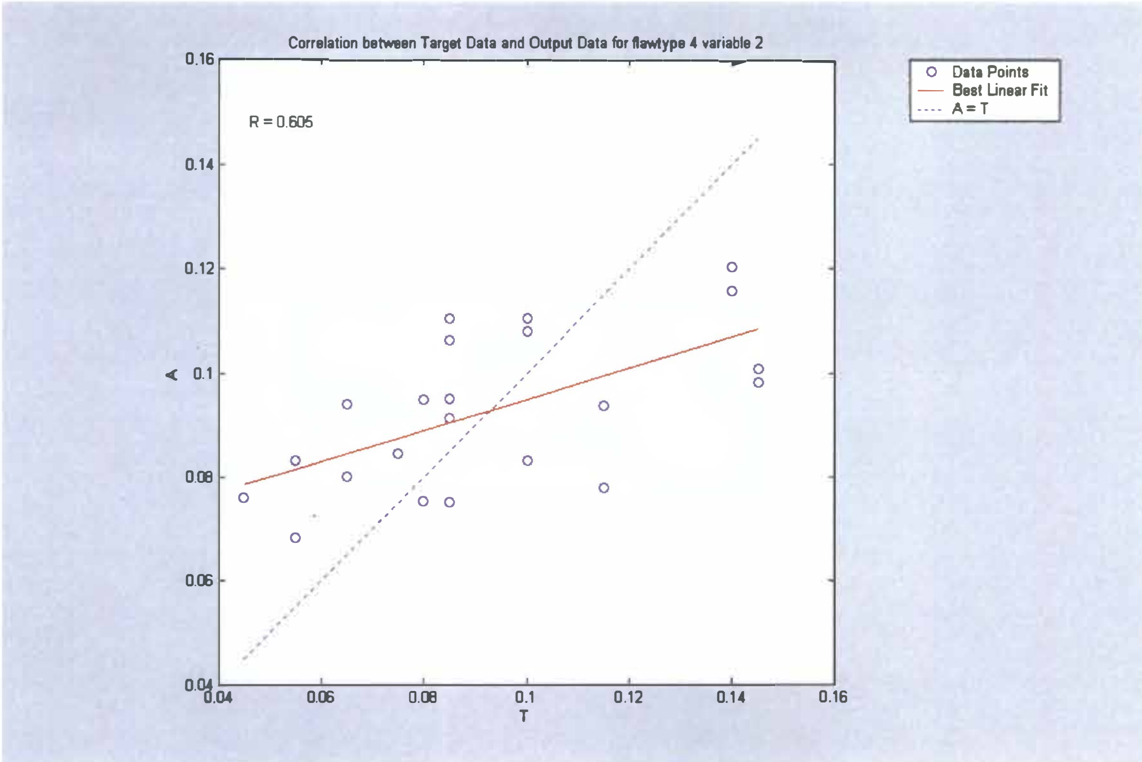

Figure F30. Plot of the Tn vs Yn for Characteristic 2 for Flaw-type 4 (with Regression Information) for All Data in TR_E_99a.

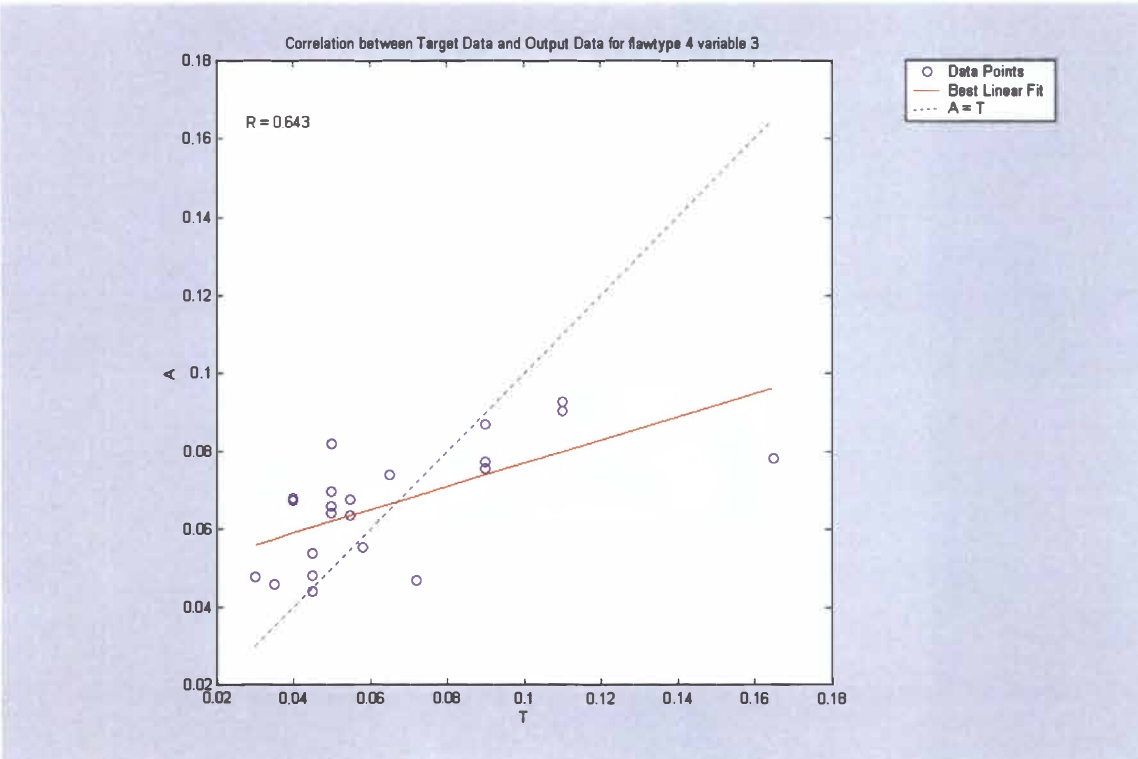

Figure F31. Plot of the Tn vs Yn for Characteristic 3 for Flaw-type 4 (with Regression Information) for All Data in TR_E_99a.

Checking a Flaw

>> eddyc

Is this "P"DD or "E"TSS data. E

Enter Manufacturer of Steam Generator (B, C or W) or ETSS #. 96001

Input ECT saved data filename (ex. T24b01_T080R025_1, no .mat needed). DHR00BC066I006_1

Is this a "S"aved data file or a command "W"indow data file? S

========== ETSS or PDD Input Information ==========

The origin (E = ETSS or P = PDD) of the data was E

The EC Data filename was DHR00BC066I006_1

The Steam Generator type or ETSS # was 96001

The PDD or ETSS location was 657 , doublecheck flaw location for the given filename!

The PDD or ETSS Flaw Magnitude was 3.9498

The PDD or ETSS Phase Angle was 98.503

======= ETSS or PDD Flaw Classification and Characterizations =======

The PDD or ETSS Flaw Type was TH

The PDD or ETSS Percent Thru-wall was 57

ETSS characteristics = 90 3

Does the data appear to be correct ("y"es or "n"o)? y

Figures F32 through F34 are generated.

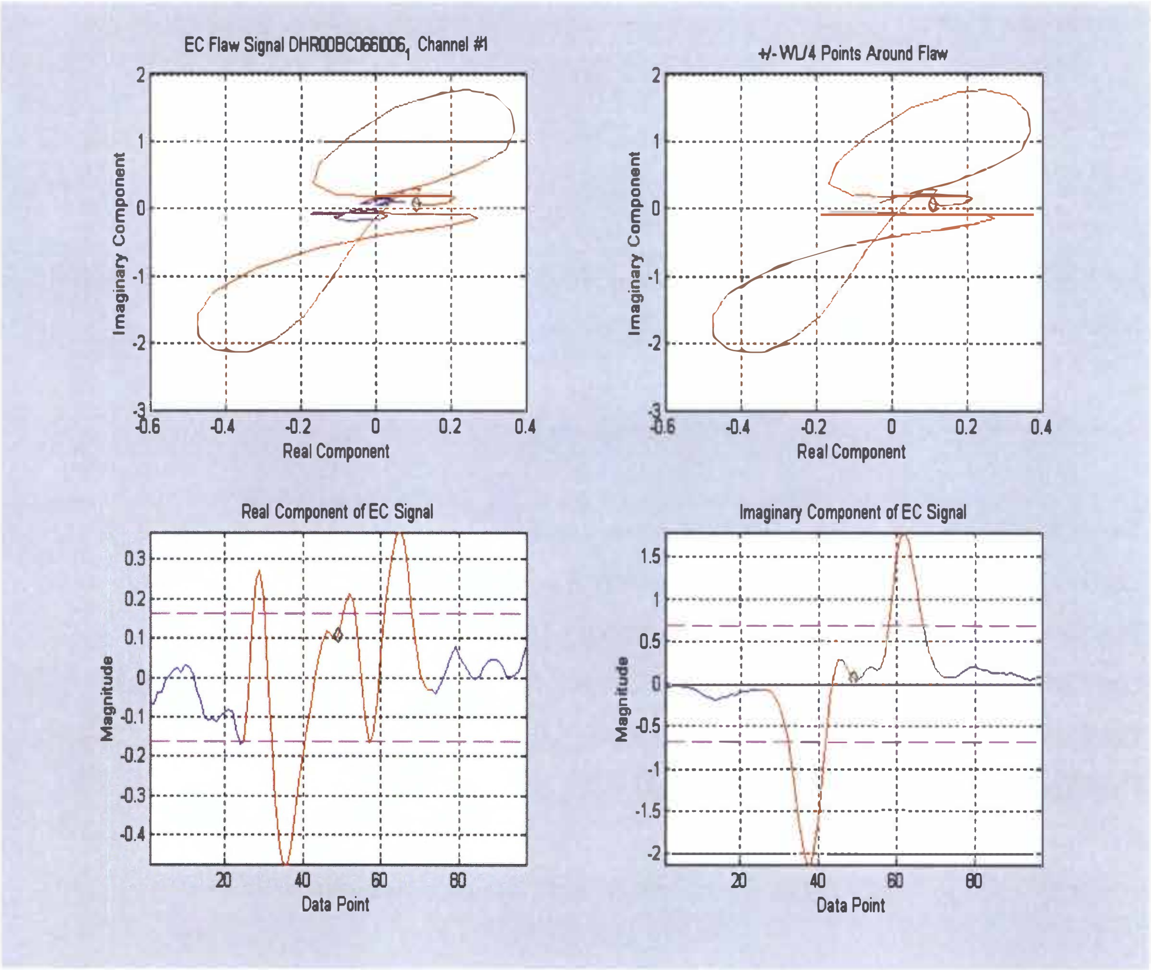

Figure F32. ECT Resistance Signal (Lissarious and Component Plots) of Flaw DHR00BC066I006_1.

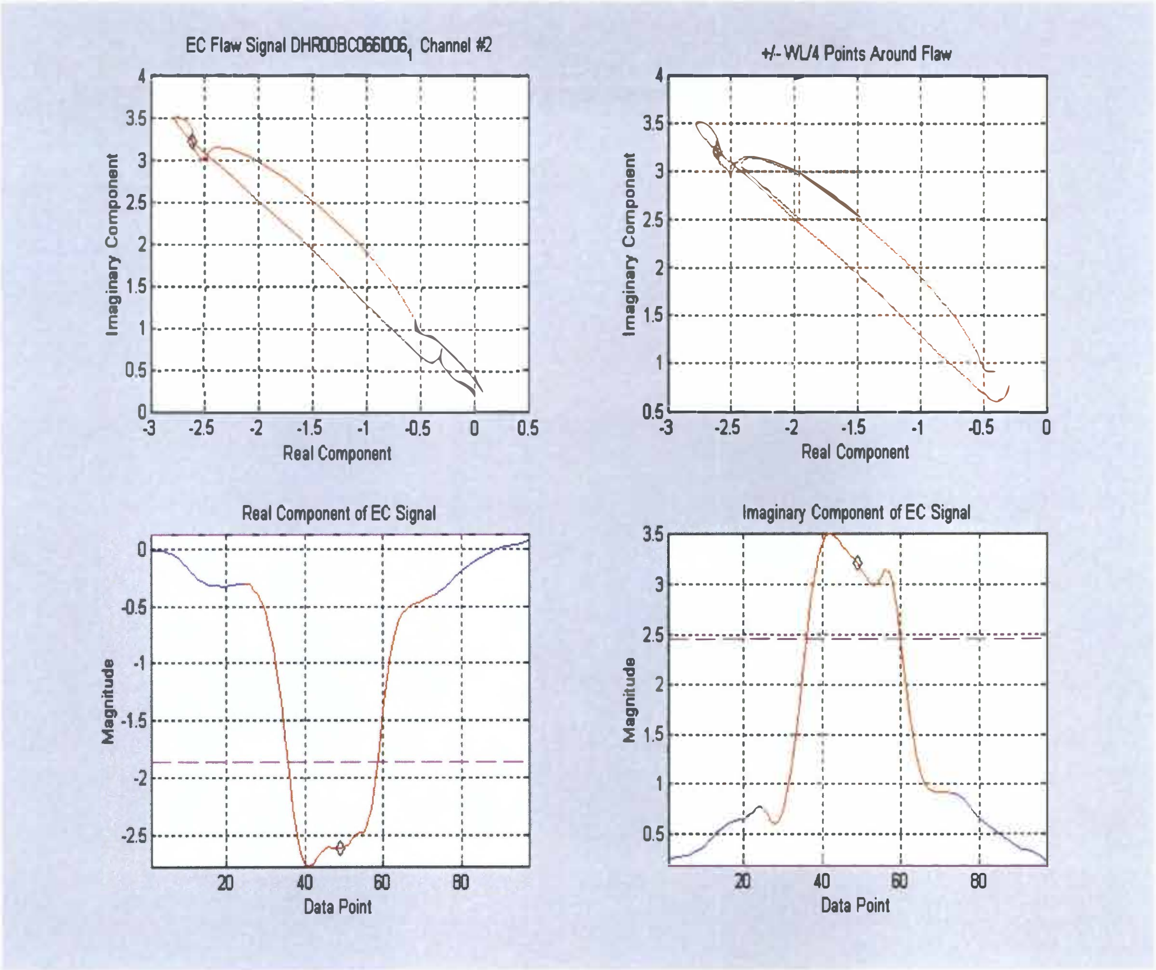

Figure F33. ECT Reactance Signal (Lissarious and Component Plots) of Flaw DHR00BC066I006_1.

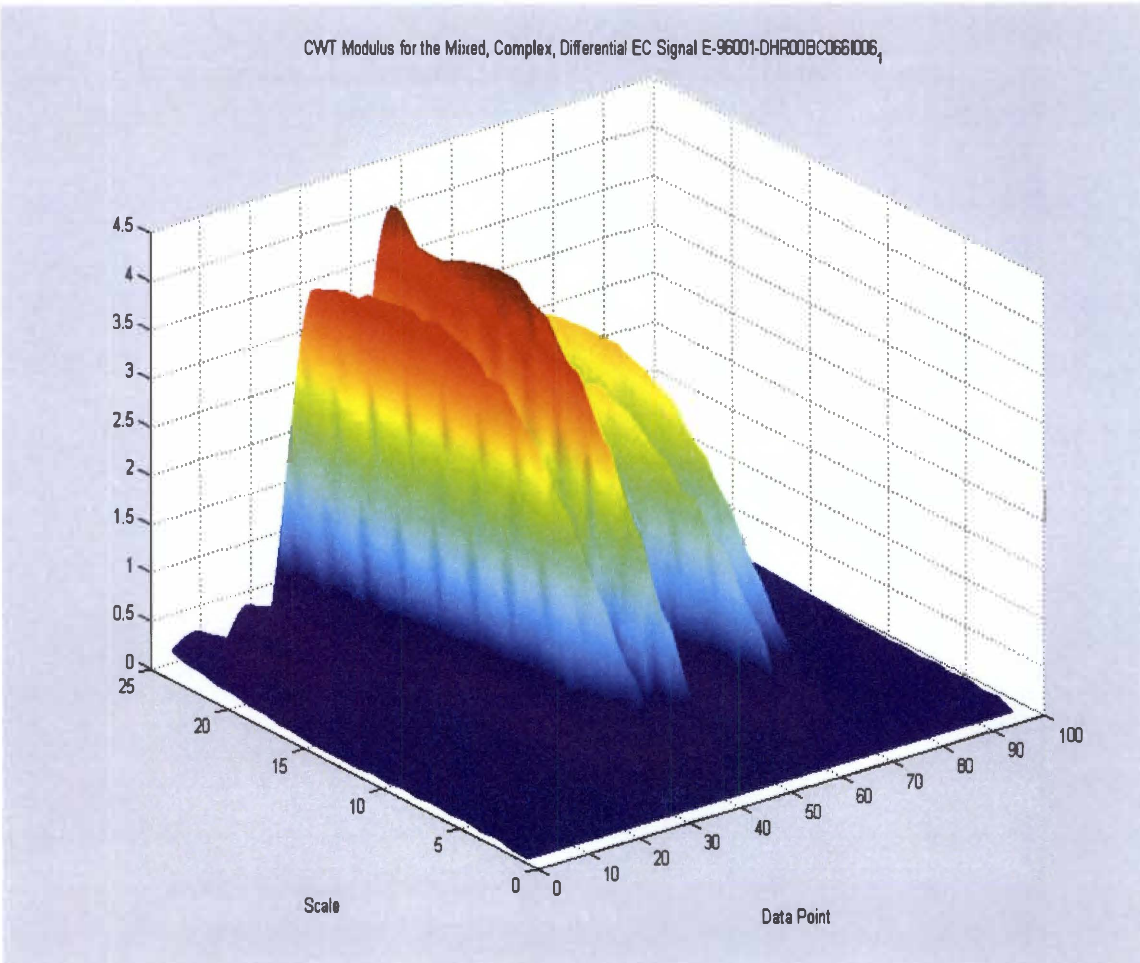

Figure F34. ECT Resistance Signal (Lissarious and Component Plots) of Flaw DHR00BC066I006_1.

Does user want to "l"oad the data cell into the uTR training cell or "c"heck flaw. c

Input the uTR run number. 2

Input the TR run number. a

Procede with Classification of flaw data ("y"es or "n"o). y

Does user want a "2"D or "3"D plot for multiple D data? 3

Figure F35 is generated.

===== Bayesian Classification Results for =====

Data origin was E

Data Group was 96001

The uTR Data run number was 2

The TR Data run number was a

Does user want to classify using original features ("y"es or "n"o). n

The flaw has been classified as flawtype 1 (1=TH, 2=IM, 3=WA and 4=PI)
using a bayesian classification system.

The Bhattacharyya Boundary (or maximum probablity of error percentage) = 0.11

Is this the correct classification ("y"es or "n"o). y

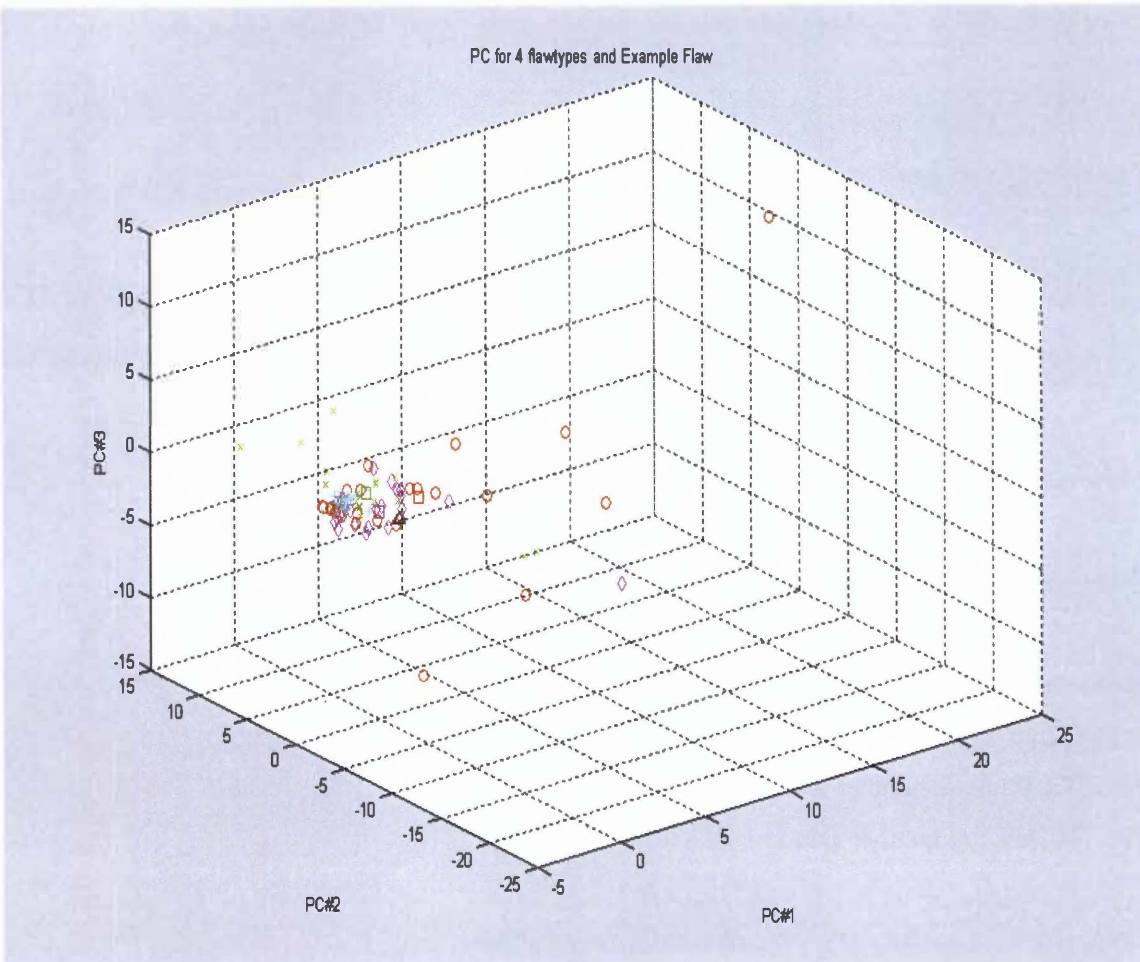

Figure F35. 3D Plot of the First three PCs of Flaw DHR00BC066I006_1.

===== Neural Network Characterization Results for =====

Data origin was E

Data Group was 96001

The Data run number was 2 a

NN char run number. 5

The calculated and actual flaw characteristics are =

ans =

```
  -14.5852   57.0000
   -0.6895   90.0000
   -1.5407    3.0000
```

The MSE between actual and calculated characterisitcs =

MSE_flaw =

4.4565e+003

If user wants to input test data into uTR and TR matrix, enter "yes". n

Does user want to continue the EddyC program ("y"es or "n"o)? n

??? Error using ==> eddyc

User did not want to continue EddyC!

>>

Appendix G. MATLAB Code for EddyC.m and Associated Programs

EddyC.m

```
%
% EddyC.m
%
%      This was the main program for the EC Data classification and
characterization routine
%
%      The program was ran after data has been preprocessed using the eddym
system or
%             the program can load preprocessed EC data saved in .mat format
(after eddym
%             preprocessing)
%

data_origin=input('Is this "P"DD or "E"TSS data. ','s');
Group=input('Enter Manufacturer of Steam Generator (B, C or W) or ETSS #.
','s');
filename=input('Input ECT saved data filename (ex. T24b01_T080R025_1, no .mat
needed). ','s');
data_type=input('Is this a "S"aved data file or a command "W"indow data file?
','s');

% This segement of programming will properly load either type of data

if data_type=='S'

[data_cell,X,flaw_phase,flaw_mag,flaw_loc,feature_vector,CWT_coef,flaw_type,pTW
,ETSS_char]=ViewDataXf(data_origin,Group,filename);
elseif data_type=='W'

[X,flaw_loc,flaw_phase,flaw_mag,flaw_type,pTW,ETSS_char]=ViewData(data_origin,G
roup,filename,x,MIDRANGE,ANGLE_MAG);
else
```

```
   error('Must enter an "W" or an "S"')
end

% Resultant variables in the command space: X (extracted flaw data), flaw_loc,
phase, mag, type, % TW,
%   Group, filename and ETSS_char (may be null set if data was PDD)

[r,c]=size(X);

% Visual conformation of data

fprintf('\r\n ============ ETSS or PDD Input Information ============ \n\r')
fprintf('\rThe origin (E = ETSS or P = PDD) of the data was %s \r',data_origin)
fprintf('The EC Data filename was %s \r',filename)
fprintf('The Steam Generator type or ETSS # was %s \r',Group)
fprintf('The PDD or ETSS location was %0.5g , doublecheck flaw location for the
given filename! \r',flaw_loc)
fprintf('The PDD or ETSS Flaw Magnitude was %0.5g \r',flaw_mag)
fprintf('The PDD or ETSS Phase Angle was %0.5g \r',flaw_phase)
fprintf('\r\n  ========  ETSS  or  PDD  Flaw  Classification  and  Characterizations
======== \n\r')
fprintf('\n\rThe PDD or ETSS Flaw Type was %s \r',flaw_type)
fprintf('The PDD or ETSS Percent Thru-wall was %.2g \r',pTW)
if data_origin == 'E'
    fprintf('ETSS characteristics =  ');fprintf('%0.5g ',ETSS_char');
end

fprintf('\r\n\n');
vis_review=input('Does the data appear to be correct ("y"es or "n"o)? ','s');
if vis_review=='n'
   error('Problem with data')
end

if data_type == 'W'

% Extract 1D feature from the Mixed Imaginary Differential Channel (Column 1 in
X)

Xdiff=imag(X(:,1));
[fext1Ddiff]=oneDfext(Xdiff,'y');
```

```
% Visual conformation of data

fprintf('\rDo 1D features for the mixed, imaginary, differential EC data\n')
vis_review=input(' appear to be correct ("y"es or "n"o)? ','s');
if vis_review=='n'
   error('Problem with data')
end

% Extract 1D features from the Imaginary Mixed Absolute Channel (Column 2 in X)

if c>1
   Xabs=imag(X(:,2));
   [fext1Dabs]=absfext(Xabs,'y');
   fprintf('\rDo the 1D features for the mixed, imaginary, absolute data\n')
   vis_review=input(' appear to be correct ("y"es, "n"o)? ','s');
   if vis_review=='n'
       error('Problem with data')
   end
else
    fext1Dabs=[];                          % NO Absolute signal from EddyM
end

% CWT calculation and feature extraction using the differential EC data

[geofext,imagefext,CWT_coef]=CWTfext(X(:,1),filename,'y');

% Visual conformation of the cwt coefs of the differential data
fprintf('\rDoes the CWT of the mixed, complex, differential EC data\n')
vis_review=input(' appear to be correct ("y"es or "n"o)? ','s');
if vis_review=='n'
   error('Problem with data')
end

%
% At this point, all features have been generated or input, none have been
normalized.  The features are:
%
%      1. fext1Ddiff (1D-diff),
%      2. fext1Dabs (1D-abs),
%      3. geofext (Geomoments),
%      4. imagefext (Image-processed) and
```

```
%      5. Input PDD (Phase and Magnitude)
%
% Also the OutputPDD vector was available.
%
% Combine Input, fext1Dabs, fext1Ddiff, geofext and imagefext to form one
feature vector
%

feature_vector=[flaw_phase flaw_mag fext1Dabs fext1Ddiff geofext imagefext];
% position of feature families [2 21 23 48 51]

% Final exit before uTR addition

fprintf('\rDo ALL features for the data appear to be correct\n')
vis_review=input('("y"es, to continue or "n"o, exit program)? ','s');
fprintf('\r\n')
if vis_review=='n'
   error('Problem with data')
end

%
% ALL the information was loaded into a nested cell called data_cell
%

% data_cells first column

data_cell{1,1}{1,1}=data_origin;
data_cell{1,1}{2,1}=Group;
data_cell{1,1}{3,1}=filename;

% data_cells second column


data_cell{1,2}{1,1}=X;
data_cell{1,2}{2,1}=[flaw_mag flaw_phase];
data_cell{1,2}{3,1}=flaw_loc;
data_cell{1,2}{4,1}=feature_vector;
data_cell{1,2}{5,1}=CWT_coef;

% data_cells third column
```

```
data_cell{1,3}{1,1}=flaw_type;
data_cell{1,3}{2,1}=pTW;
data_cell{1,3}{3,1}=ETSS_char;

% Save the Individual extracted EC data Information in Cell format.

eval(['save ' data_origin '_' Group '_' filename ' data_cell;']);
fprintf('\nData Cell %s_%s_%s has been saved.\n\r',data_origin,Group,filename);

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%
                                                                         %
%      Data  Loading,  uTR  to  TR  processing,  PCA  Processing  and  CWT  Template
calculation          %
%
                                                                         %
% User was prompted to load data into matrix if desired, then has option to
test matrix,      %
%      continue loading or test individual flaw
                                                   %
%
                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%

fprintf('\n');
load_data=input('Does user want to "l"oad the data cell into the uTR training
cell or "c"heck flaw. ','s');
fprintf('\n');

if load_data == 'l'

    uTR_run_number=input('Input the uTR run number. ','s');
   [uTR]=LoadMatrix(data_origin,Group,filename,data_cell,uTR_run_number);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%
                                 %
% Each uTR cell array page contains the information for one flaw          %
%   in a 1X3 nested cell array                                            %
%
                                         %
%                       C1                             C2
         C3                      %
%        | Origin          | Original Signal X       | flaw type       |
      %
%        | Group           | Magnitude and Phase     | % Through Wall |
      %
% R1     | filename        | flaw location           | flaw character |
      %
%        |                 | Feature Vector          |                        |
      %
%        |                     | CWT                     |                 |
      %
%
                                   %
%
                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   load_more=input('Does user want to input more data into uTR matrix, enter
"y"es. ','s');
   if load_more == 'y'
      error('Restart EddyC and Continue loading uTR') % Exit program and
continue loading data
   end

   % uTR shuffle to group like flaws together.

   [uTR,Z,index,sorting_matrix]=uTR_shuffle(uTR);

   % After uTR loading was completed, basic scatter plots and statistical
analysis may be done for
   %    each feature group (or family)

   stat_check=input('Does user want to view statistical data for uTR Feature
Matrix, enter "y"es. ','s');
```

```
   if stat_check == 'y'
      [uTR_stats]=uTR_statistics(uTR);
      stat_save=input('If uTR was fully loaded, user should "s"ave the
statistical information .','s');
      if stat_save == 's'
          if data_origin == 'P'
              eval(['save SuTR_' data_origin '_' Group '_' uTR_run_number '
SuTR;'])
          else
              eval(['save SuTR_' data_origin '_' uTR_run_number ' SuTR;'])
          end
      end
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Format of SuTR cell page, each row would be a FEATURE type
%
%          (DO NOT USE THIS INFO FOR ANY OTHER PURPOSE):
%
%
%
%              C1                        C2                                   C3
C4               C5          %
%  R1  | Load Files (cells) | Transformed Matrix |       std_mean       |
cov_matrix |  del_columns   |  %
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%

   % At this point, the user has the uTR cell.
   % PCA and cwt template compression, seperate into an array with like flaws
grouped together

   continue_program=input('Does user want to process the uTR Feature Matrix,
enter "y"es or "n"o. ','s');
   if continue_program == 'n'
      error('Exiting EddyC Program') % Exit program and continue loading data
```

```
   end

   [TR,TR_run_number]=uTR_process(uTR,data_origin,Group,uTR_run_number,'y');
             % loading was finished, process uTR into TR

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                  %
% Individual Page TR setup each page represents a flawtype.                %
%
                                    %
%                       C1                                C2
                  %
% R1     |    del_col     |       break_file for F1                       %
% R2     |    std_mean    |       cwt_comp_mat for F1
      %
% R3     |    Tn          |       flawtype_matrix for F1                 %
% R4     |    pcTR        |       flawchar_matrix for F1                 %
% R5     |    newdata     |       PCA_data{5,1} for F1                   %
% R6     |    tsquare     |       PCA_data{6,1} for F1                   %
% R7     |    QTR         |       PCA_data{7,1} for F1                   %
% R8     |    empty       |       cwts_raw for F1                        %
%
                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Note that the cwt examples are scaled from 0 to 1.

   % Plotting PCA data

   PCA_check=input('Does user want to view PCA data for TR Feature Matrix,
enter "y"es. ','s');
   if PCA_check == 'y'
      proc_TR_PCA_plot(uTR,TR,data_origin,Group,uTR_run_number,TR_run_number);
   end

   continue_p=input('Procede with Classification of TR data ("y"es or "n"o).
','s');
   if continue_p == 'n'
      error('TR processing complete, exiting EddyC program.')
   end
```

```
elseif load_data == 'c'

   %  For the test_cell: load the appropriate TR and use the processed
variables stored to likewise process
   %         the cell_data

   uTR_run_number=input('Input the uTR run number. ','s');
   TR_run_number=input('Input the TR run number. ','s');
   fprintf('\n')

  if data_origin == 'P'
      eval(['load uTR_' data_origin '_' Group '_' uTR_run_number ';'])
                 % loads appropriate uTR
      eval(['load TR_' data_origin '_' Group '_' uTR_run_number TR_run_number
';'])         % loads appropriate TR
  else
      eval(['load uTR_' data_origin '_' uTR_run_number ';'])
                        % loads appropriate uTR
      eval(['load TR_' data_origin '_' uTR_run_number TR_run_number ';'])
             % loads appropriate TR
  end

  [r,c,d]=size(TR);

   %eval(['[processed_data_' SG ']=data_process(TR,data_cell);'])      % process
test data, returns processed data

   continue_p=input('Procede with Classification of flaw data ("y"es or "n"o).
','s');
   fprintf('\n')
   if continue_p == 'n'
      error('Flaw data processing complete, exiting EddyC program.')
   end

else

   error('Programming Problem ... Entered wrong answer.')

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%             CLASSIFICATION ROUTINE
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if load_data == 'l'

   PCA_plot(TR{5,1,1},break_points);                           % Plot PCA TR

   % Bayes classification


[classnonPCA,wrongnonPCA,classPCA,wrongPCA,g,BB]=bayes_class(uTR,TR,data_origin
,Group,uTR_run_number,TR_run_number);
   fprintf('\nThe Bhattacharyya Boundary (or maximum probablity of error
percentage) = %2.2f \n\n',BB)

   % NN classification


%[net,Y,flaw_type,NN_class_run_number]=NN_class(uTR,TR,data_origin,Group,uTR_ru
n_number,TR_run_number);

   % CWT Template Matching

   %[temp_match_info,cwt_class_temp]=cwt_template_result(TR,'y');

elseif load_data == 'c'

    % data for flaw, should already be loaded from ViewdataXF output

    flaw_feature_vector=feature_vector;
    cwt_flaw=CWT_coef;

    % Data from TR

    PC=TR{4,1,1};                               % PCA transformation matrix
```

```
    std_mean=TR{2,1,1};                                   % mean and std for
preprocessing columns
    del_col=TR{1,1};
    matrix=TR{5,1,1};                                   %      Extractes      PCA
compressed TR data for all flawtypes

    for i=1:d
      if i==1
          flawtype_matrix=TR{3,2,i};                        % Flawtypes
      else
         flawtype_matrix=[flawtype_matrix;TR{3,2,i}];
      end
      flawchar_matrix{i,1}=TR{4,2,i};
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                  %
% Individual Page TR setup each page represents a flawtype.                %
%
                                    %
%                       C1                                C2
                  %
% R1     |   del_col      |       break_file for F1                    %
% R2     |   std_mean     |       cwt_comp_mat for F1
      %
% R3     |   srTR         |       flawtype_matrix for F1               %
% R4     |   pcTR         |       flawchar_matrix for F1               %
% R5     |   newdata      |       PCA_data{5,1} for F1                 %
% R6     |   tsquare      |       PCA_data{6,1} for F1                 %
% R7     |   QTR          |       PCA_data{7,1} for F1                 %
% R8     |FV_reinsertion  |       cwts_raw for F1                      %
%
                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% flaw feature vector must be pre-processed

    flaw_feature_vector(:,del_col)=[];              % delete appropriate non-
variance columns
```

```
    [rfull,cfull]=size(flaw_feature_vector);

    stdT=std_mean(1,:);meanT=std_mean(2,:);                              % std and
mean preprocessing

    flaw_feature_vector=flaw_feature_vector-meanT(ones(rfull,1),:);
    flaw_feature_vector=flaw_feature_vector./stdT(ones(rfull,1),:);

   if isempty(TR{8,1,1}) == 0                      % extracts appropriate cols
before PCA, if needed
       FV_reinsertion=TR{8,1,1};
       del_T=flaw_feature_vector(:,FV_reinsertion);
       flaw_feature_vector(:,FV_reinsertion)=[];
   end

   PCAflaw=flaw_feature_vector*PC;             % PCA transformation of flaw FV
(after extraction, if needed)

   if isempty(TR{8,1,1}) == 0                      % reinsertion of extracted FV
cols, if needed
       flaw=[PCAflaw del_T];
   else
       flaw=PCAflaw;
   end

   [break_points,num_breaks,break_file]=break_point_b(uTR);

   PCA_plot(matrix,break_points,flaw);
             % Plot PCA processed_TR and flaw

   % MATLAB classification program


[classnonPCA,wrongnonPCA,classPCA,wrongPCA,g,BB]=bayes_class(uTR,TR,data_origin
,Group,uTR_run_number,TR_run_number,flaw);

   fprintf('The flaw has been classified as flawtype %1.0f (1=TH, 2=IM, 3=WA
and 4=PI)\n',classPCA)
   fprintf('\tusing a bayesian classification system.\n')
   fprintf('\nThe Bhattacharyya Boundary (or maximum probablity of error
percentage) = %2.2f \n\n',BB)
```

```
   % Classification Correct ?

   correct_class=input('Is this the correct classification ("y"es or "n"o).
','s');
   if correct_class == 'y'
       flaw_type=classPCA;
   else
       flaw_type=input('The correct flaw classification was (1=TH, 2=IM, 3=WA
and 4=PI).');
   end

   % NN


%[net,Y,flaw_type,NN_class_run_number]=NN_class(uTR,TR,data_origin,Group,uTR_ru
n_number,TR_run_number,filename,flaw);

   % CWT template classification

   %[temp_match_info,cwt_class_temp]=cwt_template_result(TR,'y',data_cell);
   % Peak locations and value
   %info_Cn1=[norm_sumCnl peakval1 maxx1 maxy1];
   %info_Cn2=[norm_sumCn2 peakval2 minx2 miny2];

else

   error('Programming Problem ... Entered wrong answer.')

end

% STOP - PROGRAMMING FROM THIS POINT FOWARD IS NOT CORRECT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%             CHARACTERIZATION ROUTINE
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if load_data == 'l'

    % Seperate flaw using classification

    % Use compressed feature vector to train NN


[net,Y,NN_char_run_number]=NN_char(uTR,TR,data_origin,Group,uTR_run_number,TR_r
un_number);

    % Check results

    % Save trained NN

elseif load_data == 'c'


[net,Y,NN_char_run_number]=NN_char(uTR,TR,data_origin,Group,uTR_run_number,TR_r
un_number,filename,flaw,flaw_type);
    flaw_char=[data_cell{1,3}{2,1} data_cell{1,3}{3,1}]';
    [rflaw,cflaw]=size(flaw_char);
    MSE_flaw=sum(sum((flaw_char-Y).^2))/(rflaw*cflaw);
    fprintf('\nThe calculated and actual flaw characteristics are  = \n');
    [Y flaw_char]
    fprintf('\nThe  MSE  between  actual  and  calculated  characterisitcs   =
%.6f\n');
    MSE_flaw

end

% Load test data into datamatrix if desired, then continue if desired.  This
was probably OK

if load_data == 'c'
    fprintf('\n')
   load_more=input('If user wants to input test data into uTR and TR matrix,
enter "yes". ','s');
   if load_more == 'y'
       correct_flawtype=input('Is EddyC classification correct ("y"es or "n"o).
','s');
```

```
        if correct_flawtype == 'n'
            fprintf('\nWA=Wear Type 1, WB=Wear Type 2, IGA=IG, SCC=SC,
IGA/SCC=IS')
            fprintf('\n PWSCC=PW, Thin=TH, Impengement=IM, Pitting=PI,
Fatigue=FA\n')
            fprintf(' Multiple=MU\n')
            flaw_type=input('Input PDD or ETSS flaw type from above list.
','s');
        end
        pTW=input('Input %TW. ');
        flaw_char=input('Input other flaw characteristics in vector notation.
');
        data_cell{1,3}{1,1}=flaw_type;
        data_cell{1,3}{2,1}=pTW;
        data_cell{1,3}{3,1}=ETSS_char;

eval(['[uTR]=LoadMatrix(data_origin,Group,filename,data_cell,uTR_run_number);']
)  % Load test data into uTR data set

eval(['[TR,TR_run_number]=uTR_process(uTR,data_origin,Group,uTR_run_number,''y'
');'])                  % loading was finished, process uTR into TR
    end
    continue_p=input('Does user want to continue the EddyC program ("y"es or
"n"o)? ','s');
    if continue_p == 'n'
       error('User did not want to continue EddyC!')
    elseif continue_p == 'y'
       ['continue EddyC']
    else
       ['Wrong input, EddyC was continuing']
    end
end
```

ViewDataXf.m

```
function
[data_cell,X,flaw_phase,flaw_mag,flaw_loc,feature_vector,CWT_coef,flaw_type,pTW
,ETSS_char]=viewDataXf(data_origin,Group,filename);
```

```
%
% viewDataXf.m
%
%      function
[data_cell,X,flaw_phase,flaw_mag,flaw_loc,feature_vector,CWT_coef,flaw_type,pTW
,ETSS_char]=viewDataXf(data_origin,Group,filename);
%
%      Allows user to view processed ECT data saved in .mat format.
%      Save variable should be X and was also complex.
%

eval(['load ' data_origin '_' Group '_' filename '.mat;']);

% data_cell was loaded, extract information

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                  %
% Each uTR cell array page contains the information for one flaw            %
%   in a 1X3 nested cell array                                              %
%
                                          %
%                        C1                           C2
          C3                      %
%       | Origin           | Original Signal X       | flaw type       |
      %
%       | Group            | Magnitude and Phase     | % Through Wall |
      %
% R1    | filename         | flaw location           | flaw character |
      %
%       |                  | Feature Vector          |                        |
      %
%       |                      | CWT                    |                 |
      %
%
                                     %
%
                                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% data_cells second column

X=data_cell{1,2}{1,1};
flaw_mag=data_cell{1,2}{2,1}(1);
flaw_phase=data_cell{1,2}{2,1}(2);
flaw_loc=data_cell{1,2}{3,1};
feature_vector=data_cell{1,2}{4,1};
CWT_coef=data_cell{1,2}{5,1};

% data_cells third column

flaw_type=data_cell{1,3}{1,1};
pTW=data_cell{1,3}{2,1};
ETSS_char=data_cell{1,3}{3,1};

[r,c]=size(X);

% Removal of bias

%Xavg=mean(X); X=X-Xavg(ones(r,1),:);

% Determine midpoint of data and interval

if rem(r,2)==0
   m_pt=r/2;
else
   m_pt=(r-1)/2+1;
end

m_int=[m_pt-round(r/4):1:m_pt+round(r/4)];

% Visual Review of Signal

for i=1:c
std1=std(real(X(:,i)));std2=std(imag(X(:,i)));
% Adjust threshold limits with the mean of signal
threshold_R1=std1+mean(real(X(:,i)));threshold_R2=-std1+mean(real(X(:,i)));
%real limits
threshold_I1=std2+mean(imag(X(:,i)));threshold_I2=-std2+mean(imag(X(:,i)));
%imag limits
figure;
```

```
subplot(2,2,1);plot(real(X(:,i)),imag(X(:,i)),'b-
',real(X(m_int,i)),imag(X(m_int,i)),'r-', ...
   real(X(m_pt,i)),imag(X(m_pt,i)),'kd');
grid on;
title(['EC Flaw Signal ' filename ' Channel #' num2str(i)]);
xlabel('Real Component');ylabel('Imaginary Component')
subplot(2,2,2);plot(real(X(m_int,i)),imag(X(m_int,i)),'r-
',real(X(m_pt,i)),imag(X(m_pt,i)),'kd');grid on;
title(['+/- WL/4 Points Around Flaw']);
xlabel('Real Component');ylabel('Imaginary Component')
subplot(2,2,3);plot([1:r],real(X(:,i)),'b-',m_int,real(X(m_int,i)),'r-
',m_pt,real(X(m_pt,i)),'kd', ...
   [1:r],threshold_R1*ones(r,1),'m--',[1:r],threshold_R2*ones(r,1),'m--');
grid on;axis tight%([0 r 1.1*min(real(X(:,i))) 1.1*max(real(X(:,i)))]);
title('Real Component of EC Signal');
xlabel('Data Point');ylabel('Magnitude')
subplot(2,2,4);plot([1:r],imag(X(:,i)),'b-',m_int,imag(X(m_int,i)),'r-
',m_pt,imag(X(m_pt,i)),'kd', ...
   [1:r],threshold_I1*ones(r,1),'m--',[1:r],threshold_I2*ones(r,1),'m--');
grid on;axis tight%([0 r 1.1*min(imag(X(:,i))) 1.1*max(imag(X(:,i)))]);
title('Imaginary Component of EC Signal')
xlabel('Data Point');ylabel('Magnitude')
end

% CWT review

figure;
surf(CWT_coef);
colormap jet;shading interp;
xlabel('Data Point');ylabel('Scale');
title(['CWT Modulus for the Mixed, Complex, Differential EC Signal '
data_origin '-' Group '-' filename])
```

ViewData.m

```
function
[X,flaw_loc,flaw_phase,flaw_mag,flaw_type,pTW,ETSS_char]=viewData(data_origin,G
roup,filename,x,MIDRANGE,ANGLE_MAG);
```

```
%
% viewData.m
%
%   function [X,Group,flaw_loc,flaw_phase,flaw_mag,flaw_type,pTW,ETSS_char]=
%   viewData(data_origin,Group,filename,x,MIDRANGE,MAG_ANGLE);
%
%     Allows user to view loaded ECT data in conjuction with the EDDYM system.
The program prompts the
%            user to input the PDD given flaw characteristics and the Eddym
data channel number.
%

data_chan=input('Input the data channels to be viewed. ');

% Enter PDD data and flaw characterisitics

fprintf('\nDoes user want to use eddym info (midpoint, phase and mag)\n')
eddym_info=input(' for the windowed data ("y"es or "n"o)? ','s');
if eddym_info == 'y'
    flaw_loc=MIDRANGE;
    flaw_mag=ANGLE_MAG(2);
    flaw_phase=ANGLE_MAG(1);
else
    flaw_loc=input('Enter PDD or ETSS flaw location. ');
    flaw_mag=input('Input PDD or ETSS given flaw Magnitude. ');
    flaw_phase=input('Enter PDD or ETSS given Phase Angle. ');
end

fprintf('\nWA=Wear Type 1, WB=Wear Type 2, IGA=IG, SCC=SC, IGA/SCC=IS')
fprintf('\n PWSCC=PW, Thin=TH, Impengement=IM, Pitting=PI, Fatigue=FA\n')
fprintf(' Multiple=MU\n')
flaw_type=input('Input PDD or ETSS flaw type from above list. ','s');
fprintf('\n')
pTW=input('Enter PDD or ETSS given Percent Thru-wall. ');

% Input ETSS Characterization Data from Blueprints

[ETSS_char]=ETSS_input;
```

```
% Remember, data in command window in column format.  This segement extracts
and combines both real and
%      imag components, then combines to form a complex file, then mean-
subtracts.  The X file now would have
%      data_chan # of columns with 97 data points

WL=48;                                   % sets maximum 1/2 window length

% x was original MATLAB window data set, X was extracted data segment

for i=1:length(data_chan)
   X(:,i)=x(flaw_loc-WL:flaw_loc+WL,2*data_chan(i)-1)+j*x(flaw_loc-
WL:flaw_loc+WL,2*data_chan(i));
   if i==1
      X(:,i)=X(:,i)-mean(X(:,i)); % only subtract mean from the differential
signal
   end
end

[r,c]=size(X);

% Determine midpoint of data and interval

if rem(r,2)==0
   m_pt=r/2;
else
   m_pt=(r-1)/2+1;
end

m_int=[m_pt-round(WL/4):1:m_pt+round(WL/4)];

% Visual Review of Signal

for i=1:c
std1=std(real(X(:,i)));std2=std(imag(X(:,i)));
% Adjust threshold limits with the mean of signal
threshold_R1=std1+mean(real(X(:,i)));threshold_R2=-std1+mean(real(X(:,i)));
%real limits
threshold_I1=std2+mean(imag(X(:,i)));threshold_I2=-std2+mean(imag(X(:,i)));
%imag limits
figure;
```

```
subplot(2,2,1);plot(real(X(:,i)),imag(X(:,i)),'b-
',real(X(m_int,i)),imag(X(m_int,i)),'r-', ...
   real(X(m_pt,i)),imag(X(m_pt,i)),'kd');
grid on;
title(['EC Flaw Signal ' filename ' Channel #' num2str(data_chan(i))]);
xlabel('Real Component');ylabel('Imaginary Component')
subplot(2,2,2);plot(real(X(m_int,i)),imag(X(m_int,i)),'r-
',real(X(m_pt,i)),imag(X(m_pt,i)),'kd');grid on;
title(['+/- WL/4 Points Around Flaw']);
xlabel('Real Component');ylabel('Imaginary Component')
subplot(2,2,3);plot([1:r],real(X(:,i)),'b-',m_int,real(X(m_int,i)),'r-
',m_pt,real(X(m_pt,i)),'kd', ...
   [1:r],threshold_R1*ones(r,1),'m--',[1:r],threshold_R2*ones(r,1),'m--
');set(gca,'xtick',[0:5:r]);
grid on;axis tight%([0 r 1.1*min(real(X(:,i))) 1.1*max(real(X(:,i)))]);
title('Real Component of EC Signal');
xlabel('Data Point');ylabel('Magnitude')
subplot(2,2,4);plot([1:r],imag(X(:,i)),'b-',m_int,imag(X(m_int,i)),'r-
',m_pt,imag(X(m_pt,i)),'kd', ...
   [1:r],threshold_I1*ones(r,1),'m--',[1:r],threshold_I2*ones(r,1),'m--
');set(gca,'xtick',[0:5:r]);
grid on;axis tight%([0 r 1.1*min(imag(X(:,i))) 1.1*max(imag(X(:,i)))]);
title('Imaginary Component of EC Signal')
xlabel('Data Point');ylabel('Magnitude')
end

% Check 97 data points, Is another flaw located within the interval?

fprintf('\nIs window length appropriate,\n')
WL_ok=input(' "y"es or "n"o (no more than one flaw in data window)? ','s');

if WL_ok == 'n'

   WL=input('Input appropriate 1/2 window length for re-extraction of data
segment. ');
   %noise_window=input('Input smallest length noise segment of windowed signal.
');

   for i=1:length(data_chan)
```

```
      Xn(:,i)=x(flaw_loc-WL:flaw_loc+WL,2*data_chan(i)-1)+j*x(flaw_loc-
WL:flaw_loc+WL,2*data_chan(i));

      if i==1
          Xn(:,i)=Xn(:,i)-mean(Xn(:,i));       % only subtract mean from the
differential signal
      end

      %if i==2
      %    bias_abs=input('Does Imag Absolute Signal need bias adjustment ("y"es
or "n"o). ','s');
      %    if bias_abs == 'y'
      %       bias_add=input('Input amount of bias to add to signal. ');
      %       Xn(:,i)=Xn(:,i)+bias_add.*(ones(r,c)+j*ones(r,c));
      %    end
      % end

      [r,c]=size(Xn(:,i));

   end

   X=Xn;
   [r,c]=size(X);

   % Determine midpoint of data and interval

   if rem(r,2)==0
      m_pt=r/2;
   else
      m_pt=(r-1)/2+1;
   end

   m_int=[m_pt-round(WL/4):1:m_pt+round(WL/4)];

   % Visual Review of Signal

   for i=1:c
      std1=std(real(X(:,i)));std2=std(imag(X(:,i)));
      % Adjust threshold limits with the mean of signal
      threshold_R1=std1+mean(real(X(:,i)));threshold_R2=-
std1+mean(real(X(:,i))); %real limits
```

```
      threshold_I1=std2+mean(imag(X(:,i)));threshold_I2=-
std2+mean(imag(X(:,i))); %imag limits
      figure;
      subplot(2,2,1);plot(real(X(:,i)),imag(X(:,i)),'b-
',real(X(m_int,i)),imag(X(m_int,i)),'r-', ...
         real(X(m_pt,i)),imag(X(m_pt,i)),'kd');
      grid on;
      title(['EC Flaw Signal ' filename ' Channel #' num2str(data_chan(i))]);
      xlabel('Real Component');ylabel('Imaginary Component')
      subplot(2,2,2);plot(real(X(m_int,i)),imag(X(m_int,i)),'r-
',real(X(m_pt,i)),imag(X(m_pt,i)),'kd');grid on;
      title(['+or-      WL/4      Points      Around      Flaw']);xlabel('Real
Component');ylabel('Imaginary Component')
      subplot(2,2,3);plot([1:r],real(X(:,i)),'b-',m_int,real(X(m_int,i)),'r-
',m_pt,real(X(m_pt,i)),'kd', ...
         [1:r],threshold_R1*ones(r,1),'m--',[1:r],threshold_R2*ones(r,1),'m--
');
      grid on;axis tight%([0 r 1.1*min(real(X(:,i))) 1.1*max(real(X(:,i)))]);
      title('Real      Component      of      EC      Signal');xlabel('Data
Point');ylabel('Magnitude')
      subplot(2,2,4);plot([1:r],imag(X(:,i)),'b-',m_int,imag(X(m_int,i)),'r-
',m_pt,imag(X(m_pt,i)),'kd', ...
         [1:r],threshold_I1*ones(r,1),'m--',[1:r],threshold_I2*ones(r,1),'m--
');
      grid on;axis tight%([0 r 1.1*min(imag(X(:,i))) 1.1*max(imag(X(:,i)))]);
      title('Imaginary Component of EC Signal')
      xlabel('Data Point');ylabel('Magnitude')
   end

end
```

oneDfext.m

```
function [fext]=oneDfext(signal,plotyn);

%
% oneDfext.m
%
% function [fext]=oneDfext(signal,plotyn);
```

```
%
% 1D Feature Extraction program.  Will load a 1D imaginary signal and
%  detect packets of energy and characterize them (fext). Plotyn allows
%      the user to plot the individual ROIs with Peaks
%

Y=signal;
[r,c]=size(Y);

%The data vector should be in column format
% if X was a row formatted data vector, then invert the matrix

if c>r
   Y=Y';
   [r,c]=size(Y);
end

% STD Calculation

Y=Y-mean(Y);
stdi=std(Y);
thresholdI=stdi;

% Same PeakID and ROI Process for the imaginary signal

[maximaI]=peakID(Y,thresholdI,'n');
[maxrI,maxcI]=size(maximaI);

% if no ROI's are apparent, use mid point of data and assume 1 peak

if isempty(maximaI)==1
   mROIi=round(r/2);
   ROIdist=32;
   maxIavg=0;
   numROIi=1;
   fprintf('\rNo peaks above STD were detected in the Mixed Imaginary
Differential Data\n\r')
elseif maxrI == 1
   mROIi=maximaI(1);
   ROIdist=32;
   maxIavg=0;
```

```
   numROIi=1;
   fprintf('\rOne peak above STD were detected in the Mixed Imaginary
Differential Data\n\r')
elseif maxrI > 1
                                        % Must have more than one peak
   ROIdist=32;
   maxIavg=mean(abs(maximaI(:,2)));
                     % ROI centers Maximum Distance apart
   [mROIi,numROIi]=ROIcal(Y,maximaI,ROIdist);                    % Use average
peak mags /std
end

% Plot Peaks and middle of ROIs only if there are peaks

if maxrI >= 1

if plotyn=='y'
figure;
peakID(Y,thresholdI,'y');axis tight;
hold on
plot(1:r,[thresholdI*ones(r,1) -thresholdI*ones(r,1)],'r--');
plot(mROIi,0,'kd');
title(['Mixed, Imaginary, Differential EC Signal with threshold = '
num2str(thresholdI)])
xlabel('Point')
ylabel('Magnitude')
%hold off;
end

% Check for Multiple ROIs in complex data

multiROIr(Y,ROIdist,mROIi,numROIi,thresholdI);
hold off;

% Determine mean ROI positions using both Real and Imag data

pROI=mean(mROIi);
pROI=round(pROI);

% Setup ROI intervals for full signal and plot EC
```

```
%if plotyn=='y' & ( numROIr > 1 | numROIi > 1 )
%for i=1:length(pROI)
%   ROI(:,i)=Y(pROI(:,i)-ROIdist/2:pROI(:,i)+ROIdist/2);
%   figure;
%   subplot(3,1,1);plot(ROI(:,i));
%   title(['ROI #' num2str(i) ' Eddy Current Signal'])
%   ylabel('Magnitude')
%   subplot(3,1,2);plot(real(ROI(:,i)));
%   title('Real Eddy Current Signal')
%   ylabel('Magnitude')
%   subplot(3,1,3);plot(imag(ROI(:,i)));
%   title('Imaginary Eddy Current Signal')
%   xlabel('Point')
%   ylabel('Magnitude')
%end
%end

% Feature Extraction.  If there are no peaks, assign 0 values


[maxI,Imax]=max(maximaI(:,2),[],1);
[minI,Imin]=min(maximaI(:,2),[],1);
if isempty(maximaI)==0
   numpeaksI=length(maximaI(:,1));
   DpeaksI=maximaI(Imax,1)-maximaI(Imin,1);
   %distance=pdist(maxI-minI)/stdi;
   peaktopeak=(maxI-minI)/stdi;
else
   numpeaksI=0;DpeaksI=0;peaktopeak=0;
end

fext=[DpeaksI peaktopeak];

else                          % NO PEAKS DETECTED

   fext=[0 0];

end
```

peakID.m

```
function [maxima]=peakID(signal,threshold,graph);
%
%       [maxima]=peakID(signal,threshold,graph)
%
% peakID.m Locates peaks within data sets
%

yt=signal;
%yt=yt-mean(yt);

% zeroing of values in signal (less than   y) threshold level (keep original
signal

ind=(find(yt<threshold & yt>-threshold));

if yt(ind)>0
   yt(ind)=threshold;
else
   yt(ind)=-threshold;
end

% finding local maxima

datasize = length(yt);
[r,c]=size(yt);

maxima=[];
count = 0;
for i = 2 : datasize-1
   if ( ((yt(i-1) < yt(i)) & (yt(i+1) < yt(i))) ...
         | ((yt(i-1) > yt(i)) & (yt(i+1) > yt(i))) )
      count = count+1;
      maxima(count,:) = [i yt(i)];
       end
end

% Plot signal with maxima
```

```
if isempty(maxima)==0
   [rmax,cmax]=size(maxima);
   if (graph=='y')
      plot(signal);hold on;
      plot(maxima(:,1),maxima(:,2),'ro');
      plot(threshold*ones(r,1),'r--');
      plot(-threshold*ones(r,1),'r--');
      axis tight;
      aux=axis;
      if rmax<16
         for k=1:rmax
            sf=sprintf('%.2f,%.2f',maxima(k,1),maxima(k,2));
            text(aux(2)*4.5/6,aux(4)-((aux(4)-aux(3))/20)*k,sf)
         end
      end
        title(['Data Number vs. Magnitude'])
        xlabel('Data Number')
      ylabel('Magnitude')
      hold off;
   end
else
   if (graph=='y')
      plot(signal);
      hold on;
      plot(threshold*ones(r,1),'r--');
      plot(-threshold*ones(r,1),'r--');
      axis tight;
      aux=axis;
      sf=sprintf('%s','No Peaks Detected');
      text(aux(2)*4/6,aux(4)-((aux(4)-aux(3))/20),sf);
      title(['Data Number vs. Magnitude']);
      xlabel('Data Number')
      ylabel('Magnitude')
      hold off;
   end
   maxima=[];
end
```

ROIcal.m

```
function [mROI,numROI]=ROIcal(signal,maxima,ROIdist);
%
% function [mROI,numROI]=ROIcal(signal,maxima,ROIdist)
%
%      Inputs:      signal
%                          maxima = Peaks detected by peakid
%                          ROIdist = determined in oneDfext
%
%      Outputs:     mROI = Middle point of ROIs
%                          numROI = Number of ROIs
%
%      ROI center determinations for a signal
%

[maxr,maxc]=size(maxima);

% ROI region distance adjustment

ROIx=[];
mROI=[];
numROI=[];
l=1;j=1;
if maxr==1
   ROIx=maxima(:,1);
   mROI(l)=ROIx;
else
   for i=2:maxr
      P1=maxima(i-1,1);
      P2=maxima(i,1);
      distp1p2=P2-P1;
      if distp1p2>=ROIdist
         ROIx=maxima(j:i-1);
         mROI(l)=round(mean(ROIx));
         l=l+1;
         j=i;
      end
```

```
      if i==maxr
         ROIx=maxima(j:i);
         mROI(l)=round(mean(ROIx));
      end
   end
end
numROI=length(mROI);
```

multiROIr.m

```
function multiROIr(Y,ROIdist,mROIi,numROIi,thresholdI);

%
% function multiROIr(Y,ROIdist,mROIi,numROIi,thresholdI);
%
% Warning of multiple ROI's, if ROI ceneters > 32 points apart.  This was
accomplished using the real and
%     imag components seperately.
%

[r,c]=size(Y);

% for Imag data

if mROIi>=2
   for i=1:length(mROIi)-1
      d=mROIi(i+1)-mROIi(i);
      if d>ROIdist
         fprintf('\rWarning: ROI centers > 32 Points Apart.  May be seperate
Flaws!\n\r')
         plot(Y);
         text(5,1.25*min(Y),'Warning: ROI centers > 32 Points Apart.  May be
seperate Flaws!');
         hold on
         plot(thresholdI*ones(r,1),'r--');
         plot(-thresholdI*ones(r,1),'r--');
         for i=1:numROIi
            plot(mROIi(i),0,'kdiamond')
         end
```

```
        title(['Mixed, Imaginary Differential EC Signal with threshold = '
num2str(thresholdI)])
        xlabel('Point')
        ylabel('Magnitude')
        hold off
      end
   end
end
```

absfext.m

```
function [fext]=absfext(signal,plotyn);

%
% absfext.m
%
% function [fext]=absfext(signal,plotyn);
%
% Absolute Signal Feature Extraction program.  Will load an absolute signal and
%  detect packets of energsignal and characterize them (fext). Plotsignaln
allows the user
%     to plot the individual ROIs with Peaks.
%     The loop index tells the user which signal (loop) causes the problem
%

Y=signal;
[r,c]=size(Y);

%The data vector should be in column format
% if X was a row formatted data vector, then invert the matrix

if c>r
   Y=Y';
   [r,c]=size(Y);
end

% STD Calculation

stdfull=std(Y);
```

```
threshold1=stdfull+mean(Y);threshold2=-stdfull+mean(Y);

% Visual Review of Abs singal

figure;
plot(Y);hold on;axis tight;
plot(threshold1*ones(r,1),'r--');
plot(threshold2*ones(r,1),'r--');
plot(mean(Y)*ones(r,1),'g--');
title(['Mixed,   Imaginary,   Absolute   EC   Signal   with   thresholds   =   '
num2str([threshold1 threshold2])])
xlabel('Point')
ylabel('Magnitude')
hold off;

fprintf('\nIs there information contained in the ');
vis_review=input('\n Mixed, Imaginary Absolute Signal ("y"es or "n"o)? ','s');

% Feature Extraction only to be done if abs signal has information

if vis_review == 'y'

% Peak Identification

[maxima,num_int,IP,FP]=peakIDabs(Y,threshold1,threshold2,'n');

if isempty(maxima) == 1

    ROIdist=round(r/2)-1;
    maxavg=0;                                                         %   Use
average peak mags /std
    mROI=round(length(Y));
    numROI=0;
    IP=1;FP=length(Y);
    intervals=[IP;FP];

else

ROIdist=round(r/2)-1;
maxavg=mean(abs(maxima(:,2)));
              % Use average peak mags /std
```

```
[mROI,numROI]=ROIcal(Y,maxima,ROIdist);
intervals=[IP';FP'];

end


% Plot Peaks and middle of ROIs

if plotyn=='y'
   peakIDabs(Y,threshold1,threshold2,'y');axis tight;
   hold on;
   plot(mean(Y)*ones(r,1),'g--');
   aux=axis;
   if isempty(maxima)==0
      plot(mROI,aux(3),'kd');
      for i=1:num_int
         plot(IP(i),aux(3),'gd');plot(FP(i),aux(3),'gd');
      end
      sf=sprintf('Intervals above (or below) STD \n (IP , FP) \n');
      text(aux(2)*1/25,aux(4)-((aux(4)-aux(3))/15),sf);
      sf=sprintf(' %.0f , %.0f  \n',intervals);
      text(aux(2)*1/20,aux(4)-((aux(4)-aux(3))/5),sf);
      title(['Mixed, Imaginary, Absolute EC Signal with thresholds = '
num2str([threshold1 threshold2])])
      xlabel('Point')
      ylabel('Magnitude')
      hold off
   end
end

% Check for Multiple ROIs in data

multiROIa(Y,mROI,numROI,ROIdist,threshold1,threshold2);

% Extract portion of data above threshold around maximum peak

if isempty(maxima)==0
[maxI,Imax]=max(maxima(:,2),[],1);
end

% Slect appropriate data interval
```

```
interval=input('\nInput interval start and finish numbers for the data in
MATLAB [ ] format. ');

intial_point=interval(1);final_point=interval(2);
newsignal=Y(intial_point:final_point);
newsignal=newsignal-newsignal(1);        % This sets the first point of
newsignal at (0,0)
lengthNS=length(newsignal);
Xsignal=[0:lengthNS-1]';
lengthXs=length(Xsignal);

% Polyfit Feature Extraction

if lengthNS < 20 & lengthNS >=3
   npolycoef=lengthNS-2;
   [coeff,S]=polyfit(Xsignal,newsignal,npolycoef);
   [Y,delta]=polyconf(coeff,Xsignal,S,0.1);
   residuals=newsignal-Y;
   ErrorSqr=sum(residuals.^2);
elseif lengthNS <= 2
   coeffabs=newsignal;Y=newsignal;residuals=0;ErrorSqr=0;npolycoef=lengthNS-2;
else
   npolycoef=18;
   [coeff,S]=polyfit(Xsignal,newsignal,npolycoef);
   [Y,delta]=polyconf(coeff,Xsignal,S,0.1);
   residuals=newsignal-Y;
   ErrorSqr=sum(residuals.^2);
end

% Set coeffabs vector length to 18, then pad with 0 was nessicary

if lengthNS < 20
   coeff=[zeros(1,18-npolycoef) coeff];
end

% Plot original data and polyfitted data

if plotyn=='y'

   if lengthNS<2
```

```
      ['No plot needed, only one point above threshold.  Use y value of
point.']
   else
      figure;
      subplot(2,1,1);plot(Xsignal,newsignal,'bs-',Xsignal,Y,'ro-');axis tight;
      title('Extracted Mixed, Imaginary, Absolute EC Signal and Polyfitted
Approximation');
      legend([' = Original'],[' = Fitted'],0);
      ylabel('Magnitude')
      subplot(2,1,2);plot(residuals.^2);axis tight;
      title('Absolute Value of Residuals');
      aux=axis;
      sf=sprintf('Sum Error^2 = %.7f',ErrorSqr);
      text(aux(2)*4/6,aux(4)-((aux(4)-aux(3))/15),sf);
      ylabel('Magnitude');
      xlabel('Point')
   end

end

% Sort maxima by magnitude

%maxima=[maxima(:,2) maxima(:,1) maxima(:,3) maxima(:,4)];
%[maxsort,ind]=sort(maxima);
%maxima=maxima(ind(:,1),:);
%maxima=[maxima(:,2) maxima(:,1) maxima(:,3) maxima(:,4)];%

% Feature Vector Generation

fext=[coeff];

elseif vis_review == 'n'                % NO INFORMATION IN SIGNAL

   fprintf('\nNo Information detected in Mixed Imaginary Absolute Data\n\r')
   coeff=zeros(1,19);
   fext=[coeff];

end
```

PeakIDabs.m

```
function [maxima,k,IP,FP]=peakIDabs(signal,threshold1,threshold2,graph);
%
%      [maxima,k,IP,FP]=peakIDabs(signal,threshold1,,threshold2,graph)
%
% peakIDabs.m Locates peaks and peak intervals within abs imag data sets
%

Y=signal;
datasize = length(Y);
[r,c]=size(Y);

% finding local maxima

maxima=[];
countA = 0; countB = 0;
for i = 2 : datasize-1
   if Y(i) > threshold1 | Y(i) < threshold2
      countB = countB+1;
      if ((Y(i-1) < Y(i)) & ((Y(i+1) < Y(i)))) | ((Y(i-1) > Y(i)) & ((Y(i+1) >
Y(i))))
         countA = countA+1;
         maxima(countA,:) = [i Y(i)];
      end
      points(countB)=i;
   end
end

% Intervals around peaks that are above threshold

l=length(points);
k=1;FP=[];
IP(1)=points(1);
for i=1:l-1
   if (points(i+1) - points(i)) ~= 1
      FP(k)=points(i);
      IP(k+1)=points(i+1);
      k=k+1;
   end
```

```
   if i == l-1
      FP(k)=points(i+1);
   end
end

points;                             % points above threshold
k;                                  % k = number of data intervals above threshold
maxima;                             % peaks above threshold
IP=IP';                             % IP = Initial Point of intervals
FP=FP';                             % FP = Final Point of intervals

% Insert 0's for no peaks

if (graph=='y')

   figure;

if isempty(maxima)==1
   points=[];maxima=[];
   plot(Y);hold on;
   plot(threshold1*ones(r,1),'r--');
   plot(threshold2*ones(r,1),'r--');
   axis tight;
   aux=axis;
   sf=sprintf('%s','No Peaks Were Detected');
   text(aux(2)*4/6,aux(4)-((aux(4)-aux(3))/20),sf)
       title(['Data Number vs. Magnitude'])
       xlabel('Data Number')
   ylabel('Magnitude')
   hold off;

else

   [rmax,cmax]=size(maxima);
   plot(Y);hold on;
   plot(maxima(:,1),maxima(:,2),'ro')
   plot(threshold1*ones(r,1),'r--');
   plot(threshold2*ones(r,1),'r--');
   axis tight;
   aux=axis;
       if rmax<16
```

```
        for k=1:rmax
          sf=sprintf('%.2f,%.2f',maxima(k,1),maxima(k,2));
          text(aux(2)*4.5/6,aux(4)-((aux(4)-aux(3))/20)*k,sf)
               end
      end
        title(['Data Number vs. Magnitude'])
        xlabel('Data Number')
     ylabel('Magnitude')
     hold off;

end

end
```

MultiROIabs.m

```
function multiROIa(Y,mROI,numROI,ROIdist,threshold1,threshold2);

%
% function multiROIa(Y,mROI,numROI,ROIdist,threshold);
%
% Warning of multiple ROI's, if ROI ceneters > 32 points apart.
%

[r,c]=size(Y);
if mROI>=2
   for i=1:length(mROI)-1
      d=mROI(i+1)-mROI(i);
      if d>ROIdist
         fprintf('\rWarning: ROI centers > 32 Points Apart.   May be seperate
Flaws!\n\r')
         figure;
         peakIDabs(Y,threshold1,threshold2,'y');
         text(5,0.95*max(Y),'Warning: ROI centers > 32 Points Apart.   May be
seperate Flaws!');
         hold on
         plot(threshold1*ones(r,1),'r');plot(threshold2*ones(r,1),'r');
         for i=1:numROI
```

```
            plot(mROI(i),0,'kdiamond')
         end
         title(['Mixed Imaginary Absolute Eddy Current Signal with thresholds =
' num2str([thresholdl threshold2])])
         xlabel('Point')
         ylabel('Magnitude')
         hold off
      end
   end
end
```

CWTfext.m

```
function [geofext,imagefext,coef]=CWTfext(signal,filename,plotyn);
%
% CWTfext.m
%
% function [geofext,imagefext,coef]=CWTfext(signal,plotyn,filename);
%
% CWT Feature Extraction program.  Will load a signal, exicute a 24-level CWT
with the
% specified wavelet on the selected EC freqs.  The CWT coeff are then sent to a
geomoments
% and image-processing feature extraction routine
%
% Plotyn allows the user to plot the individual EC flaws
%

signal=signal-mean(signal);
coef=cwt(signal,1:24,'biorl.5');

% Generate cwt modulas coefficients

coef=abs(coef);
[r,c]=size(coef);

% Visual Examintaion of the CWT

if plotyn=='y'
```

```
   figure,
   surfc(coef);shading interp;
   axis([0 c 0 r 0 1.1*max(max(abs(coef)))]);
   title(['CWT Modulus for the Mixed, Complex, Differential EC Signal '
filename])
   xlabel('Distance')
   ylabel('Scale or Frequency')
end

% Feature Extraction using Geometric Moments

[geofext]=geomomentfext(coef);

% Feature Extraction using Image Processing

[imagefext]=imfext(coef,'n');

% Pad cwt_mag with 0's was signal length was < 97 (1/2 Window of 48)

if c<97
   pad=zeros(r,round((97-c)/2));
   coef=[pad coef pad];
       [r,c]=size(coef);
end
```

geomomentfext.m

```
function [geofext]=geomomentfext(coef);

% Feature Extraction from CWT coeff using geometric moments

[rcwt,ccwt]=size(coef);

[G,GN]=geomoment(coef,4,4,[0 ccwt],[1 rcwt]);
[r,c]=size(G);

% Restacking from matrix to vector
```

```
n=1;
for i=1:r
   for j=1:c
      Gvect(n)=G(i,j);
      n=n+1;
   end
end

geofext=Gvect;

% Omit x0 y0 geometric moment if desired

%omit_00=input('Does User want to omit the 00 moment (y or n). ','s');
%if omit_00=='y'
%   omit=[1 26 51 76 101];
%   geofext(omit)=[];
%end
```

geomoment.m

```
function [G,C,Cn,HU] = geomoment(M,xp,yq,Xif,Yif)
%
%Geometric Moments Calcualtion [G,C,Cn,HU]=Geomoment(M,xp,yq,Xif,Yif)
%
%This program calculates for digitial images:
%   1. geometric moments .  The x moments will be translation invariant.
%   2. Centralized Moments.  X and Y are translation invariant.
%   3. Normalized Central Moments
%   4. Hu's 7 Invariant Moments
%
%Inputs: magnitude matrix of digital image, M;
%      starting and final value row vector for x (x initial and x final), Xif;
%      starting and final value row vector for y (y initial and y final), Yif;
%      vector of geometric moments to be calculated, xp and yq.
%
%Outputs:geometric moments vector (G), Central moments (C) and Normalized C
(Cn) and HU's Moments (HU).
%
```

```
[rM,cM]=size(M);            % rM = # of row, cM = # of columns

% Extract initial and final values for x and y

xi=Xif(1,1);yi=Yif(1,1);xf=Xif(1,2);yf=Yif(1,2);

% Set up x and y value vectors

rangexold=xf-xi;
rangeyold=yf-yi;

deltax=rangexold/cM;
deltay=rangeyold/rM;

xinit=xi+deltax; % First step from initial values
yinit=yi+deltay;

x=xinit:deltax:xf;
y=yinit:deltay:yf;

% Ranging x and y axis:  x~[-1,+1] and y~[-1,+1]

rangex=(1-(-1));rangey=(1-(-1));

newx=(rangex/rangexold).*x+(1-(rangex*xf)/rangexold);
newy=(rangey/rangeyold).*y+(1-(rangey*yf)/rangeyold);

% Calculation of xbar (and ybar, though not used) for translation invariant
%             x axis of the central moment

m00=sum(sum(M));
m10=sum(sum(newx*M'));
m01=sum(sum(newy*M));
xbar=m10/m00;                     % Notice that xbar and ybar are not = meanx
and meany
ybar=m01/m00;

% Calculation of geometric moments Matrix (G), Central Moments (C) and
Normalized Central Moments (Cn)

for i=0:1:xp
```

```
   for j=0:1:yq
      Z1=((newx-xbar).^i)*M'*(newy.^j)';                    % Invariant in the x
direction, Not in the y.
      %Z=(newx.^i)*M'*(newy.^j)';
      G(i+1,j+1)=Z1;
      Z2=((newx-xbar).^i)*M'*((newy-ybar).^j)';         % Central Moment calc,
invariant in x and y directions
      C(i+1,j+1)=Z2;
      gamma=(i+j+2)/2;
      Cn(i+1,j+1)=Z2/(C(1,1)^gamma);                  % Normalized Central Moment
calc, invariant to translation and scaling
   end
end

% Zero out extremely small + and - numbers for G

for i=0:1:xp
   for j=0:1:yq
      if abs(G(i+1,j+1))<0.0000009
         G(i+1,j+1)=0;
      end
   end
end

% 7 Moments of Hu, first define needed coefficients (xp and yq must be > 3)

if xp > 3 & yq > 3

nu00=Cn(1,1);

% 1st and 2nd Moments of HU

nu11=Cn(2,2);
nu02=Cn(1,3);nu20=Cn(3,1);

phi_1=nu20+nu02;
phi_2=(nu20-nu02)^2+4*nu11^2;

% 3rd - 7th Moments of HU

nu01=Cn(1,2);nu10=Cn(2,1);
```

```
nu03=Cn(1,4);nu30=Cn(4,1);
nu12=Cn(2,3);nu21=Cn(3,2);

phi_3=(nu30-3*nu12)^2+(nu03-3*nu21)^2;
phi_4=(nu30+nu12)^2+(nu03+nu21)^2;
phi_5=(nu30-3*nu12)*(nu30+nu12)*((nu30+nu12)^2-3*(nu21+nu03)^2)+(nu03-
3*nu21)*(nu03+nu21)*((nu03+nu21)^2-3*(nu12+nu30)^2);
phi_6=(nu20-nu02)*((nu30+nu12)^2-(nu21+nu03)^2)+4*nu11*(nu30+nu12)*(nu03+nu21);
phi_7=(3*nu21-nu03)*(nu30+nu12)*((nu30+nu12)^2-3*(nu21+nu03)^2)+(nu30-
3*nu12)*(nu21+nu03)*((nu03+nu21)^2-3*(nu30+nu12)^2);

% Form HU vector with the 7 moment

HU=[phi_1 phi_2 phi_3 phi_4 phi_5 phi_6 phi_7];

else

    HU=['Degree Of Moments Not greater than 4 ... Hu"s Moments could NOT be
Calculated']

end
```

Imfext.m

```
function [imagefext]=imfext(mag,plotyn);
%
% imfext.m : Calcuates the weighted area, euler number,
%      RR and Geometric moments.  The area, euler and RR are calculated
%      after normalizationwith a specified threshold.
%
%                    plotyn = plotting, yes or no.
%
% function [imagefext]=imfext(mag,plotyn);
%

x=mag;
[r,c]=size(x);
```

```
if plotyn=='y'
surf(x);shading interp;
title('Transformation')
xlabel('Time (or Distance)')
ylabel('Scale or Frequency')
end

max=1;min=0;
[normx]=matnorm(x,max,min);

% Use 2Dstd to set threshold

threshold=1;
[magthresh,stdx,thresholdstd]=matthresh(normx,threshold);

if plotyn=='y'
figure,
surf(magthresh);shading interp;
title('Transformation')
xlabel('Time (or Distance)')
ylabel('Scale or Frequency')
end

% Greyscale image of normalized cwt

if plotyn=='y'
   figure,imshow(x);
end

% Edge detection using greyscale

bwedge=edge(magthresh,'sobel');
bw1=bwmorph(bwedge,'clean');

if plotyn=='y'
   figure,imshow(bw1);
end

% Use IP to process in Binary

bwx=im2bw(x,thresholdstd);
```

```
if plotyn=='y'
   figure,imshow(bwx)
end

bwperimx=bwperim(bwx,8);

if plotyn=='y'
   figure,imshow(bwperimx)
end

% Calc # ON pixels for total area, perimeter and Roundness
%      Ratio (RR)

areasum=sum(sum(bwx));
perimsum=sum(sum(bwperimx));

% Calculate Features

xarea=bwarea(bwx);
xeuler=bweuler(bwx,8);
RR=(perimsum^2)/(4*pi*areasum);

imagefext=[xarea xeuler RR];
```

Loadmatrix.m

```
function [uTR]=LoadMatrix(data_origin,Group,filename,data_cell,uTR_run_number);

%
% function [uTR]=LoadMatrix(data_origin,Group,filename,data_cell);
%
% This program loads the processsed feature vectors into individual feature
matrices according to feature
%      type and SG manufacturer
%

[r,c,d]=size(data_cell);
```

```
% Load each feature vector into feature vector matrices (for each type)



first=input('Is this the first cell added to the uTR cell array? ','s');
if first=='y'
   uTR=data_cell;
   if data_origin == 'P'
        eval(['save uTR_' data_origin '_' Group '_' uTR_run_number ' uTR;'])
    else
        eval(['save uTR_' data_origin '_' uTR_run_number ' uTR;'])
    end
else
    if data_origin == 'P'
        eval(['load uTR_' data_origin '_' Group '_' uTR_run_number ';'])
        uTR=cat(3,uTR,data_cell);                                  % generate the
updated uTR using data_cell and old uTR
        eval(['save uTR_' data_origin '_' Group '_' uTR_run_number ' uTR;'])
    else
        eval(['load uTR_' data_origin '_' uTR_run_number ';'])
        uTR=cat(3,uTR,data_cell);                            %    generate    the
updated uTR using data_cell and old uTR
        eval(['save uTR_' data_origin '_' uTR_run_number ' uTR;'])
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%
%
% User may save the matrix or continue load or save the constructed matrix.
%
%                        The       Train       matrix       example       name:
%
%
%
%     uTR_P_b_1 = unprocessed Training cell of PDD data for a B&W SG, run
number 1.          %
%
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                    %
% Each uTR cell array page contains the information for one flaw            %
%   in a 1X3 nested cell array                                              %
%
                                             %
%                         C1                             C2
          C3                       %
%        | Origin            | Original Signal X       | flaw type       |
       %
%        | Group             | Magnitude and Phase     | % Through Wall |
       %
% R1     | filename          | flaw location           | flaw character |
       %
%        |                   | Feature Vector          |                          |
       %
%        |                       | CWT                     |                  |
       %
%
                                       %
%
                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

uTR_shuffle.m

```
function [uTR_sorted,Z,index,sorting_matrix]=uTR_shuffle(uTR);
%
% uTR_shuffle
%
% This program shuffles rows of uTR data such that, first data_origin, then,
flaw_type
%   thenGroup and finally filename are used to shuffle the uTR pages.
%

[r,c,d]=size(uTR);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                  %
% Each uTR cell array page contains the information for one flaw            %
%   in a 1X3 nested cell array.                                             %
%
                                          %
%                        C1                             C2
          C3                      %
%       | Origin            | Original Signal X       | flaw type      |
      %
%       | Group             | Magnitude and Phase     | % Through Wall |
      %
% R1    | filename          | flaw location           | flaw character |
      %
%       |                   | Feature Vector          |                        |
      %
%       |                       | CWT                     |                  |
      %
%
                                    %
%
                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generate a cell with the pertainent shuffling info

for i=1:d

    sorting_matrix{i,1}=uTR{1,3,i}{1,1};        % flaw type
    sorting_matrix{i,2}=uTR{1,1,i}{1,1};        % Origin
    sorting_matrix{i,3}=uTR{1,1,i}{2,1};        % Group
    sorting_matrix{i,4}=uTR{1,1,i}{3,1};        % filename

end

% Flaws are sorted by the order of columns in the sorting matrix, the first col
was given highest priority
```

```
[Z,index]=sortrows(sorting_matrix);                % sorting_matrix has 4 columns
and multiple rows.
uTR_sorted=uTR(:,:,index);                          %  Each row represents a page
(example flaw)
```

uTR_statistics.m

```
function [uTR_stats,Ts]=uTR_statistics(uTR);

%
% uTR_statistics
%
%   function [uTR_stats]=uTR_statistics(uTR);
%
% This program will extract feature data in groups and perform staistical
analysis
%   on each group seperately and together
%

if nargin == 0  % No uTR loaded
    data_origin=input('Data origin ("P"DD or "E"TSS). ','s');
    run_number=input('Input uTR run number. ','s');
    if data_origin == 'P'
        Group=input('Enter steam generator type (B, C or W). ','s');
        eval(['load uTR_' data_origin '_' Group '_' run_number ';']);
    else
        eval(['load uTR_' data_origin '_' run_number ';']);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                  %
% Each uTR cell array page contains the information for one flaw             %
%   in a 1X3 nested cell array                                               %
%
                                          %
%                        C1                             C2
          C3                      %
```

```
%       | Origin          | Original Signal X      | flaw type      |
        %
%       | Group           | Magnitude and Phase    | % Through Wall |
        %
% R1    | filename        | flaw location          | flaw character |
        %
%       |                 | Feature Vector         |                |
        %
%       |                     | CWT                    |            |
        %
%
                                    %
%
                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[r,c,d]=size(uTR);

% Extracts each loadfile, FV and output from each page of the uTR

for i=1:d
    load_files{i,1}=uTR{1,1,i}{3,1};    % individual cells for each filename
    T(i,:)=uTR{1,2,i}{4,1};             % feature_vector matrices
    Output(i,:)=uTR{1,3,i}{1,1};        % flawtype
end

number_feature=input('Input the number feature families in the feature vector
(usually 5). ');
feature_cutoff=input('Enter the last position for each of the above feature
families in MATLAB format ([2 21 23 48 51]). ');

% feature_vector=[flaw_phase flaw_mag fext1Dabs fext1Ddiff geofext imagefext];
position of feature families [2 21 23 48 51]

%geomoment_extract=input('Input which geometric moments to view (in vector form
... [1 7 13 19 25]). ');

% Statistical Proccesing using only the related feature groups 1 at a time

for i=1:number_feature
```

```
    % Extract feature families

if i==1
    Tp=T(:,1:feature_cutoff(i));
else
    Tp=T(:,feature_cutoff(i-1)+1:feature_cutoff(i));
end

% Look at Geomoments 11, 22, 33, 44 and 55
%if i == 3
%    Tp=Tp(:,[1 7 13 19 25]);
%end

% finds the zero columns

del_col_1=[find(var(Tp)==0)];

% Process

[rfull,cfull]=size(Tp);
stdT=std(Tp);varT=var(Tp);
Tp=Tp./stdT(ones(rfull,1),:);
meanT=mean(Tp);
Tp=Tp-meanT(ones(rfull,1),:);

% finds low variance columns from the processed
%      data for each feature type

variance_level=0.01;
del_col_2=[find(abs(var(Tp))<=variance_level)];%del_col_v=[find(var(Tp)==0)];
del_col=[del_col_1 del_col_2];

if isempty(del_col) == 0
    del_col=sort(del_col);
    del_col(:,diff(del_col)==0)=[];
    fprintf('\nThe non-variance (defined as <= %0.6f) deleted columns for the
Flaw-type # %1.0f Feature Matrix\n',variance_level,i)
    fprintf(' was/are: ')
    fprintf(' %2.0f ',del_col')
    fprintf('\r\n')
else
```

```
    del_col=[];
end

% previous line remembers which cols =[];

Tp(:,del_col)=[];[rfull,cfull]=size(Tp);
fprintf('\nNumber of columns (variables) for feature group %1.0f = %1.0f
\n',i,cfull)

% Usefull stats

%[mu,sigma,muci,sigmaci]=normfit(Tp);
%norm_param(i,:)={mu sigma muci sigmaci};
std_mean=[stdT;meanT];
cov_matrix=cov(Tp);

% Plotting

labelxy=['Feature#1';'Feature#2';'Feature#3';'Feature#4';'Feature#5'];
flaw_color=['rgmcbk'];flaw_mark=['oxd+p.'];
if cfull >= 5 % Only for geomoments
    fprintf('\nEnter absolute coeff groupings in cell format {1:5 6:10 11:15
16:19}\n')
    groupings=input(' or geometric groupings in cell format {1:5 6:9 10:14
15:19 20:24}. ');
    [r_group,c_group]=size(groupings);
    for j=1:c_group

figure;gplotmatrix(Tp(:,groupings{j}),[],Output(:,1:2),flaw_color,flaw_mark,'',
'on','hist', ...

labelxy(1:length(groupings{j}),:),labelxy(1:length(groupings{j}),:));
        title(['Scatter Plots of Feature Group #' num2str(i) ' subgroup '
num2str(j) ' for each Feature Variable']);
        gname(load_files);
    end
else

figure;gplotmatrix(Tp,[],Output(:,1:2),flaw_color,flaw_mark,'','on','hist',labe
lxy(1:cfull,:),labelxy(1:cfull,:));
```

```
    title(['Scatter Plots of Feature Group #' num2str(i) ' for each Feature
Variable']);gname(load_files);
end

% Baysiean pdf approximation using each information for each flawtype

%for j=1:length(cfull)
%    normalpdf=normpdf(-10:0.1:10,norm_param{i,1}(j),norm_param{i,2}(j));
%    figure;plot(normalpdf);hold on;
%end
%hold off;

% Save statistics

if nargout >= 1

uTR_stats{i,1}=load_files;uTR_stats{i,2}=Tp;uTR_stats{i,3}=std_mean;uTR_stats{i
,4}=cov_matrix;uTR_stats{i,5}=del_col;
end

if i==1
    Ts=Tp;
else
    Ts=[Ts Tp];
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%   Format   of   last   cell   page,   each   row   would   be   a   FEATURE   type
%
%            (DO    NOT    USE    THIS    INFO    FOR    ANY    OTHER    PURPOSE):
%
%
%
%                 C1                       C2                                C3
C4                C5         %
```

```
%    R1    | Load  Files  (cells)  |  Transformed  Matrix  |       std_mean      |
cov_matrix  |   del_columns   |   %
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
```

uTR_process.m

```
function
[TR,TR_run_number]=uTR_process(uTR,data_origin,Group,uTR_run_number,plotyn);

%
% function [TR]=uTR_process(uTR,data_origin,Group,uTR_run_number,plotyn);
%
% At this point the user has the uTR cell.
%
%      For the uTR cell: compress and perform PCA and cwt template compression,
seperate into an array
%             with like flaws grouped together
%

[r,c,d]=size(uTR);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                  %
% Each uTR cell array page contains the information for one flaw            %
%   in a 1X3 nested cell array.                                             %
%
                                          %
%                        C1                             C2
           C3                      %
%       | Origin           | Original Signal X      | flaw type       |
       %
%       | Group            | Magnitude and Phase    | % Through Wall |
       %
% R1    | filename         | flaw location          | flaw character |
       %
```

```
%       |                    | Feature Vector         |                        |
       %
%       |                        | CWT                    |                  |
       %
%
                                         %
%
                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Breakpoint detection

[break_points,num_breaks,break_file]=break_point_b(uTR);

for i=1:d
    load_files{i,1}=uTR{1,1,i}{3,1};
    Group_matrix{i,1}=uTR{1,1,i}{2,1};
    CWT_mag(:,:,i)=uTR{1,2,i}{5,1};
    feature_vector_matrix(i,:)=uTR{1,2,i}{4,1};
    flawtype_matrix(i,:)=uTR{1,3,i}{1,1};
    flawchar_matrix{i,1}=[uTR{1,3,i}{2,1} uTR{1,3,i}{3,1}];
end

% PCA Process feature vector matrix
%
%PCA_data{1,1}=del_col;  columns are deleted from FV, no variance
%PCA_data{2,1}=std_mean;
%PCA_data{3,1}=srTR;
%PCA_data{4,1}=pcTR;
%PCA_data{5,1}=newdata;
%PCA_data{6,1}=tsquare;
%PCA_data{7,1}=QTR;
%PCA_data{8,1}=FV_reinsertion;   features   are   extracted   before   PCA   then
reinserted after PCA

[PCA_data]=PCAmatrix(feature_vector_matrix,'y');

% CWT template calculation, cwt_comp_mat was compresed template in cell array
format, cwts_raw was all processed
%      cwts (not compressed template) before compression.
```

```
[cwt_comp_mat,cwts_raw]=cwt_compress(CWT_mag,uTR);  %  cwt_comp_mat  was  cell
array 1 x # of flawtypes

% construct TR cell using the PCA results and the CWT results, raw cwts, load
files and outputPDD

if num_breaks==0    % NO FLAWTYPE SUBSETS

   col_one{1,1}=PCA_data{1,1};
   col_one{2,1}=PCA_data{2,1};
   col_one{3,1}=PCA_data{3,1};
   col_one{4,1}=PCA_data{4,1};
   col_one{5,1}=PCA_data{5,1};
   col_one{6,1}=PCA_data{6,1};
   col_one{7,1}=PCA_data{7,1};
   col_one{8,1}=PCA_data{8,1};

   col_two{1,1}=break_file;
   col_two{2,1}=cwt_comp_mat;
   col_two{3,1}=flawtype_matrix;
   col_two{4,1}=flawchar_matrix;
   col_two{5,1}=[];
   col_two{6,1}=[];
   col_two{7,1}=[];
   col_two{8,1}=cwts_raw;

   TR=cat(2,col_one,col_two);

else

for i=1:num_breaks
   if i == 1

        col_one{1,1}=PCA_data{1,1};
        col_one{2,1}=PCA_data{2,1};
        col_one{3,1}=PCA_data{3,1};
        col_one{4,1}=PCA_data{4,1};
        col_one{5,1}=PCA_data{5,1};
        col_one{6,1}=PCA_data{6,1};
        col_one{7,1}=PCA_data{7,1};
        col_one{8,1}=PCA_data{8,1};
```

```
        col_two{1,1}=break_file(1,i);
        col_two{2,1}=cwt_comp_mat(1,i);
        col_two{3,1}=flawtype_matrix(1:break_points(1),:);
        col_two{4,1}=flawchar_matrix(1:break_points(1),:);
        col_two{5,1}=PCA_data{5,1}(1:break_points(1),:);
        col_two{6,1}=PCA_data{6,1}(1:break_points(1));
        col_two{7,1}=PCA_data{7,1}(1:break_points(1));
        col_two{8,1}=cwts_raw{1,1};

        TR1=cat(2,col_one,col_two);
        TR=TR1;

    else

       col_one{1,1}=PCA_data{1,1};
        col_one{2,1}=PCA_data{2,1};
        col_one{3,1}=PCA_data{3,1};
        col_one{4,1}=PCA_data{4,1};
        col_one{5,1}=PCA_data{5,1};
        col_one{6,1}=PCA_data{6,1};
        col_one{7,1}=PCA_data{7,1};
        col_one{8,1}=PCA_data{8,1};

        col_two{1,1}=break_file(1,i);
        col_two{2,1}=cwt_comp_mat(1,i);
        col_two{3,1}=flawtype_matrix(break_points(i-1)+1:break_points(i),:);
        col_two{4,1}=flawchar_matrix(break_points(i-1)+1:break_points(i),:);
        col_two{5,1}=PCA_data{5,1}(break_points(i-1)+1:break_points(i),:);
        col_two{6,1}=PCA_data{6,1}(break_points(i-1)+1:break_points(i));
        col_two{7,1}=PCA_data{7,1}(break_points(i-1)+1:break_points(i));
        col_two{8,1}=cwts_raw{1,i};

        eval(['TR' num2str(i) '=cat(2,col_one,col_two);'])
        eval(['TR=cat(3,TR,TR' num2str(i) ');'])
    end

end

end
```

```
%   PCA_data{1,1}=del_col;
%   PCA_data{2,1}=std_mean;
%   PCA_data{3,1}=srTR;
%   PCA_data{4,1}=pcTR;
%   PCA_data{5,1}=newdata;
%   PCA_data{6,1}=tsquare;
%   PCA_data{7,1}=QTR;
%   PCA_data{8,1}=FV_reinsertion;

%   col_one{1,1}=PCA_data{1,1};
%   col_one{2,1}=PCA_data{2,1};
%   col_one{3,1}=PCA_data{3,1};
%   col_one{4,1}=PCA_data{4,1};
%   col_one{5,1}=PCA_data{5,1};
%   col_one{6,1}=PCA_data{6,1};
%   col_one{7,1}=PCA_data{7,1};
%   col_one{8,1}=FV_reinsertion;

%   col_two{1,1}=break_file(1,i);
%   col_two{2,1}=cwt_comp_mat(1,i);
%   col_two{3,1}=flawtype_matrix(1:break_points(1),:);
%   col_two{4,1}=flawchar_matrix(1:break_points(1),:);
%   col_two{5,1}=PCA_data{5,1}(1:break_points(1),:);
%   col_two{6,1}=PCA_data{6,1}(1:break_points(1),:);
%   col_two{7,1}=PCA_data{7,1}(1:break_points(1),:);
%   col_two{8,1}=cwts_raw{1,i};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                   %
% Individual Page TR setup each page represents a flawtype.                  %
%
                                      %
%                      C1                                C2
                  %
% R1    |   del_col      |      break_file for F1                        %
% R2    |   std_mean     |      cwt_comp_mat for F1
      %
% R3    |   srTR         |      flawtype_matrix for F1                  %
% R4    |   pcTR         |      flawchar_matrix for F1                  %
% R5    |   newdata      |      PCA_data{5,1} for F1                    %
```

```
% R6     |   tsquare      |       PCA_data{6,1} for F1                   %
% R7     |   QTR          |       PCA_data{7,1} for F1                   %
% R8     |FV_reinsertion  |       cwts_raw for F1                        %
%
                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if plotyn == 'y'
    for i=1:num_breaks
        figure;plot(TR{3,1,i});axis tight;
        title(['Mean-centered,  STD  normalized  Data  Features  for  flawtype  #'
num2str(i)]);
        xlabel('Data Number');ylabel('Magnitude');
        figure;plot(TR{3,1,i}');axis tight;
        title(['Mean-centered,  STD  normalized  Data  Features  for  flawtype  #'
num2str(i)]);
        xlabel('Feature Number');ylabel('Magnitude');
    end
end

% Save TR data files

TR_run_number=input('Input  TR  run  number  (actually  a  letter;  a  through  z).
','s');
if data_origin == 'P'
    eval(['save  TR_'  data_origin  '_'  Group  '_'  uTR_run_number  TR_run_number  '
TR;'])                              % save TR cell
else
    eval(['save TR_' data_origin '_' uTR_run_number TR_run_number ' TR;'])
                                      % save TR cell
end
```

PCAmatrix.m

```
function [PCA_data]=PCAmatrix(feature_vector_matrix,plotyn);

%
% PCAmatix.m
%
```

```
% PCA Calculations for a specified manufacturer (all flaw types are included).
%

[r,c,d]=size(feature_vector_matrix);

% feature_vector

T=feature_vector_matrix;

% deletes 0 variance columns

del_col=[find(var(T)==0)];
T(:,del_col)=[];

fprintf('\nThe non-variance (defined as == 0) deleted columns for the Feature
Matrix\n')
fprintf(' was/are: ')
fprintf(' %2.0f ',del_col')
fprintf('\r\n')

% mean-centering

meanT=mean(T);
T=T-meanT(ones(r,1),:);
stdT=std(T);
T=T./stdT(ones(r,1),:);
std_mean=[stdT;meanT];   % Save std and mean

%if isempty(del_col_m) == 0 | isempty(del_col_v) == 0
%    del_col=[del_col_m del_col_v];
%    del_col=sort(del_col);
%    del_col(:,diff(del_col)==0)=[];
%     fprintf('\nThe non-variance (defined as <= %0.6f) deleted columns for the
Feature Matrix\n',variance_level)
%    fprintf(' was/are: ')
%    fprintf(' %2.0f ',del_col')
%    fprintf('\r\n')
%else
%    del_col=[];
%end
%T(:,del_col)=[];
```

```
% PCA Calculations

[Tn,newdata,pcT,tsquare,QT,FV_col_del,del_T,FV_reinsertion]=PCAmatrixcalc(T);

% Saves each featuretype newdata, tsquare and QTR into one cell.
%
%      del_col ... Deleted Columns for each feature type
%      newdata ... New data matrix
%      std_mean ... std and mean for the TR featuretype
%   pcT ... Transformation matrices
%      tsquare ...
%      QT ...

if isempty(del_T)==0 & isempty(FV_col_del)==1
    newdata=[newdata del_T];                                  % Reinserts nonPCA
processed columns
elseif isempty(del_T)==0 & isempty(FV_col_del)==0
    del_col=[del_col FV_col_del];                             % Also add deleted
columns from editing
end

% Plotting

if plotyn == 'y'
    figure;plot(newdata);title('PCA         Data         Features');xlabel('Data
Number');ylabel('Magnitude');
    figure;plot(tsquare','b+');gname;title('T-squared');xlabel('Data
Number');ylabel('Magnitude');
    figure;plot(QT,'b+');gname;title('QT');xlabel('Data
Number');ylabel('Magnitude');
end

%PCA_data{1,1}=del_col;
%PCA_data{2,1}=std_mean;
%PCA_data{3,1}=srTR;
%PCA_data{4,1}=pcTR;
%%PCA_data{5,1}=newdata;
%PCA_data{6,1}=tsquare;
%PCA_data{7,1}=QTR;
%PCA_data{8,1}=FV_reinsertion;
```

```
if nargout>=1
PCA_data{1,1}=del_col;
PCA_data{2,1}=std_mean;
PCA_data{3,1}=Tn;
PCA_data{4,1}=pcT;
PCA_data{5,1}=newdata;
PCA_data{6,1}=tsquare;
PCA_data{7,1}=QT;
PCA_data{8,1}=FV_reinsertion;
end
```

PCAmatrixcalc.m

```
function
[Tn,newdata,PC,tsquare,QT,FV_col_del,del_T,FV_reinsertion]=PCAmatrixcalc(matrix
);

%
% PCAmatrixcalc.m
%
%                                                                       function
[T,newdata,PC,tsquare,QT,std_mean,FV_col_del,del_T,FV_reinsertion]=PCAmatrixcal
c(matrix);
%

T=matrix;    % feature_vector=[flaw_phase flaw_mag fext1Dabs fext1Ddiff geofext
imagefext];
            %position of feature families [2 21 23 48 51]

fprintf('\n')
edit_FV=input('Does user want to edit feature vector ("y"es or "n"o). ','s');
if edit_FV =='y'
    fprintf('\nFeature Vector families are [1:2 3:21 22:23 24:48 49:51] \n')
    fprintf(' use deleted columns to adjust families. \n\n')
    FV_col_del=input('Input FV columns for deletion (in [] format). ');
    del_T=T(:,FV_col_del);
    T(:,FV_col_del)=[];
```

```
    feature_reinsertion=input('Reinsert extracted features after PCA ("y"es or
"n"o). ','s');
    if feature_reinsertion=='y'
        FV_reinsertion=FV_col_del;
        FV_col_del=[];
    end
else
    FV_reinsertion=[];
    FV_col_del=[];
    del_T=[];
end

Tn=T;
[r,c]=size(Tn);

% Z-scores for T

[Z]=zscore(Tn);

%PCA calculations

[PC,SCORE,LATENT,tsquare]=princomp(Tn);
[pcT,varT,expT]=pcacov(cov(Tn));

% PCA explaied variances

fprintf('\n Percent Explained for TR Matrix = \n')
if c<20
    explained=100*LATENT/sum(LATENT);
    fprintf('\t\t%.6f\r',explained)
   else
    explained=100*LATENT(1:20,:)/sum(LATENT(1:20,:));
    fprintf('\t\t%.6f\r',explained)
end

% Number of PC to retain

fprintf('\r\n\n')
PCA_num=input('Input the number of PC"s to retain. ');
fprintf('\r\n')
```

```
% % retained variance

fprintf('Percent Explained for kept PCs = %.6f \n\n',sum(explained(1:PCA_num)))

% Keep selected PC's

SCORE=SCORE(:,1:PCA_num);
PC=PC(:,1:PCA_num);
newdata=SCORE;
[rnd,cnd]=size(newdata);
QT=zeros(1,r);
for i=1:r
    QT(i)=Tn(i,:)*(eye(c,c)-PC*PC')*Tn(i,:)';
end

   %pcT=pcT(:,1:PCA_num);
   %newdata=T*pcT;
   %[rnd,cnd]=size(newdata);
   %tsquare=zeros(cnd,rnd);
   %for i=1:cnd
   %    for j=1:rnd
   %
tsquare(i,j)=srT(j,:)*pcT(:,i)*(inv(diag(eigcovT(i))))*pcT(:,i)'*srT(j,:)';
   %   end
   %end
   %sprintf('\tPercent Explained for TR or TE Matrix = ')
   %if cfull<10
   %    sprintf('\t\t%.6f\r',expT)
   %else
   %sprintf('\t\t%.6f\r',expT(1:10,:))
   %end
```

proc_TR_PCA_plot.m

```
function
[Ysqr,Z,T]=proc_TR_PCA_plot(uTR,TR,data_origin,Group,uTR_run_number,TR_run_numb
er);

% proc_TR_PCA_plot.m
```

```
%
% function [Ysqr,Z,T]=proc_TR_PCA_plot(TR);
%
% Extracts TR feature matrix data (post PCA) and plots
%   PC's together allowing for outlier IDing also basic cluster
%   analysis
%

if nargin == 0
    run_number=input('Input TR data number. ','s');
    data_origin=input('Enter data origin, "P"DD or "E"TSS. ','s');
    if data_origin == 'E'
        eval(['load TR_' data_origin '_' uTR_run_number TR_run_number ';']);
    else
        Group=input('Enter data group (b, c or w for PDD). ','s');
        eval(['load TR_' data_origin '_' Group '_' uTR_run_number TR_run_number
';']);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                   %
% Individual Page TR setup each page represents a flawtype.                 %
%
                                     %
%                      C1                                C2
                  %
% R1    |   del_col     |       break_file for F1                        %
% R2    |   std_mean    |       cwt_comp_mat for F1
      %
% R3    |   srTR        |       flawtype_matrix for F1                 %
% R4    |   pcTR        |       flawchar_matrix for F1                 %
% R5    |   newdata     |       PCA_data{5,1} for F1                   %
% R6    |   tsquare     |       PCA_data{6,1} for F1                   %
% R7    |   QTR         |       PCA_data{7,1} for F1                   %
% R8    |   empty       |       cwts_raw for F1                        %
%
                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% PCA Process feature vector matrix
%
%PCA_data{1,1}=del_col;
%PCA_data{2,1}=std_mean;
%PCA_data{3,1}=srTR;
%PCA_data{4,1}=pcTR;
%PCA_data{5,1}=newdata;
%PCA_data{6,1}=tsquare;
%PCA_data{7,1}=QTR;


[r,c,d]=size(TR);
for i=1:d
    if i==1
        load_files=[TR{1,2,i}{:,:}];
        PDDoutput=[TR{3,2,i}];
        FVmatrix=[TR{5,2,i}];
    else
        load_files=[load_files;TR{1,2,i}{:,:}];
        PDDoutput=[PDDoutput;TR{3,2,i}];
        FVmatrix=[FVmatrix;TR{5,2,i}];
    end
end

[break_points,num_breaks,break_file]=break_point_b(uTR);

% 3D or 2D plot of all flaw examples IDed by flawtype.

PCA_plot(FVmatrix,break_points);
title(['FV PC Plot for processed - TR Data Set ' data_origin '_' Group '_'
uTR_run_number TR_run_number ' with ' num2str(num_breaks) ' Flawtypes'])

%PCA_plot(absmatrix,break_points_abs);
%title(['Abs. Poly Coeff Plot for processed - TR Data Set # '
num2str(data_file_number) ' with ' num2str(num_breaks) ' Flawtypes'])

% 2D PC plots for 1 & 2, 2 & 3, 1 & 3

flaw_color=['rgmcbk'];flaw_mark=['oxd+p.'];
numPC=['PC#1';'PC#2';'PC#3';'PC#4';'PC#5';'PC#6'];
```

```
figure;gplotmatrix(FVmatrix(:,1:5),[],PDDoutput(:,1:2),flaw_color,flaw_mark,'',
'on','hist',numPC(1:5,:),numPC(1:5,:));
%plot(matrix(:,1),matrix(:,2),'bo');%axis tight;xlabel('PC#1');ylabel('PC#2');
title(['PC''s #1, 2, 3, 4 & 5 Plotted for processed - TR Data Set ' data_origin
'_' Group '_'  uTR_run_number TR_run_number ' with ' num2str(num_breaks) '
Flawtypes']);
gname(load_files);
figure;gplotmatrix(FVmatrix(:,1:2),[],PDDoutput(:,1:2),flaw_color,flaw_mark,'',
'on','hist',numPC(1:2,:),numPC(1:2,:));
%plot(matrix(:,1),matrix(:,2),'bo');%axis tight;xlabel('PC#1');ylabel('PC#2');
title(['PC #1 & 2 Plot for processed - TR Data Set ' data_origin '_' Group '_'
uTR_run_number TR_run_number ' with ' num2str(num_breaks) ' Flawtypes']);
gname(load_files);
figure;gplotmatrix(FVmatrix(:,[1
3]),[],PDDoutput(:,1:2),flaw_color,flaw_mark,'','on','hist',numPC([1
3],:),numPC([1 3],:));
%plot(matrix(:,1),matrix(:,3),'bo');%axis tight;xlabel('PC#1');ylabel('PC#3');
title(['PC #1 & 3 Plot for processed - TR Data Set ' data_origin '_' Group '_'
uTR_run_number TR_run_number ' with ' num2str(num_breaks) ' Flawtypes']);
gname(load_files);
figure;gplotmatrix(FVmatrix(:,2:3),[],PDDoutput(:,1:2),flaw_color,flaw_mark,'',
'on','hist',numPC(2:3,:),numPC(2:3,:));
%plot(matrix(:,2),matrix(:,3),'bo');%axis tight;xlabel('PC#2');ylabel('PC#3');
title(['PC #2 & 3 Plot for processed - TR Data Set ' data_origin '_' Group '_'
uTR_run_number TR_run_number ' with ' num2str(num_breaks) ' Flawtypes']);
gname(load_files);

% 2D plots for 1st 5 coeffs

%flaw_color=['rgmcbk'];flaw_mark=['oxd+p.'];
%numPC=['coef#1';'coef#2';'coef#3';'coef#4';'coef#5'];
%figure;gplotmatrix(absmatrix(:,1:5),[],PDDoutput_abs(:,1:2),flaw_color,flaw_ma
rk,'','on','hist',numPC(1:5,:),numPC(1:5,:));
%title(['Coefs #1, 2, 3, 4 & 5 Plotted for processed - TR Data Set # '
num2str(data_file_number) ' for all Flawtypes']);
%gname(load_files_abs);


% CLUSTER ANALYSIS

pdist_type='Mahal';
```

```
Y=pdist(FVmatrix,pdist_type);
Ysqr=squareform(Y);
linkage_type='centeriod';
Z=linkage(Y,linkage_type);
graph_title=['Automated Linkage between Flaw Examples using ' pdist_type '
distance and ' linkage_type ' linkage'];
dendrogram(Z,0);title(graph_title);xlabel('Flaw Example #');ylabel('Level of
Linkage');
T=cluster(Z,2);

% END CLUSTER ANALYSIS
```

PCA_plot.m

```
function PCA_plot(matrix,break_points,flaw);

%
% PCA_plot.m
%
% function PCA_plot(matrix,break_points,flaw);
%
% Calculate new data flaw centers (individually and all together)
%

for i=1:length(break_points)
    if i==1
        flaw_data{1,i}=matrix(1:break_points(1),:);
    else
        flaw_data{1,i}=matrix(break_points(i-1)+1:break_points(i),:);
    end
end

if (nargin == 2) % TR file (without flaw) processing

   % Processed_TR files need to only have the following cells for each page:
   %
   %  cwt_examples, cwt_template, Feature_vector compressed data, fext1Dabs
data, PDDoutput and filenames
```

```
[a,b]=size(matrix);      % Number of PC's retianed
[r,c]=size(flaw_data);   % c = Number of flaws
flaw_mark=['ro';'gx';'md';'c+';'bp';'k.'];
flaw_center_mark=['rs';'gs';'ms';'cs';'bs';'ks'];
markerID={'Flaw  #1';'#1  Center';'Flaw  #2';'#2  Center';'Flaw  #3';'#3
Center';'Flaw #4'; ...
        '#4  Center';'Flaw  #5';'#5  Center';'Flaw  #6';'#6  Center';'Flaw
#7';'#7 Center';;'Flaw #8';'#8 Center'};

data_plot_dim=input('Does user want a "2"D or "3"D plot for multiple D data?
');

figure;

for i=1:c

   F_center=mean(flaw_data{1,i});                % center of feature type
cluster for a specified flawtype
   F_variance=var(flaw_data{1,i});               % variance of feature type
cluster for a specified flawtype

   if b == 1           % Dimension of data

   plot(flaw_data{1},flaw_mark(i,:));hold on;
   plot(F_center,flaw_center_mark(i,:));legend(markerID(1:2*c,:),-1);

   elseif b == 2

   plot(flaw_data{i}(:,1),flaw_data{i}(:,2),flaw_mark(i,:));hold on;

plot(F_center(1),F_center(2),flaw_center_mark(i,:));legend(markerID(1:2*c,:),-
1);

   elseif b > 2

       if data_plot_dim==3

plot3(flaw_data{i}(:,1),flaw_data{i}(:,2),flaw_data{i}(:,3),flaw_mark(i,:));hol
d on;grid on;
```

```
plot3(F_center(1),F_center(2),F_center(3),flaw_center_mark(i,:));legend(markerI
D(1:2*c,:),-1);
              xlabel('PC#1');ylabel('PC#2');zlabel('PC#3');title(['PC    for    '
num2str(c) ' flawtypes']);
          else
              plot(flaw_data{i}(:,1),flaw_data{i}(:,2),flaw_mark(i,:));hold on;

plot(F_center(1),F_center(2),flaw_center_mark(i,:));legend(markerID(1:2*c,:),-
1);
              xlabel('PC#1');ylabel('PC#2');title(['PC   for   '   num2str(c)   '
flawtypes']);
          end

      end

   end

   hold off;

elseif (nargin == 3) % Flaw data and TR data already processed

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
                                %
    % Individual Page TR setup each page represents a flawtype.
%
    %
                                  %
    %                    C1                                C2
                  %
    % R1     |   del_col      |       break_file for F1                  %
    % R2     |   std_mean     |       cwt_comp_mat for F1
      %
    % R3     |   srTR         |       flawtype_matrix for F1               %
    % R4     |   pcTR         |       flawchar_matrix for F1               %
    % R5     |   newdata      |       PCA_data{5,1} for F1                 %
    % R6     |   tsquare      |       PCA_data{6,1} for F1                 %
    % R7     |   QTR          |       PCA_data{7,1} for F1                 %
    % R8     |FV_reinsertion  |       cwts_raw for F1                      %
```

```
%
                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[a,b]=size(matrix);      % Number of PC's retianed
[r,c]=size(flaw_data);   % c = Number of flaws

data_plot_dim=input('Does user want a "2"D or "3"D plot for multiple D data?
');

figure;
flaw_mark=['ro';'gx';'md';'c+';'bp'];
flaw_center_mark=['rs';'gs';'ms';'cs';'bs'];
markerID={'Flaw';'Flaw #1';'#1 Center';'Flaw #2';'#2 Center';'Flaw #3';'#3
Center';'Flaw #4'; ...
        '#4 Center';'Flaw #5';'#5 Center'};

for i=1:c

   F_center=mean(flaw_data{1,i});                 % center of feature type
cluster for a specified flawtype
   F_variance=var(flaw_data{1,i});                % variance of feature type
cluster for a specified flawtype

   if b == 1           % Dimension of data

   plot(flaw(1),'k^');hold on;grid on;
   plot(flaw_data{1},flaw_mark(i,:));hold on;
   plot(F_center,flaw_center_mark(i,:));%legend(markerID(1:2*c,:),-1);

   elseif b == 2

       plot(flaw(1),flaw(2),'k^');hold on;grid on;
   plot(flaw_data{i}(:,1),flaw_data{i}(:,2),flaw_mark(i,:));hold on;

plot(F_center(1),F_center(2),flaw_center_mark(i,:));%legend(markerID(1:2*c,:),-
1);

   elseif b > 2

       if data_plot_dim==3
```

```
            plot3(flaw(1),flaw(2),flaw(3),'k^');hold on;grid on;

plot3(flaw_data{i}(:,1),flaw_data{i}(:,2),flaw_data{i}(:,3),flaw_mark(i,:));hol
d on;grid on;

plot3(F_center(1),F_center(2),F_center(3),flaw_center_mark(i,:));%legend(marker
ID(1:2*c,:),-1);
            xlabel('PC#1');ylabel('PC#2');zlabel('PC#3');title(['PC   for   '
num2str(c) ' flawtypes and Example Flaw']);
        else
            plot(flaw(1),flaw(2),'k^');hold on;grid on;
            plot(flaw_data{i}(:,1),flaw_data{i}(:,2),flaw_mark(i,:));hold on;

plot(F_center(1),F_center(2),flaw_center_mark(i,:));%legend(markerID(1:2*c,:),-
1);
            xlabel('PC#1');ylabel('PC#2');title(['PC   for   '   num2str(c)   '
flawtypes and Example Flaw']);
        end

      end
  end
else

   ['Wrong number of input arguments']

end
```

bayes_class.m

```
function
[classnonPCA,wrongnonPCA,classPCA,wrongPCA,g,BB]=bayes_class(uTR,TR,data_origin
,Group,uTR_run_number,TR_run_number,flaw);

%
% bayes_class.m
%
%                                                                      function
[classnonPCA,wrongnonPCA,classPCA,wrongPCA,g]=bayes_class(uTR,TR,data_origin,Gr
oup,run_number);
```

```
%
% Classifies Y given data X.  Number of columns  denotes number of variables,
number
%   of row for X was number of examples.  X may contain many classes. Send one
Y at
%   a time.
%
fprintf('\r\n===== Bayesian Classification Results for ===== \n\n');

if nargin == 0
    data_origin=input('Input TR data origin ("P"DD or "E"TSS). ','s');
    Group=input('Enter  steam  generator  type  (b,  c  or  w)  or  ETSS  Group  #.
','s');
    uTR_run_number=input('Input uTR run number. ','s');
    TR_run_number=input('Input TR run number. ','s');
    eval(['load  TR_'  data_origin  '_'  Group  '_'  uTR_run_number  TR_run_number
';']);
    eval(['load  uTR_'  data_origin  '_'  Group  '_'  uTR_run_number  TR_run_number
';']);
else
    fprintf('Data origin was %s',data_origin);
    fprintf('\nData Group was %s',Group);
    fprintf('\nThe uTR Data run number was %s',uTR_run_number);
    fprintf('\nThe TR Data run number was %s\n\n',TR_run_number);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                   %
% Individual Page TR setup each page represents a flawtype.                  %
%
                                      %
%                        C1                                 C2
                   %
% R1     |   del_col     |       break_file for F1                        %
% R2     |   std_mean    |       cwt_comp_mat for F1
      %
% R3     |   srTR        |       flawtype_matrix for F1                  %
% R4     |   pcTR        |       flawchar_matrix for F1                  %
% R5     |   newdata     |       PCA_data{5,1} for F1                    %
% R6     |   tsquare     |       PCA_data{6,1} for F1                    %
```

```
% R7    |   QTR          |       PCA_data{7,1} for F1                    %
% R8    |   empty        |       cwts_raw for F1                         %
%
                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[r,c,d]=size(TR);
%for i=1:d
%    if i==1
%        load_files=[TR{1,2,i}{:,:}];
%        PDDoutput=[TR{3,2,i}];
%    else
%        load_files=[load_files;TR{1,2,i}{:,:}];
%        PDDoutput=[PDDoutput;TR{3,2,i}];
%    end
%end

PCAmatrix=TR{5,1,1};

% Bhatacharyya Bounds Calculation

% k=NCHOOSEK(d,2), perms(1:4)

P1=1/d;P2=P1;sP12=sqrt(P1*P2);                    % assume equal probability for
any flawtype
k=1;
for i=1:d
    PCAdata1=TR{5,2,i};
    mean1=mean(PCAdata1);
    cov1=cov(PCAdata1);cov1=diag(diag(cov1,0));
    detcov(i)=det(cov1);
    for j=1:d
        if (j ~= i) & (j > i)
            PCAdata2=TR{5,2,j};
            mean2=mean(PCAdata2);
            cov2=cov(PCAdata2);cov2=diag(diag(cov2,0));
            mean12=mean2-mean1;cov12=(cov1+cov2)./2;
            %k12=1/8*(mean12)*(inv(cov12))*(mean12)'

k12=1/8*(mean12)*(inv(cov12))*(mean12)'+1/2*log(det(cov12)/(sqrt(det(cov1)*det(
cov2))));
```

```
            BBij(k)=sP12*exp(-k12);
            detcov12(k)=det(cov12);
            k=k+1;
        end
    end
end
BB=sum(BBij);

% Classify using non_PCA features

nonPCAclass=input('Does user want to classify using original features ("y"es or
"n"o). ','s');
fprintf('\n')
if nonPCAclass=='y'
    feature_columns=input('Define feature columns in MATLAB vector format. ');
    feature_vector=TR{3,1,1}(:,feature_columns);
    %feature_vector_bas=TR{3,1,1}(:,[1:2  14:15  40:42]);        %  Deletes  the
abspoly's and geos features
    %feature_vector_abs=TR{3,1,1}(:,[3:13]);                  % Deletes the geo
and basic features
    %feature_vector_geo=TR{3,1,1}(:,[16:39]);                 % Deletes the abs
and basic features
end

% break information

[break_points,num_breaks,break_file]=break_point_b(uTR);

% Must convert PDDoutput from string to number

for j=1:num_breaks
    [rb,cb]=size(break_file{1,j});
    if j==1
        g=j*ones(rb,1);
    else
        g=[g;j*ones(rb,1)];
    end
end

% Check what? TR or data_cell
```

```
% testing TR load data or a data_cell

if nargin == 6              % Testing a TR

    check_all=input('Does user want to check a "s"ingle flaw from file or
"a"ll? ','s');
    if check_all == 's'
        which_flaw=input('Enter which flaw [page and stack position] to check
against each FV data. ');
        flaw=TR{5,2,which_flaw(1)}(which_flaw(2),:);
    else
        flaw=PCAmatrix;
    end

    % Now classify
    classPCA=classify(flaw,PCAmatrix,g);
    wrongPCA=find(abs(diff([classPCA g],1,2))~=0)';
    fprintf('\nWrong classifications for the PCA data:\n')
    fprintf(' %2.0f ',wrongPCA)
    fprintf('\n')
    fprintf('\nPercentage of Wrong Classifications for the PCA data = %2.2f
\n\n',100*length(wrongPCA)/length(classPCA));

    if nonPCAclass == 'y'
        classnonPCA=classify(feature_vector,feature_vector,g);
        wrongnonPCA=find(abs(diff([classnonPCA g],1,2))~=0)';
        fprintf('\nWrong classifications for the basic features:\n')
        fprintf(' %2.0f ',wrongnonPCA)
        fprintf('\n')
        fprintf('\nPercentage of Wrong Classifications for the nonPCA data =
%2.2f \n\n',100*length(wrongnonPCA)/length(classnonPCA));
    else
        classnonPCA=[];wrongnonPCA=[];
    end


elseif nargin == 7   % Testing a data cell


classPCA=classify(flaw,PCAmatrix,g);classnonPCA=[];wrongnonPCA=[];wrongPCA=[];
```

```
end
```

break_point_b.m

```
function [break_points,num_breaks,break_file]=break_point_b(uTR);

% break_point_b.m
%
%  function [break_points,num_breaksbreak_file]=break_point_b(uTR);
%
% Break Point determination.
%

[r,c,d]=size(uTR);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                  %
% Each uTR cell array page contains the information for one flaw           %
%   in a 1X3 nested cell array.                                            %
%
                                          %
%                        C1                             C2
         C3                         %
%       | Origin          | Original Signal X      | flaw type      |
      %
%       | Group           | Magnitude and Phase    | % Through Wall |
      %
% R1    | filename        | flaw location          | flaw character |
      %
%       |                 | Feature Vector         |                        |
      %
%       |                     | CWT                    |                 |
      %
%
                                           %
%
                                                            %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generate a cell with the pertainent shuffling info

for i=1:d

    sorting_matrix{i,1}=uTR{1,3,i}{1,1};        % flaw type
    sorting_matrix{i,2}=uTR{1,1,i}{1,1};        % Origin
    sorting_matrix{i,3}=uTR{1,1,i}{2,1};        % Group
    sorting_matrix{i,4}=uTR{1,1,i}{3,1};        % filename

end

for i=1:d
    flaw_type_vector(i,1)=sorting_matrix{i,1}(1);
    flaw_type_vector(i,2)=sorting_matrix{i,1}(2);
end

file_number_diff=diff(flaw_type_vector',2);
[r,c]=size(sorting_matrix(:,1));

if r==1 | sum(abs(file_number_diff)) == 0
    break_points=0;num_breaks=0;break_file=sorting_matrix;
else
    break_points=find(file_number_diff~=0);
    break_points=[break_points r];
    num_breaks=length(find(file_number_diff~=0))+1;
end

for i=1:num_breaks
    if i == 1
         break_file{1,i}=sorting_matrix(1:break_points(1),:);
     else
         break_file{1,i}=sorting_matrix(break_points(i-1)+1:break_points(i),:);
     end
 end
```

NN_char.m

```
function
[net,Y,NN_char_run_number]=NN_char(uTR,TR,data_origin,Group,uTR_run_number,TR_r
un_number,filename,flaw,flaw_type);

% [net,Y]=NN_char(uTR,TR,data_origin,Group,run_number,flaw);

fprintf('\r\n===== Neural Network Characterization Results for ===== \n\n');

if nargin == 0
    data_origin=input('Input TR data origin ("P"DD or "E"TSS). ','s');
    Group=input('Enter steam generator type (b, c or w) or ETSS Group #.
','s');
    run_number=input('Input TR run number. ','s');
    eval(['load TR_' data_origin '_' Group '_' uTR_run_number TR_run_number
';']);
    eval(['load uTR_' data_origin '_' Group '_' uTR_run_number TR_run_number
';']);
else
    fprintf('\nData origin was %s',data_origin);
    fprintf('\nData Group was %s',Group);
    fprintf('\nThe Data run number was %s %s\n',uTR_run_number,TR_run_number);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                   %
% Individual Page TR setup each page represents a flawtype.            %
%
                                      %
%                        C1                                 C2
                  %
% R1     |   del_col      |       break_file for F1                 %
% R2     |   std_mean     |       cwt_comp_mat for F1
      %
% R3     |   srTR         |       flawtype_matrix for F1           %
% R4     |   pcTR         |       flawchar_matrix for F1           %
% R5     |   newdata      |       PCA_data{5,1} for F1             %
% R6     |   tsquare      |       PCA_data{6,1} for F1             %
% R7     |   QTR          |       PCA_data{7,1} for F1             %
% R8     |   empty        |       cwts_raw for F1                  %
```

```
%
                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[rTR,cTR,dTR]=size(TR);
[ruTR,cuTR,duTR]=size(uTR);

% break information

[break_points,num_breaks,break_file]=break_point_b(uTR);

if nargin == 6  % Train NN

    for i=1:dTR                         % loop number = flawtype

        PCAmatrix=TR{5,2,i};
        [a,b]=size(TR{5,2,i});
        if i>1
        clear T;
        end
        for j=1:a
            T(j,:)=TR{4,2,i}{j,1}(:,:);
        end

        % Corelation Analysis of P and T

        fprintf('\n===== Correlation Analysis for Flawtype %1.0f =====\n',i)
        eval(['CA=corrcoef([PCAmatrix T]);[rCA,cCA]=size(CA);'])
        eval(['CA' num2str(i) '=CA(1:rCA,16:cCA)'])

        % Neural Network

        fprintf('===== Neural Network Analysis for Flawtype %1.0f =====\n\n',i)
        S1=input('Number of neurons for the hidden layer (5). ');
        goal=input('Desire SSE goal (0.05). ');
        fprintf('\n')

        P = PCAmatrix';
        T = T';
```

```
        [Pn,minp,maxp,Tn,mint,maxt]=premnmx(P,T);                        % T
must be scaled
        %[Pn,meanp,stdp,Tn,meant,stdt]=prestd(P,T);
        [R,Q]=size(Pn);[S2,Q]=size(Tn);                                 % Tn
and T are the same size
        %      NEWFF(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF) takes,
        %        PR  - Rx2 matrix of min and max values for R input elements.
        %        Si  - Size of ith layer, for Nl layers.
        %        TFi - Transfer function of ith layer, default = 'tansig'.
        %        BTF - Backprop network training function, default = 'trainlm'.
        %        BLF  -  Backprop  weight/bias  learning  function,  default  =
'learngdm'.
        %        PF  - Performance function, default = 'mse'.
        %      and returns an N layer feed-forward backprop network.
        net = newff(minmax(Pn),[S1 S2],{'tansig' 'purelin'},'trainbr'); % Setup
NN
        net.trainParam.goal = goal;
        %net.trainParam.mc = 0.95;
        net.trainParam.show = 10;
        net.trainParam.epochs = 200;
        net = train(net,Pn,Tn);                                          % use
Tn
        Yn = sim(net,Pn);
        Y = postmnmx(Yn,mint,maxt);                                    % using
this since Tn was scaled
        %Y = poststd(Yn,meant,stdt);

        MSE_TnYn=sum(sum((Tn-Yn).^2))/(S2*Q);                           % MSE
between Tn and Yn matrices

        fprintf('\nTarget  Flaw  characterization  vector  for  flawtype  #  %1.0f
\n',i)
        T
        fprintf('\nNN  Flaw  characterization  vector  for  flawtype  #  =  %1.0f
\n',i)
        Y
        fprintf('\nThe  MSE  between  Tn  and  Yn  for  flawtype  #  %1.0f  =  %.4f
\n\n',i,MSE_TnYn)

        for k=1:S2
```

```
            figure;[m,b,r]   =   postreg(Y(k,:),T(k,:));%   Performs   a   linear
regression between the network and the target
            title(['Correlation   between   Target   Data   and   Output   Data   for
flawtype ' num2str(i) ' variable ' num2str(k)])
            fprintf('Correlation Coeff between T and Y for flawtype # %1.0f
variable %1.0f = %1.4f \n',i,k,r)
        end

        fprintf('\n')

        save_net=input('Does user want to save the generated NN and info ("y"es
or "n"o)? ','s');
        if save_net == 'y'
            NN_char_run_number=input('NN char run number (usually 1, 2 ... with
5al being full run ID). ','s');
            eval(['save net_char_' data_origin '_' uTR_run_number TR_run_number
NN_char_run_number '_' num2str(i) ' net minp maxp mint maxt;']);
            %eval(['save   net_class_'   data_origin   '_'   NN_run_number   '_'
num2str(i) ' net meanp stdp meant stdt;']);
        else
            NN_char_run_number=[];
        end
        fprintf('\n')

    end

end

if nargin == 9      % Characterize flaw with NN

    NN_char_run_number=input('NN char run number. ','s');
    eval(['load   net_char_'   data_origin   '_'   uTR_run_number   TR_run_number
NN_char_run_number '_' num2str(flaw_type) ';']);
    [r,c]=size(flaw);

    % mnmx scaling

    scaled_flaw         =         2.*(flaw-minp(ones(r,1),:))./(maxp(ones(r,1),:)-
minp(ones(r,1),:)) - ones(r,1);

    unscaled_Y = sim(net,scaled_flaw');
```

```
    Y = postmnmx(unscaled_Y,mint,maxt);

end
```

Xvalidate.m

```
% Xvalidate.m

clear predicted_class actual_class;
[r,c,d]=size(uTR);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                  %
% Each uTR cell array page contains the information for one flaw          %
%   in a 1X3 nested cell array                                            %
%
                                          %
%                       C1                            C2
          C3                        %
%       | Origin          | Original Signal X      | flaw type      |
      %
%       | Group           | Magnitude and Phase    | % Through Wall |
      %
% R1    | filename        | flaw location          | flaw character |
      %
%       |                 | Feature Vector         |                       |
      %
%       |                     | CWT                    |                |
      %
%
                                    %
%
                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% feature_vector=[flaw_phase flaw_mag fext1Dabs fext1Ddiff geofext imagefext];
% position of feature families [2 21 23 48 51]
```

```
% redivided and assemble 23 subgroups or sg %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

k=1;
for i=1:d
    if mod(i,23)~=0
        %eval(['sg' num2str(mod(i,4)) '(k,:)=uTR{1,2,i}{4,1};'])
        %eval(['sgf' num2str(mod(i,4)) '(k,:)=uTR{1,1,i}{3,1} ;'])
        %eval(['sgt' num2str(mod(i,4)) '(k,:)=uTR{1,3,i}{1,1} ;'])
        eval(['sg' num2str(mod(i,23)) '{k,1}=uTR{1,2,i}{4,1};'])
        eval(['sg' num2str(mod(i,23)) '{k,2}=uTR{1,1,i}{3,1};'])
        eval(['sg' num2str(mod(i,23)) '{k,3}=uTR{1,3,i}{1,1};'])
    else
        %eval(['sg' num2str(4) '(k,:)=uTR{1,2,i}{4,1};'])
        %eval(['sgf' num2str(4) '(k,:)=uTR{1,1,i}{3,1};'])
        %eval(['sgt' num2str(4) '(k,:)=uTR{1,3,i}{1,1};'])
        eval(['sg' num2str(23) '{k,1}=uTR{1,2,i}{4,1};'])
        eval(['sg' num2str(23) '{k,2}=uTR{1,1,i}{3,1};'])
        eval(['sg' num2str(23) '{k,3}=uTR{1,3,i}{1,1};'])
        k=k+1;
    end
end

feature_breaks=[2 21 23 48 51];
D=1;
for i=1:length(feature_breaks)  % feature families

    if i==1
        del_group=1:feature_breaks(1);
    else
        del_group=feature_breaks(i-1)+1:feature_breaks(i);
    end

    for j=1:23                    % subgroup formations
        clear T X C Y gC gT newdataC newdataT Del_GroupX Del_GroupY;
        z=1:23;
        z(j)=[];                  % deletes number j from Z
        eval(['X=cat(1,sg'    num2str(z(1))    ',sg'    num2str(z(2))    ',sg'
num2str(z(3)) ',sg' num2str(z(4)) ...
                ',sg'  num2str(z(5))  ',sg'  num2str(z(6))  ',sg'  num2str(z(7))
',sg' num2str(z(8)) ...
```

```
                ',sg' num2str(z(9)) ',sg' num2str(z(10)) ',sg' num2str(z(11))
',sg' num2str(z(12)) ...
                ',sg' num2str(z(13)) ',sg' num2str(z(14)) ',sg' num2str(z(15))
',sg' num2str(z(16)) ...
                ',sg' num2str(z(17)) ',sg' num2str(z(18)) ',sg' num2str(z(19))
',sg' num2str(z(20)) ...
                ',sg' num2str(z(21)) ',sg' num2str(z(22)) ');'])        % X =
Training
        eval(['Y=sg' num2str(j) ';'])                                   % Y =
Checking
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));             % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            Del_GroupX(k,:)=X{k,1}(:,del_group);      % Retains the extracted
feature group
            X{k,1}(:,del_group)=[];                 % Extracts feature family
            T(k,:)=X{k,1};                          % extracts Training data
        end
        for k=1:rY
            Del_GroupY(k,:)=Y{k,1}(:,del_group);      % Retains the extracted
feature group
            Y{k,1}(:,del_group)=[];                 % Extracts feature family
            C(k,:)=Y{k,1};                          % extracts Checking data
        end

        % Pre-Processing C and T
        del_col=[find(var(T)==0)];                  % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
        [rC,cC]=size(C);
        % mean-centering
        meanT=mean(T);
        T=T-meanT(ones(rT,1),:);
        C=C-meanT(ones(rC,1),:);
        stdT=std(T);
```

```
        T=T./stdT(ones(rT,1),:);
        C=C./stdT(ones(rC,1),:);

        % Pre-Processing Deleted Groups
        del_col_R=[find(var(Del_GroupX)==0)];                        % 0 Varaince
cols
        Del_GroupX(:,del_col_R)=[];
        Del_GroupY(:,del_col_R)=[];
        [rT,cT]=size(Del_GroupX);
        [rC,cC]=size(Del_GroupY);
        % mean-centering
        meanT=mean(Del_GroupX);
        Del_GroupX=Del_GroupX-meanT(ones(rT,1),:);
        Del_GroupY=Del_GroupY-meanT(ones(rC,1),:);
        stdT=std(Del_GroupX);
        Del_GroupX=Del_GroupX./stdT(ones(rT,1),:);
        Del_GroupY=Del_GroupY./stdT(ones(rC,1),:);

        % Create classification vectors, must use numbers

        for L=1:rT
            if prod(double(X{L,3}))==5621   % IM
                gT(L)=1;
           elseif prod(double(X{L,3}))==5655   % WA
                gT(L)=2;
            elseif prod(double(X{L,3}))==5840   % PI
                gT(L)=3;
            elseif prod(double(X{L,3}))==6048   % TH
                gT(L)=4;
            end
        end
        for L=1:rC
            if prod(double(Y{L,3}))==5621   % IM
                gC(L)=1;
            elseif prod(double(Y{L,3}))==5655   % WA
                gC(L)=2;
            elseif prod(double(Y{L,3}))==5840   % PI
                gC(L)=3;
            elseif prod(double(Y{L,3}))==6048   % TH
                gC(L)=4;
            end
```

```
        end

        % classify using raw features

        if length(del_group)<18
            predicted_class_R{D,1}=classify(Del_GroupY,Del_GroupX,gT)';
            actual_class_R{D,1}=gC;
            incorrect_R{D,1}=find(abs(diff([predicted_class_R{D,1}'
actual_class_R{D,1}'],1,2))~=0)';
            if                    isempty(find(abs(diff([predicted_class_R{D,1}'
actual_class_R{D,1}'],1,2))~=0)')==1
                incorrect_R{D,1}=0;
                family_incorrect_R{j,1}=0;
            else
                incorrect_R{D,1}=find(abs(diff([predicted_class_R{D,1}'
actual_class_R{D,1}'],1,2))~=0)';
                family_incorrect_R{j,1}=find(abs(diff([predicted_class_R{D,1}'
actual_class_R{D,1}'],1,2))~=0)';
            end
        end

        %PCA calculations
        [PC,SCORE,LATENT,tsquare]=princomp(T);
        [pcT,varT,expT]=pcacov(cov(T));
        % PCA explaied variances
        PCA_num=15;
        %fprintf('\n Percent Explained for TR Matrix = \n')
        explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
        %fprintf('\t\t%.6f\r',explained)
        % retained variance
        %fprintf('\nPercent    Explained    for    kept    PCs    =    %.6f
\n\n',sum(explained(1:PCA_num)))
        % Keep selected PC's
        SCORE=SCORE(:,1:PCA_num);
        PC=PC(:,1:PCA_num);
        newdataT=SCORE;            % Transformed T
        newdataC=C*PC;             % Transformed C

        % classify using PCs

        predicted_class{D,1}=classify(newdataC,newdataT,gT)';
```

```
        actual_class{D,1}=gC;

        % Keeps up with incorrects

        if                              isempty(find(abs(diff([predicted_class{D,1}'
actual_class{D,1}'],1,2))~=0)')==1
            incorrect{D,1}=0;
            family_incorrect{j,1}=0;
        else
            incorrect{D,1}=find(abs(diff([predicted_class{D,1}'
actual_class{D,1}'],1,2))~=0)';
            family_incorrect{j,1}=find(abs(diff([predicted_class{D,1}'
actual_class{D,1}'],1,2))~=0)';
        end



        D=D+1;

    end

    % family incorrects using raw data
    if length(del_group)<18
        [r,c]=size(family_incorrect_R);
        for n=1:r
            if family_incorrect_R{n,1}==0
              family_Incorrect_total_R(n)=0;
            else
              family_Incorrect_total_R(n)=length(incorrect_R{n,1});
            end
        end
        deleted_family=i;
        family_Incorrect_percentage_R=sum(family_Incorrect_total_R)/(r*4)*100;
        fprintf('The incorrect percentage for raw deleted family %1.0f = %2.2f
\n',i,family_Incorrect_percentage_R)
        clear family_incorrect_R family_Incorrect_total_R;
    end

    % family incorrects
    [r,c]=size(family_incorrect);
     for n=1:r
         if family_incorrect{n,1}==0
```

```
            family_Incorrect_total(n)=0;
        else
            family_Incorrect_total(n)=length(incorrect{n,1});
        end
    end
    deleted_family=i;
    family_Incorrect_percentage=sum(family_Incorrect_total)/(r*4)*100;
    fprintf('The incorrect percentage without deleted family %1.0f = %2.2f
\n',i,family_Incorrect_percentage)
    clear family_incorrect family_Incorrect_total;

end

% all results

[r,c]=size(incorrect);
for i=1:r
    if incorrect{i,1}==0
        Incorrect_total(i)=0;
    else
        Incorrect_total(i)=length(incorrect{i,1});
    end
end

Incorrect_percentage=sum(Incorrect_total)/(r*4)*100;
fprintf('The average incorrect percentage for deleted families = %2.2f
\n',Incorrect_percentage)

% All Feature Families included %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D=1;
for j=1:23                    % subgroup formations
        clear T X C Y gC gT newdataC newdataT;
        z=1:23;
        z(j)=[];                % deletes number j from Z
        eval(['X=cat(1,sg' num2str(z(1)) ',sg' num2str(z(2)) ',sg'
num2str(z(3)) ',sg' num2str(z(4)) ...
                ',sg' num2str(z(5)) ',sg' num2str(z(6)) ',sg' num2str(z(7))
',sg' num2str(z(8)) ...
                ',sg' num2str(z(9)) ',sg' num2str(z(10)) ',sg' num2str(z(11))
',sg' num2str(z(12)) ...
```

```
                ',sg' num2str(z(13)) ',sg' num2str(z(14)) ',sg' num2str(z(15))
',sg' num2str(z(16)) ...
                ',sg' num2str(z(17)) ',sg' num2str(z(18)) ',sg' num2str(z(19))
',sg' num2str(z(20)) ...
                ',sg' num2str(z(21)) ',sg' num2str(z(22)) ');'])        % X =
Training
        eval(['Y=sg' num2str(j) ';'])                                   % Y =
Checking
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));             % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            T(k,:)=X{k,1};                          % extracts Training data
        end
        for k=1:rY
            C(k,:)=Y{k,1};                          % extracts Checking data
        end
        % Pre-Processing
        del_col=[find(var(T)==0)];                  % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
        [rC,cC]=size(C);
        % mean-centering
        meanT=mean(T);
        T=T-meanT(ones(rT,1),:);
        C=C-meanT(ones(rC,1),:);
        stdT=std(T);
        T=T./stdT(ones(rT,1),:);
        C=C./stdT(ones(rC,1),:);
        % Create classification vectors, must use numbers

        for L=1:rT
            if prod(double(X{L,3}))==5621   % IM
                gT(L)=1;
          elseif prod(double(X{L,3}))==5655   % WA
                gT(L)=2;
```

```
        elseif prod(double(X{L,3}))==5840   % PI
            gT(L)=3;
        elseif prod(double(X{L,3}))==6048   % TH
            gT(L)=4;
        end
    end
    for L=1:rC
        if prod(double(Y{L,3}))==5621   % IM
            gC(L)=1;
        elseif prod(double(Y{L,3}))==5655   % WA
            gC(L)=2;
        elseif prod(double(Y{L,3}))==5840   % PI
            gC(L)=3;
        elseif prod(double(Y{L,3}))==6048   % TH
            gC(L)=4;
        end
    end

    %PCA calculations
    [PC,SCORE,LATENT,tsquare]=princomp(T);
    [pcT,varT,expT]=pcacov(cov(T));
    % PCA explaied variances
    PCA_num=15;
    %fprintf('\n Percent Explained for TR Matrix = \n')
    explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
    %fprintf('\t\t%.6f\r',explained)
    % retained variance
    %fprintf('\nPercent     Explained     for     kept     PCs     =     %.6f
\n\n',sum(explained(1:PCA_num)))
    % Keep selected PC's
    SCORE=SCORE(:,1:PCA_num);
    PC=PC(:,1:PCA_num);
    newdataT=SCORE;              % Transformed T
    newdataC=C*PC;               % Transformed C

    % classify using PCs

    predicted_class_all_4{D,1}=classify(newdataC,newdataT,gT)';
    actual_class_all_4{D,1}=gC;

    % Keeps up with incorrects
```

```
        wrong_flaw=find(abs(diff([predicted_class_all_4{D,1}'
actual_class_all_4{D,1}'],1,2))~=0)';
        if isempty(wrong_flaw)==1
            incorrect_all_4{D,1}=0;
        else
            incorrect_all_4{D,1}=wrong_flaw;
            [t,u]=size(wrong_flaw);
            for a=1:u
                incorrect_all_4{D,1+a}=Y{wrong_flaw(a),2};
            end
        end

        D=D+1;

end

% all results

[r,c]=size(incorrect_all_4);
for i=1:r
    if incorrect_all_4{i,1}==0
        Incorrect_total_all_4(i)=0;
    else
        Incorrect_total_all_4(i)=length(incorrect_all_4{i,1});
    end
end

Incorrect_percentage_all_4=sum(Incorrect_total_all_4)/(r*4)*100;
fprintf('The incorrect percentage using all feature families (4 extracted) =
%2.2f \n',Incorrect_percentage_all_4)

% Extract One %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D=1;
for j=1:92                   % subgroup formations

        clear T X C Y gT gC newdataC newdataT Del_GroupX Del_GroupY;
        % extract Y and X
        Y{1,1}=uTR{1,2,j}{4,1};Y{1,2}=uTR{1,1,j}{3,1};Y{1,3}=uTR{1,3,j}{1,1};
        X=uTR;X(:,:,j)=[];
```

```
for P=1:91
    X1{P,1}=X{1,2,P}{4,1};X1{P,2}=X{1,1,P}{3,1};X1{P,3}=X{1,3,P}{1,1};
end
X=X1;clear X1;
[rY,cY]=size(Y);
[rX,cX]=size(X);
[Z,index]=sortrows(X(:,3));                % sorting_matrix has 1 columns
and multiple rows.
X=X(index,:);
[Z,index]=sortrows(Y(:,3));
Y=Y(index,:);clear Z;
for k=1:rX
    T(k,:)=X{k,1};                          % extracts Training data
end
for k=1:rY
    C(k,:)=Y{k,1};                          % extracts Checking data
end
% Pre-Processing C and T
del_col=[find(var(T)==0)];                  % 0 Varaince cols
T(:,del_col)=[];
C(:,del_col)=[];
[rT,cT]=size(T);
[rC,cC]=size(C);
% mean-centering
meanT=mean(T);
T=T-meanT(ones(rT,1),:);
C=C-meanT(ones(rC,1),:);
stdT=std(T);
T=T./stdT(ones(rT,1),:);
C=C./stdT(ones(rC,1),:);

% Create classification vectors, must use numbers

for L=1:rT
    if prod(double(X{L,3}))==5621   % IM
        gT(L)=1;
   elseif prod(double(X{L,3}))==5655   % WA
        gT(L)=2;
    elseif prod(double(X{L,3}))==5840   % PI
        gT(L)=3;
    elseif prod(double(X{L,3}))==6048   % TH
```

```
                gT(L)=4;
            end
        end

        if prod(double(Y{1,3}))==5621   % IM
                gC=1;
        elseif prod(double(Y{1,3}))==5655   % WA
                gC=2;
        elseif prod(double(Y{1,3}))==5840   % PI
                gC=3;
        elseif prod(double(Y{1,3}))==6048   % TH
                gC=4;
        end

        %PCA calculations
        [PC,SCORE,LATENT,tsquare]=princomp(T);
        [pcT,varT,expT]=pcacov(cov(T));
        % PCA explaied variances
        PCA_num=15;
        %fprintf('\n Percent Explained for TR Matrix = \n')
        explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
        %fprintf('\t\t%.6f\r',explained)
        % retained variance
        %fprintf('\nPercent     Explained     for     kept     PCs     =     %.6f
\n\n',sum(explained(1:PCA_num)))
        % Keep selected PC's
        SCORE=SCORE(:,1:PCA_num);
        PC=PC(:,1:PCA_num);
        newdataT=SCORE;            % Transformed T
        newdataC=C*PC;             % Transformed C

        % classify using PCs

        predicted_class_all_1{D,1}=classify(newdataC,newdataT,gT)';
        actual_class_all_1{D,1}=gC;

        % Keeps up with incorrects

        wrong_flaw_all_1=find(abs(diff([predicted_class_all_1{D,1}
actual_class_all_1{D,1}]))~=0)';
        if isempty(wrong_flaw_all_1)==1
```

```
            incorrect_all_1{D,1}=0;
            incorrect_all_1{D,2}=0;
        else
            incorrect_all_1{D,1}=wrong_flaw_all_1;
            incorrect_all_1{D,2}=Y{1,2};
        end


        D=D+1;

end


% all results extracting one

[r,c]=size(incorrect_all_1);
for i=1:r
    if incorrect_all_1{i,1}==0
        Incorrect_total_all_1(i)=0;
    else
        Incorrect_total_all_1(i)=length(incorrect_all_1{i,1});
    end
end

Incorrect_percentage_all_1=sum(Incorrect_total_all_1)/r*100;
fprintf('The average incorrect percentage (extracting one) using all feature
families = %2.2f \n',Incorrect_percentage_all_1)
```

Xvalidate_B.m

```
% Xvalidate_B.m

clear    predicted_class    actual_class    predicted_class_R    actual_class_R
predicted_class_R1 actual_class_R1;

[r,c,d]=size(uTR);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                 %
```

```
% Each uTR cell array page contains the information for one flaw           %
%   in a 1X3 nested cell array                                             %
%
                                         %
%                      C1                            C2
          C3                       %
%       | Origin          | Original Signal X      | flaw type      |
      %
%       | Group           | Magnitude and Phase    | % Through Wall |
      %
% R1    | filename        | flaw location          | flaw character |
      %
%       |                 | Feature Vector         |                       |
      %
%       |                     | CWT                    |               |
      %
%
                                    %
%
                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% feature_vector=[flaw_phase flaw_mag fext1Dabs fext1Ddiff geofext imagefext];
% position of feature families [2 21 23 48 51]



% Extract One, Use one feaute group at a time %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

feature_breaks=[2 21 23 48 51];
D=1;E=1;
for i=1:length(feature_breaks)  % feature families

    D=1;

    if i==1
        group=1:feature_breaks(1);
    else
        group=feature_breaks(i-1)+1:feature_breaks(i);
    end
```

```
    if length(group) < 20

    for j=1:92                    % subgroup formations
        clear T X X1 C Y gC gT;
        % extract Y and X
        Y{1,1}=uTR{1,2,j}{4,1};Y{1,2}=uTR{1,1,j}{3,1};Y{1,3}=uTR{1,3,j}{1,1};
        X1=uTR;
        X1(:,:,j)=[];
        for P=1:91
            X{P,1}=X1{1,2,P}{4,1};X{P,2}=X1{1,1,P}{3,1};X{P,3}=X1{1,3,P}{1,1};
        end
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));              % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            T(k,:)=X{k,1}(:,group);    % Retains the extracted feature group
        end
        for k=1:rY
            C(k,:)=Y{k,1}(:,group);    % Retains the extracted feature group
        end

        % Pre-Processing C and T
        del_col=[find(var(T)==0)];                  % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
        [rC,cC]=size(C);
        % mean-centering
        meanT=mean(T);
        T=T-meanT(ones(rT,1),:);
        C=C-meanT(ones(rC,1),:);
        stdT=std(T);
        T=T./stdT(ones(rT,1),:);
        C=C./stdT(ones(rC,1),:);


        % Create classification vectors, must use numbers
```

```
        for L=1:rT
            if prod(double(X{L,3}))==5621   % IM
                gT(L)=1;
           elseif prod(double(X{L,3}))==5655   % WA
                gT(L)=2;
            elseif prod(double(X{L,3}))==5840   % PI
                gT(L)=3;
            elseif prod(double(X{L,3}))==6048   % TH
                gT(L)=4;
            end
        end
        for L=1:rC
            if prod(double(Y{L,3}))==5621   % IM
                gC(L)=1;
            elseif prod(double(Y{L,3}))==5655   % WA
                gC(L)=2;
            elseif prod(double(Y{L,3}))==5840   % PI
                gC(L)=3;
            elseif prod(double(Y{L,3}))==6048   % TH
                gC(L)=4;
            end
        end


        % classify using raw features

            predicted_class_R1{D,1}=classify(C,T,gT)';
            actual_class_R1{D,1}=gC;
            wrong_flaw_R1=find(abs(diff([predicted_class_R1{D,1}
    actual_class_R1{D,1}]))~=0)';

            if isempty(wrong_flaw_R1)==1
                incorrect_R1{D,1}=0;
                incorrect_R1{D,2}=0;
                incorrect_R1_all{E,1}=0;
                incorrect_R1_all{E,2}=0;
            else
                incorrect_R1{D,1}=wrong_flaw_R1;
                incorrect_R1{D,2}=Y{1,2};
                incorrect_R1_all{E,1}=wrong_flaw_R1;
```

```
                incorrect_R1_all{E,2}=Y{1,2};
            end



        D=D+1;E=E+1;

    end

    % family incorrects using raw data

        [r,c]=size(incorrect_R1);
        for n=1:r
            if isempty(incorrect_R1{n,1}) == 1 | incorrect_R1{n,1} == 0
              Incorrect_total_R1(n)=0;
            else
              Incorrect_total_R1(n)=length(incorrect_R1{n,1});
            end
        end

        Incorrect_percentage_R1=sum(Incorrect_total_R1)/r*100;
        fprintf('The incorrect percentage (extract one, NO PCA) for family
%1.0f = %2.2f \n',i,Incorrect_percentage_R1)

        clear incorrect_R1;
        clear Incorrect_total_R1;
        clear Incorrect_percentage_R1;
        clear predicted_class_R1;
        clear actual_class_R1;


    end

    clear D;

end

% all results for that feature family

[r,c]=size(incorrect_R1_all);
for i=1:r
    if incorrect_R1_all{i,1}==0
```

```
        Incorrect_total_R1_all(i)=0;
    else
        Incorrect_total_R1_all(i)=length(incorrect_R1_all{i,1});
    end
end

Incorrect_percentage_R1_all=sum(Incorrect_total_R1_all)/r*100;
fprintf('The average incorrect percentage (extract one, NO PCA) for all
families = %2.2f \n',Incorrect_percentage_R1_all)


% Extract 1, use CWT information %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D=1;
CWT=[24:51];
for j=1:92                    % subgroup formations
        clear T X X1 C Y gC gT newdataC newdataT;
        % extract Y and X
        Y{1,1}=uTR{1,2,j}{4,1};Y{1,2}=uTR{1,1,j}{3,1};Y{1,3}=uTR{1,3,j}{1,1};
        X1=uTR;
        X1(:,:,j)=[];
        for P=1:91
            X{P,1}=X1{1,2,P}{4,1};X{P,2}=X1{1,1,P}{3,1};X{P,3}=X1{1,3,P}{1,1};
        end
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));              % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            T(k,:)=X{k,1}(:,CWT);    % Retains the extracted feature group
        end
        for k=1:rY
            C(k,:)=Y{k,1}(:,CWT);    % Retains the extracted feature group
        end
        % Pre-Processing C and T
        del_col=[find(var(T)==0)];                % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
```

```
[rT,cT]=size(T);
[rC,cC]=size(C);
% mean-centering
meanT=mean(T);
T=T-meanT(ones(rT,1),:);
C=C-meanT(ones(rC,1),:);
stdT=std(T);
T=T./stdT(ones(rT,1),:);
C=C./stdT(ones(rC,1),:);
%PCA calculations
[PC,SCORE,LATENT,tsquare]=princomp(T);
[pcT,varT,expT]=pcacov(cov(T));
% PCA explaied variances
PCA_num=15;
%fprintf('\n Percent Explained for TR Matrix = \n')
explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
%fprintf('\t\t%.6f\r',explained)
% retained variance
%fprintf('\nPercent    Explained    for    kept    PCs    =    %.6f
\n\n',sum(explained(1:PCA_num)))
% Keep selected PC's
SCORE=SCORE(:,1:PCA_num);
PC=PC(:,1:PCA_num);
newdataT=SCORE;              % Transformed T
newdataC=C*PC;               % Transformed C

% Create classification vectors, must use numbers

for L=1:rT
    if prod(double(X{L,3}))==5621   % IM
        gT(L)=1;
   elseif prod(double(X{L,3}))==5655   % WA
        gT(L)=2;
    elseif prod(double(X{L,3}))==5840   % PI
        gT(L)=3;
    elseif prod(double(X{L,3}))==6048   % TH
        gT(L)=4;
    end
end
for L=1:rC
    if prod(double(Y{L,3}))==5621   % IM
```

```
            gC(L)=1;
        elseif prod(double(Y{L,3}))==5655   % WA
            gC(L)=2;
        elseif prod(double(Y{L,3}))==5840   % PI
            gC(L)=3;
        elseif prod(double(Y{L,3}))==6048   % TH
            gC(L)=4;
        end
    end

    % classify using raw features

        predicted_class_CWT{D,1}=classify(newdataC,newdataT,gT)';
        actual_class_CWT{D,1}=gC;
        wrong_flaw_CWT=find(abs(diff([predicted_class_CWT{D,1}
actual_class_CWT{D,1}]))~=0)';

        if isempty(wrong_flaw_CWT)==1 | wrong_flaw_CWT == 0
            incorrect_CWT{D,1}=0;
            incorrect_CWT{D,2}=0;
        else
            incorrect_CWT{D,1}=wrong_flaw_CWT;
            incorrect_CWT{D,2}=Y{1,2};
        end



    D=D+1;

end

  % family incorrects using raw data

    [r,c]=size(incorrect_CWT);
    for n=1:r
        if isempty(incorrect_CWT{n,1}) == 1 | incorrect_CWT{n,1} == 0
          Incorrect_total_CWT(n)=0;
        else
          Incorrect_total_CWT(n)=length(incorrect_CWT{n,1});
        end
    end
    Incorrect_percentage_CWT=sum(Incorrect_total_CWT)/r*100;
```

```
        fprintf('The incorrect percentage (extract one, PCA) only using CWT =
%2.2f \n',Incorrect_percentage_CWT)

        %clear incorrect_CWT Incorrect_total_CWT Incorrect_percentage_CWT;


% Extract One without CWT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D=1;
del_CWT=[24:51];
for j=1:92                   % subgroup formations

        clear T X C Y gT gC newdataC newdataT;
        % extract Y and X
        Y{1,1}=uTR{1,2,j}{4,1};Y{1,2}=uTR{1,1,j}{3,1};Y{1,3}=uTR{1,3,j}{1,1};
        X=uTR;X(:,:,j)=[];
        for P=1:91
            X1{P,1}=X{1,2,P}{4,1};X1{P,2}=X{1,1,P}{3,1};X1{P,3}=X{1,3,P}{1,1};
        end
        X=X1;clear X1;
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));              % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            X{k,1}(:,del_CWT)=[];                % Extracts feature family
            T(k,:)=X{k,1};                         % extracts Training data
        end
        for k=1:rY
            Y{k,1}(:,del_CWT)=[];                % Extracts feature family
            C(k,:)=Y{k,1};                         % extracts Checking data
        end

        % Pre-Processing C and T
        del_col=[find(var(T)==0)];                 % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
```

```
[rC,cC]=size(C);
% mean-centering
meanT=mean(T);
T=T-meanT(ones(rT,1),:);
C=C-meanT(ones(rC,1),:);
stdT=std(T);
T=T./stdT(ones(rT,1),:);
C=C./stdT(ones(rC,1),:);

% Create classification vectors, must use numbers

for L=1:rT
    if prod(double(X{L,3}))==5621   % IM
        gT(L)=1;
   elseif prod(double(X{L,3}))==5655   % WA
        gT(L)=2;
    elseif prod(double(X{L,3}))==5840   % PI
        gT(L)=3;
    elseif prod(double(X{L,3}))==6048   % TH
        gT(L)=4;
    end
end

if prod(double(Y{1,3}))==5621   % IM
        gC=1;
elseif prod(double(Y{1,3}))==5655   % WA
        gC=2;
elseif prod(double(Y{1,3}))==5840   % PI
        gC=3;
elseif prod(double(Y{1,3}))==6048   % TH
        gC=4;
end

%PCA calculations
[PC,SCORE,LATENT,tsquare]=princomp(T);
[pcT,varT,expT]=pcacov(cov(T));
% PCA explaied variances
PCA_num=15;
%fprintf('\n Percent Explained for TR Matrix = \n')
explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
%fprintf('\t\t%.6f\r',explained)
```

```
        % retained variance
        %fprintf('\nPercent    Explained    for    kept    PCs    =    %.6f
\n\n',sum(explained(1:PCA_num)))
        % Keep selected PC's
        SCORE=SCORE(:,1:PCA_num);
        PC=PC(:,1:PCA_num);
        newdataT=SCORE;             % Transformed T
        newdataC=C*PC;              % Transformed C

        % classify using PCs

        predicted_class_delCWT{D,1}=classify(newdataC,newdataT,gT)';
        actual_class_delCWT{D,1}=gC;

        % Keeps up with incorrects

        wrong_flaw_delCWT=find(abs(diff([predicted_class_delCWT{D,1}
actual_class_delCWT{D,1}]))~=0)';
        if isempty(wrong_flaw_delCWT)==1
            incorrect_delCWT{D,1}=0;
            incorrect_delCWT{D,2}=0;
        else
            incorrect_delCWT{D,1}=wrong_flaw_delCWT;
            incorrect_delCWT{D,2}=Y{1,2};
        end



        D=D+1;

end


% all results extracting one

[r,c]=size(incorrect_delCWT);
for i=1:r
    if incorrect_delCWT{i,1}==0
        Incorrect_total_delCWT(i)=0;
    else
        Incorrect_total_delCWT(i)=length(incorrect_delCWT{i,1});
    end
```

```
end

Incorrect_percentage_delCWT=sum(Incorrect_total_delCWT)/r*100;
fprintf('The incorrect percentage (extract one, PCA) NO CWT info = %2.2f
\n',Incorrect_percentage_delCWT)

% Extract One %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D=1;
for j=1:92                    % subgroup formations

        clear T X X1 C Y gT gC newdataC newdataT;
        % extract Y and X
        Y{1,1}=uTR{1,2,j}{4,1};Y{1,2}=uTR{1,1,j}{3,1};Y{1,3}=uTR{1,3,j}{1,1};
        X=uTR;X(:,:,j)=[];
        for P=1:91
            X1{P,1}=X{1,2,P}{4,1};X1{P,2}=X{1,1,P}{3,1};X1{P,3}=X{1,3,P}{1,1};
        end
        X=X1;clear X1;
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));               % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            T(k,:)=X{k,1};                          % extracts Training data
        end
        for k=1:rY
            C(k,:)=Y{k,1};                          % extracts Checking data
        end
        % Pre-Processing C and T
        del_col=[find(var(T)==0)];                  % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
        [rC,cC]=size(C);
        % mean-centering
        meanT=mean(T);
        T=T-meanT(ones(rT,1),:);
```

```
C=C-meanT(ones(rC,1),:);
stdT=std(T);
T=T./stdT(ones(rT,1),:);
C=C./stdT(ones(rC,1),:);

% Create classification vectors, must use numbers

for L=1:rT
    if prod(double(X{L,3}))==5621   % IM
        gT(L)=1;
   elseif prod(double(X{L,3}))==5655   % WA
        gT(L)=2;
    elseif prod(double(X{L,3}))==5840   % PI
        gT(L)=3;
    elseif prod(double(X{L,3}))==6048   % TH
        gT(L)=4;
    end
end

if prod(double(Y{1,3}))==5621   % IM
        gC=1;
elseif prod(double(Y{1,3}))==5655   % WA
        gC=2;
elseif prod(double(Y{1,3}))==5840   % PI
        gC=3;
elseif prod(double(Y{1,3}))==6048   % TH
        gC=4;
end

%PCA calculations
[PC,SCORE,LATENT,tsquare]=princomp(T);
[pcT,varT,expT]=pcacov(cov(T));
% PCA explaied variances
PCA_num=15;
%fprintf('\n Percent Explained for TR Matrix = \n')
explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
%fprintf('\t\t%.6f\r',explained)
% retained variance
%fprintf('\nPercent    Explained    for    kept    PCs    =    %.6f
\n\n',sum(explained(1:PCA_num)))
% Keep selected PC's
```

```
        SCORE=SCORE(:,1:PCA_num);
        PC=PC(:,1:PCA_num);
        newdataT=SCORE;             % Transformed T
        newdataC=C*PC;              % Transformed C

        % classify using PCs

        predicted_class_all_1{D,1}=classify(newdataC,newdataT,gT)';
        actual_class_all_1{D,1}=gC;

        % Keeps up with incorrects

        wrong_flaw_all_1=find(abs(diff([predicted_class_all_1{D,1}
actual_class_all_1{D,1}]))~=0)';
        if isempty(wrong_flaw_all_1)==1
            incorrect_all_1{D,1}=0;
            incorrect_all_1{D,2}=0;
        else
            incorrect_all_1{D,1}=wrong_flaw_all_1;
            incorrect_all_1{D,2}=Y{1,2};
        end


        D=D+1;

end


% all results extracting one

[r,c]=size(incorrect_all_1);
for i=1:r
    if incorrect_all_1{i,1}==0
        Incorrect_total_all_1(i)=0;
    else
        Incorrect_total_all_1(i)=length(incorrect_all_1{i,1});
    end
end

Incorrect_percentage_all_1=sum(Incorrect_total_all_1)/r*100;
```

```
fprintf('The average incorrect percentage (extract one, PCA) using all feature
families = %2.2f \n\n',Incorrect_percentage_all_1)

% Extract 4, look at ind. feature families %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

k=1;
for i=1:d
    if mod(i,23)~=0
        %eval(['sg' num2str(mod(i,4)) '(k,:)=uTR{1,2,i}{4,1};'])
        %eval(['sgf' num2str(mod(i,4)) '(k,:)=uTR{1,1,i}{3,1} ;'])
        %eval(['sgt' num2str(mod(i,4)) '(k,:)=uTR{1,3,i}{1,1} ;'])
        eval(['sg' num2str(mod(i,23)) '{k,1}=uTR{1,2,i}{4,1};'])
        eval(['sg' num2str(mod(i,23)) '{k,2}=uTR{1,1,i}{3,1};'])
        eval(['sg' num2str(mod(i,23)) '{k,3}=uTR{1,3,i}{1,1};'])
    else
        %eval(['sg' num2str(4) '(k,:)=uTR{1,2,i}{4,1};'])
        %eval(['sgf' num2str(4) '(k,:)=uTR{1,1,i}{3,1};'])
        %eval(['sgt' num2str(4) '(k,:)=uTR{1,3,i}{1,1};'])
        eval(['sg' num2str(23) '{k,1}=uTR{1,2,i}{4,1};'])
        eval(['sg' num2str(23) '{k,2}=uTR{1,1,i}{3,1};'])
        eval(['sg' num2str(23) '{k,3}=uTR{1,3,i}{1,1};'])
        k=k+1;
    end
end

feature_breaks=[2 21 23 48 51];
D=1;
for i=1:length(feature_breaks)  % feature families

    if i==1
        del_group=1:feature_breaks(1);
    else
        del_group=feature_breaks(i-1)+1:feature_breaks(i);
    end

    for j=1:23                    % subgroup formations
        clear T X C Y gC gT newdataC newdataT Del_GroupX Del_GroupY;
        z=1:23;
        z(j)=[];                  % deletes number j from Z
        eval(['X=cat(1,sg'    num2str(z(1))    ',sg'    num2str(z(2))    ',sg'
num2str(z(3)) ',sg' num2str(z(4)) ...
```

```
                ',sg'  num2str(z(5))  ',sg'  num2str(z(6))  ',sg'  num2str(z(7))
',sg' num2str(z(8)) ...
                ',sg'  num2str(z(9))  ',sg'  num2str(z(10))  ',sg'  num2str(z(11))
',sg' num2str(z(12)) ...
                ',sg'  num2str(z(13))  ',sg'  num2str(z(14))  ',sg'  num2str(z(15))
',sg' num2str(z(16)) ...
                ',sg'  num2str(z(17))  ',sg'  num2str(z(18))  ',sg'  num2str(z(19))
',sg' num2str(z(20)) ...
                ',sg'  num2str(z(21))  ',sg'  num2str(z(22))  ');'])        % X =
Training
        eval(['Y=sg' num2str(j) ';'])                                        % Y =
Checking
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));             % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            Del_GroupX(k,:)=X{k,1}(:,del_group);       % Retains the extracted
feature group
            X{k,1}(:,del_group)=[];                   % Extracts feature family
            T(k,:)=X{k,1};                            % extracts Training data
        end
        for k=1:rY
            Del_GroupY(k,:)=Y{k,1}(:,del_group);       % Retains the extracted
feature group
            Y{k,1}(:,del_group)=[];                   % Extracts feature family
            C(k,:)=Y{k,1};                            % extracts Checking data
        end

        % Pre-Processing C and T
        del_col=[find(var(T)==0)];                    % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
        [rC,cC]=size(C);
        % mean-centering
        meanT=mean(T);
        T=T-meanT(ones(rT,1),:);
```

```
C=C-meanT(ones(rC,1),:);
stdT=std(T);
T=T./stdT(ones(rT,1),:);
C=C./stdT(ones(rC,1),:);

% Pre-Processing Deleted Groups
del_col_R=[find(var(Del_GroupX)==0)];                    % 0 Varaince
cols
Del_GroupX(:,del_col_R)=[];
Del_GroupY(:,del_col_R)=[];
[rT,cT]=size(Del_GroupX);
[rC,cC]=size(Del_GroupY);
% mean-centering
meanT=mean(Del_GroupX);
Del_GroupX=Del_GroupX-meanT(ones(rT,1),:);
Del_GroupY=Del_GroupY-meanT(ones(rC,1),:);
stdT=std(Del_GroupX);
Del_GroupX=Del_GroupX./stdT(ones(rT,1),:);
Del_GroupY=Del_GroupY./stdT(ones(rC,1),:);

% Create classification vectors, must use numbers

for L=1:rT
    if prod(double(X{L,3}))==5621   % IM
        gT(L)=1;
   elseif prod(double(X{L,3}))==5655   % WA
        gT(L)=2;
    elseif prod(double(X{L,3}))==5840   % PI
        gT(L)=3;
    elseif prod(double(X{L,3}))==6048   % TH
        gT(L)=4;
    end
end
for L=1:rC
    if prod(double(Y{L,3}))==5621   % IM
        gC(L)=1;
    elseif prod(double(Y{L,3}))==5655   % WA
        gC(L)=2;
    elseif prod(double(Y{L,3}))==5840   % PI
        gC(L)=3;
    elseif prod(double(Y{L,3}))==6048   % TH
```

```
            gC(L)=4;
        end
    end

    % classify using raw features

    if length(del_group)<18
        predicted_class_R{D,1}=classify(Del_GroupY,Del_GroupX,gT)';
        actual_class_R{D,1}=gC;
        incorrect_R{D,1}=find(abs(diff([predicted_class_R{D,1}'
actual_class_R{D,1}'],1,2))~=0)';
        if                      isempty(find(abs(diff([predicted_class_R{D,1}'
actual_class_R{D,1}'],1,2))~=0)')==1
            incorrect_R{D,1}=0;
            family_incorrect_R{j,1}=0;
        else
            incorrect_R{D,1}=find(abs(diff([predicted_class_R{D,1}'
actual_class_R{D,1}'],1,2))~=0)';
            family_incorrect_R{j,1}=find(abs(diff([predicted_class_R{D,1}'
actual_class_R{D,1}'],1,2))~=0)';
        end
    end

    %PCA calculations
    [PC,SCORE,LATENT,tsquare]=princomp(T);
    [pcT,varT,expT]=pcacov(cov(T));
    % PCA explaied variances
    PCA_num=15;
    %fprintf('\n Percent Explained for TR Matrix = \n')
    explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
    %fprintf('\t\t%.6f\r',explained)
    % retained variance
    %fprintf('\nPercent     Explained     for     kept     PCs     =     %.6f
\n\n',sum(explained(1:PCA_num)))
    % Keep selected PC's
    SCORE=SCORE(:,1:PCA_num);
    PC=PC(:,1:PCA_num);
    newdataT=SCORE;             % Transformed T
    newdataC=C*PC;              % Transformed C

    % classify using PCs
```

```
        predicted_class{D,1}=classify(newdataC,newdataT,gT)';
        actual_class{D,1}=gC;

        % Keeps up with incorrects

        if                            isempty(find(abs(diff([predicted_class{D,1}'
actual_class{D,1}'],1,2))~=0)')==1
            incorrect{D,1}=0;
            family_incorrect{j,1}=0;
        else
            incorrect{D,1}=find(abs(diff([predicted_class{D,1}'
actual_class{D,1}'],1,2))~=0)';
            family_incorrect{j,1}=find(abs(diff([predicted_class{D,1}'
actual_class{D,1}'],1,2))~=0)';
        end


        D=D+1;

    end

    % family incorrects using raw data
    if length(del_group)<18
        [r,c]=size(family_incorrect_R);
        for n=1:r
            if family_incorrect_R{n,1}==0
              family_Incorrect_total_R(n)=0;
            else
              family_Incorrect_total_R(n)=length(incorrect_R{n,1});
            end
        end
        deleted_family=i;
        family_Incorrect_percentage_R=sum(family_Incorrect_total_R)/(r*4)*100;
        fprintf('The incorrect percentage (extract 4, NO PCA) for raw deleted
family %1.0f = %2.2f \n',i,family_Incorrect_percentage_R)
        clear family_incorrect_R family_Incorrect_total_R;
    end

    % family incorrects
    [r,c]=size(family_incorrect);
```

```
    for n=1:r
        if family_incorrect{n,1}==0
            family_Incorrect_total(n)=0;
        else
            family_Incorrect_total(n)=length(incorrect{n,1});
        end
    end
    deleted_family=i;
    family_Incorrect_percentage=sum(family_Incorrect_total)/(r*4)*100;
    fprintf('The incorrect percentage (extract 4, PCA) without deleted family
%1.0f = %2.2f \n',i,family_Incorrect_percentage)
    clear family_incorrect family_Incorrect_total;

end

% all results

[r,c]=size(incorrect);
for i=1:r
    if incorrect{i,1}==0
        Incorrect_total(i)=0;
    else
        Incorrect_total(i)=length(incorrect{i,1});
    end
end

Incorrect_percentage=sum(Incorrect_total)/(r*4)*100;
fprintf('The average incorrect percentage (extract 4, NO PCA) for deleted
families = %2.2f \n',Incorrect_percentage)

% Extract 4, All Feature Families included
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D=1;
for j=1:23                    % subgroup formations
        clear T X C Y gC gT newdataC newdataT;
        z=1:23;
        z(j)=[];              % deletes number j from Z
        eval(['X=cat(1,sg' num2str(z(1)) ',sg' num2str(z(2)) ',sg'
num2str(z(3)) ',sg' num2str(z(4)) ...
```

```
            ',sg' num2str(z(5)) ',sg' num2str(z(6)) ',sg' num2str(z(7))
',sg' num2str(z(8)) ...
            ',sg' num2str(z(9)) ',sg' num2str(z(10)) ',sg' num2str(z(11))
',sg' num2str(z(12)) ...
            ',sg' num2str(z(13)) ',sg' num2str(z(14)) ',sg' num2str(z(15))
',sg' num2str(z(16)) ...
            ',sg' num2str(z(17)) ',sg' num2str(z(18)) ',sg' num2str(z(19))
',sg' num2str(z(20)) ...
            ',sg' num2str(z(21)) ',sg' num2str(z(22)) ');'])      % X =
Training
        eval(['Y=sg' num2str(j) ';'])                              % Y =
Checking
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));              % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            T(k,:)=X{k,1};                         % extracts Training data
        end
        for k=1:rY
            C(k,:)=Y{k,1};                         % extracts Checking data
        end
        % Pre-Processing
        del_col=[find(var(T)==0)];                 % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
        [rC,cC]=size(C);
        % mean-centering
        meanT=mean(T);
        T=T-meanT(ones(rT,1),:);
        C=C-meanT(ones(rC,1),:);
        stdT=std(T);
        T=T./stdT(ones(rT,1),:);
        C=C./stdT(ones(rC,1),:);
        % Create classification vectors, must use numbers

        for L=1:rT
```

```
    if prod(double(X{L,3}))==5621   % IM
        gT(L)=1;
   elseif prod(double(X{L,3}))==5655   % WA
        gT(L)=2;
    elseif prod(double(X{L,3}))==5840   % PI
        gT(L)=3;
    elseif prod(double(X{L,3}))==6048   % TH
        gT(L)=4;
    end
end
for L=1:rC
    if prod(double(Y{L,3}))==5621   % IM
        gC(L)=1;
    elseif prod(double(Y{L,3}))==5655   % WA
        gC(L)=2;
    elseif prod(double(Y{L,3}))==5840   % PI
        gC(L)=3;
    elseif prod(double(Y{L,3}))==6048   % TH
        gC(L)=4;
    end
end

%PCA calculations
[PC,SCORE,LATENT,tsquare]=princomp(T);
[pcT,varT,expT]=pcacov(cov(T));
% PCA explaied variances
PCA_num=15;
%fprintf('\n Percent Explained for TR Matrix = \n')
explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
%fprintf('\t\t%.6f\r',explained)
% retained variance
%fprintf('\nPercent    Explained    for    kept    PCs    =    %.6f
\n\n',sum(explained(1:PCA_num)))
% Keep selected PC's
SCORE=SCORE(:,1:PCA_num);
PC=PC(:,1:PCA_num);
newdataT=SCORE;             % Transformed T
newdataC=C*PC;              % Transformed C

% classify using PCs
```

```
        predicted_class_all_4{D,1}=classify(newdataC,newdataT,gT)';
        actual_class_all_4{D,1}=gC;

        % Keeps up with incorrects

        wrong_flaw=find(abs(diff([predicted_class_all_4{D,1}'
actual_class_all_4{D,1}'],1,2))~=0)';
        if isempty(wrong_flaw)==1
            incorrect_all_4{D,1}=0;
        else
            incorrect_all_4{D,1}=wrong_flaw;
            [t,u]=size(wrong_flaw);
            for a=1:u
                incorrect_all_4{D,1+a}=Y{wrong_flaw(a),2};
            end
        end

        D=D+1;

end

% all results

[r,c]=size(incorrect_all_4);
for i=1:r
    if incorrect_all_4{i,1}==0
        Incorrect_total_all_4(i)=0;
    else
        Incorrect_total_all_4(i)=length(incorrect_all_4{i,1});
    end
end

Incorrect_percentage_all_4=sum(Incorrect_total_all_4)/(r*4)*100;
fprintf('The average incorrect percentage (extract 4, PCA) using all feature
families = %2.2f \n',Incorrect_percentage_all_4)
```

Xvalidate_B1.m

```
% Xvalidate_B1.m
```

```
clear    predicted_class    actual_class    predicted_class_R    actual_class_R
predicted_class_R1 actual_class_R1;

[r,c,d]=size(uTR);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                                 %
% Each uTR cell array page contains the information for one flaw           %
%   in a 1X3 nested cell array                                             %
%
                                         %
%                       C1                             C2
         C3                      %
%      | Origin           | Original Signal X      | flaw type       |
      %
%      | Group            | Magnitude and Phase    | % Through Wall |
      %
% R1   | filename         | flaw location          | flaw character |
      %
%      |                  | Feature Vector         |                     |
      %
%      |                      | CWT                    |                |
      %
%
                                    %
%
                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% feature_vector=[flaw_phase flaw_mag fext1Dabs fext1Ddiff geofext imagefext];
% position of feature families [2 21 23 48 51]



% Extract One, Use one feaute group at a time %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

feature_breaks=[2 21 23 48 51];
D=1;E=1;
for i=1:length(feature_breaks)  % feature families
```

```
    D=1;

    if i==1
        group=1:feature_breaks(1);
    else
        group=feature_breaks(i-1)+1:feature_breaks(i);
    end

    %if length(group) < 20

    for j=1:92                    % subgroup formations
        clear T X X1 C Y gC gT;
        % extract Y and X
        Y{1,1}=uTR{1,2,j}{4,1};Y{1,2}=uTR{1,1,j}{3,1};Y{1,3}=uTR{1,3,j}{1,1};
        X1=uTR;
        X1(:,:,j)=[];
        for P=1:91
            X{P,1}=X1{1,2,P}{4,1};X{P,2}=X1{1,1,P}{3,1};X{P,3}=X1{1,3,P}{1,1};
        end
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));              % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            T(k,:)=X{k,1}(:,group);    % Retains the extracted feature group
        end
        for k=1:rY
            C(k,:)=Y{k,1}(:,group);    % Retains the extracted feature group
        end

        if length(group) > 20          % Geometric moments

            T(:,[5 10 15 20 21 22 23 24 25])=[];C(:,[5 10 15 20 21 22 23 24
25])=[];
            % extracts the 4th order moments

        end
```

```
% Pre-Processing C and T
del_col=[find(var(T)==0)];                    % 0 Varaince cols
T(:,del_col)=[];
C(:,del_col)=[];
[rT,cT]=size(T);
[rC,cC]=size(C);
% mean-centering
meanT=mean(T);
T=T-meanT(ones(rT,1),:);
C=C-meanT(ones(rC,1),:);
stdT=std(T);
T=T./stdT(ones(rT,1),:);
C=C./stdT(ones(rC,1),:);


% Create classification vectors, must use numbers

for L=1:rT
    if prod(double(X{L,3}))==5621   % IM
        gT(L)=1;
   elseif prod(double(X{L,3}))==5655   % WA
        gT(L)=2;
    elseif prod(double(X{L,3}))==5840   % PI
        gT(L)=3;
    elseif prod(double(X{L,3}))==6048   % TH
        gT(L)=4;
    end
end
for L=1:rC
    if prod(double(Y{L,3}))==5621   % IM
        gC(L)=1;
    elseif prod(double(Y{L,3}))==5655   % WA
        gC(L)=2;
    elseif prod(double(Y{L,3}))==5840   % PI
        gC(L)=3;
    elseif prod(double(Y{L,3}))==6048   % TH
        gC(L)=4;
    end
end
```

```
        % classify using raw features

            predicted_class_R1{D,1}=classify(C,T,gT)';
            actual_class_R1{D,1}=gC;
            wrong_flaw_R1=find(abs(diff([predicted_class_R1{D,1}
actual_class_R1{D,1}]))~=0)';

            if isempty(wrong_flaw_R1)==1
                incorrect_R1{D,1}=0;
                incorrect_R1{D,2}=0;
                incorrect_R1_all{E,1}=0;
                incorrect_R1_all{E,2}=0;
            else
                incorrect_R1{D,1}=wrong_flaw_R1;
                incorrect_R1{D,2}=Y{1,2};
                incorrect_R1_all{E,1}=wrong_flaw_R1;
                incorrect_R1_all{E,2}=Y{1,2};
            end


        D=D+1;E=E+1;

    end

    % family incorrects using raw data

        [r,c]=size(incorrect_R1);
        for n=1:r
            if isempty(incorrect_R1{n,1}) == 1 | incorrect_R1{n,1} == 0
              Incorrect_total_R1(n)=0;
            else
              Incorrect_total_R1(n)=length(incorrect_R1{n,1});
            end
        end

        Incorrect_percentage_R1=sum(Incorrect_total_R1)/r*100;
        fprintf('The incorrect percentage (extract one, NO PCA) for family
%1.0f = %2.2f \n',i,Incorrect_percentage_R1)

        clear incorrect_R1;
        clear Incorrect_total_R1;
```

```
        clear Incorrect_percentage_R1;
        clear predicted_class_R1;
        clear actual_class_R1;


        %end

    clear D;

end

% all results for that feature family

[r,c]=size(incorrect_R1_all);
for i=1:r
    if incorrect_R1_all{i,1}==0
        Incorrect_total_R1_all(i)=0;
    else
        Incorrect_total_R1_all(i)=length(incorrect_R1_all{i,1});
    end
end

Incorrect_percentage_R1_all=sum(Incorrect_total_R1_all)/r*100;
fprintf('The average incorrect percentage (extract one, NO PCA) for all
families = %2.2f \n',Incorrect_percentage_R1_all)


% Extract 1, use only CWT information %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D=1;
CWT=[24:51];
for j=1:92                     % subgroup formations
        clear T X X1 C Y gC gT newdataC newdataT;
        % extract Y and X
        Y{1,1}=uTR{1,2,j}{4,1};Y{1,2}=uTR{1,1,j}{3,1};Y{1,3}=uTR{1,3,j}{1,1};
        X1=uTR;
        X1(:,:,j)=[];
        for P=1:91
            X{P,1}=X1{1,2,P}{4,1};X{P,2}=X1{1,1,P}{3,1};X{P,3}=X1{1,3,P}{1,1};
        end
        [rY,cY]=size(Y);
```

```
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));             % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            T(k,:)=X{k,1}(:,CWT);    % Retains the extracted feature group
        end
        for k=1:rY
            C(k,:)=Y{k,1}(:,CWT);    % Retains the extracted feature group
        end
        T(:,[5  10  15  20  21  22  23  24  25])=[];C(:,[5  10  15  20  21  22  23  24
25])=[];    % extracts the 4th order moments
        % Pre-Processing C and T
        del_col=[find(var(T)==0)];                 % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
        [rC,cC]=size(C);
        % mean-centering
        meanT=mean(T);
        T=T-meanT(ones(rT,1),:);
        C=C-meanT(ones(rC,1),:);
        stdT=std(T);
        T=T./stdT(ones(rT,1),:);
        C=C./stdT(ones(rC,1),:);
        %PCA calculations
        [PC,SCORE,LATENT,tsquare]=princomp(T);
        [pcT,varT,expT]=pcacov(cov(T));
        % PCA explaied variances
        PCA_num=15;
        %fprintf('\n Percent Explained for TR Matrix = \n')
        explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
        %fprintf('\t\t%.6f\r',explained)
        % retained variance
        %fprintf('\nPercent    Explained    for    kept    PCs    =    %.6f
\n\n',sum(explained(1:PCA_num)))
        % Keep selected PC's
        SCORE=SCORE(:,1:PCA_num);
        PC=PC(:,1:PCA_num);
```

```
newdataT=SCORE;                % Transformed T
newdataC=C*PC;                 % Transformed C

% Create classification vectors, must use numbers

for L=1:rT
    if prod(double(X{L,3}))==5621   % IM
        gT(L)=1;
   elseif prod(double(X{L,3}))==5655   % WA
        gT(L)=2;
    elseif prod(double(X{L,3}))==5840   % PI
        gT(L)=3;
    elseif prod(double(X{L,3}))==6048   % TH
        gT(L)=4;
    end
end
for L=1:rC
    if prod(double(Y{L,3}))==5621   % IM
        gC(L)=1;
    elseif prod(double(Y{L,3}))==5655   % WA
        gC(L)=2;
    elseif prod(double(Y{L,3}))==5840   % PI
        gC(L)=3;
    elseif prod(double(Y{L,3}))==6048   % TH
        gC(L)=4;
    end
end

% classify using raw features

    predicted_class_CWT{D,1}=classify(newdataC,newdataT,gT)';
    actual_class_CWT{D,1}=gC;
    wrong_flaw_CWT=find(abs(diff([predicted_class_CWT{D,1}
actual_class_CWT{D,1}]))~=0)';

    if isempty(wrong_flaw_CWT)==1 | wrong_flaw_CWT == 0
        incorrect_CWT{D,1}=0;
        incorrect_CWT{D,2}=0;
    else
        incorrect_CWT{D,1}=wrong_flaw_CWT;
        incorrect_CWT{D,2}=Y{1,2};
```

```
            end


        D=D+1;

end

    % family incorrects using raw data

        [r,c]=size(incorrect_CWT);
        for n=1:r
            if isempty(incorrect_CWT{n,1}) == 1 | incorrect_CWT{n,1} == 0
              Incorrect_total_CWT(n)=0;
            else
              Incorrect_total_CWT(n)=length(incorrect_CWT{n,1});
            end
        end
        Incorrect_percentage_CWT=sum(Incorrect_total_CWT)/r*100;
        fprintf('The incorrect percentage (extract one, PCA) only using CWT =
%2.2f \n',Incorrect_percentage_CWT)

        %clear incorrect_CWT Incorrect_total_CWT Incorrect_percentage_CWT;


% Extract One without CWT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D=1;
del_CWT=[24:51];
for j=1:92                    % subgroup formations

        clear T X C Y gT gC newdataC newdataT;
        % extract Y and X
        Y{1,1}=uTR{1,2,j}{4,1};Y{1,2}=uTR{1,1,j}{3,1};Y{1,3}=uTR{1,3,j}{1,1};
        X=uTR;X(:,:,j)=[];
        for P=1:91
            X1{P,1}=X{1,2,P}{4,1};X1{P,2}=X{1,1,P}{3,1};X1{P,3}=X{1,3,P}{1,1};
        end
        X=X1;clear X1;
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
```

```
        [Z,index]=sortrows(X(:,3));               % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            X{k,1}(:,del_CWT)=[];                  % Extracts feature family
            T(k,:)=X{k,1};                           % extracts Training data
        end
        for k=1:rY
            Y{k,1}(:,del_CWT)=[];                  % Extracts feature family
            C(k,:)=Y{k,1};                           % extracts Checking data
        end

        % Pre-Processing C and T
        del_col=[find(var(T)==0)];                   % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
        [rC,cC]=size(C);
        % mean-centering
        meanT=mean(T);
        T=T-meanT(ones(rT,1),:);
        C=C-meanT(ones(rC,1),:);
        stdT=std(T);
        T=T./stdT(ones(rT,1),:);
        C=C./stdT(ones(rC,1),:);

        % Create classification vectors, must use numbers

        for L=1:rT
            if prod(double(X{L,3}))==5621    % IM
                gT(L)=1;
           elseif prod(double(X{L,3}))==5655    % WA
                gT(L)=2;
            elseif prod(double(X{L,3}))==5840    % PI
                gT(L)=3;
            elseif prod(double(X{L,3}))==6048    % TH
                gT(L)=4;
            end
        end
```

```
        if prod(double(Y{1,3}))==5621   % IM
                gC=1;
        elseif prod(double(Y{1,3}))==5655   % WA
                gC=2;
        elseif prod(double(Y{1,3}))==5840   % PI
                gC=3;
        elseif prod(double(Y{1,3}))==6048   % TH
                gC=4;
        end

        %PCA calculations
        [PC,SCORE,LATENT,tsquare]=princomp(T);
        [pcT,varT,expT]=pcacov(cov(T));
        % PCA explaied variances
        PCA_num=15;
        %fprintf('\n Percent Explained for TR Matrix = \n')
        explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
        %fprintf('\t\t%.6f\r',explained)
        % retained variance
        %fprintf('\nPercent     Explained     for     kept     PCs     =     %.6f
\n\n',sum(explained(1:PCA_num)))
        % Keep selected PC's
        SCORE=SCORE(:,1:PCA_num);
        PC=PC(:,1:PCA_num);
        newdataT=SCORE;             % Transformed T
        newdataC=C*PC;              % Transformed C

        % classify using PCs

        predicted_class_delCWT{D,1}=classify(newdataC,newdataT,gT)';
        actual_class_delCWT{D,1}=gC;

        % Keeps up with incorrects

        wrong_flaw_delCWT=find(abs(diff([predicted_class_delCWT{D,1}
actual_class_delCWT{D,1}]))~=0)';
        if isempty(wrong_flaw_delCWT)==1
            incorrect_delCWT{D,1}=0;
            incorrect_delCWT{D,2}=0;
        else
```

```
            incorrect_delCWT{D,1}=wrong_flaw_delCWT;
            incorrect_delCWT{D,2}=Y{1,2};
        end


        D=D+1;

end



% all results extracting one

[r,c]=size(incorrect_delCWT);
for i=1:r
    if incorrect_delCWT{i,1}==0
        Incorrect_total_delCWT(i)=0;
    else
        Incorrect_total_delCWT(i)=length(incorrect_delCWT{i,1});
    end
end

Incorrect_percentage_delCWT=sum(Incorrect_total_delCWT)/r*100;
fprintf('The incorrect percentage (extract one, PCA) NO CWT info = %2.2f
\n',Incorrect_percentage_delCWT)

% Extract One %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D=1;
for j=1:92                    % subgroup formations

        clear T X X1 C Y gT gC newdataC newdataT;
        % extract Y and X
        Y{1,1}=uTR{1,2,j}{4,1};Y{1,2}=uTR{1,1,j}{3,1};Y{1,3}=uTR{1,3,j}{1,1};
        X=uTR;X(:,:,j)=[];
        for P=1:91
            X1{P,1}=X{1,2,P}{4,1};X1{P,2}=X{1,1,P}{3,1};X1{P,3}=X{1,3,P}{1,1};
        end
        X=X1;clear X1;
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
```

```
        [Z,index]=sortrows(X(:,3));              % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            T(k,:)=X{k,1};                          % extracts Training data
        end
        for k=1:rY
            C(k,:)=Y{k,1};                          % extracts Checking data
        end
        % Pre-Processing C and T
        del_col=[find(var(T)==0)];                  % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
        [rC,cC]=size(C);
        % mean-centering
        meanT=mean(T);
        T=T-meanT(ones(rT,1),:);
        C=C-meanT(ones(rC,1),:);
        stdT=std(T);
        T=T./stdT(ones(rT,1),:);
        C=C./stdT(ones(rC,1),:);

        % Create classification vectors, must use numbers

        for L=1:rT
            if prod(double(X{L,3}))==5621   % IM
                gT(L)=1;
           elseif prod(double(X{L,3}))==5655   % WA
                gT(L)=2;
            elseif prod(double(X{L,3}))==5840   % PI
                gT(L)=3;
            elseif prod(double(X{L,3}))==6048   % TH
                gT(L)=4;
            end
        end

        if prod(double(Y{1,3}))==5621   % IM
                gC=1;
```

```
        elseif prod(double(Y{1,3}))==5655   % WA
                gC=2;
        elseif prod(double(Y{1,3}))==5840   % PI
                gC=3;
        elseif prod(double(Y{1,3}))==6048   % TH
                gC=4;
        end

        %PCA calculations
        [PC,SCORE,LATENT,tsquare]=princomp(T);
        [pcT,varT,expT]=pcacov(cov(T));
        % PCA explaied variances
        PCA_num=15;
        %fprintf('\n Percent Explained for TR Matrix = \n')
        explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
        %fprintf('\t\t%.6f\r',explained)
        % retained variance
        %fprintf('\nPercent     Explained     for     kept     PCs     =     %.6f
\n\n',sum(explained(1:PCA_num)))
        % Keep selected PC's
        SCORE=SCORE(:,1:PCA_num);
        PC=PC(:,1:PCA_num);
        newdataT=SCORE;             % Transformed T
        newdataC=C*PC;              % Transformed C

        % classify using PCs

        predicted_class_all_1{D,1}=classify(newdataC,newdataT,gT)';
        actual_class_all_1{D,1}=gC;

        % Keeps up with incorrects

        wrong_flaw_all_1=find(abs(diff([predicted_class_all_1{D,1}
actual_class_all_1{D,1}]))~=0)';
        if isempty(wrong_flaw_all_1)==1
            incorrect_all_1{D,1}=0;
            incorrect_all_1{D,2}=0;
        else
            incorrect_all_1{D,1}=wrong_flaw_all_1;
            incorrect_all_1{D,2}=Y{1,2};
        end
```

```
        D=D+1;

end


% all results extracting one

[r,c]=size(incorrect_all_1);
for i=1:r
    if incorrect_all_1{i,1}==0
        Incorrect_total_all_1(i)=0;
    else
        Incorrect_total_all_1(i)=length(incorrect_all_1{i,1});
    end
end

Incorrect_percentage_all_1=sum(Incorrect_total_all_1)/r*100;
fprintf('The average incorrect percentage (extract one, PCA) using all feature
families = %2.2f \n\n',Incorrect_percentage_all_1)

% Extract 4, look at ind. feature families %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

k=1;
for i=1:d
    if mod(i,23)~=0
        %eval(['sg' num2str(mod(i,4)) '(k,:)=uTR{1,2,i}{4,1};'])
        %eval(['sgf' num2str(mod(i,4)) '(k,:)=uTR{1,1,i}{3,1} ;'])
        %eval(['sgt' num2str(mod(i,4)) '(k,:)=uTR{1,3,i}{1,1} ;'])
        eval(['sg' num2str(mod(i,23)) '{k,1}=uTR{1,2,i}{4,1};'])
        eval(['sg' num2str(mod(i,23)) '{k,2}=uTR{1,1,i}{3,1};'])
        eval(['sg' num2str(mod(i,23)) '{k,3}=uTR{1,3,i}{1,1};'])
    else
        %eval(['sg' num2str(4) '(k,:)=uTR{1,2,i}{4,1};'])
        %eval(['sgf' num2str(4) '(k,:)=uTR{1,1,i}{3,1};'])
        %eval(['sgt' num2str(4) '(k,:)=uTR{1,3,i}{1,1};'])
        eval(['sg' num2str(23) '{k,1}=uTR{1,2,i}{4,1};'])
        eval(['sg' num2str(23) '{k,2}=uTR{1,1,i}{3,1};'])
        eval(['sg' num2str(23) '{k,3}=uTR{1,3,i}{1,1};'])
        k=k+1;
```

```
    end
end

feature_breaks=[2 21 23 48 51];
D=1;
for i=1:length(feature_breaks)  % feature families

    if i==1
        del_group=1:feature_breaks(1);
    else
        del_group=feature_breaks(i-1)+1:feature_breaks(i);
    end

    for j=1:23                    % subgroup formations
        clear T X C Y gC gT newdataC newdataT Del_GroupX Del_GroupY;
        z=1:23;
        z(j)=[];                  % deletes number j from Z
        eval(['X=cat(1,sg'    num2str(z(1))    ',sg'    num2str(z(2))    ',sg'
num2str(z(3)) ',sg' num2str(z(4)) ...
                ',sg'  num2str(z(5))  ',sg'  num2str(z(6))  ',sg'  num2str(z(7))
',sg' num2str(z(8)) ...
                ',sg'  num2str(z(9))  ',sg'  num2str(z(10))  ',sg'  num2str(z(11))
',sg' num2str(z(12)) ...
                ',sg'  num2str(z(13))  ',sg'  num2str(z(14))  ',sg'  num2str(z(15))
',sg' num2str(z(16)) ...
                ',sg'  num2str(z(17))  ',sg'  num2str(z(18))  ',sg'  num2str(z(19))
',sg' num2str(z(20)) ...
                ',sg'  num2str(z(21))  ',sg'  num2str(z(22))  ');'])        % X =
Training
        eval(['Y=sg' num2str(j) ';'])                                        % Y =
Checking
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));              % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            Del_GroupX(k,:)=X{k,1}(:,del_group);       % Retains the extracted
feature group
```

```
            X{k,1}(:,del_group)=[];                  % Extracts feature family
            T(k,:)=X{k,1};                           % extracts Training data
        end
        for k=1:rY
            Del_GroupY(k,:)=Y{k,1}(:,del_group);       %  Retains  the  extracted
feature group
            Y{k,1}(:,del_group)=[];                  % Extracts feature family
            C(k,:)=Y{k,1};                           % extracts Checking data
        end

        % Pre-Processing C and T
        del_col=[find(var(T)==0)];                   % 0 Varaince cols
        T(:,del_col)=[];
        C(:,del_col)=[];
        [rT,cT]=size(T);
        [rC,cC]=size(C);
        % mean-centering
        meanT=mean(T);
        T=T-meanT(ones(rT,1),:);
        C=C-meanT(ones(rC,1),:);
        stdT=std(T);
        T=T./stdT(ones(rT,1),:);
        C=C./stdT(ones(rC,1),:);

        % Pre-Processing Deleted Groups
        del_col_R=[find(var(Del_GroupX)==0)];                      %  0  Varaince
cols
        Del_GroupX(:,del_col_R)=[];
        Del_GroupY(:,del_col_R)=[];
        [rT,cT]=size(Del_GroupX);
        [rC,cC]=size(Del_GroupY);
        % mean-centering
        meanT=mean(Del_GroupX);
        Del_GroupX=Del_GroupX-meanT(ones(rT,1),:);
        Del_GroupY=Del_GroupY-meanT(ones(rC,1),:);
        stdT=std(Del_GroupX);
        Del_GroupX=Del_GroupX./stdT(ones(rT,1),:);
        Del_GroupY=Del_GroupY./stdT(ones(rC,1),:);

        % Create classification vectors, must use numbers
```

```
        for L=1:rT
            if prod(double(X{L,3}))==5621    % IM
                gT(L)=1;
           elseif prod(double(X{L,3}))==5655    % WA
                gT(L)=2;
            elseif prod(double(X{L,3}))==5840    % PI
                gT(L)=3;
            elseif prod(double(X{L,3}))==6048    % TH
                gT(L)=4;
            end
        end
        for L=1:rC
            if prod(double(Y{L,3}))==5621    % IM
                gC(L)=1;
            elseif prod(double(Y{L,3}))==5655    % WA
                gC(L)=2;
            elseif prod(double(Y{L,3}))==5840    % PI
                gC(L)=3;
            elseif prod(double(Y{L,3}))==6048    % TH
                gC(L)=4;
            end
        end

        % classify using raw features

        if length(del_group)<18
            predicted_class_R{D,1}=classify(Del_GroupY,Del_GroupX,gT)';
            actual_class_R{D,1}=gC;
            incorrect_R{D,1}=find(abs(diff([predicted_class_R{D,1}'
actual_class_R{D,1}'],1,2))~=0)';
            if isempty(incorrect_R{D,1})==1
                incorrect_R{D,1}=0;
                family_incorrect_R{j,1}=0;
            else
                family_incorrect_R{j,1}=incorrect_R{D,1};
            end
        end

        %PCA calculations
        [PC,SCORE,LATENT,tsquare]=princomp(T);
        [pcT,varT,expT]=pcacov(cov(T));
```

```
        % PCA explaied variances
        PCA_num=15;
        %fprintf('\n Percent Explained for TR Matrix = \n')
        explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
        %fprintf('\t\t%.6f\r',explained)
        % retained variance
        %fprintf('\nPercent     Explained     for     kept     PCs     =     %.6f
\n\n',sum(explained(1:PCA_num)))
        % Keep selected PC's
        SCORE=SCORE(:,1:PCA_num);
        PC=PC(:,1:PCA_num);
        newdataT=SCORE;             % Transformed T
        newdataC=C*PC;              % Transformed C

        % classify using PCs

        predicted_class{D,1}=classify(newdataC,newdataT,gT)';
        actual_class{D,1}=gC;
        incorrect{D,1}=find(abs(diff([predicted_class{D,1}'
actual_class{D,1}'],1,2))~=0)';

        % Keeps up with incorrects

        if isempty(incorrect{D,1})==1
            incorrect{D,1}=0;
            family_incorrect{j,1}=0;
        else
            family_incorrect{j,1}=incorrect{D,1};
        end



        D=D+1;

    end

    % family incorrects using raw data
    if length(del_group)<18
        [r,c]=size(family_incorrect_R);
        for n=1:r
            if family_incorrect_R{n,1}==0
              family_Incorrect_total_R(n)=0;
```

```
            else
              family_Incorrect_total_R(n)=length(incorrect_R{n,1});
            end
        end
        deleted_family=i;
        family_Incorrect_percentage_R=sum(family_Incorrect_total_R)/(r*4)*100;
        fprintf('The incorrect percentage (extract 4, NO PCA) for raw deleted
family %1.0f = %2.2f \n',i,family_Incorrect_percentage_R)
        clear family_incorrect_R family_Incorrect_total_R
family_Incorrect_percentage_R;
    end

    % family incorrects
    [r,c]=size(family_incorrect);
     for n=1:r
         if family_incorrect{n,1}==0
              family_Incorrect_total(n)=0;
         else
              family_Incorrect_total(n)=length(incorrect{n,1});
         end
     end
     deleted_family=i;
     family_Incorrect_percentage=sum(family_Incorrect_total)/(r*4)*100;
     fprintf('The incorrect percentage (extract 4, PCA) without deleted family
%1.0f = %2.2f \n',i,family_Incorrect_percentage)
     clear family_incorrect family_Incorrect_total family_Incorrect_percentage;

end

% all results

[r,c]=size(incorrect);
for i=1:r
    if incorrect{i,1}==0
        Incorrect_total(i)=0;
    else
        Incorrect_total(i)=length(incorrect{i,1});
    end
end

Incorrect_percentage=sum(Incorrect_total)/(r*4)*100;
```

```
fprintf('The average incorrect percentage (extract 4, NO PCA) for deleted
families = %2.2f \n',Incorrect_percentage)

%     Extract     4,     All     Feature     Families     included
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

D=1;
for j=1:23                    % subgroup formations
        clear T X C Y gC gT newdataC newdataT;
        z=1:23;
        z(j)=[];                % deletes number j from Z
        eval(['X=cat(1,sg'   num2str(z(1))   ',sg'   num2str(z(2))   ',sg'
num2str(z(3)) ',sg' num2str(z(4)) ...
                ',sg' num2str(z(5)) ',sg' num2str(z(6)) ',sg' num2str(z(7))
',sg' num2str(z(8)) ...
                ',sg' num2str(z(9)) ',sg' num2str(z(10)) ',sg' num2str(z(11))
',sg' num2str(z(12)) ...
                ',sg' num2str(z(13)) ',sg' num2str(z(14)) ',sg' num2str(z(15))
',sg' num2str(z(16)) ...
                ',sg' num2str(z(17)) ',sg' num2str(z(18)) ',sg' num2str(z(19))
',sg' num2str(z(20)) ...
                ',sg' num2str(z(21)) ',sg' num2str(z(22)) ');'])       % X =
Training
        eval(['Y=sg' num2str(j) ';'])                                   % Y =
Checking
        [rY,cY]=size(Y);
        [rX,cX]=size(X);
        [Z,index]=sortrows(X(:,3));            % sorting_matrix has 1 columns
and multiple rows.
        X=X(index,:);
        [Z,index]=sortrows(Y(:,3));
        Y=Y(index,:);clear Z;
        for k=1:rX
            T(k,:)=X{k,1};                         % extracts Training data
        end
        for k=1:rY
            C(k,:)=Y{k,1};                         % extracts Checking data
        end
        % Pre-Processing
        del_col=[find(var(T)==0)];                 % 0 Varaince cols
        T(:,del_col)=[];
```

```
C(:,del_col)=[];
[rT,cT]=size(T);
[rC,cC]=size(C);
% mean-centering
meanT=mean(T);
T=T-meanT(ones(rT,1),:);
C=C-meanT(ones(rC,1),:);
stdT=std(T);
T=T./stdT(ones(rT,1),:);
C=C./stdT(ones(rC,1),:);
% Create classification vectors, must use numbers

for L=1:rT
    if prod(double(X{L,3}))==5621   % IM
        gT(L)=1;
   elseif prod(double(X{L,3}))==5655   % WA
        gT(L)=2;
    elseif prod(double(X{L,3}))==5840   % PI
        gT(L)=3;
    elseif prod(double(X{L,3}))==6048   % TH
        gT(L)=4;
    end
end
for L=1:rC
    if prod(double(Y{L,3}))==5621   % IM
        gC(L)=1;
    elseif prod(double(Y{L,3}))==5655   % WA
        gC(L)=2;
    elseif prod(double(Y{L,3}))==5840   % PI
        gC(L)=3;
    elseif prod(double(Y{L,3}))==6048   % TH
        gC(L)=4;
    end
end

%PCA calculations
[PC,SCORE,LATENT,tsquare]=princomp(T);
[pcT,varT,expT]=pcacov(cov(T));
% PCA explaied variances
PCA_num=15;
%fprintf('\n Percent Explained for TR Matrix = \n')
```

```
        explained=100*LATENT(1:PCA_num,:)/sum(LATENT(1:PCA_num,:));
        %fprintf('\t\t%.6f\r',explained)
        % retained variance
        %fprintf('\nPercent    Explained    for    kept    PCs    =    %.6f
\n\n',sum(explained(1:PCA_num)))
        % Keep selected PC's
        SCORE=SCORE(:,1:PCA_num);
        PC=PC(:,1:PCA_num);
        newdataT=SCORE;             % Transformed T
        newdataC=C*PC;              % Transformed C

        % classify using PCs

        predicted_class_all_4{D,1}=classify(newdataC,newdataT,gT)';
        actual_class_all_4{D,1}=gC;
        wrong_flaw=find(abs(diff([predicted_class_all_4{D,1}'
actual_class_all_4{D,1}'],1,2))~=0)';

        % Keeps up with incorrects

        if isempty(wrong_flaw)==1
            incorrect_all_4{D,1}=0;
        else
            incorrect_all_4{D,1}=wrong_flaw;
            [t,u]=size(wrong_flaw);
            for a=1:u
                incorrect_all_4{D,1+a}=Y{wrong_flaw(a),2};
            end
        end

        D=D+1;

end

% all results

[r,c]=size(incorrect_all_4);
for i=1:r
    if incorrect_all_4{i,1}==0
        Incorrect_total_all_4(i)=0;
    else
```

```
        Incorrect_total_all_4(i)=length(incorrect_all_4{i,1});
    end
end

Incorrect_percentage_all_4=sum(Incorrect_total_all_4)/(r*4)*100;
fprintf('The average incorrect percentage (extract 4, PCA) using all feature
families = %2.2f \n',Incorrect_percentage_all_4)
```

VITA

James Patrick McClanahan was born in Richlands, Virginia on January 30, 1964. He attended elementary school (grades K through 7th) and junior high (8th and 9th grades) in the public system for Buchanan County, Virginia. He then transferred to the public system of Washington County, Virginia where he graduated from Abingdon High School in 1982. He entered The University of Tennessee in the fall of 1982 majoring in Nuclear Engineering. He then transferred in the spring of 1986 to Emory and Henry College located in Emory, Virginia. He graduated from Emory and Henry College in May 1988 receiving Bachelor of Science degrees in Physics and Applied Mathematics. He then began working at Nuclear Fuel Services, Inc. in July of 1989. While working at Nuclear Fuel Services, he obtained a Master of Business Administration in March 1992 from Bristol University, Bristol, Tennessee.

Then, in August of 1994, he left Nuclear Fuel Services to re-enter The University of Tennessee's Nuclear Engineering Department. He obtained his Bachelor of Science degree in Nuclear Engineering from The University of Tennessee in the spring of 1996. He immediately began pursuit of his Master of Science degree in Nuclear Engineering, at The University of Tennessee. During his pursuit of his Master of Science, he worked as a Research Assistant helping Ali Erbay assemble a motor testing laboratory for Emerson. The Master of Science in Nuclear Engineering was finished in December 1998. During this time, he continued as a research assistant, installing a NDT laboratory for UTK's MRC. After this project was completed in May 2001, he became a teaching assistant employed by the ENGAGE (Freshman Engineering) Program. This employment lasted until he finished his Ph.D. He obtained his Ph.D. in Nuclear Engineering at The University of Tennessee in August 2003.