Spring 2011

Neural Network Prediction of Ultimate Compression After Impact Loads in Graphite-Epoxy Coupons from Ultrasonic C-Scan Images

Nikolas L. Geiselman
Embry-Riddle Aeronautical University - Daytona Beach

NEURAL NETWORK PREDICTION OF ULTIMATE COMPRESSION AFTER IMPACT LOADS IN GRAPHITE-EPOXY COUPONS FROM ULTRASONIC C-SCAN IMAGES

by

Nikolas L. Geiselman

A Thesis Submitted to the Graduate Studies Office in
Partial Fulfillment of the Requirements for the Degree of
Master of Science in Aerospace Engineering

Embry-Riddle Aeronautical University
Daytona Beach, Florida
Spring 2011

NEURAL NETWORK PREDICTION OF ULTIMATE COMPRESSION AFTER IMPACT LOADS IN GRAPHITE-EPOXY COUPONS FROM ULTRASONIC C-SCAN IMAGES

by

Nikolas L. Geiselman

This thesis was prepared under the direction of the candidate's thesis committee chairmen, Dr Eric Hill, Department of Aerospace Engineering, and has been approved by the members of his thesis committee. It was submitted to the School of Graduate Studies and Research and was accepted in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Engineering.

THESIS COMMITTEE:

Eric v. K. Hill

Dr. Eric v. K. Hill
Chairman

[illegible]

Dr. William C. Barott
Member

[illegible]

Christopher D. Hess
Member

[illegible]

Executive Vice President & Chief Academic Officer, ERAU

[illegible]

Department Chair, Aerospace Engineering

5/11/11
Date

ACKNOWLEDGMENTS

Throughout my studies I have encountered many difficulties and headaches, but the completion of this research has instilled in me a sense of perseverance. Numerous people have helped me during the years I have spent in college and have to be thanked. To my loving parents, Paul and Laurie Geiselman, to whom I owe everything; if not for their undying love and support, I would not have accomplished as much as I have. They enforced the importance of higher education on me from a very young age and sacrificed much to make college a possibility; for that I will be eternally thankful. A special thanks must go to my sister, Gina Geiselman, who has provided both financial support and encouragement throughout my graduate studies.

I would like to thank my thesis advisor Dr. Eric Hill for leading by example and offering his guidance from the beginning of my graduate degree, helping me develop into a true master of engineering. I would also like to thank Dr. Hill for his strict presentation methods which have molded me into a confident presenter. I am very thankful to Dr. William Barott for his persistence in instructing me about radio frequency and antenna theory as an undergraduate student and signal processing and filtering as a graduate student. I would like to thank both Andrew Pacific and Christopher Hess for laying the groundwork for this research through many tireless hours of research of their own.

Finally, I would also like to thank several of my very close friends who helped me throughout both my undergraduate and graduate degrees. Thanks to Daniel Lane and his family opened their home to me on many occasions and has been such a loyal and supportive friend throughout the years I have lived in Florida. Finally, a special thanks must also go to my good friends and roommates, Evan Colquhoun and Carl Flanagan, for all the good times we have had during my time spent in Florida.

ABSTRACT

Author:	Nikolas L. Geiselman
Title:	Neural Network Prediction of Ultimate Compression After Impact Loads in Graphite-Epoxy Coupons from Ultrasonic C-Scan Images
Institution:	Embry-Riddle Aeronautical University
Degree:	Master of Science in Aerospace Engineering
Year:	2011

The purpose of this project was to investigate how accurately an artificial neural network could predict the ultimate compressive loads of impact damaged 24-ply graphite-epoxy coupons from ultrasonic C-scan images. The 24-ply graphite-epoxy coupons were manufactured with bidirectional preimpregnated tape and cut into 21 coupons, 4 inches by 6 inches each. The coupons were impacted at known impact energies of 10, 12, 14, 16, 18, and 20 Joules in order to create barely visible impact damage (BVID). The coupons were then scanned with an ultrasonic C-scan system to create an image of the damaged area. Each coupon was then compressed to failure to determine its ultimate compressive load.

Numeric values for each pixel were determined from the C-scan image. Since the image was represented as a red-green-blue (RGB) map, each pixel had three numbers associated with it, one for each of the three colors. To make the image readable to the artificial neural network the columns of the resulting matrix were then summed, and these numbers were used as inputs for a backpropagation neural network (BPNN) to generate accurate predictions of the ultimate compressive loads. The BPNN was trained and optimized on 15 of the 21 sample data sets and tested on the remaining 6 sample data sets. The optimized BPNN was able to produce ultimate compression after impact (CAI) load predictions for the BVID composite coupons with a worst case error of -8.98%. This was within the ±10% goal for this research and comfortably within the B-basis allowables commonly applied to composite structures.

The ultrasonic C-scan images were then preprocessed using Fast Fourier Transforms (FFTs) in an effort to remove any image noise present. The results of the BPNN that was trained and tested on the green color data only were then compared to the results yielded by the BPNN trained and tested on the images that were processed through the FFT. It was found that the FFT processed images had a worst case BPNN prediction error of 8.65%, which was only slightly lower than the -8.98% error that was generated by the unprocessed green layer only C-scan image data. This improvement suggested that the added work involved in FFT preprocessing of the worst case error was not as productive as had been hoped, leading to a few suggestions for future noise removal research. This also reinforced the notion that BPNNs, being an iterative optimization scheme, can provide accurate predictions in the presence of at least small amounts of noise. Thus, image filtering methods coupled with the iterative optimization technique that comprises a BPNN have demonstrated the ability to generate accurate CAI load predictions in composite coupons that have experienced BVID.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

OVERVIEW

Over the past decade, composite materials have become increasingly prevalent in aerospace structures because they provide excellent stiffness properties and high strength-to-weight ratios. This major advantage of composites over metals has led to a huge increase in their use, especially as major structural elements. Both the military and commercial aviation sectors have adopted composites heavily as shown in Table 1.

In certain circumstances the weight savings of composites dictates their use, such as during the development of the B-2 stealth bomber. The addition of the radar absorbing paint on the fuselage caused the aircraft to be overweight; in order to reduce this weight penalty, composites were used over the majority of the aircraft [1]. Weight savings is also a crucial factor in the design of commercial aircraft because it controls the number of passengers and fuel economy; as such, weight savings lowers operating costs and increases profits. Airbus utilized a metal matrix composite in the design of the A380 called GLARE, which is comprised of fiberglass fibers embedded within an aluminum matrix, offering a weight savings of between 15% and 30% over standard aerospace aluminums [1].

Although they have high strength-to-weight ratios and excellent stiffness properties, polymer matrix composites are very susceptible to barely visible impact damage (BVID). BVID can be caused by many different things that aircraft encounter on a daily basis such as being struck with runway debris, tools being dropped by aircraft mechanics, or bird strikes. Although damage may be barely visible to the naked eye on the surface, significant damage may exist underneath the surface in the form of matrix cracking, delamination between plies, or occasional fiber breaks, weakening the composite substantially and causing the part to fail at a load lower than it was designed to withstand. Because of the inherent danger of BVID existing in structures such as aircraft wings or other load bearing parts, it becomes necessary to develop a method of quantitatively evaluating the severity of BVID without reliance on visual inspection.

Table 1: Composites by Weight of Various Aircraft [1]

Aircraft Name	Composites by Weight
Boeing 787	50%
Boeing V-22 Osprey	50%
Eurofighter	40%
Airbus A320	28%
Dassault Rafale	26%
Lockheed F-22 Raptor	24%
Airbus A380	22%
Boeing 777	20%
FA-18 Hornet	19%

PREVIOUS RESEARCH

Artificial neural networks have been used to predict ultimate compressive loads of impact damaged composite laminates from ultrasonic C-scan image data. Hess [2] originated the project in 2003 when he obtained a worst case error of 16.62% in graphite-epoxy coupons. He used three sets of 16-ply graphite-epoxy coupons and damaged them with known impact energies ranging from 0-20 ft-lb$_f$. Each coupon was then ultrasonically C-scanned, and a 16 color image was generated of the impact damage. In the image file, each pixel was assigned a numerical value between 0 and 15, with 0 corresponding to black and 15 to white. The coupons were then compressed to failure in a compression after impact (CAI) test fixture to determine their ultimate compressive load. The numerical value of each pixel was input into the artificial neural network in order to predict the ultimate CAI load of the coupons. In 2005, Nguyen [3] improved upon the Hess's results slightly with a worst case error of 14.61% using fiberglass-epoxy coupons. Subsequently, Gunasekera [4] in 2009 obtained a worst case error of -11.53% using the acoustic emission data taken during compression of the same graphite-epoxy coupons that were used in the current project.

CURRENT RESEARCH

The current approach was to explore the ability for an artificial neural network to predict the ultimate compressive load of graphite-epoxy coupons that have BVID using improved ultrasonic C-scan images as inputs. Twenty-one test specimen coupons were impacted at known energies, scanned using an ultrasonic C-scan machine, and then compressed to failure with their ultimate CAI loads being recorded for use in the artificial neural network. After the C-scan images were recorded for each damaged coupon, the images were cropped into 100 by 100 pixel squares centered on the damaged area and input into a MATLAB code for image preprocessing. The MATLAB code quantified the damaged area of the image and created a 1 row by 300 column matrix of the image which contained three colors: red, green, and blue. By inspection, the red and blue color layers contained only noise; thus, only the green color layer was used as the input to the neural network. Fifteen coupons were employed for training the neural network with the remaining six being used to test the network. The target goal of the artificial neural network was twofold: (1) the ability to predict the ultimate compressive load of each composite coupon within ±10% of the actual failure load, and hopefully, (2) also predicting within the statistical B-basis allowables of the graphite-epoxy composite coupons as well.

CHAPTER 2

BACKGROUND

ULTRASONIC C-SCAN

Ultrasound is a volumetric method of nondestructive testing that uses high frequency sound waves to analyze a part, point by point, without destroying it. Ultrasonic waves are high frequency sound waves that are outside the range of human hearing, normally well above 20 kHz [5]. A typical ultrasonic system consists of a pulser/receiver transducer that contains a piezoelectric ceramic crystal which converts an electric signal into a sound wave; conversely, the piezoelectric element will produce an electrical signal in response to an incident sound wave. To scan a part, the ultrasonic transducer emits a sound wave, then switches to listen mode to receive the echoes from the part. Whenever the wave encounters a change in density, it is both reflected and refracted; these changes in density can be caused by a defect under the surface. As the ultrasonic sound wave enters the part, it reflects and refracts. The reflected wave returns to the transducer first as the "initial pulse" (Figure 1). Then the wave propagates through the material until it reaches the crack or discontinuity and is again reflected and refracted, whereupon the reflected wave returns to the transducer as the "crack echo". The wave continues to propagate through the part and eventually reaches the back surface and is again reflected and refracted. Here the reflected wave returns to the transducer as the "back surface echo". Thus, the three echoes in Figure 1 are formed.

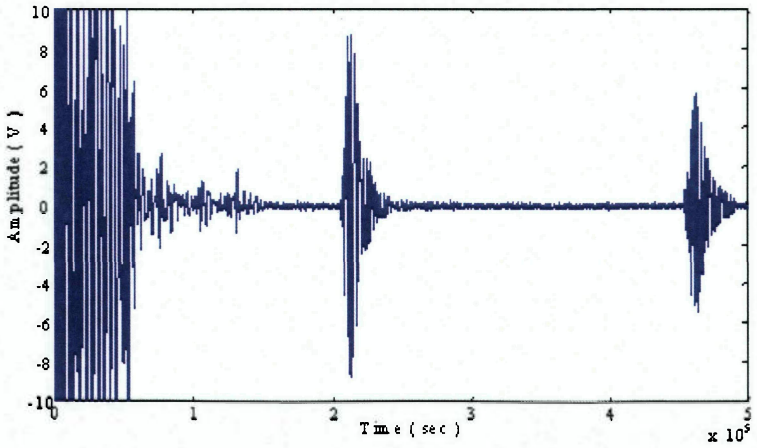


Figure 1: Ultrasonic Waveform [6]

The specific application of ultrasonic nondestructive testing used in this research was the ultrasonic C-scan, which is shown in Figure 2. A C-scan is a planar image generated by compiling all of the ultrasonic echoes for each point along the surface of the part. This allows the user to see the size and location of flaws both underneath and atop the surface. This image is generated by moving the ultrasonic transducer in a sweeping pattern over the part while recording the amplitude and time-of-flight of the received pulses as it proceeds. These signals are displayed on a screen at each position of the transducer using either a color or grayscale. The C-scan system used in this research was a water coupled immersion scanning machine, which means that both the transducer and the part being inspected were underwater, allowing the water to transmit the sound waves across the distance from the transducer to the part and back again.

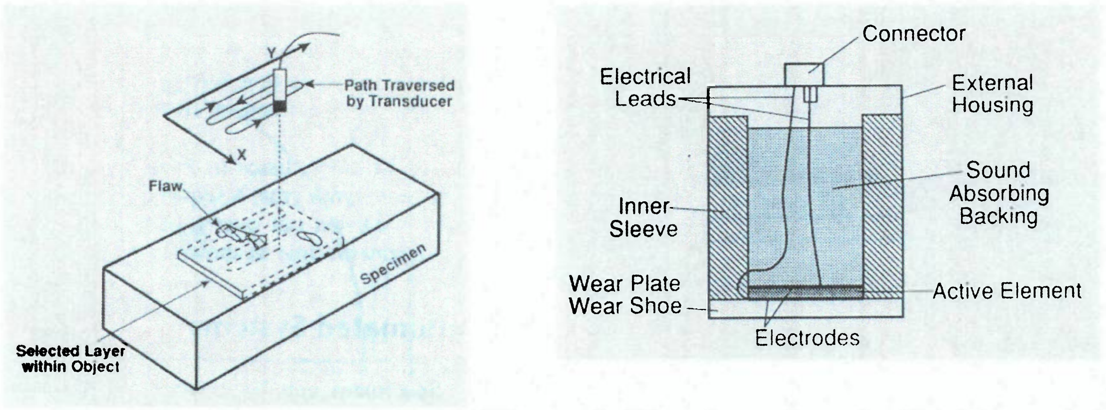


Figure 2: C-Scan Pattern (Left), Ultrasonic Transducer Cutaway (Right)

NEURAL NETWORKS

Artificial neural networks were derived from the processing of the human brain, utilizing many different neurons to make complex calculations very quickly. A neural network consists of a group of interconnected neurons, or processing elements, which change weighted connections in response to an output error, and through multiple iterations converge to the desired output or answer. Many different kinds of artificial neural networks exist for special tasks, but the network that was used for this research was a backpropagation neural network (BPNN).

A typical BPNN has an input layer, one or more hidden layers, and an output layer. Networks with more than one hidden layer are generally used to solve more complex problems such as those that require both classification and prediction. Each layer is fully connected to the neighboring layers with information passing from the input layer through the hidden layers and on to the output layer. During the learning phase, the output error is propagated back through the network, and the connection weights are updated accordingly -- this is where the BPNN gets its name.

The backpropagation neural network is a feed-forward, multilayered, supervised learning system. Feed-forward refers to the direction of movement of information in the network. In the BPNN of Figure 3, information enters the network through the input layer, is then passed forward (from left to right) through one or more hidden layers, and finally exits the network through the output layer. Information is not passed backwards through the network, and it is therefore designated as a feed-forward network. Multilayered alludes to the number of layers of a neural network. The number of hidden layers can be varied as needed. A supervised network requires the operator to give it a desired result or output, which the network trains toward during the training phase.

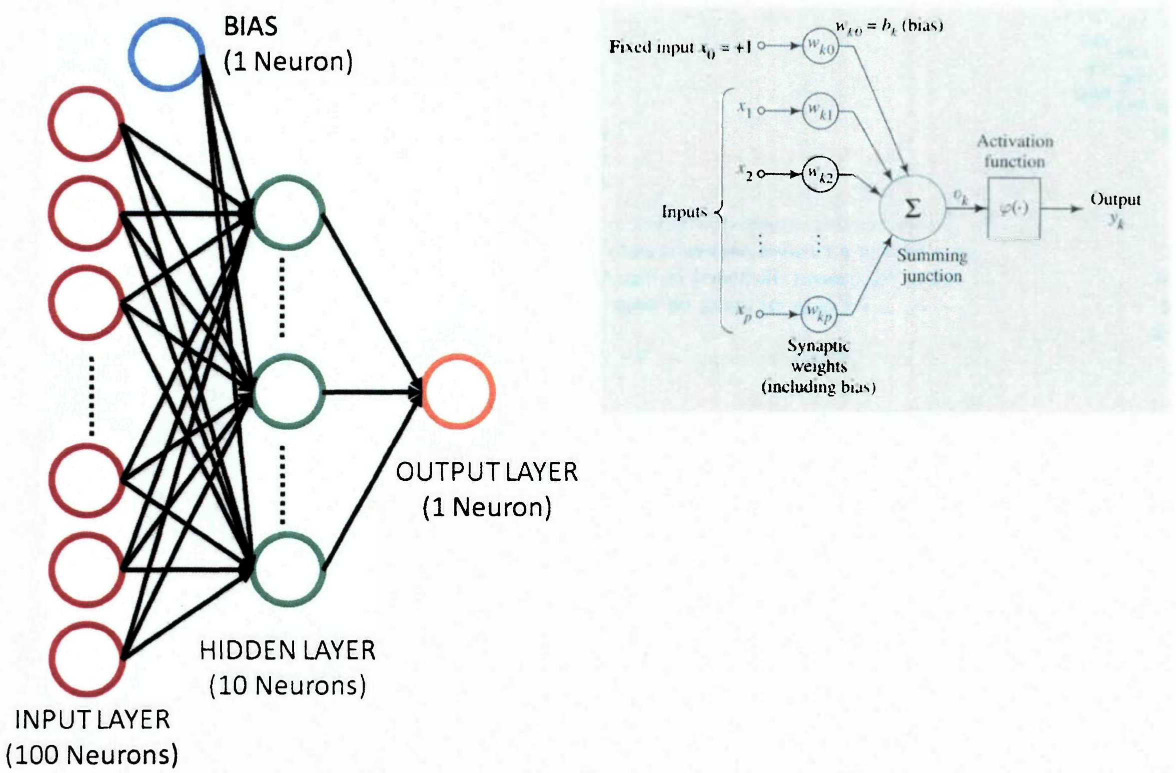


Figure 3: Backpropagation Neural Network Architecture (Left), Neuron Architecture (Right)

The BPNN initializes by assigning random weights between 0 and 1 to all the neuron connections. The inputs are then multiplied by these weights and passed through a transfer or squashing function, which normalizes the data before it is passed on to the next layer. After each training iteration, the resulting answer from the single output layer neuron is compared to the desired answer, and the error is determined. This error is used to calculate weight adjustments which are then propagated back through all the network connections, after which the next training iteration begins. This process continues iteratively until the answer in the output layer approaches the desired answer, at which point training is considered to be complete. Once the network has been trained, the network weights are held fixed and are no longer changed. The testing phase begins when the network is presented another input file that it has not seen before; the data from this file are then run through the network. The resulting answer given by the output layer is compared to the desired answer, and the prediction error is calculated. This final prediction error is what is being minimized for this research. The desired result is to optimize the BPNN such that it will predict the ultimate CAI loads of barely visible impact damaged graphite-epoxy coupons to within a ±10% worst case error from the ultrasonic C-scan images of the coupons.

The learning algorithm used for the BPNN was the Normalized-Cumulative-Delta rule. This rule adds up all the squared errors over the training set or epoch, then takes the square root of the sum and divides this value by the epoch size to normalize it. As mentioned previously, this normalized RMS error is then used to update the weighted connections between the output and the hidden layer(s), and the hidden layer(s) and the input. Thus, all network weights are updated at the end of each training epoch. When this RMS output error reaches a user defined value, typically 5% or less, or the network completes a specified number of training cycles, the training is considered complete, the network weights are fixed, and testing or prediction phase can begin on data inputs that have not yet been considered.

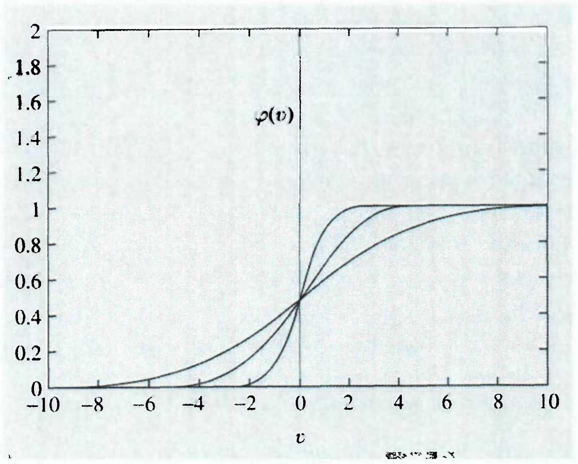


Figure 4: Sigmoid Functions at Different “a” Values

The transfer function that was used for the BPNN in this research was the sigmoid function as seen in Figure 4. The sigmoid transfer function is described by the following equation:

$$\varphi(v) = \frac{1}{1+e^{-av}}$$

where $\varphi(v)$ = value of sigmoid function with input v
a = slope parameter.

Varying the slope parameter “a” yields sigmoid functions with different slopes. At the origin, the slope of the sigmoid transfer function is “a/4”. At the extremities of the function, as v approaches infinity, the slope of the sigmoid function becomes infinitely small and training is very slow. For higher positive values of input v the transfer function scales the output value to 1, whereas for higher negative values of input v the transfer function scales the output value to 0. This scaling of values effectively squashes the input data within each neuron such that its output ranges from 0 to 1. This makes the larger numbers less significant and smaller numbers more significant such that the data fed into subsequent neurons can be more easily processed by the neural network. The bias neuron has a constant output of 1 which when multiplied by the connection weight acts as a translation term to shift the sigmoid activation function $\varphi(v)$ such that it operates near its highest slope and therefore trains as quickly as possible.

FAST FOURIER TRANSFORM

A Fast Fourier Transform (FFT) transforms a real image into the frequency domain, which enables the image to be filtered. This is accomplished by taking the FFT of each pixel in the image and transforming its value into a complex number in the frequency domain. The real image consists of the summation of several different sinusoids with different frequencies and amplitudes. When an image is transformed using a FFT, it represents each sinusoid as a pair of conjugate symmetric points, each with a frequency and a phase. Figure 5 shows the correlation between a sinusoidal image and its frequency transform. Measured with relation to the center of the FFT image, the distance to each point denotes the frequency of the sinusoidal wave. The angle that the position vector from the center of the FFT image to the point, measured from the horizontal, denotes the phase angle of the sinusoidal wave. In order to transform a real two dimensional image into the frequency domain, the equation below must be calculated for each pixel in the real image:

$$F(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-j2\pi\left(x\frac{m}{M}+y\frac{n}{N}\right)}$$

where F(x,y) = value of each pixel at position (x,y) in the frequency domain
f(m,n) = value of each pixel at location (m,n) in the space domain
M = width of the image
N = height of the image.

After the image has been filtered and the noise eliminated, the FFT must be inverted to return the transformed image to a real image in the spatial domain. By employing the equation below at each point and its corresponding conjugate in the frequency image, it is reverted back to a real image in the spatial domain:

$$f(m,n) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F(x,y) e^{j2\pi\left(x\frac{m}{M}+y\frac{n}{N}\right)}$$

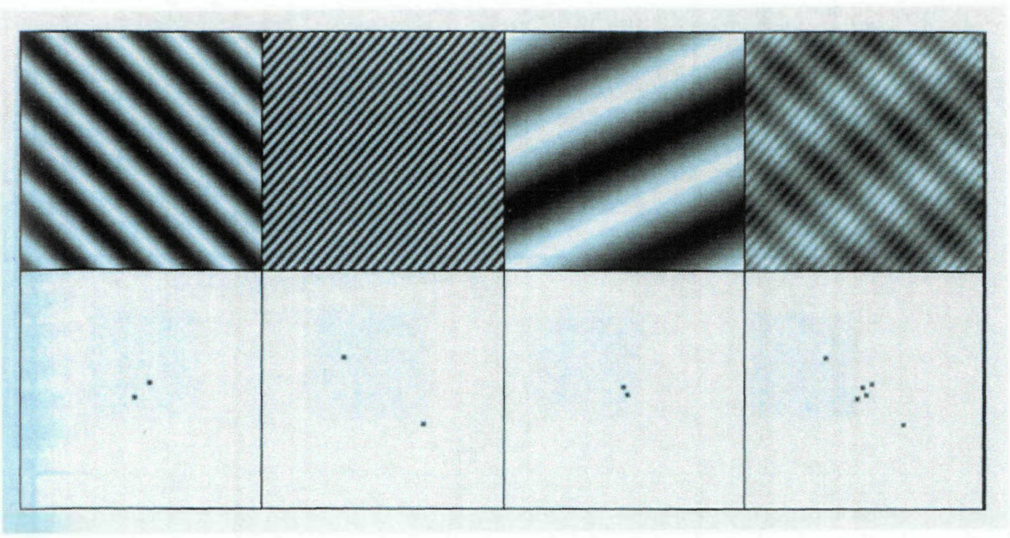


Figure 5: Sinusoidal Pattern with corresponding Frequency Transform [7]

It is important to notice in Figure 5 that as the distance of the conjugate symmetric points grows from the center point of the frequency image (bottom row), the frequency of the corresponding real image increases (top row). The importance of this is discussed later in the image filtering section, but an important generalization can be reached from this: the higher frequencies of the real image are focused on the edges of its corresponding frequency transform. Also, as a general rule, the lower frequencies determine the overall shape of the image, while the higher frequencies sharpen the edges and control the fine details. The images on the far right column of Figure 5 are the result of the summation of the three signals in the preceding three columns. The image (top row) of column four is analogous to the images analyzed in this research; the C-scan image can be considered the summation of many different sinusoids of differing phases. The theory of noise cancellation of the C-scan images is predicated on the correct selection of the high frequency noise and its cancellation.

CHAPTER 3
EXPERIMENTAL PROCEDURE

COUPON MANUFACTURE

The test coupons were manufactured using six 24-ply panels made from Cycom 985 GF3070PW graphite 3070 plain weave preimpregnated tape (with a fiber volume fraction of 0.63) by Gunasekera [4] and Pacific, et al. [14]. All of the coupons had misaligned fibers, but it was decided to move forward with testing because this defect existed in every coupon which at least provided consistency for the testing. The number of plies per coupon was determined by the ASTM standard D7137/D 7137M-07 which covers CAI testing and mandates that test coupons be 0.20 inches thick [8-9]. Considering the thickness of each individual ply and the thickness of the plain weave tape, it was determined that a 24-ply layup would yield a coupon thickness of the required 0.20 inches. The composite panels were laid up in a wooden jig, and the resulting laminate panels were cured at 355^{o}F for two hours while being clamped with four C-clamps between two aluminum caul plates to prevent warping. After curing, the oven was shut down, and the laminate panels were left to cool down to room temperature in the oven. Once cooled, each panel was cut into four (4 in x 6 in) coupons using a diamond tip wet-saw, yielding a total of 21 useable coupons. Each coupon was labeled with a number and a letter designating from which plate the coupon originated, with different coupons coming from the same plate given the same number but different letters [4].

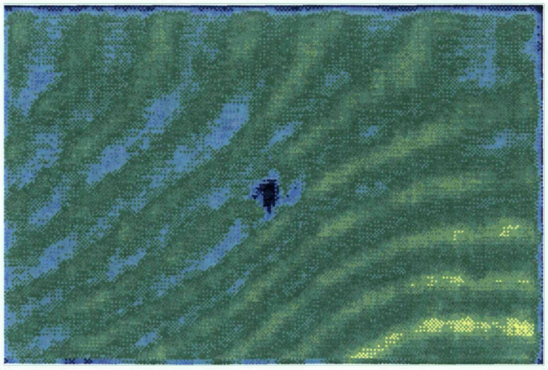


Figure 6: Misaligned Fibers in Test Coupon

EXPERIMENTAL PROCEDURE

The 21 coupons were impacted at known energies of 10, 12, 14, 16, 18, 20 Joules using an Instron Dynatup 9200 impacter with a blunt 0.5 inch hemispherical tup (Figure 7) to create barely visible impact damage (BVID). The impacter simulated a low velocity impact similar to a tool dropping on the coupon. Pneumatic brakes on the impacter were utilized to avoid multiple impacts from the tup, since the impacter bounced after the initial impact. The coupon was marked with a silver metallic marker to more accurately determine the center of the coupon as the impact site for the tup. Pneumatic clamps held the sample in place to avoid movement during the test and ensure a precise impact location in the center of the coupon.

After impacting the samples, the BVID coupons were C-scanned using a Physical Acoustics Corporation (PAC) ULTRAPAC II water immersion C-Scanner (Figure 7). The ULTRAPAC II system employed a 0.25 inch diameter piezoelectric crystal ultrasonic transducer with a characteristic frequency of 5 MHz for scanning. Figure 8 shows the hardware setup menu of the C-Scan system, detailing all the settings used for this research. The ULTRAPAC II system outputs several images including amplitude and time-of-flight; however, only the amplitude image was used for this research. The horizontal lines on the A-scan output represent the gates of the scan. These gates control what information is recorded, which allows the computer to ignore the initial pulse as the wave enters the water couplant. The gate time selection also allows the user to select data specifically from the damaged layers of the composite, essentially looking inside the composite only at the depth of the damaged area.

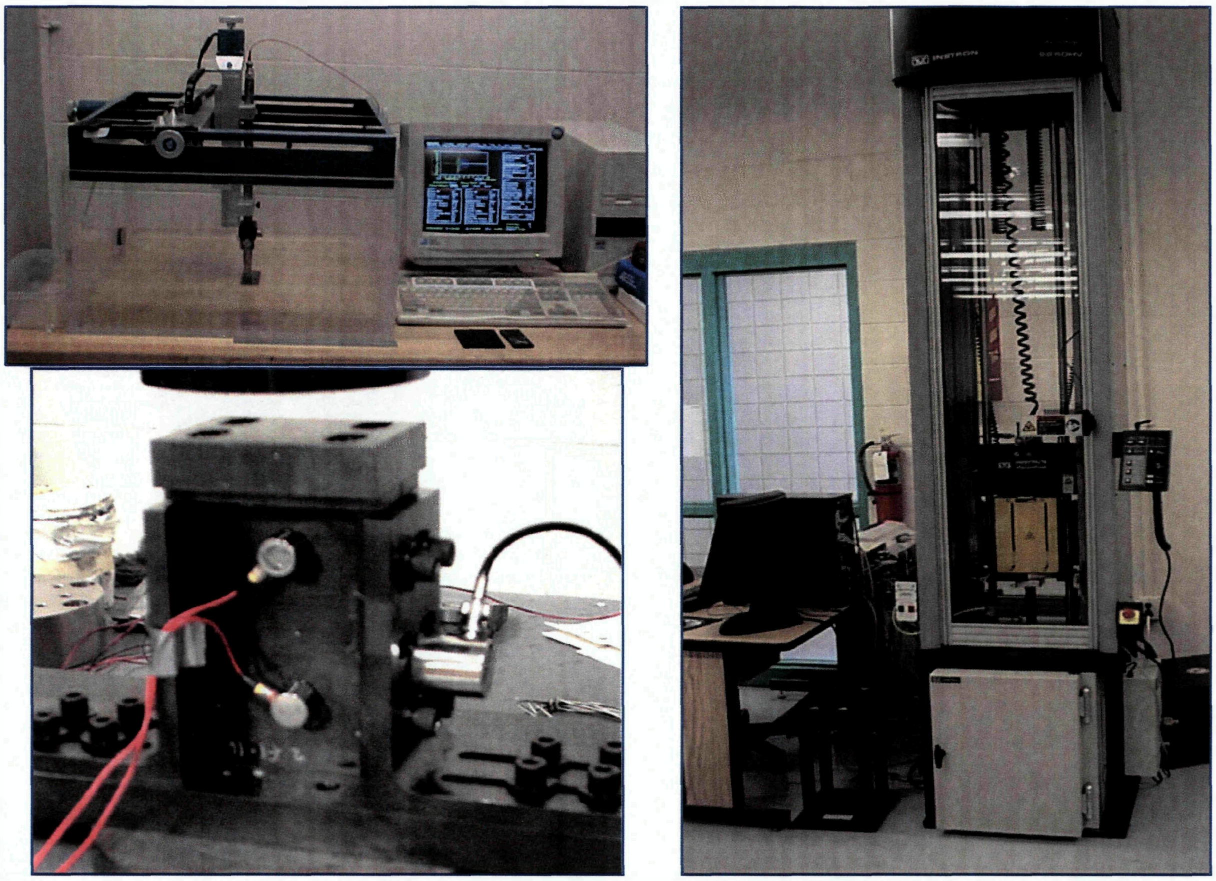
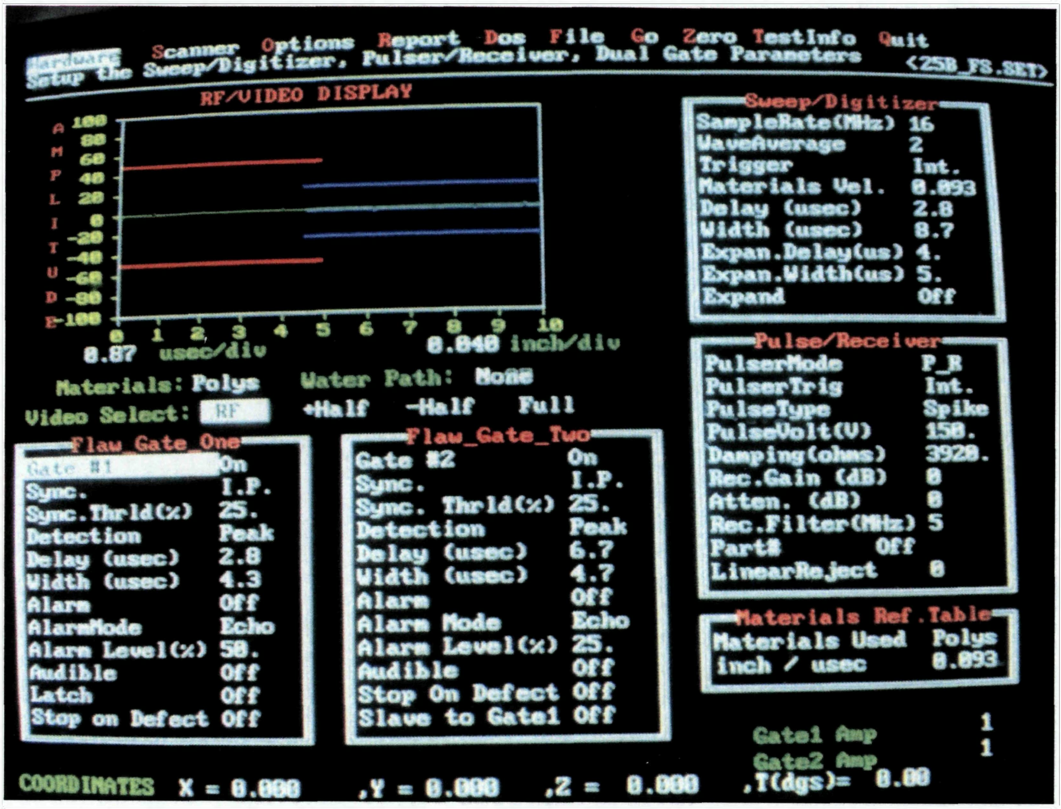


Figure 7: UltraPAC II C-Scan imaging system (Top left), Boeing Compression After Impact (CAI) test fixture (Bottom left), Instron Dynatup 9200 Impacter (Right)

The samples were then compressed to failure to determine their ultimate CAI loads using a Tinius-Olsen model 290 Lo Cap testing machine (Figure 7). A Boeing compression after impact test fixture was used to keep the coupons from buckling in accordance with ASTM standards D 7137/D 7137M-07 [8]. The coupons were secured in the Boeing CAI fixture and compressed to failure, with failure always occurring at the BVID impact location, indicating that the levels of BVID selected had adequately compromised the strength of the composite.



```
Hardware  Scanner  Options  Report  Dos  File  Go  Zero  TestInfo  Quit
Setup the Sweep/Digitizer, Pulser/Receiver, Dual Gate Parameters      <25B_FS.SET>

           RF/VIDEO DISPLAY                          Sweep/Digitizer
                                                     SampleRate(MHz)  16
                                                     WaveAverage      2
                                                     Trigger          Int.
                                                     Materials Vel.   0.093
                                                     Delay (usec)     2.8
                                                     Width (usec)     8.7
                                                     Expan.Delay(us)  4.
                                                     Expan.Width(us)  5.
                                                     Expand           Off
 0.87 usec/div              0.040 inch/div
                                                     Pulse/Receiver
Materials: Polys    Water Path: None                 PulserMode       P_R
Video Select: RF    +Half   -Half   Full             PulserTrig       Int.
                                                     PulseType        Spike
   Flaw Gate One             Flaw_Gate_Two           PulseVolt(V)     150.
Gate #1          On     Gate #2           On         Damping(ohms)    3920.
Sync.            I.P.   Sync.             I.P.       Rec.Gain (dB)    0
Sync.Thrld(%)    25.    Sync. Thrld(%)    25.        Atten. (dB)      0
Detection        Peak   Detection         Peak       Rec.Filter(MHz)  5
Delay (usec)     2.8    Delay (usec)      6.7        Part#         Off
Width (usec)     4.3    Width (usec)      4.7        LinearReject     0
Alarm            Off    Alarm             Off
AlarmMode        Echo   Alarm Mode        Echo       Materials Ref.Table
Alarm Level(%)   50.    Alarm Level(%)    25.        Materials Used   Polys
Audible          Off    Audible           Off        inch / usec      0.093
Latch            Off    Stop On Defect    Off
Stop on Defect   Off    Slave to Gate1    Off
                                                     Gate1 Amp            1
                                                     Gate2 Amp            1
COORDINATES  X = 0.000   ,Y = 0.000   ,Z =  0.000   ,T(dgs)=  0.00
```

Figure 8: C-Scan Hardware Setup Screen

Once the coupons were compressed to failure and testing had concluded, the C-scan images were cropped to a 100 by 100 pixel square around the damaged area. This damaged square image was a 256 color RGB image consisting of 3 layers -- a red layer, a green layer, and a blue layer -- with the pixel of each layer assigned a number from 0 to 255. Finally, the image was put into a MATLAB processing program developed by Hess [2] and converted into a 1 row by 300 column matrix which was used as the artificial neural network input for ultimate CAI load prediction.

IMAGE MANIPULATION

The C-scan image files output from the UltraPAC II are in ".PCX" format and needed to be converted to ".BMP" format so they could be analyzed as 256 color RGB images. Once all the images were converted to 256 color BMP files, they were loaded into the MATLAB program. Figure 9 shows the MATLAB output for an impacted sample displaying all 3 color layers. The graph below the image displays the color value of each pixel (AMP) and the x position of the pixel. Positions 0 to 100 correspond to the red color layer, positions 100 to 200 correspond to the green color layer, and positions 200 to 300 correspond to the blue color layer.

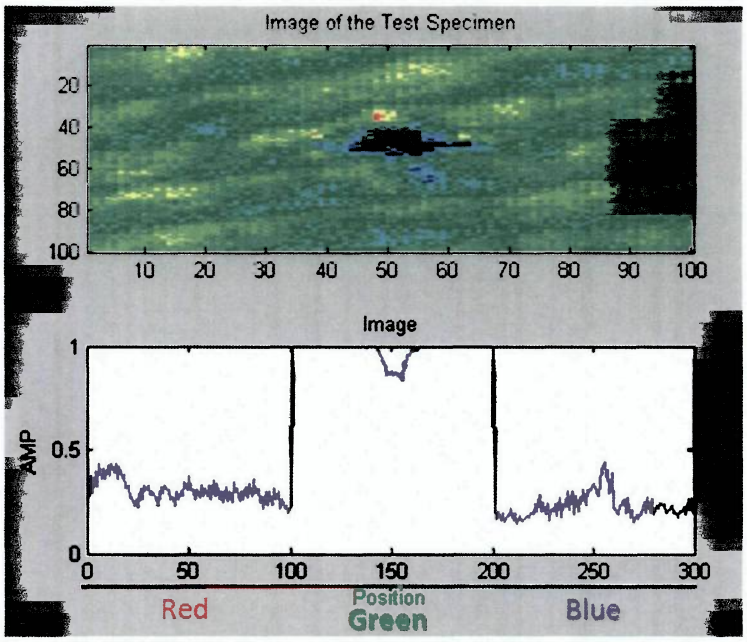


Figure 9: MATLAB Program Output Including all 3 color layers

By observation, the red and the blue layers include mostly noise, whereas the green layer clearly displays a dip corresponding to the impact location. Not all images are as free of noise as shown in Figure 9. Several of the images that were C-scanned contained significant noise, such as sample 3B in Figure 10. When these noisy images were input to the BPNN, the network had difficulty determining the location of the damaged section, thereby increasing the error significantly, as seen in Table 2. In an effort to minimize noise, the red and blue layers were removed from each image (Figure 11), and only the green layer was used as input to the backpropagation neural network.

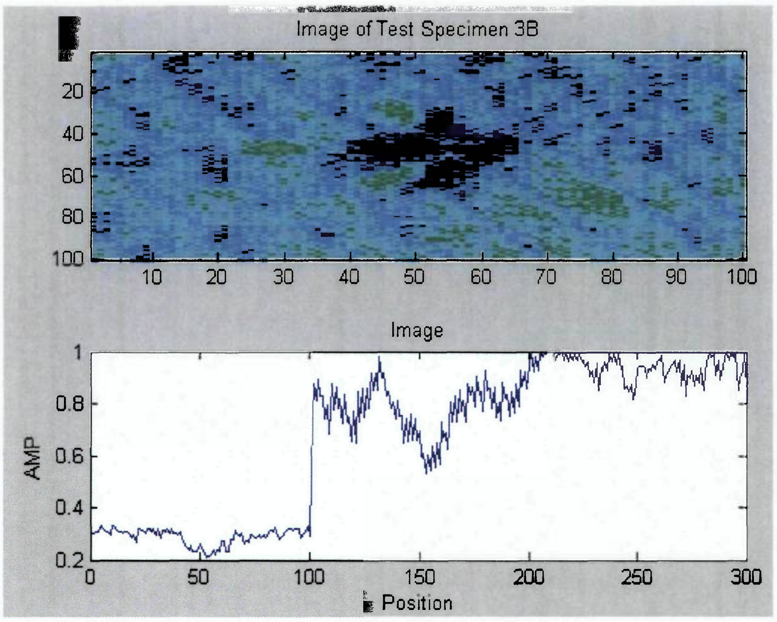


Figure 10: Noisy C-Scan Image

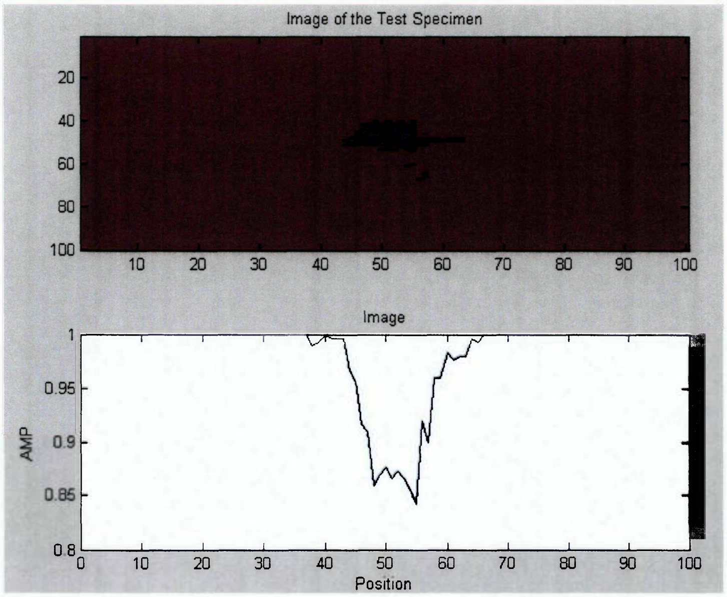


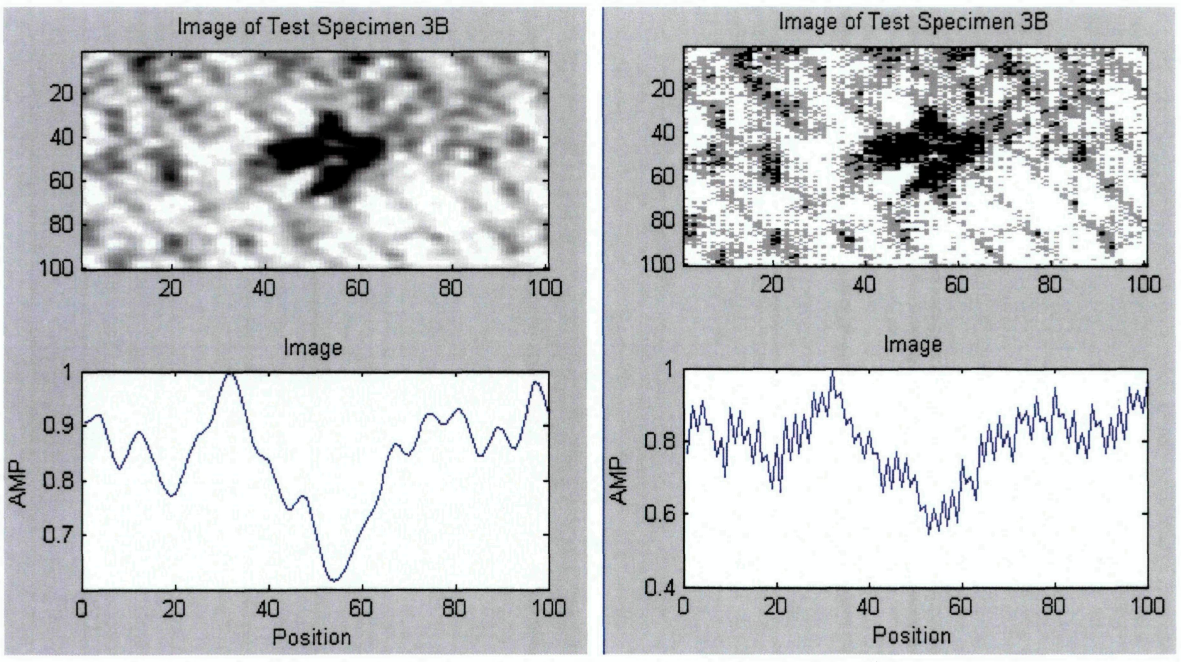
Figure 11: MATLAB Program Output Containing Green Layer Only

Table 2: Worst Case Error using Red, Green, and Blue color layers

Coupon ID	**Impact Energy (J)**	**Ultimate Compressive Load (lb_f)**	**Predicted Ultimate Compressive Load (lb_f)**	**Percent Error**
27B	16	18,825	20,338.64	8.04%
25D	20	17,249	16,638.96	-3.54%
26C	18	20,729	20,079.15	-3.14%
3A	14	19,156	21,583.95	$\boxed{12.67\%}$
24B	12	19,782	17,856.1	-9.74%
26A	10	22,190	23,274.83	4.89%

IMAGE FILTERING

While the majority of the green layer images were mostly free of noise, there existed a few images which were particularly noisy, such as Figure 10. It is difficult to identify the impact location and damage of the coupon in Figure 10; therefore, a filter had to be implemented to remove the higher frequency noise. Figure 12 shows a side-by-side view of test specimen 3B, a particularly noisy image, before it was filtered and after it was filtered to remove the high frequency noise.



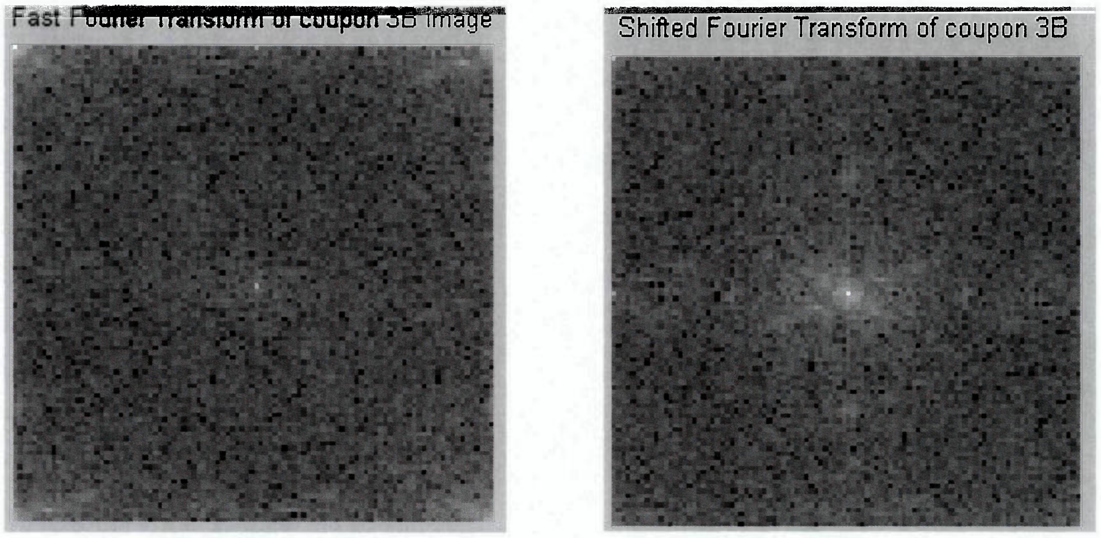
(a) (b)

Figure 12: Green Layer of Noisy C-Scan image Filtered (Left), Unfiltered (Right)

This image filtering process took place in several steps. Initially the MATLAB output indicated that the green layer of the RGB image contained the least amount of noise (Figure 9) and therefore the cleanest signal; the green layer for each impact specimen was then extracted from each image (Figure 11). Thus, the BPNN was trained and tested solely on the green layer of the C-scan images which is discussed in the Neural Network Optimization section.

While the noise of most images was completely eliminated by removing the red and blue layers, significant noise still remained in some of the images, as can be seen in Figure 12(b). In an effort to improve the BPNN's prediction of the test specimens' ultimate compressive strength, a Fast Fourier Transform (FFT) was implemented. The FFT decomposes an image into real and complex parts that represent the image in the frequency domain. The FFT allows the higher frequency noise to be eliminated, removing the pixilation in the more noisy images and leaving the noiseless images relatively unchanged.

After the FFT of the image has been taken, the pixels that constitute the high frequency noise in the image are located near the center of the FFT image, while the low frequency pixels are around the edges as seen in Figure 13(a). The FFT was shifted to move the high frequency noise to the edges of the FFT where they could then be filtered out (Figure 13(b)). While operating in the frequency domain, a simple MATLAB code was written to remove the high frequency noise by setting their values equal to 0. The MATLAB code set the values of pixels to 0 that were outside of a square around the center of the image as seen in Figure 14.


Fast Fourier Transform of coupon 3B image
Shifted Fourier Transform of coupon 3B

(a) (b)

Figure 13: Test Specimen 3B After Fast Fourier Transform (Left), and After Shifting the Fast Fourier Transform (Right)

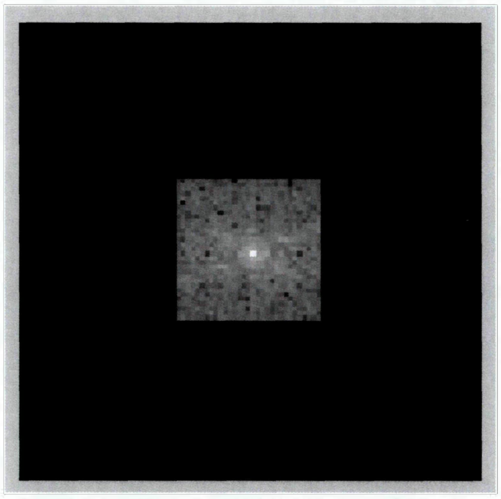


Figure 14: Fast Fourier Transform of Specimen 3B after Filter Applied

The size of the square around the center of the FFT controls how much of the image is filtered. A larger area square filters less of the image, while a smaller area square will filter more of the image. The square filter needs to be positioned at the center of the FFT image; its location is important because the FFT of a real image is "a conjugate symmetric". Conjugate symmetric refers to when a real signal is transformed to the frequency domain; each pixel is represented as a complex number with conjugate values located symmetrically at each corner of the image. To avoid getting a bad inverse, and therefore getting complex numbers as pixel values when the image is returned to the space domain, both of the pixels' conjugate values must be set equal to 0. This is accomplished by creating a square filter which is symmetric about the center of the image as seen in Figure 14.

CHAPTER 4
RESULTS

TRAINING AND TESTING SELECTION

The initial selection of coupons to be included in the training set and the testing set was random. This, however, led to testing a data set containing the highest ultimate compressive load, which confused the neural network because it encountered a higher load during testing than it had seen during training. This induced some error into the calculation and was avoided in future training and testing sets.

Therefore, the selection of coupons for the training set began by including both the coupons with the highest and lowest ultimate compressive loads. The loads were then organized from highest to lowest, and two coupons at each impact energy level were selected for training, ensuring that the coupons selected were evenly spaced to give the BPNN an optimal selection of data points. Because the sample size was quite small, as only 21 coupons were C-scanned and compressed to failure, a method referred to as "Bootstrapping Data" was utilized to increase the apparent sample size for the BPNN. The method of bootstrapping data is a technique that uses the same composite coupon multiple times in random positions in the training file to fool the neural network into believing there are more coupons in the data set than are actually present. Bootstrapping does not skew the data; it only increases the size of a small data set. Bootstrapping data was employed in this experiment by using each coupon in the training set three times in a random order. Once the training and testing data sets and files were written, the neural network needed to be optimized to yield the lowest possible worst case error. Appendix A contains the index of each coupon in the training and testing files which yielded the most promising worst case error.

NEURAL NETWORK OPTIMIZATION

The only way to find the optimal network parameters is through trial and error using the computer program NeuralWorks Professional II Plus. The architecture chosen for the BPNN began with one hidden layers. Varying the number of hidden layer neurons, the hidden layer learning coefficient, the output layer learning coefficient, the momentum, the learning ratio, the F' Offset, and the transition point through trial and error is required to arrive at the optimal neural network. This trial and error procedure can be very time consuming; thus, two methods were used to expedite the optimization process.

The two optimization techniques employed to find the optimal network were series optimization and parallel optimization. Parallel optimization entailed varying all neural network parameters independently in different networks, then once the network was optimized, all optimum parameters were placed into the same neural network. Parallel optimization is the most time efficient method of BPNN optimization; however, it can only be used when a team of people is available to optimize the neural network. It was found in this project that both parallel and series optimization yielded identical BPNN parameters.

The first optimization technique employed herein was series optimization, which entailed varying all parameters individually, selecting the value that yielded the lowest worst case error, and using that parameter value in the BPNN. The first parameter value that was varied was the number of hidden layer neurons. Figure 15 shows the variation of the worst case error as the number of hidden layer 1 neurons was increased. A closer view of the area of interest can be seen in Figure 16 which focused around 10 hidden layer 1 neurons; this was the optimal number of neurons which yielded an absolute value worst case error of 11.61%.

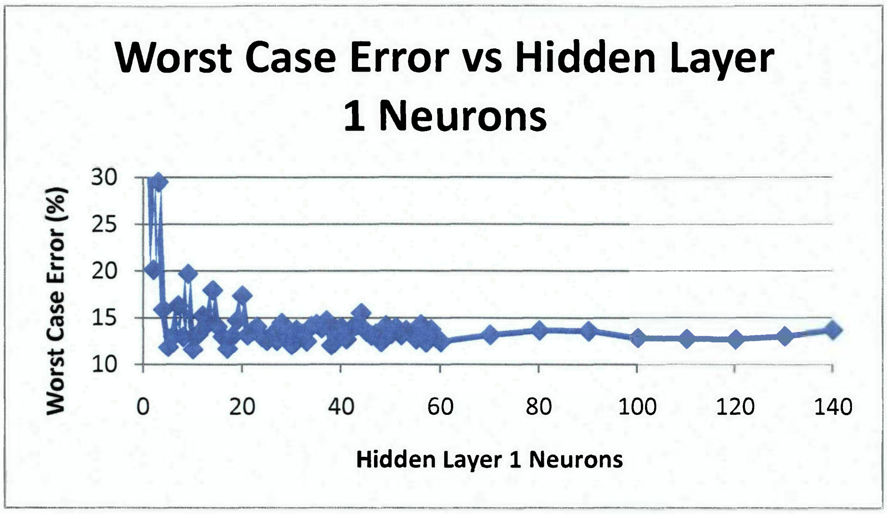


Figure 15: Hidden Layer 1 Worst case error vs. Hidden Layer 1 Neurons

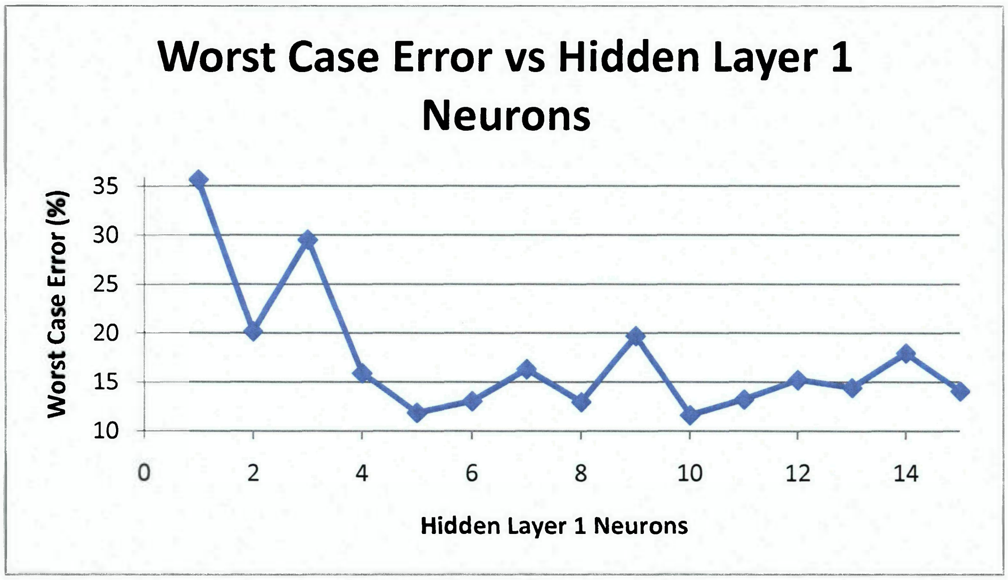


Figure 16: Zoomed in Hidden Layer 1 Worst case error vs. Hidden Layer 1 Neurons

With an absolute value worst case error of 11.61%, the number of hidden layer 2 neurons was varied to explore the need for a two hidden layer network. After varying the number of hidden layer 2 neurons between 10 and 100, Figure 17 shows that a second hidden layer was not necessary, as all the worst case errors were greater than the 11.61% value obtained for one hidden layer.

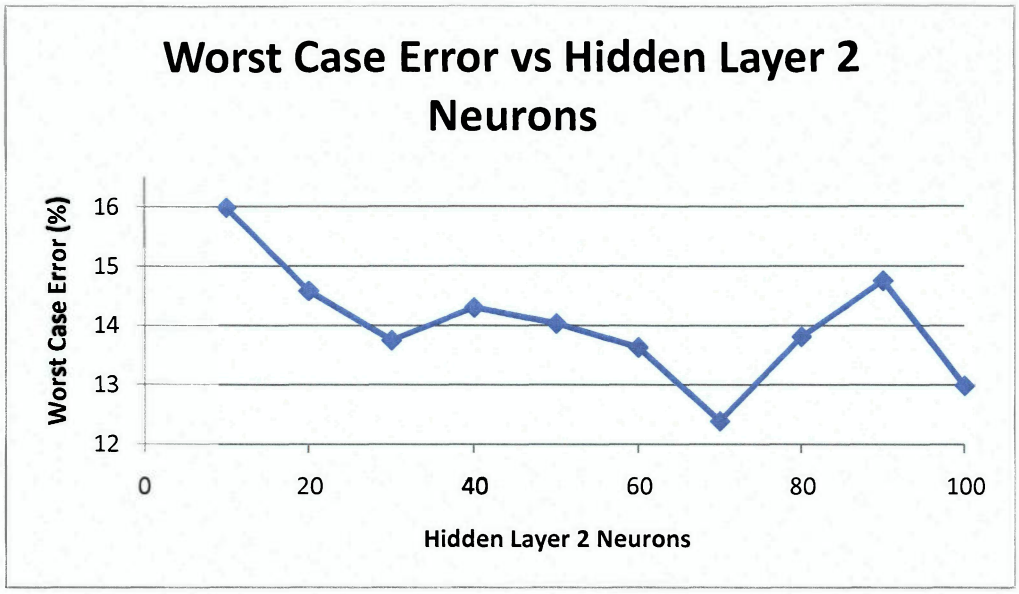


Figure 17: Worst Case Error vs. Hidden Layer 2 Neurons

As the network only had one hidden layer, only the hidden layer 1 learning coefficient needed to be varied. Hence, the hidden layer 1 learning coefficient was varied between 0.001 and 0.5, and the optimal value was found to be 0.001, as seen on Figure 18. This value yielded a slight reduction in the absolute value worst case error of from 11.61% down to 11.5%.

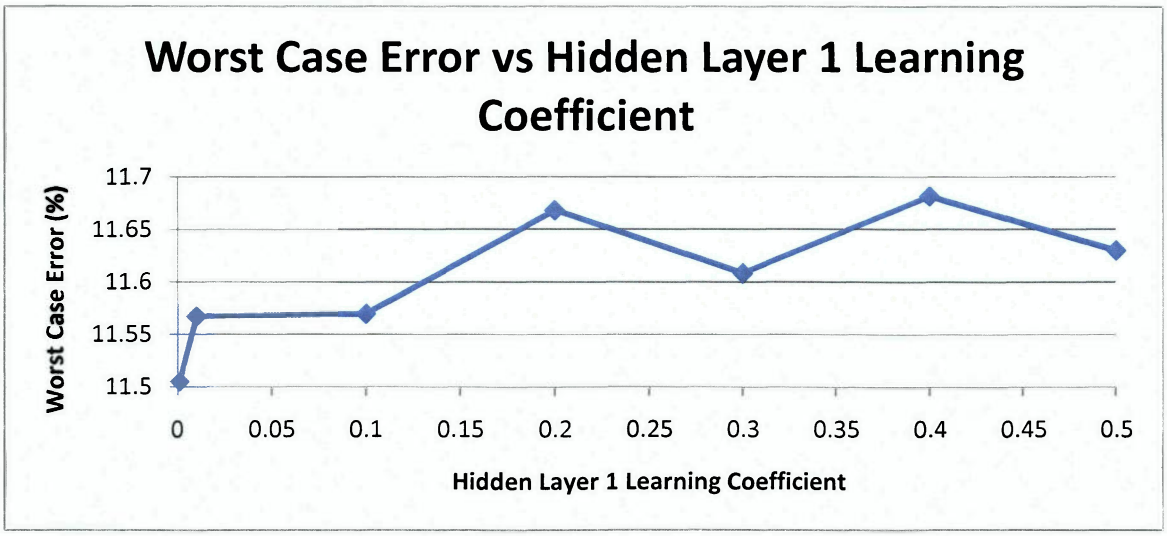


Figure 18: Worst Case Error vs. Hidden Layer 1 Learning Coefficient

The next logical step was to vary the output layer learning coefficient. This proved to be the most successful in decreasing the worst case error. By decreasing the output layer coefficient to 0.017, the error reduced from 11.5% to 5.19%, a decrease of over 50% (Figure 19).

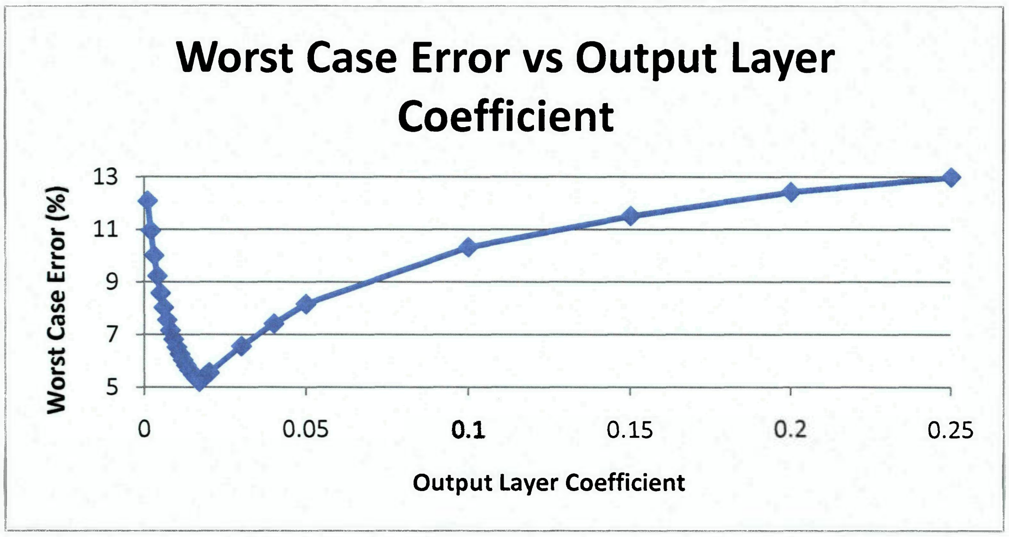


Figure 19: Worst Case Error vs. Output Layer Coefficient

Figure 20 through Figure 23 show the variation of the remaining four parameters as they were modified and optimized using the same techniques.

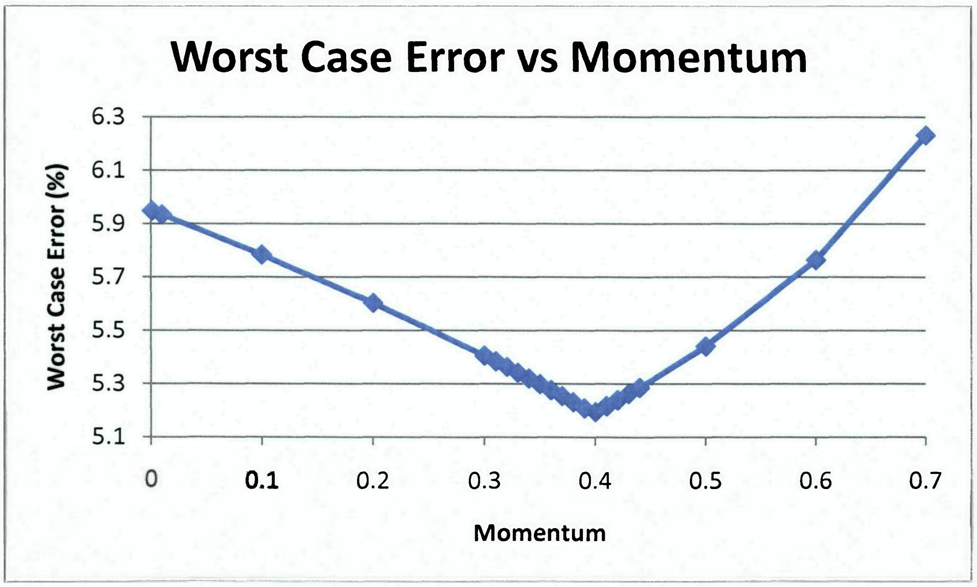


Figure 20: Worst Case Error vs. Momentum

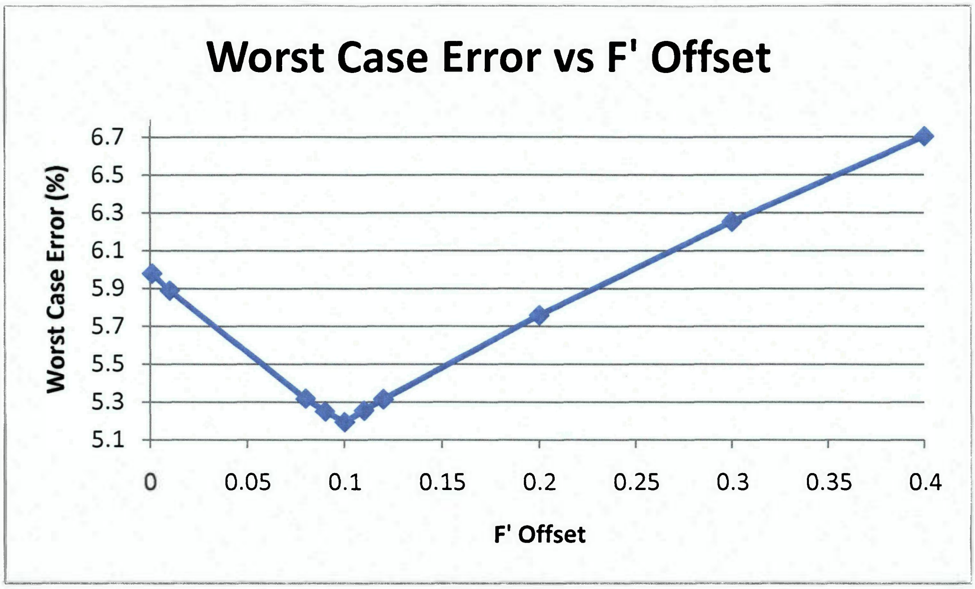


Figure 21: Worst Case Error vs. F' Offset

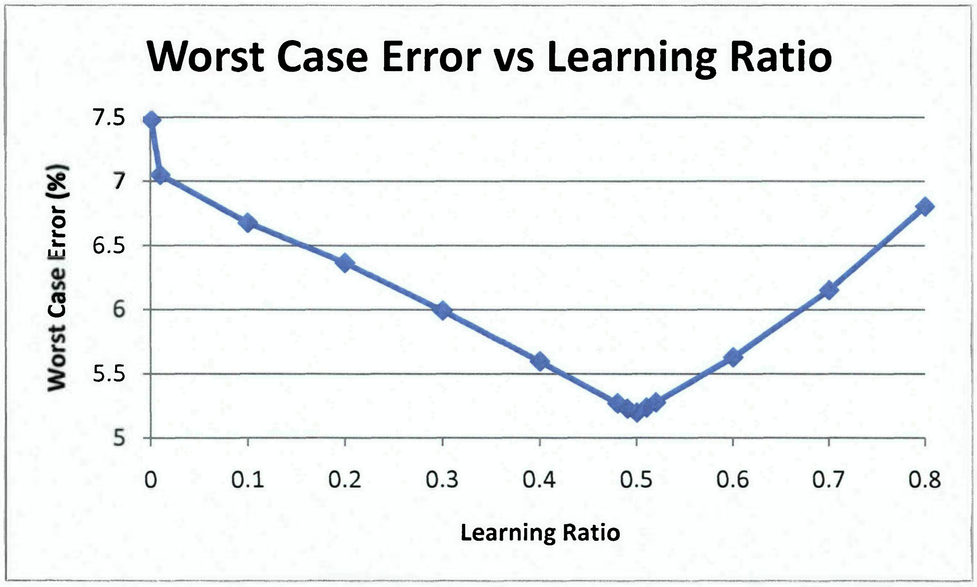


Figure 22: Worst Case Error vs. Learning Ratio

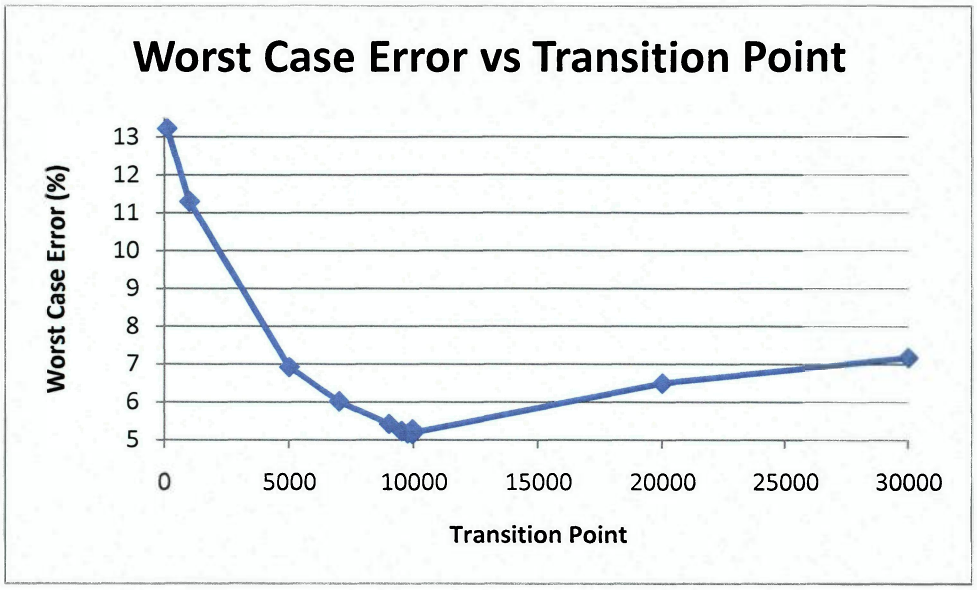


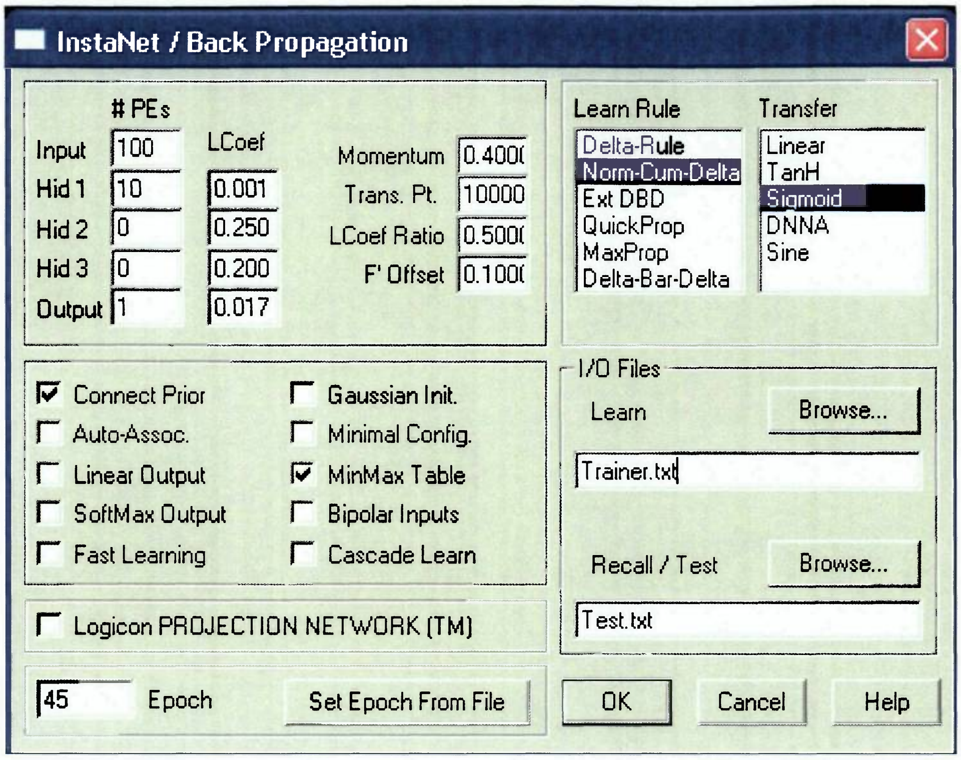
Figure 23: Worst Case Error vs. Transition Point

Table 3 summarizes the optimal parameter settings found for the neural network. Using these values, the BPNN was able to predict the ultimate compressive load with a worst case testing error of -5.16%, well within the ±10% target range.

Table 3: Optimized Neural Network Configuration

Number of Input Neurons	100
Hidden Layer 1 Neurons	10
Hidden Layer Coefficient	0.001
Number of Output Neurons	1
Output Layer Coefficient	0.017
Learning Coefficient	0.5
Momentum Value	0.4
Transition Point	10,000
Transfer Function	Sigmoid
Learning Rule	Normalized-Cumulative-Delta
F' Offset	0.1

The NeuralWorks Professional II Plus software BPNN setup screen with optimal settings can be seen below in Figure 24.


InstaNet / Back Propagation

	# PEs	LCoef
Input	100	
Hid 1	10	0.001
Hid 2	0	0.250
Hid 3	0	0.200
Output	1	0.017

Momentum	0.400(
Trans. Pt.	10000
LCoef Ratio	0.500(
F' Offset	0.100(

Learn Rule: Delta-Rule, [✓ Norm-Cum-Delta], Ext DBD, QuickProp, MaxProp, Delta-Bar-Delta

Transfer: Linear, TanH, [✓ Sigmoid], DNNA, Sine

- [✓ Connect Prior]
- Auto-Assoc.
- Linear Output
- SoftMax Output
- Fast Learning
- Gaussian Init.
- Minimal Config.
- [✓ MinMax Table]
- Bipolar Inputs
- Cascade Learn
- Logicon PROJECTION NETWORK (TM)

I/O Files

Learn: Trainer.txt

Recall / Test: Test.txt

Epoch: 45

Figure 24: Optimal BPNN Settings

The compiled results, including both training and prediction errors can be seen in Table 4. The rows highlighted in yellow are coupons which were included in the testing file; the other rows were coupons used in the training file. It should be noted that a worst case testing error of within ±10% was the target of this research; unfortunately, by using optimal settings, the worst case training error was found to be -12.52%, slightly outside of this range. Therefore, further optimization was sought.

Table 4: Unfiltered Image BPNN Results

Coupon ID	**Impact Energy (J)**	**Ultimate Compressive Load (lb_f)**	**Predicted Ultimate Compressive Load (lb_f)**	**Percent Error**
26D	20	20,024	19,054.8145	-4.84%
24D	20	17,250	17,640.0879	2.26%
25D	20	17,249	17,470.6836	1.29%
26C	18	20,729	19,663.1777	-5.14%
25C	18	20,010	19,657.5039	-1.76%
27C	18	18,986	19,049.0488	0.33%
27B	16	18,825	19,643.6719	4.35%
27D	16	18,742	19,276.1270	2.85%
24C	16	17,944	18,925.3066	5.47%
3A	14	19,156	19,802.0117	3.37%
3D	14	19,152	21,107.6641	10.21%
2C	14	16,200	16,996.6348	4.92%
2A	12	21,750	21,078.8828	-3.09%
25B	12	21,749	20,173.1055	-7.25%
2B	12	18,900	20,036.0273	6.01%
3B	12	20,250	20,682.1895	2.13%
24B	12	19,782	19,763.3281	-0.09%
27A	12	17,249	19,385.4375	12.39%
24A	10	24,195	21,166.8926	-12.52%
26A	10	22,190	21,044.6621	-5.16%
25A	10	21,815	20,603.8965	-5.55%

The significant difference in testing and training errors can be attributed to the relatively small output layer learning coefficient; this could possibly have caused some overtraining to occur on the data. Figure 25 shows the comparison between the worst case training error and the worst case testing error as the output layer learning coefficient was varied. It can be clearly observed from the plot that as the output layer learning coefficient approaches zero, the training error steadily increases. Thus, the optimal value of the output layer learning coefficient is not 0.017, as had been earlier supposed, where the BPNN produces the minimal testing error, but rather is where the training and testing error curves intersect at an output layer learning coefficient of 0.063. Here the worst case training and testing errors should be approximately equal.

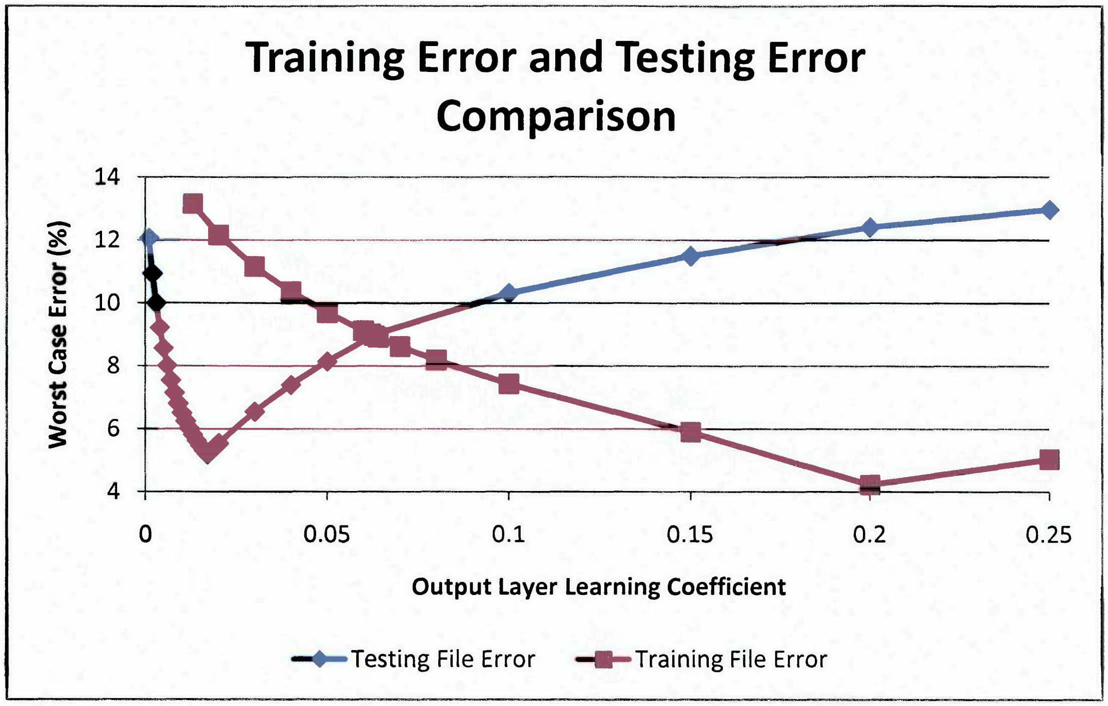


Figure 25: Comparison Between Prediction and Training Errors

After comparing the training and testing errors and finding the optimal output learning coefficient, the final BPNN prediction errors could be calculated. The predicted ultimate CAI loads of the unfiltered image data can be seen in Table 5. These values were calculated using the optimal output learning coefficient of 0.063, which yielded a worst case error for training and testing of -8.96% and -8.98%, respectively. As noted, these two values are approximately equal and both are within the ±10% worst case prediction error goal.

Table 5: BPNN Results using 0.063 Output Layer Coefficient

Coupon ID	Impact Energy (J)	Ultimate Compressive Load (lb_f)	Predicted Ultimate Compressive Load (lb_f)	Percent Error
26D	20	20,024	19,764.19	-1.30%
24D	20	17,250	16,981.84	-1.55%
25D	20	17,249	17,328.87	0.46%
26C	18	20,729	18,866.51	$\boxed{-8.98\%}$
25C	18	20,010	19,773.29	-1.18%
27C	18	18,986	19,326.75	1.79%
27B	16	18,825	19,422.61	3.17%
27D	16	18,742	18,726.62	-0.08%
24C	16	17,944	18,332.10	2.16%
3A	14	19,156	20,328.60	6.12%
3D	14	19,152	20,404.80	6.54%
2C	14	16,200	16,447.73	1.53%
2A	12	21,750	21,502.41	-1.14%
25B	12	21,749	21,371.35	-1.74%
2B	12	18,900	19,805.28	4.79%
3B	12	20,250	20,323.34	0.36%
24B	12	19,782	19,218.11	-2.85%
27A	12	17,249	18,452.20	6.98%
24A	10	24,195	22,026.11	$\boxed{-8.96\%}$
26A	10	22,190	21,571.86	-2.79%
25A	10	21,815	21,028.18	-3.61%

FILTERED AND UNFILTERED IMAGE COMPARISON

The C-scan images were placed into the BPNN for ultimate CAI load prediction after they were FFT filtered, and a significant amount of high frequency noise was removed. Following the same optimization procedure as with the unfiltered images, the BPNN was optimized, yielding the results tabulated in Table 6. Here it can be seen that FFT image filtering did improve the worst case prediction error down from -8.98% to 8.65%. This reduction in error was not nearly as much as was expected.

Table 6: Filtered Image BPNN Results

Coupon ID	Impact Energy (J)	Ultimate Compressive Load (lb_f)	Predicted Ultimate Compressive Load (lb_f)	Percent Error
26D	20	20,024	19,795.97	-1.14%
24D	20	17,250	17,045.25	-1.19%
25D	20	17,249	18,306.68	6.13%
26C	18	20,729	18,942.68	-8.62%
25C	18	20,010	19,858.97	-0.75%
27C	18	18,986	19,351.67	1.93%
27B	16	18,825	18,478.77	-1.84%
27D	16	18,742	18,471.83	-1.44%
24C	16	17,944	18,155.08	1.18%
3A	14	19,156	20,735.68	8.25%
3D	14	19,152	20,063.35	4.76%
2C	14	16,200	16,608.53	2.52%
2A	12	21,750	21,512.81	-1.09%
25B	12	21,749	21,400.50	-1.60%
2B	12	18,900	19,886.24	5.22%
3B	12	20,250	20,379.29	0.64%
24B	12	19,782	18,632.73	-5.81%
27A	12	17,249	18,741.61	8.65%
24A	10	24,195	22,124.94	-8.56%
26A	10	22,190	21,316.96	-3.93%
25A	10	21,815	20,981.32	-3.82%

From the summarized results of Table 6 it is clear that the BPNN was able to predict accurately with both noisy and FFT filtered images. Utilizing only its iterative optimization scheme, the BPNN was able to remove most of the high frequency noise, as the weights of the respective neurons which contained most of the image noise approached zero. The FFT image filtering process did provide some reduction in worst case error; however, in this case, the rectangular filter used to remove the high frequency noise may have removed too much frequency information in the image, thereby resulting in a higher than expected worst case error.

Table 7: Filtered and Unfiltered Image Comparison

	Worst Case Testing Error	**Worst Case Training Error**
Unfiltered	-8.98%	-8.96%
Filtered	-8.62%	8.65%

MATERIAL ALLOWABLES

The tolerance interval within which the BPNN predictions should fall is referred to as the B-basis material allowables for the composite coupons. B-basis allowables are defined as the tolerance interval within which there is a 95% confidence that 90% of all future ultimate compression after impact (CAI) ultimate loads will fall [10]. The B-basis tolerance interval may be calculated from the following equation:

$$Interval = \pm K(n, P, c)s_x$$

where	Interval = tolerance interval for B-basis allowables
	K = factor dependant on N, P, C parameters
	n = number of samples in a certain group
	P = fraction of population
	c = confidence interval
	s_x = standard deviation.

Table 8: B-basis Allowables Tolerance Interval

Impact Energy (J)	**Number of Coupons**	**Mean Ultimate Compressive Load (lb_f)**	**Standard Deviation (lb_f)**	**K factor**	**B-basis Allowables Interval (lb_f)**	**Upper Limit (lb_f)**	**Lower Limit (lb_f)**
10	3	22,733.33	1,279.65	6.919	±8,853.91	31,587.24	13,879.42
12	6	19,946.67	1,731.62	3.723	±6,446.80	26,393.47	13,499.86
14	3	18,169.33	1,705.49	6.919	±11,800.31	29,969.64	6,369.021
16	3	18,503.67	486.46	6.919	±3,365.81	21,869.48	15,137.86
18	3	19,908.33	875.94	6.919	±6,060.60	25,968.94	13,847.73
20	3	18,174.33	1,601.86	6.919	±11,083.26	29,257.59	7,091.075

It can be clearly observed in Figure 26 that all the ultimate compressive loads predicted by the optimized BPNN fall well within the B-basis allowables of the graphite/epoxy coupon sample group.

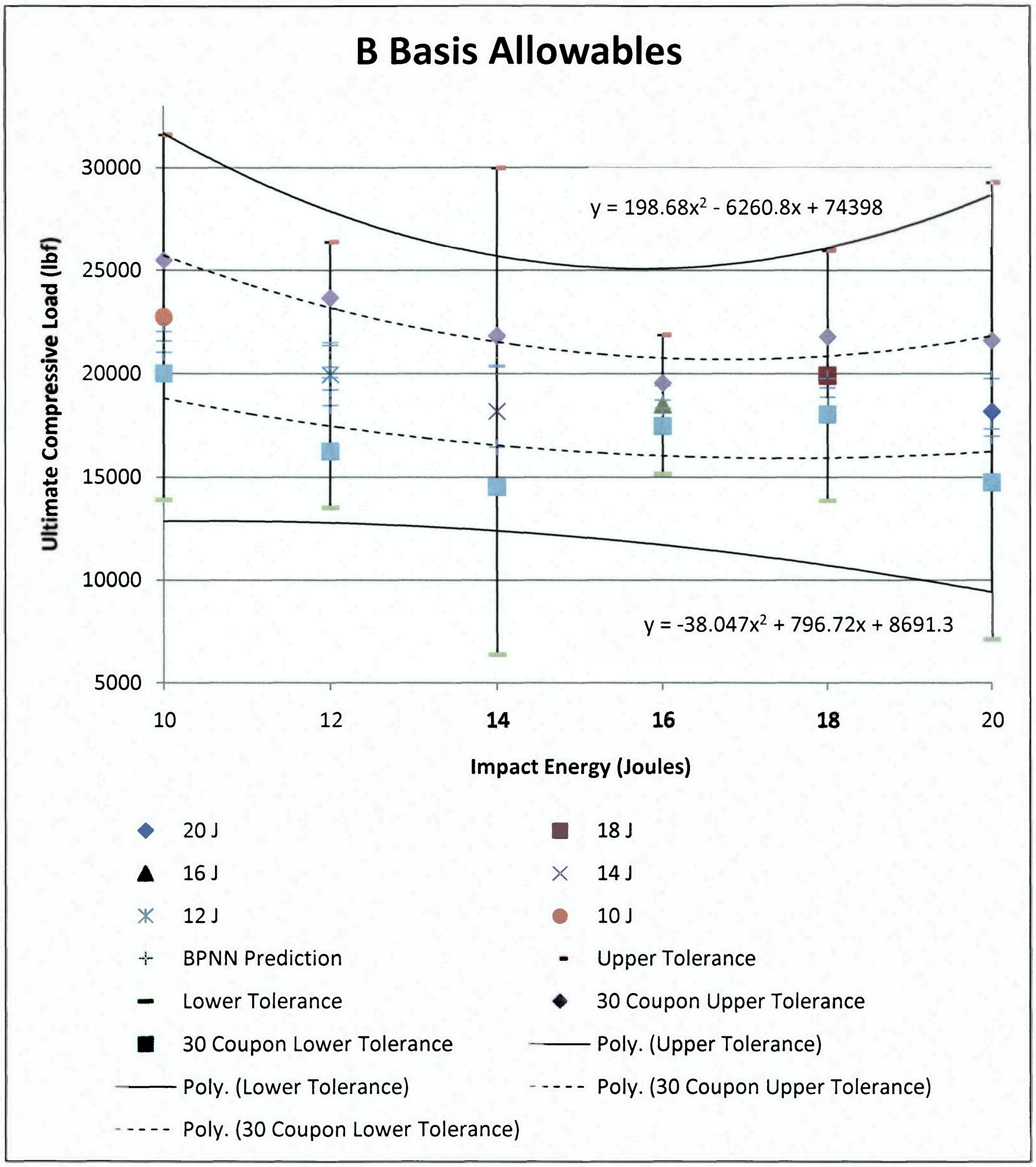


Figure 26: B-Basis Allowables

Increasing the number of coupons at each impact damage energy level would obviously decrease the B-basis allowables. If instead of the 3 or 6 samples at each energy level, there were 30 coupons at each energy level, the K values would all decrease from 6.919 (3 samples) or 3.723 (6 samples), as seen in Table 8, to 2.140 (30 samples). Assuming the same mean and standard deviation values for the increased sample size, the

B-basis allowables would decrease significantly to the values shown by the dashed lines in Figure 26. Note that all the BPNN predictions are well within these more conservative values as well. This is significant because the B-basis allowables are typically calculated for composites based on a sample size of 30 or more test specimens. Unfortunately, this research did not have the resources available to generate such a large sample size.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

CONCLUSIONS

- After initial image preprocessing, a backpropagation neural network (BPNN) using the green layer only data from the ultrasonic C-scan image of barely visible impact damage (BVID) in graphite-epoxy composite laminates was able to accurately predict the ultimate compression after impact (CAI) load with a worst case error of -8.98%, which was within the ±10% goal for this research and comfortably within the B-basis allowable for composites.
- Because the Fast-Fourier Transform (FFT) noise removal routine resulted in a slight improvement in the prediction capability of the BPNN, down from a worst case error of -8.98% to 8.65%, it can be concluded that some high frequency image noise was removed by the FFT, which aided the BPNN in making more accurate CAI load predictions.
- This research has demonstrated the viability of an ultrasonics based nondestructive evaluation (NDE) technique that could save aircraft manufacturers and maintenance companies thousands of dollars on unnecessary repairs by giving a trained technician the ability to objectively evaluate the effect of BVID on any composite part and predict with confidence the effect of the damage on the ultimate CAI load.

RECOMMENDATIONS

- Manually eliminating individual points in the FFT frequency image and setting their values to zero, as was done here, would be far too time consuming for an operational assessment of impact damage. Moreover, the square filter used in this research for FFT noise removal may have been too aggressive in eliminating pixels in the frequency domain. Future research might investigate the effect of different filters on image noise removal.

- Future research might also inquire into the possibility of using the raw reflectivity of the C-scan image, rather than using solely the green layer data, for image manipulation and ultimate CAI load prediction.

REFERENCES

1. Quilter, Adam. Composites in Aerospace Applications. [Online] [Cited: April 12, 2011.] http://cis.ihs.com/NR/rdonlyres/AEF9A38E-56C3-4264-980C-D8D6980A4C84/0/444.pdf.

2. Hess, Christopher D. *Residual Compressive Strength Prediction of Carbon/Epoxy laminates Subjected to Low Velocity Impact Damage.* Daytona Beach : Embry-Riddle Aeronautical University, 2003. M.S. Aerospace Engineering Thesis.

3. Nguyen, T-K.D. *Damage Assessment and Strength Prediction in S2-Glass/Epoxy Laminates Subjected to Low Energy Impact.* Daytona Beach : Embry-Riddle Aeronautical University, 2005. M.S. Aerospace Engineering Thesis.

4. Gunasekera, Anthony M. *Compression After Impact Strength Prediction in Graphite/Epoxy Laminates Using Acoustic Emission and Artificial Neural Networks.* Daytona Beach : Embry-Riddle Aeronautical University, 2009. M.S. Aerospace Engineering Thesis.

5. American Society for Nondestructive Testing. *Nondestructive Testing Handbook, Ultrasonic Testing.* [ed.] Gary L Workman, Doron Kishoni and Patrick O Moore. 3rd Edition. Columbus : s.n., 2007. Vol. 7.

6. *Experimental Wavelet Analysis and Applications to Ultrasonic Non-destructive Evaluation.* Park, Ik Keun, Park, Un Su and Ahn, Hyung Keun. [ed.] Sook In Kwun and Jai Won Byeon. Seoul, Korea : Seoul National University of Technology. 15th World Conference on Nondestructive Testing. http://www.ndt.net/article/wcndt00/papers/idn347/idn347.htm.

7. Russ, John C. *The Image Processing Handbook.* 4th Edition. Raleigh : CRC Press, 2002. Materials Science and Engineering Department, North Carolina State University.

8. *Standard Test Method for Compressive Residual Strength Properties of Damaged Polymer Matrix Composite Plates. D7137/D 7137 M.* ASTM Standards. Conshohocken, PA : ASTM International, 2007.

9. *Standard Test Method for Measuring the Damage Resistance of a Fiber-Reinforced Polymer Matrix Composite to a Drop-Weight Impact Event. D7136/D 7136 M.* ASTM Standards. Conshohocken, PA : ASTM International, 2007.

10. Grégoire, Alexandre David. *Ultimate Compression After Impact Load Prediction in Graphite/Epoxy Coupons Using Neural Network and Multivariate Statistical Analyses.* Daytona Beach : Embry-Riddle Aeronautical University, 2011. M.S. Aerospace Engineering Thesis.

11. Cartz, Louis. *Nondestructive Testing, Radiography, Ultrasonics, Liquid Penetrant, Magnetic Particle, Eddy Current.* Materials Park : ASM International, 1995.

12. Haykin, Simon. *Neural Networks A Comprehensive Foundation.* [ed.] John Griffin. Hamilton : Macmillan College Publishing Company, 1994.

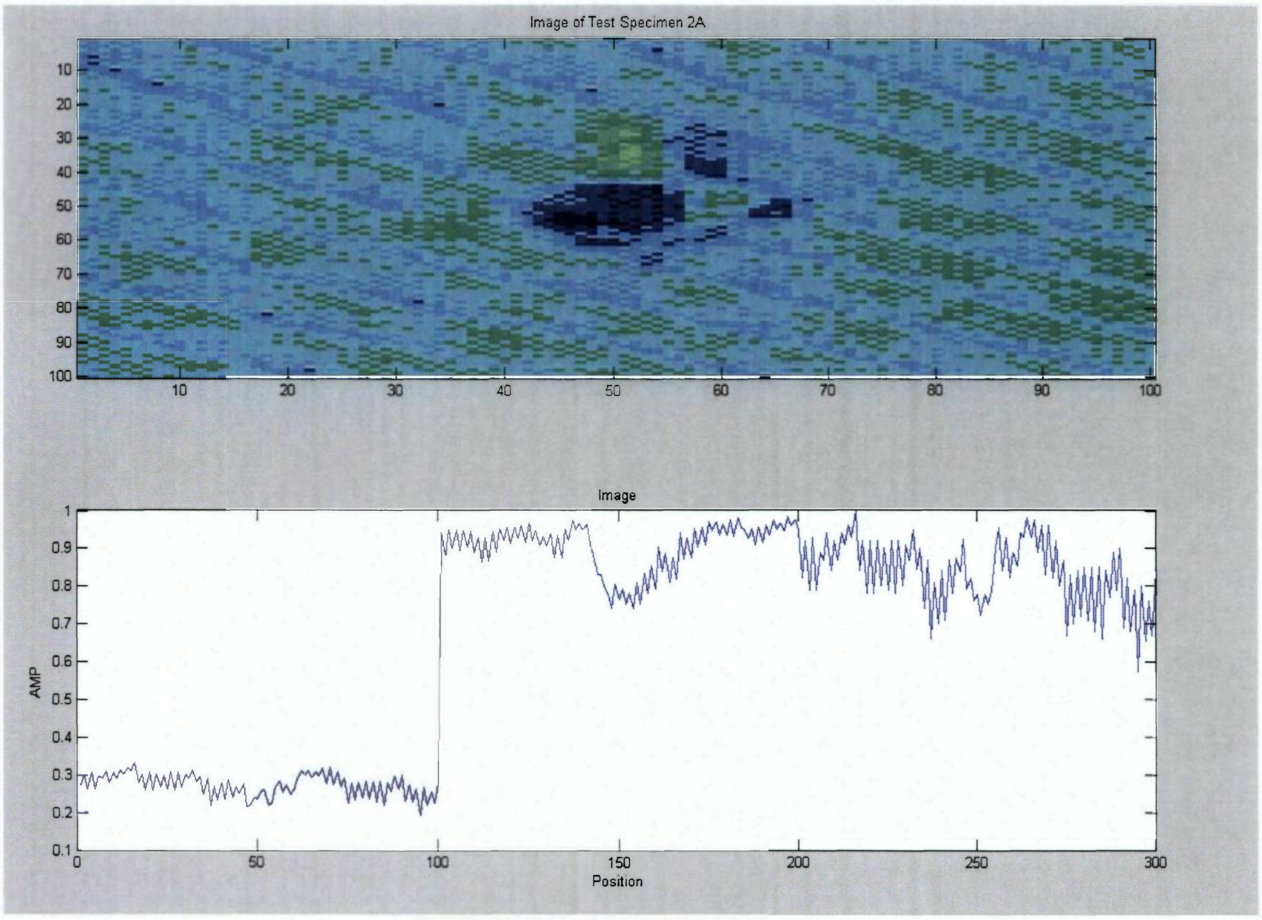
13. Dorfman, Michele D. *Ultimate Strength Prediction in Fiberglass/Epoxy Beams Subjected to Three Point Bending Using Acoustic Emission and Neural Networks.* Daytona Beach : Embry-Riddle Aeronautical University, 2004. M.S. Aerospace Engineering Thesis.

14. A.B. Pacific, E.v.K. Hill, Nikolas L. Geiselman, Christopher J. Foti, Matthew D. Gonitzke and Hannah L. Surber, "Neural Network Prediction of Ultimate Compression After Impact Loads in Graphite-Epoxy Coupons from Ultrasonic C-Scan Images," ASNT Fall Conference & Quality Testing Show 2010, American Society for Nondestructive Testing, Columbus, OH, 2010, 8 pages.

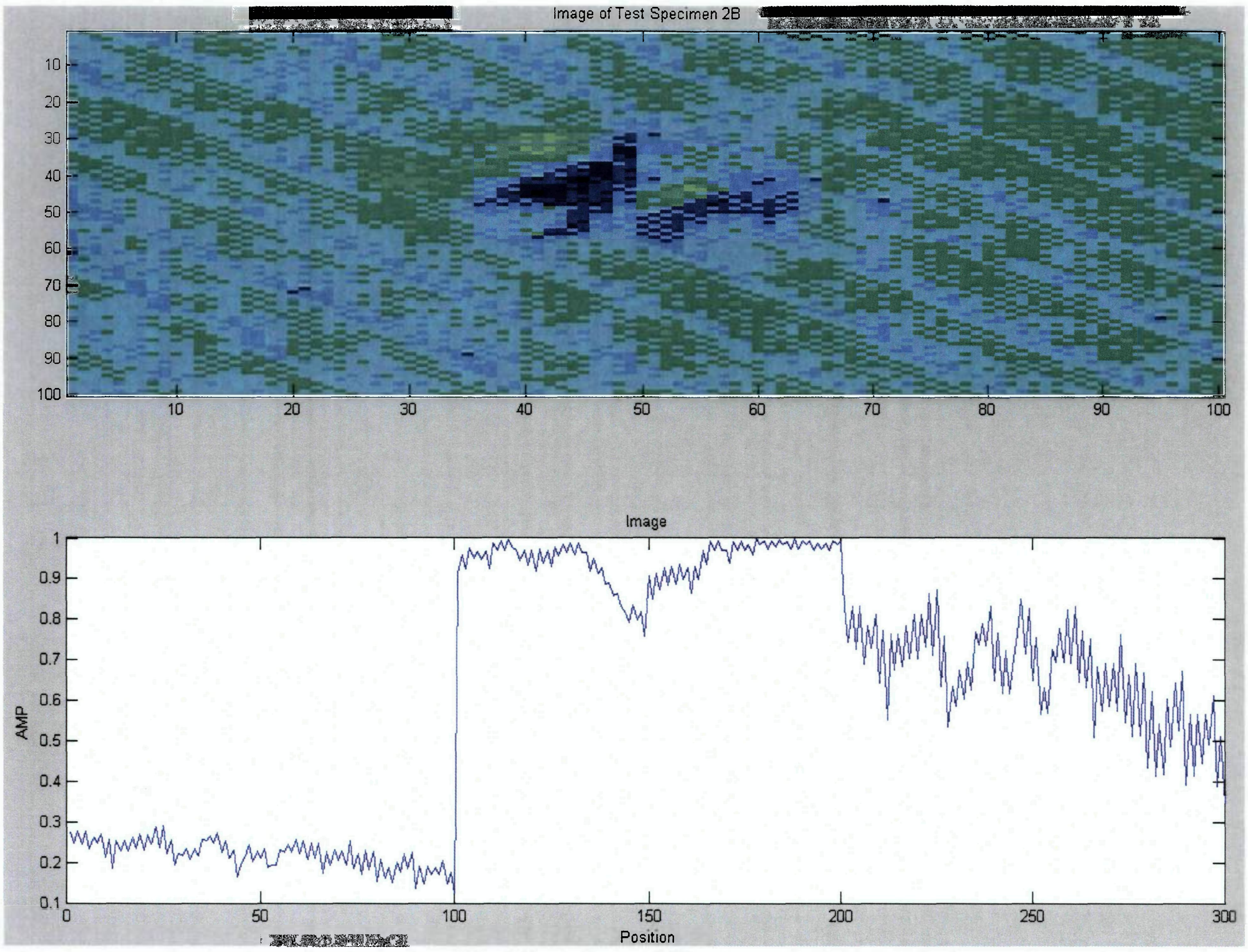
APPENDIX A: TRAINING AND TESTING FILE INDEX

Coupon ID	Impact Energy (J)	Ultimate Compressive Strength (lbf)	Testing File Index	Training File Index		
26D	20	20,024		6	29	39
24D	20	17,250		9	27	34
25D	20	17,249	2			
26C	18	20,729	3			
25C	18	20,010		2	22	25
27C	18	18,986		13	26	37
27B	16	18,825	1			
27D	16	18,742		1	7	18
24C	16	17,944		5	14	42
3A	14	19,156	4			
3D	14	19,152		19	30	41
2C	14	16,200		4	11	31
2A	12	21,750		43	44	45
25B	12	21,749		17	21	40
3B	12	20,250		8	24	36
24B	12	19,782	5			
2B	12	18,900		3	23	32
27A	12	17,249		16	20	38
24A	10	24,195		10	33	35
26A	10	22,190	6			
25A	10	21,815		12	15	28

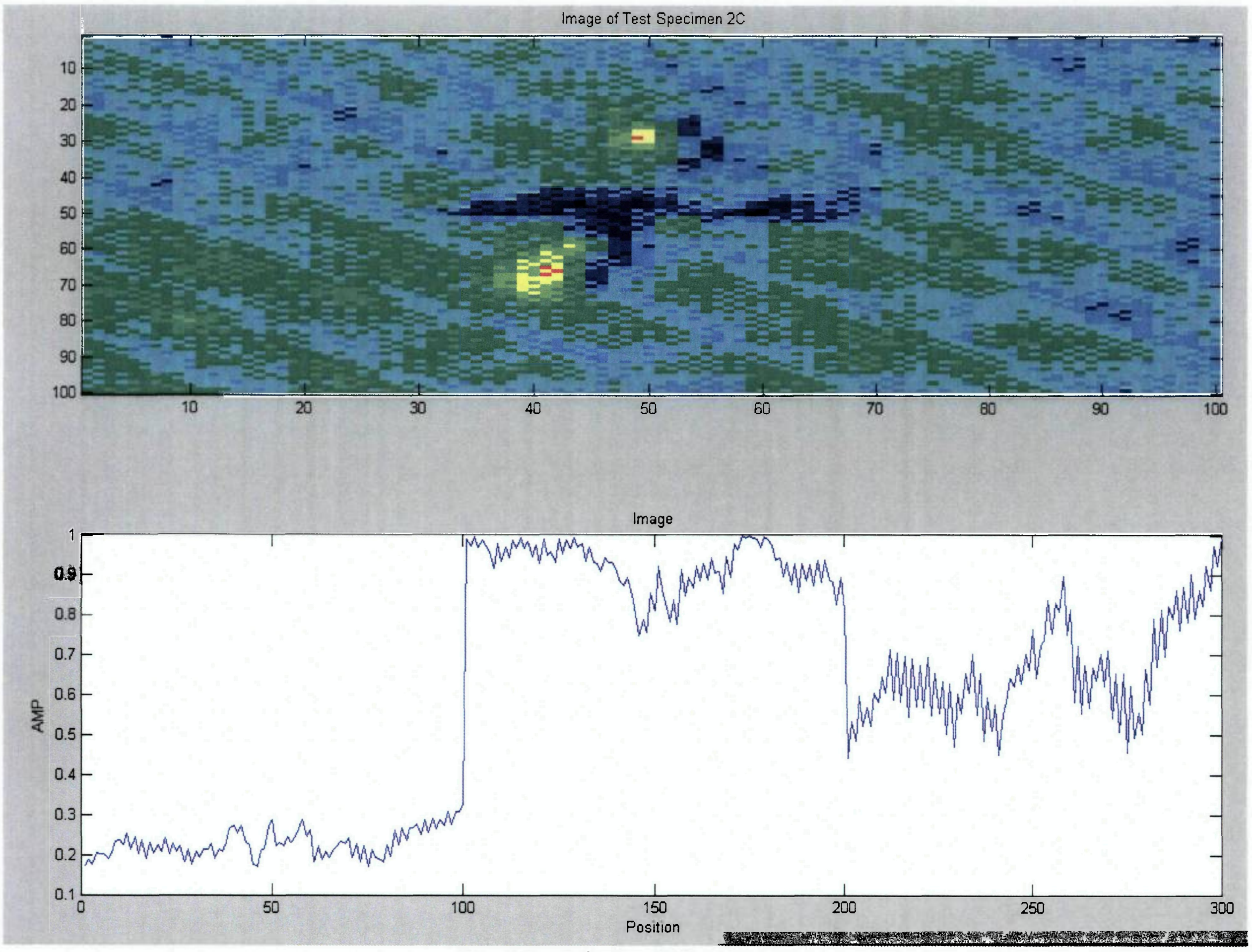
APPENDIX B: C-SCAN IMAGES WITH MATLAB OUTPUT



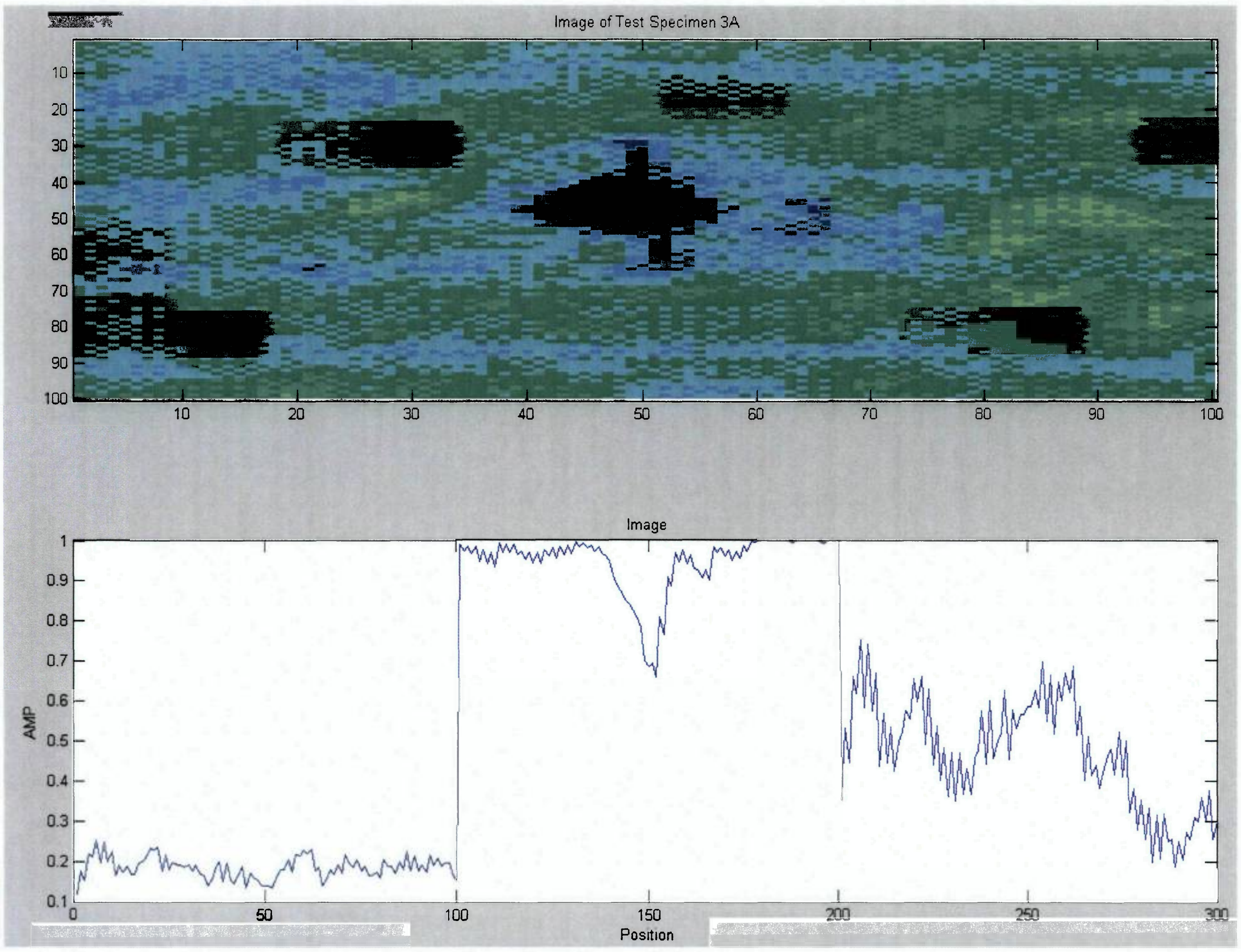
Test Specimen 2A



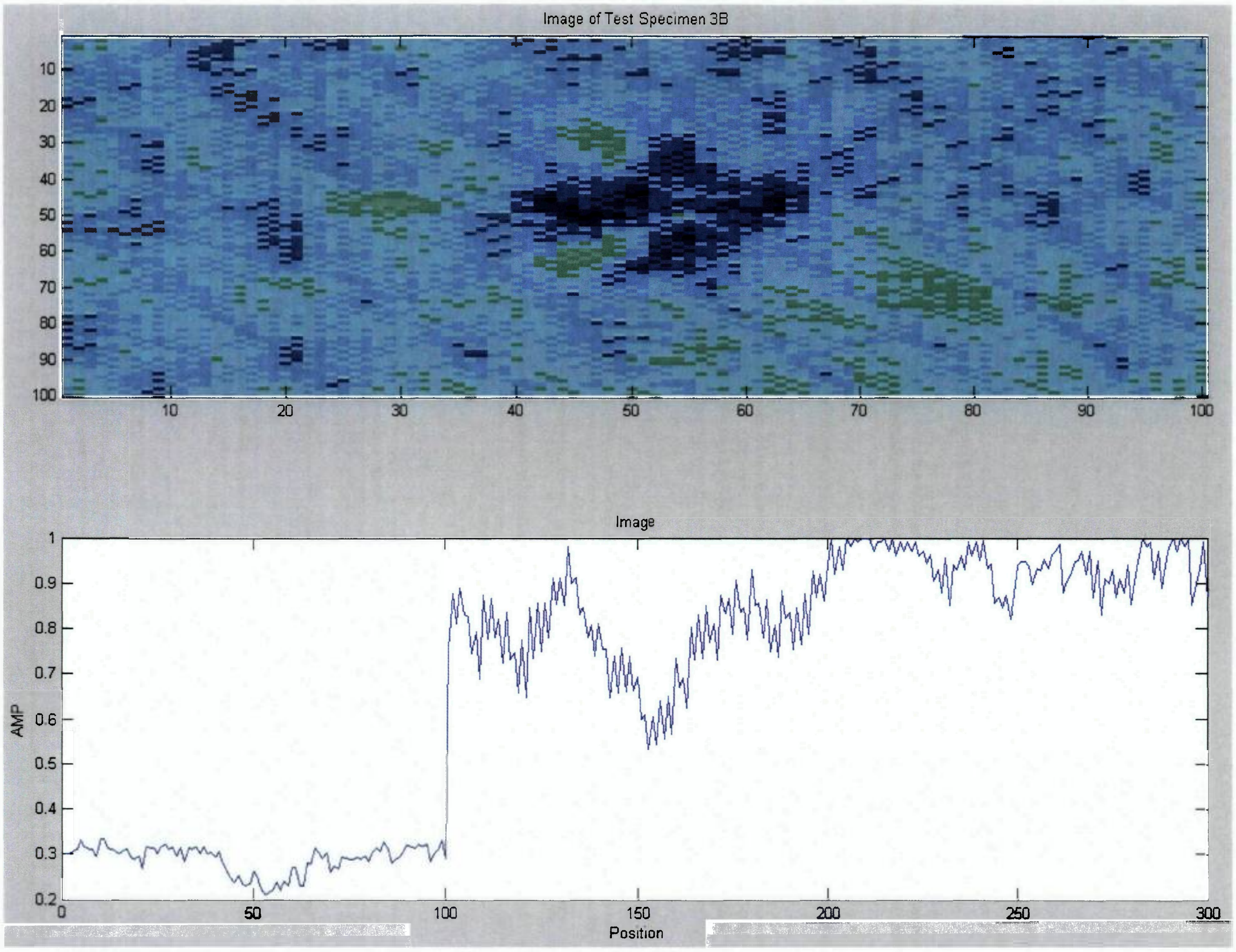
Test Specimen 2B



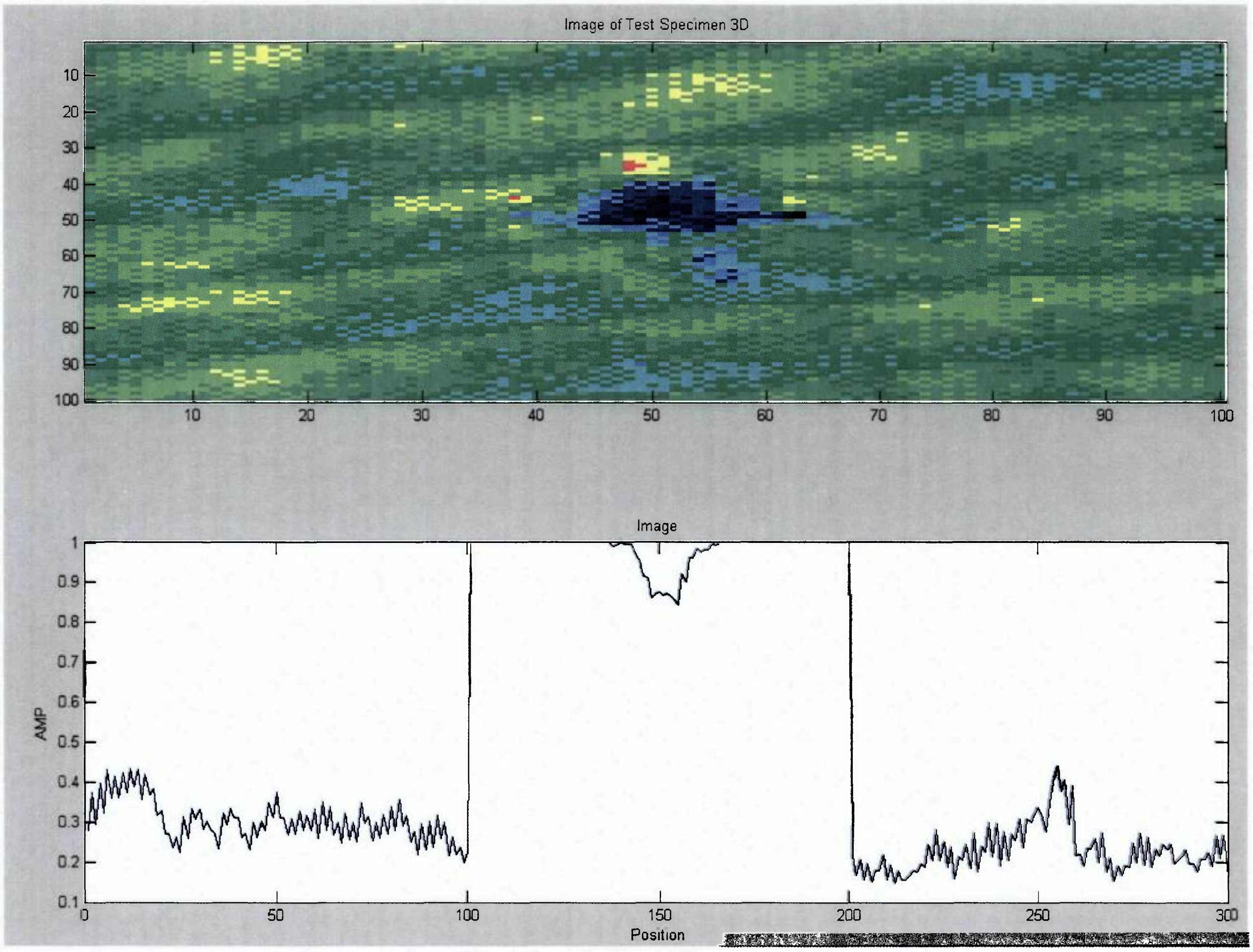
Test Specimen 2C



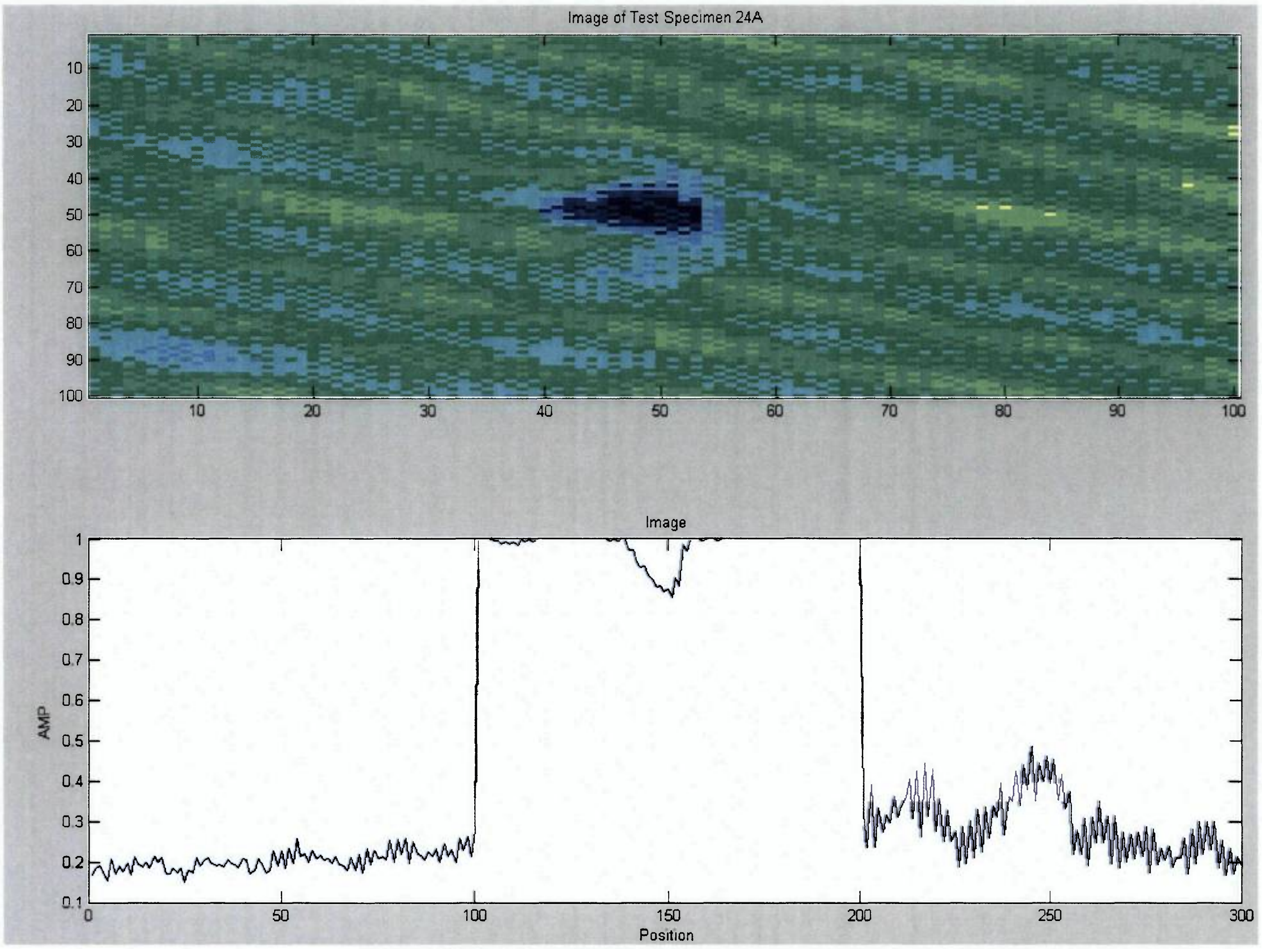
Test Specimen 3A



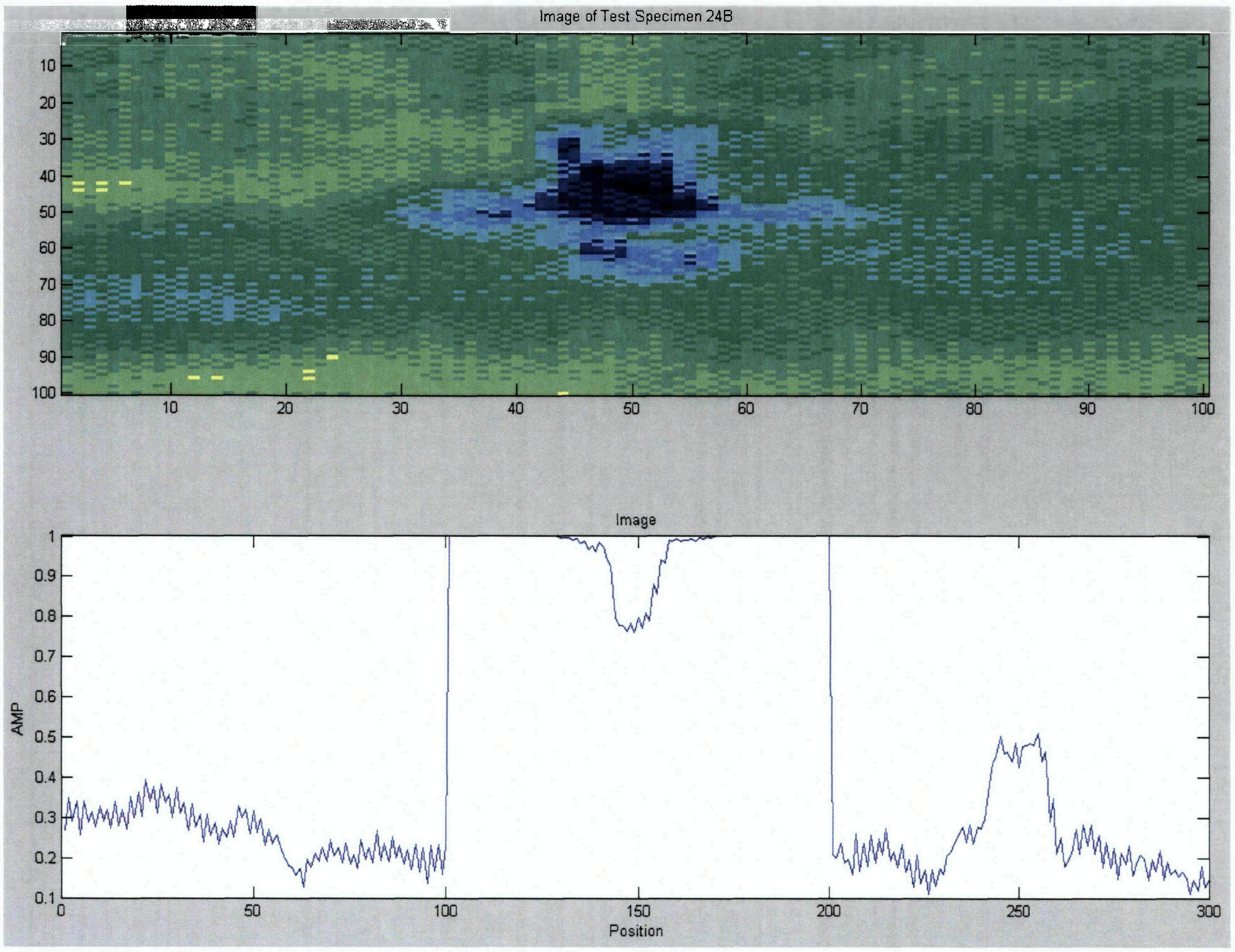
Test Specimen 3B



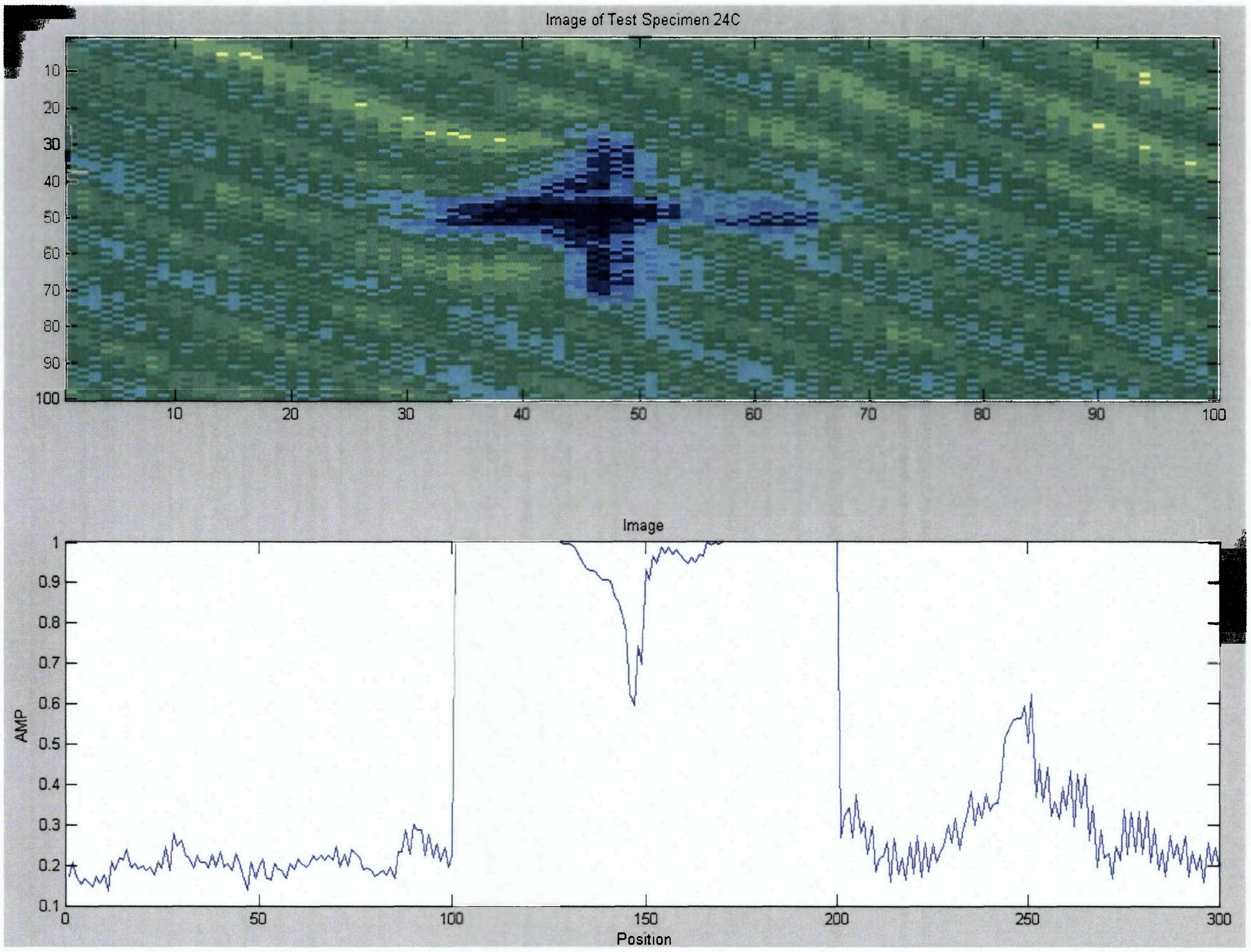
Test Specimen 3D



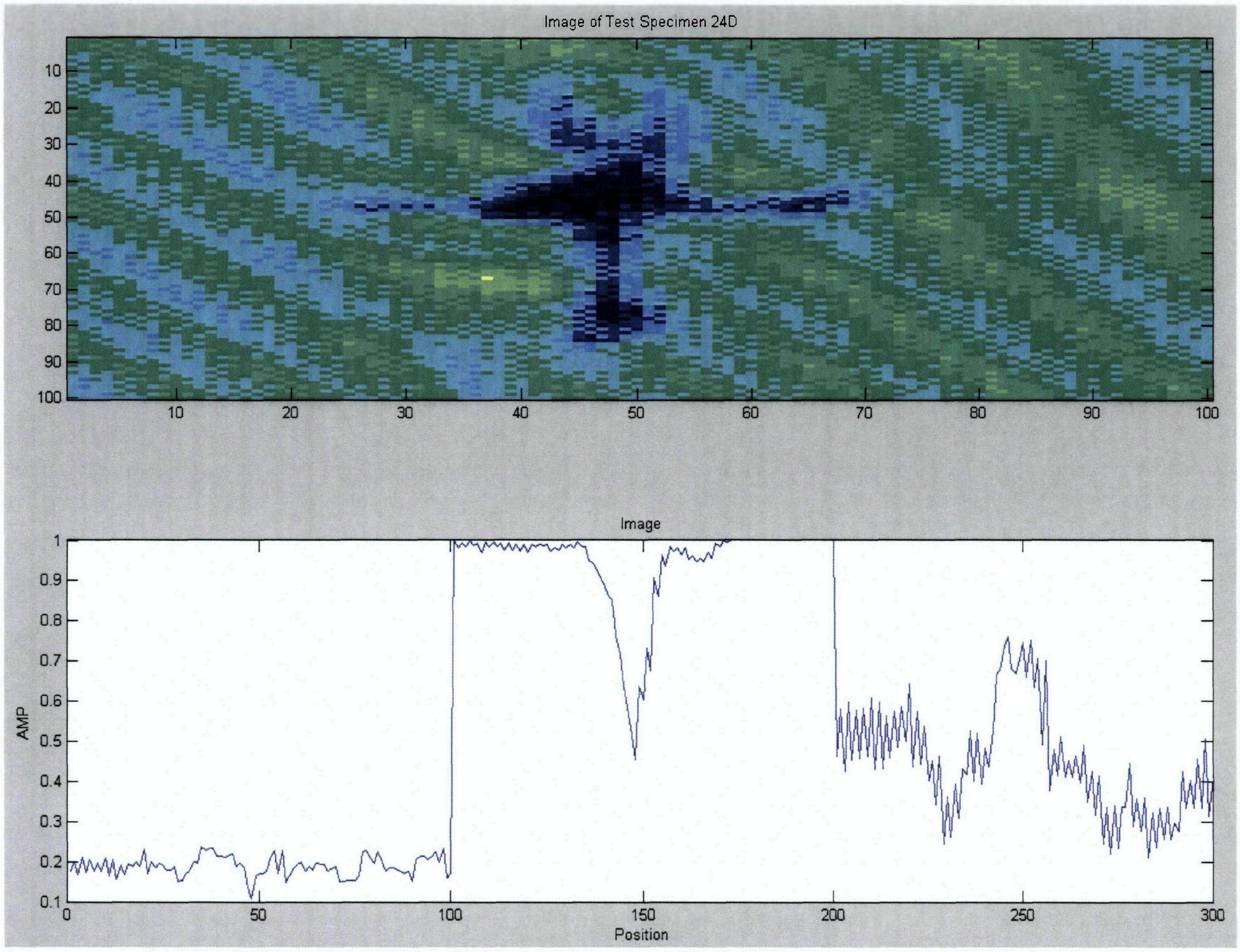
Test Specimen 24A



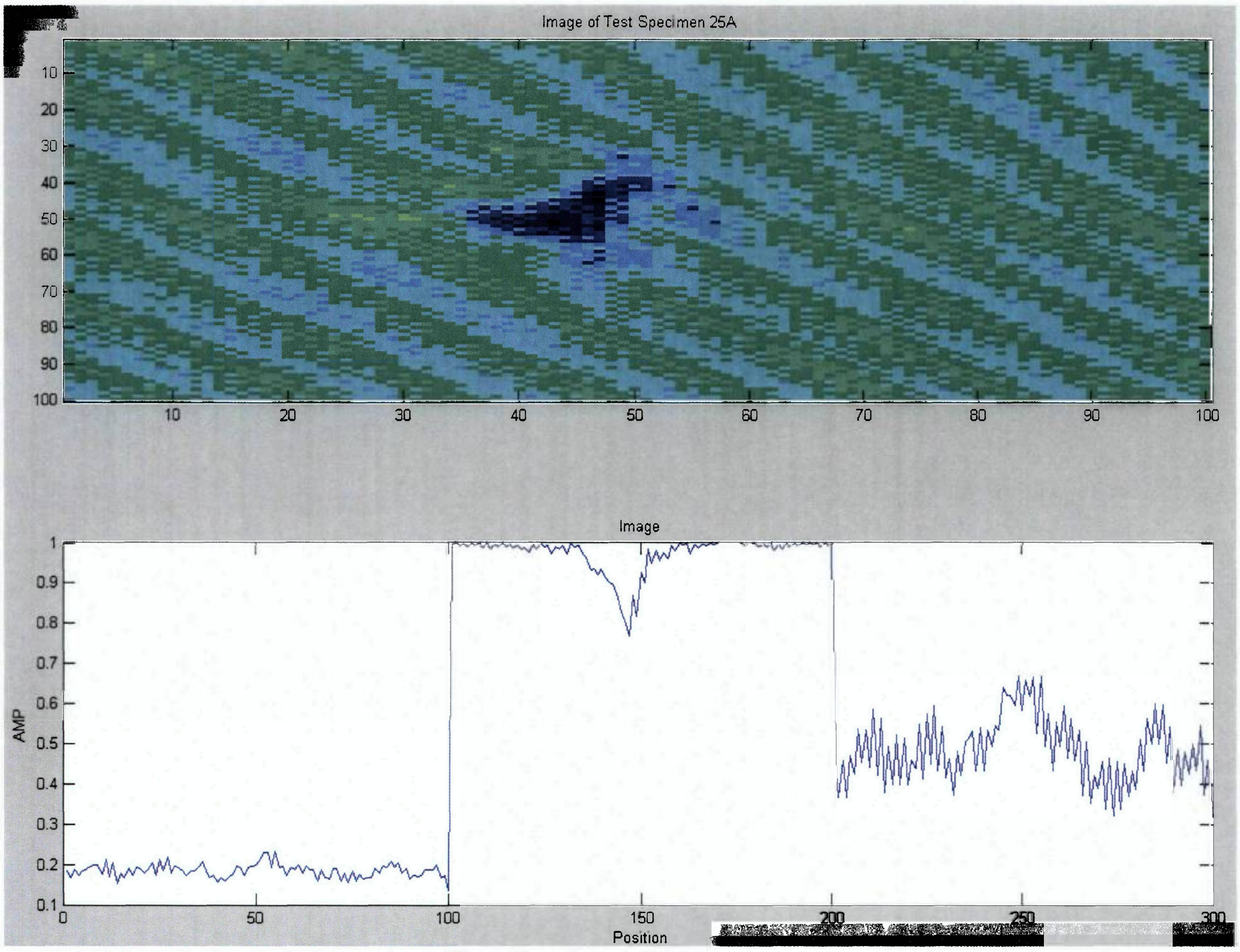
Test Specimen 24B



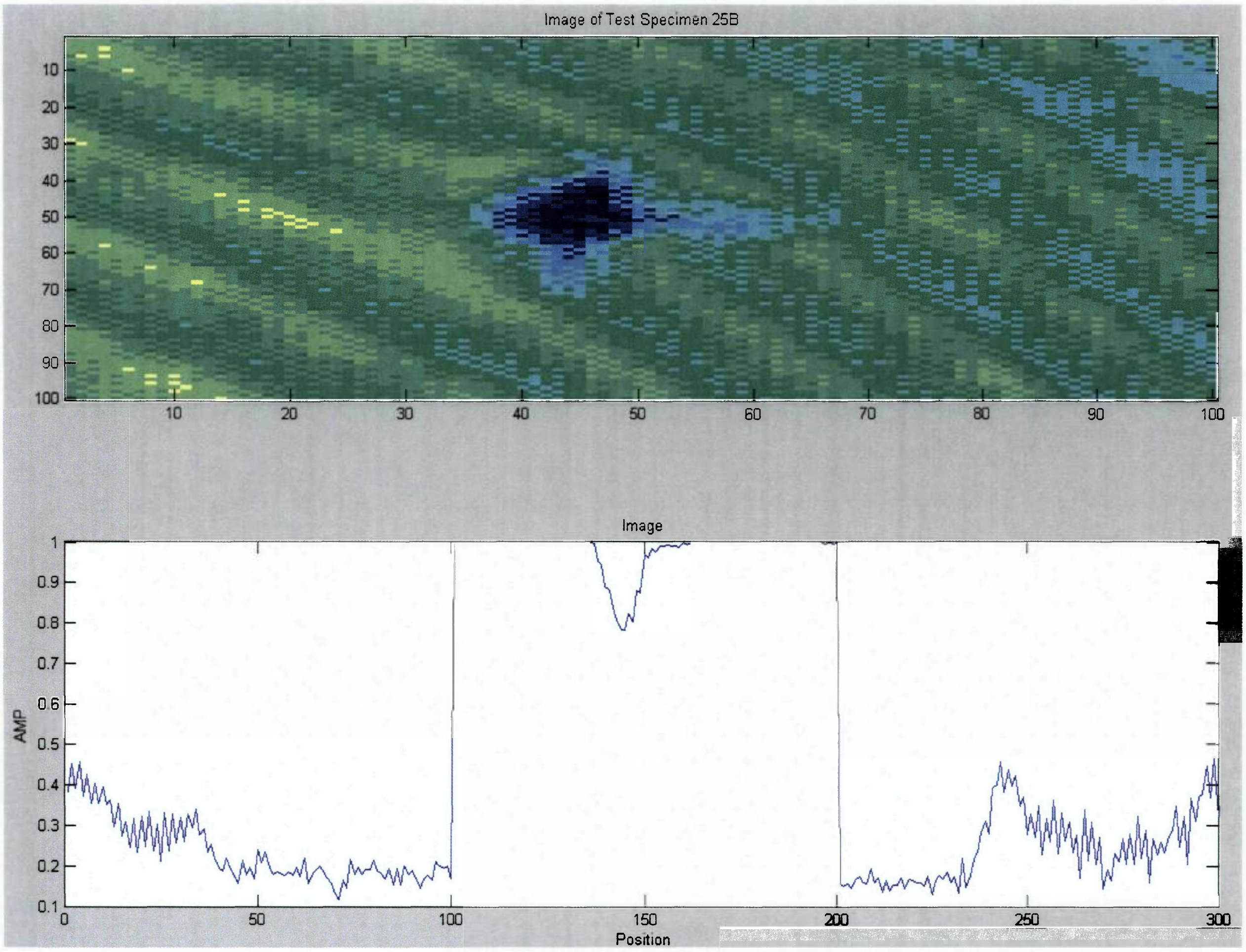
Test Specimen 24C



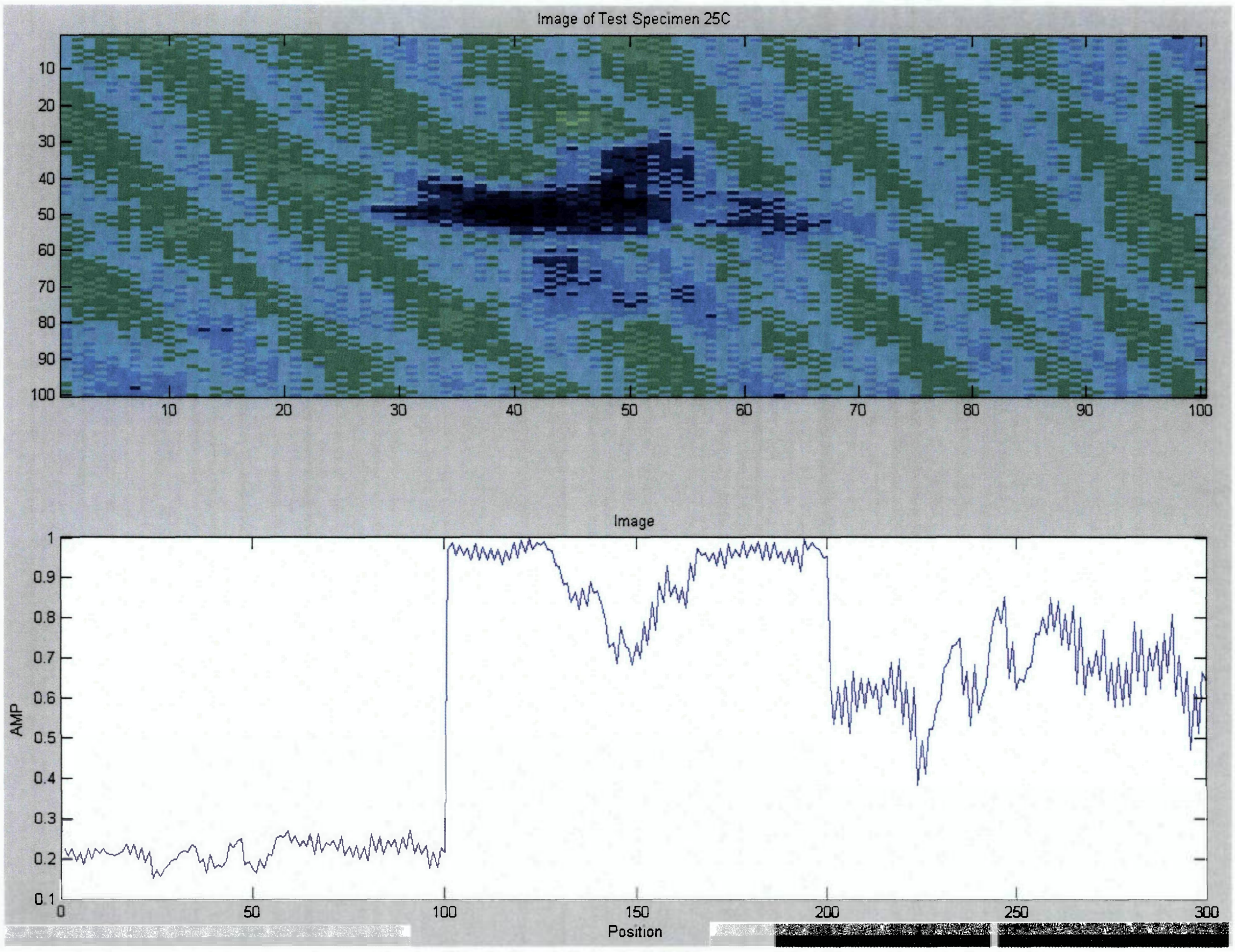
Test Specimen 24D



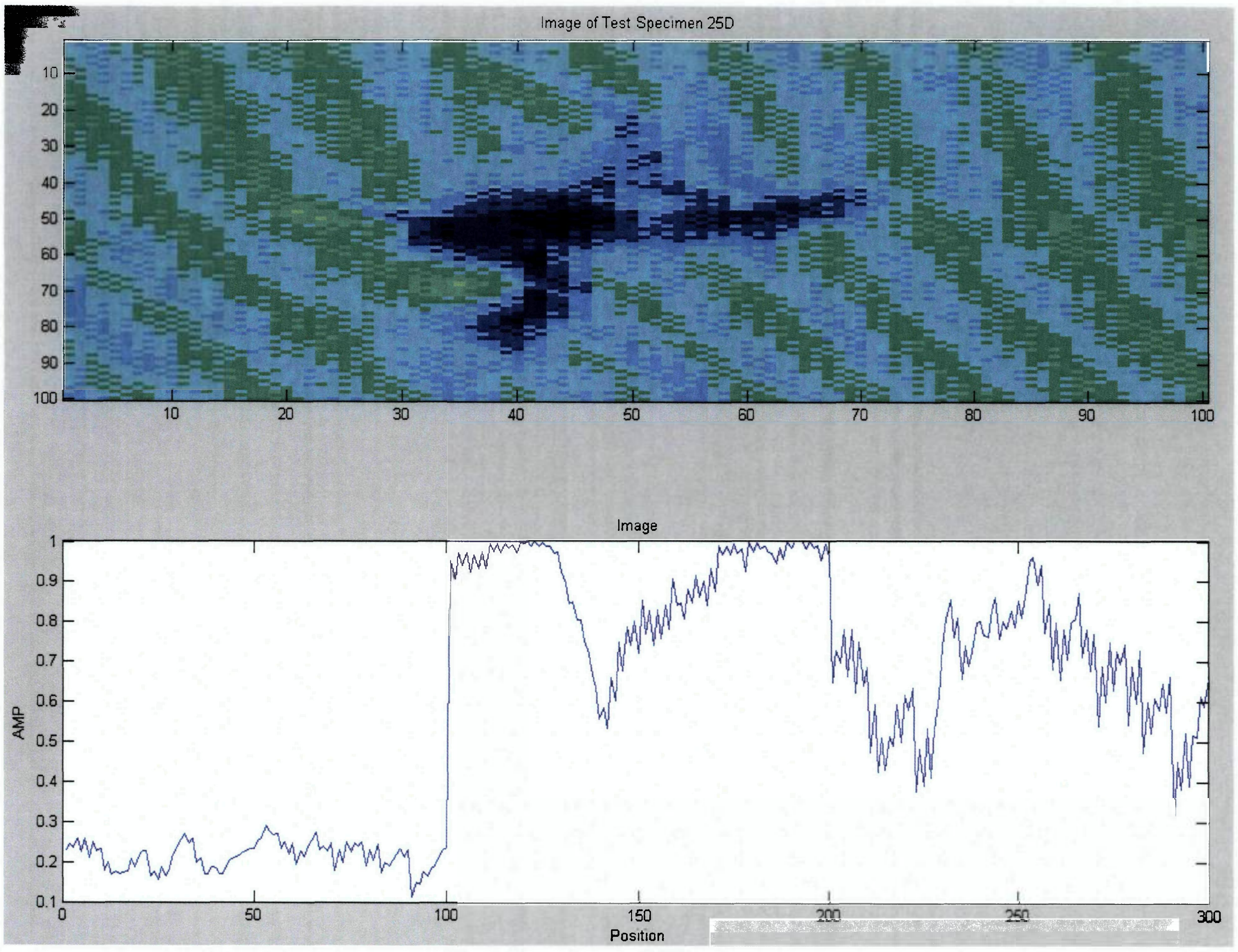
Test Specimen 25A



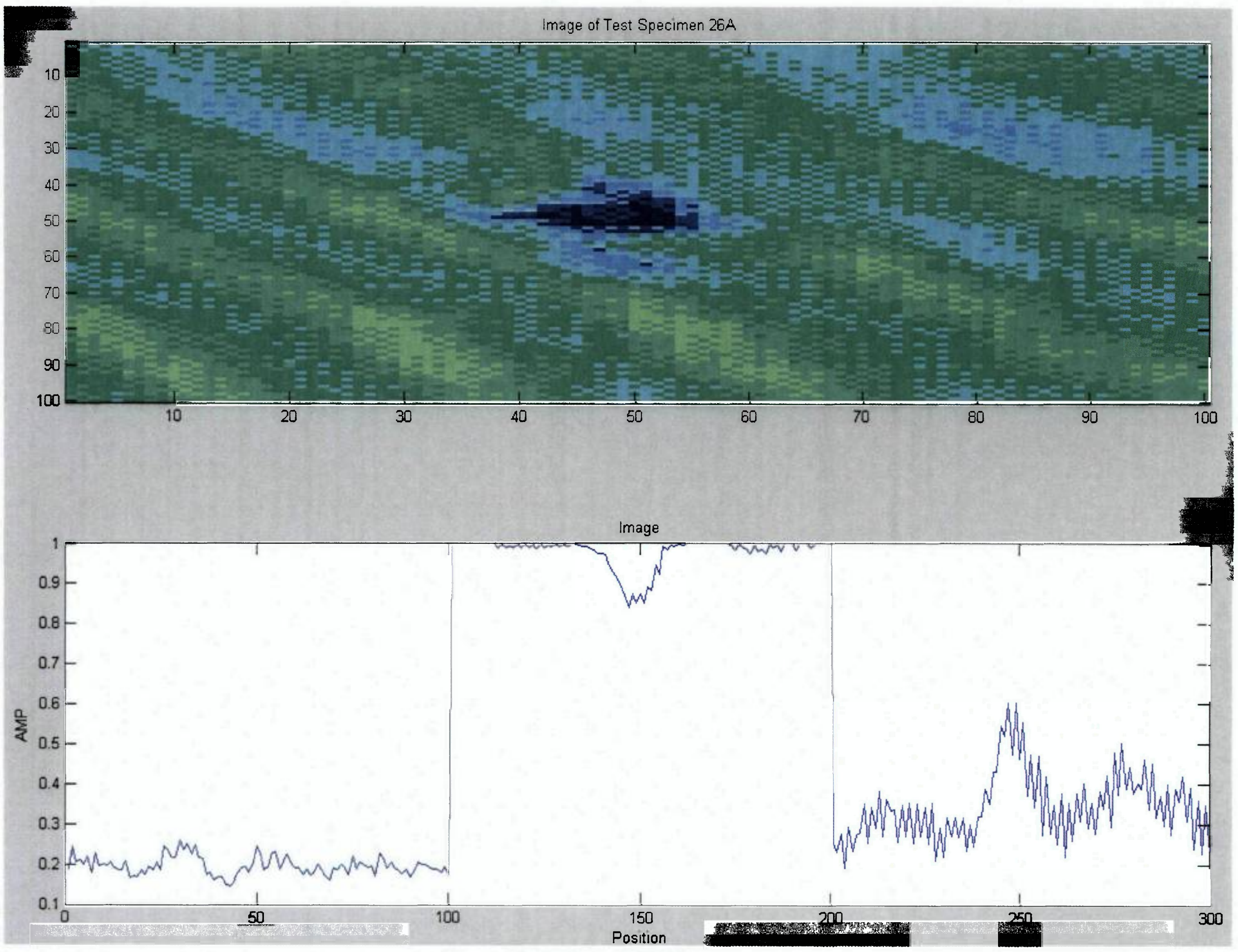
Test Specimen 25B



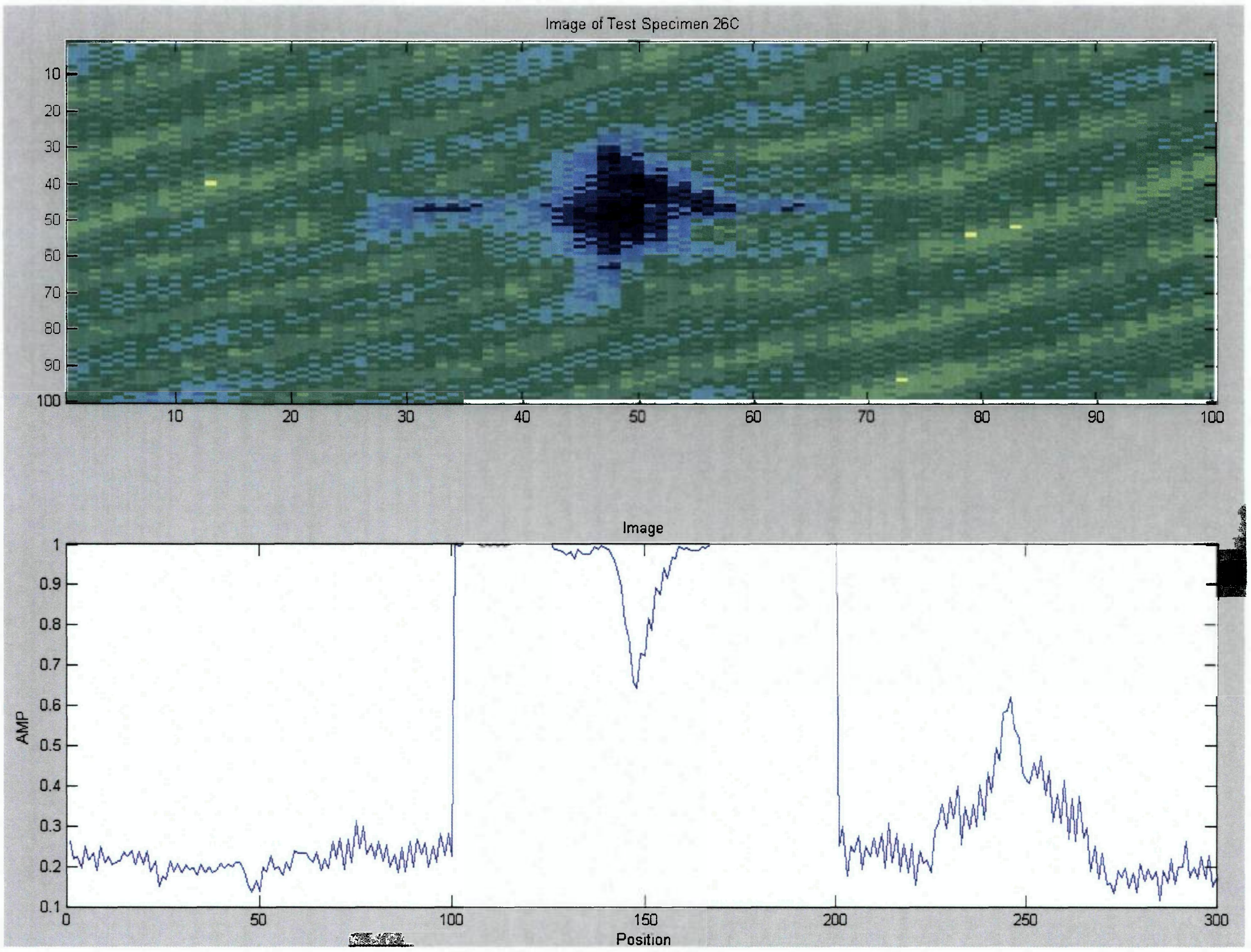
Test Specimen 25C



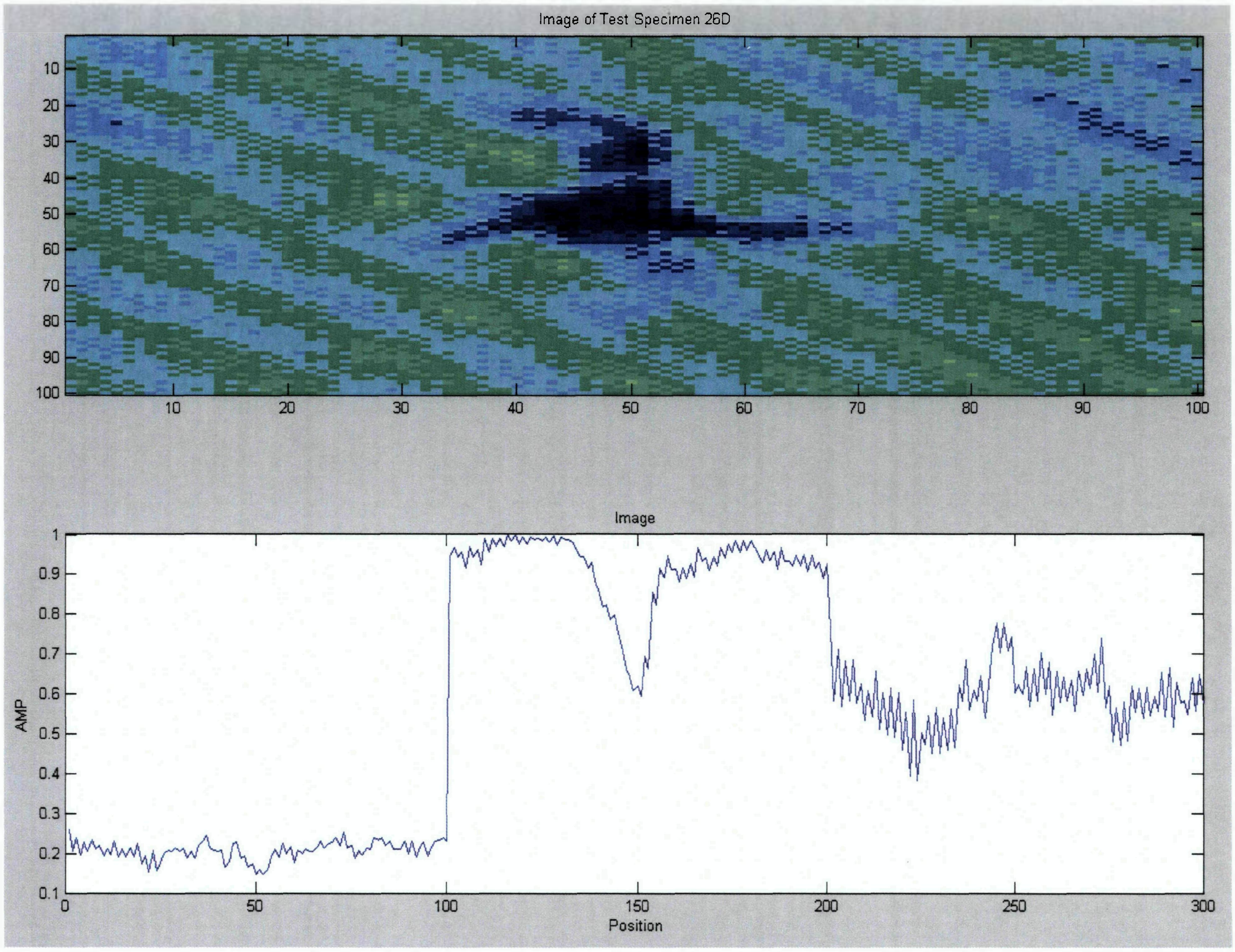
Test Specimen 25D



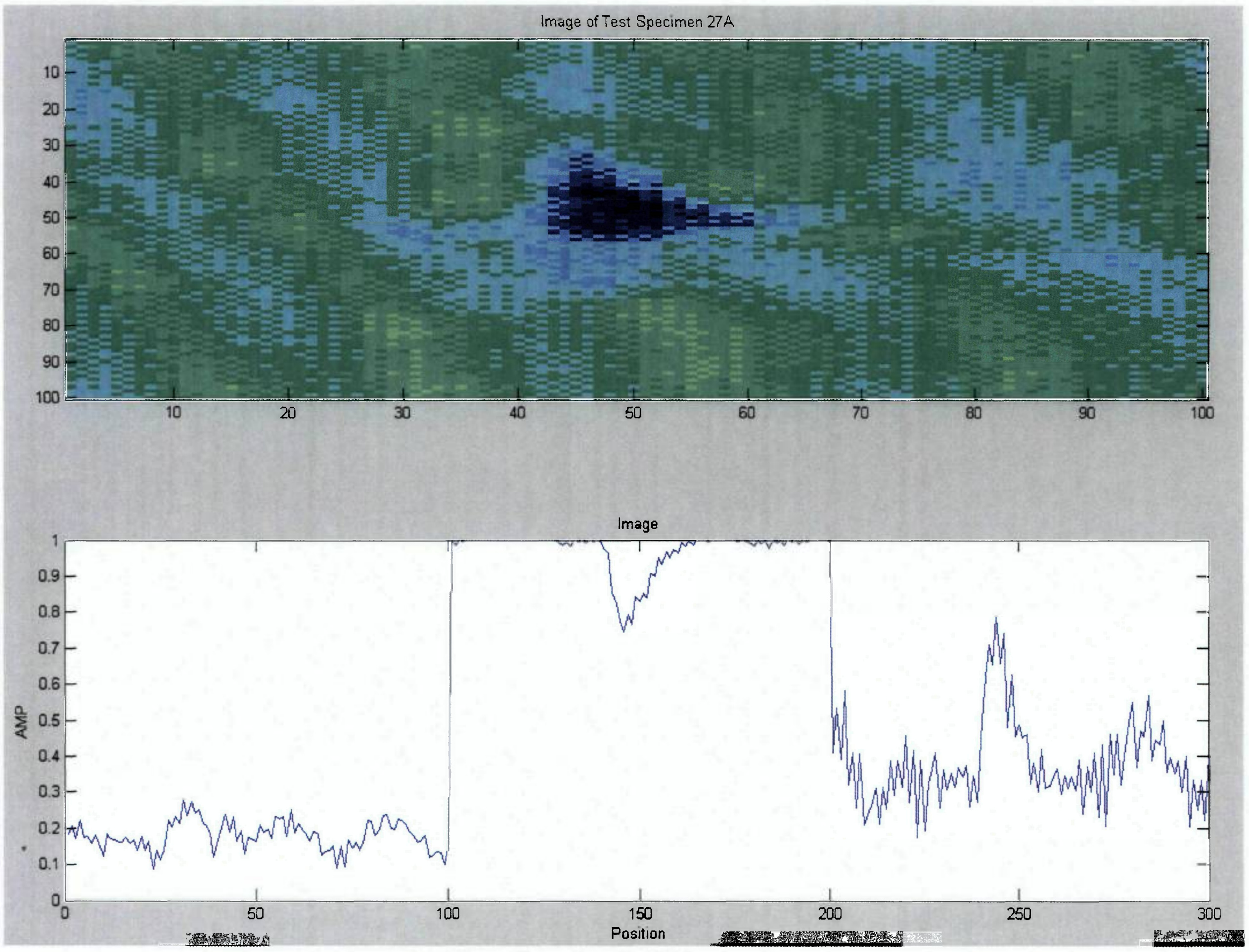
Test Specimen 26A



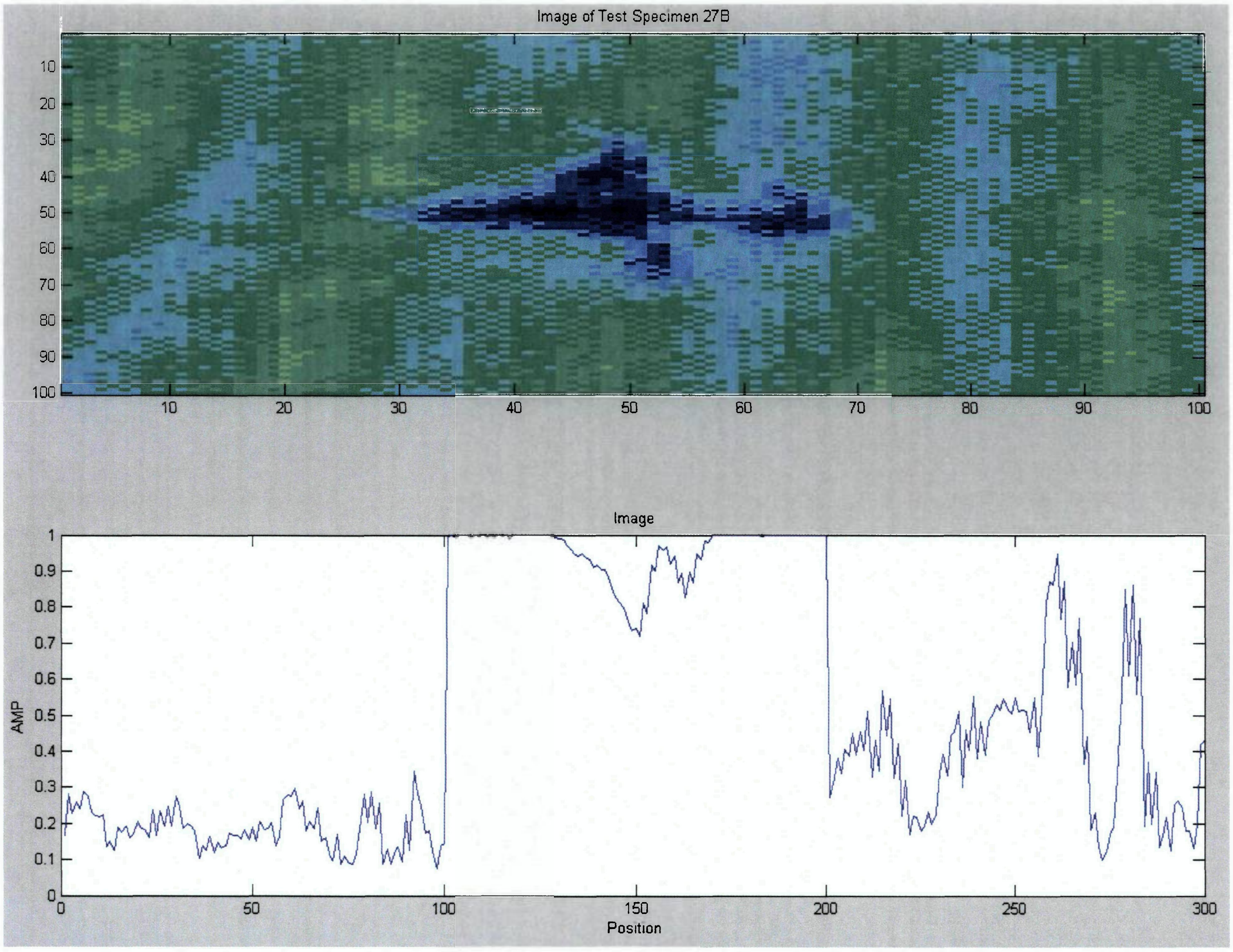
Test Specimen 26C



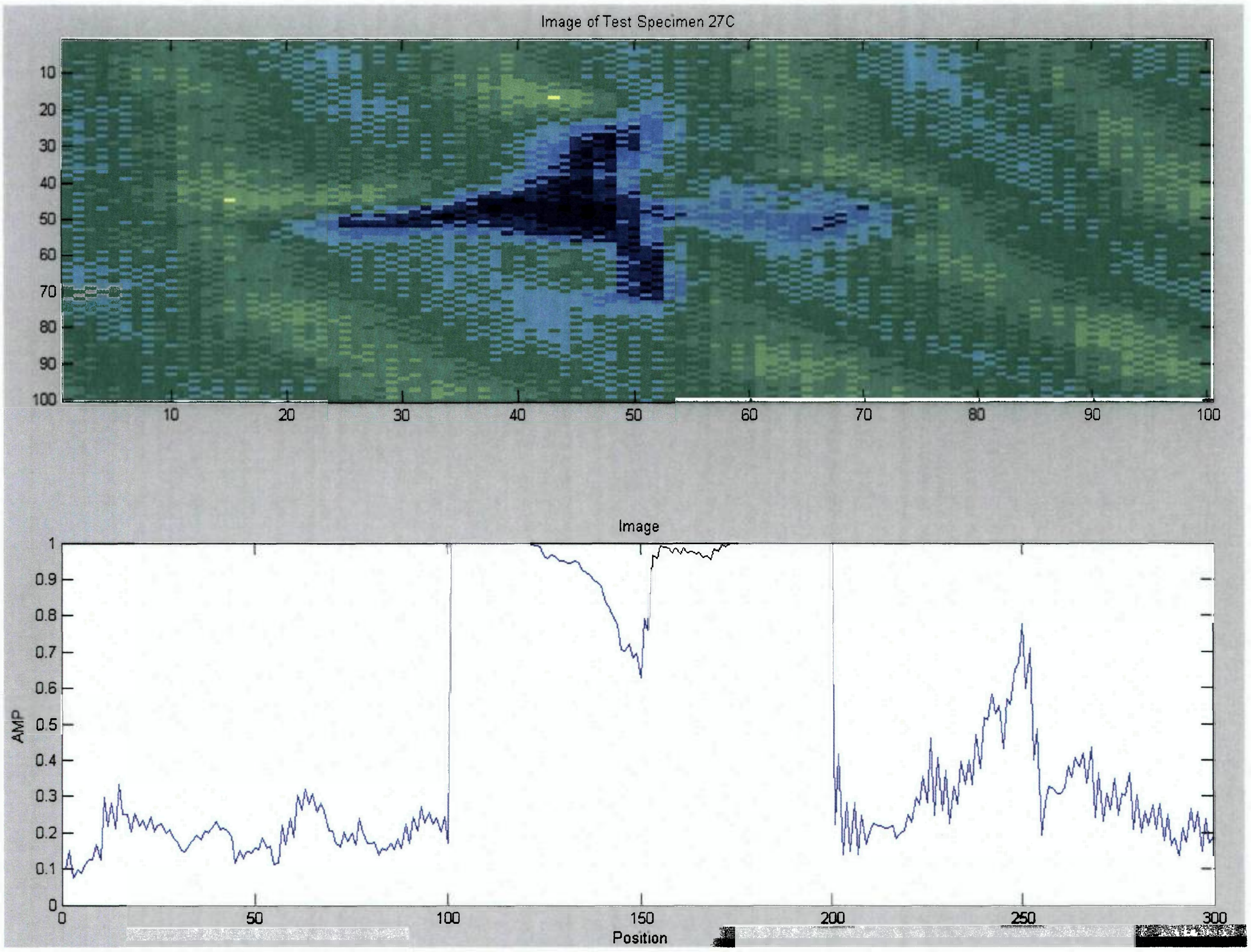
Test Specimen 26D



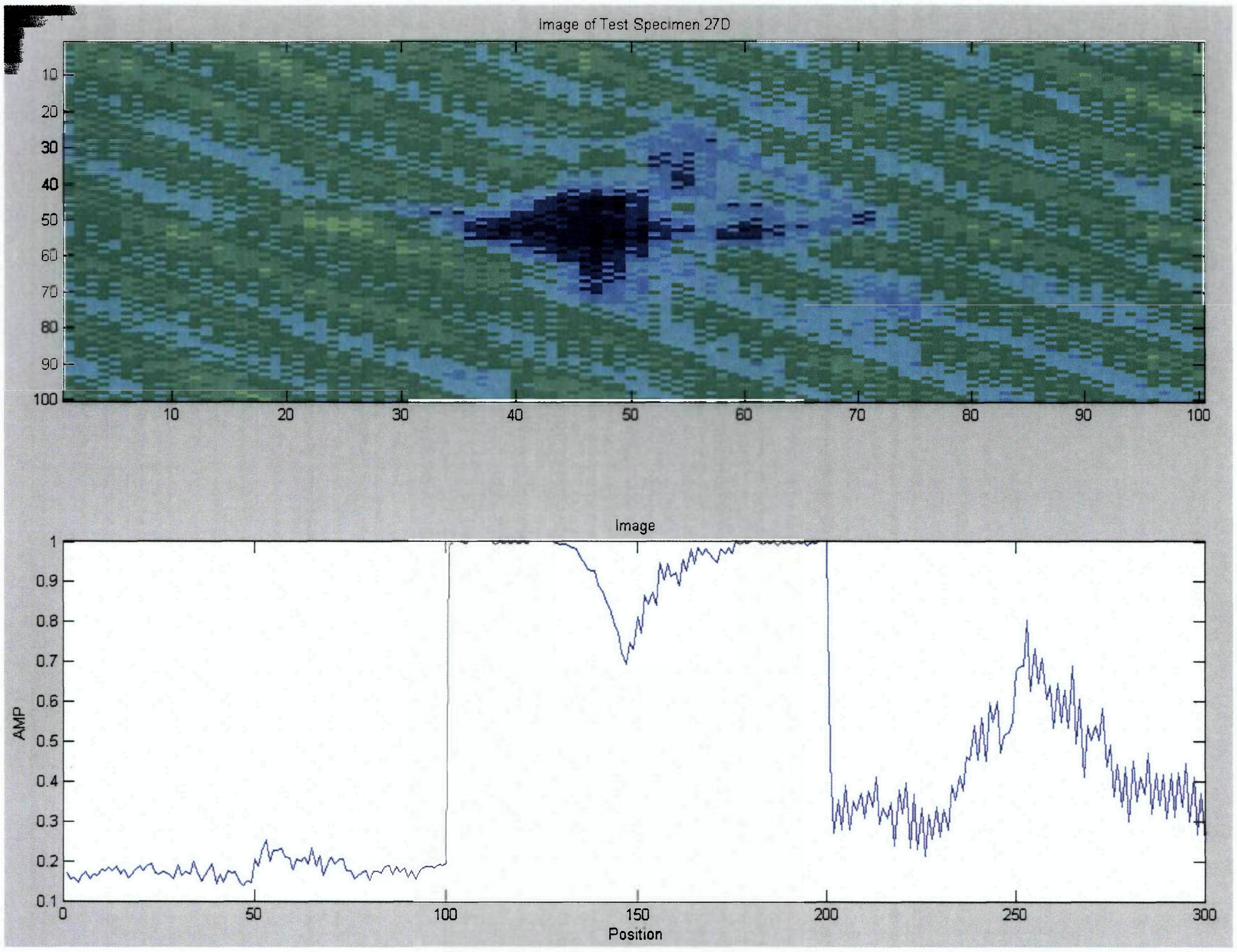
Test Specimen 27A



Test Specimen 27B



Test Specimen 27C



Test Specimen 27D

APPENDIX C: MATLAB CODE

MATLAB IMAGE PREPROCESSING PROGRAM

```
% To run this program, copy and the file name into the 'readimage'
command. To test an image, make it's in bitmap format

% Revision Date: April 15,2003
% Revised by: Christopher Hess

% Revision Date: November 19, 2010
% Revised by: Andrew Pacific

% Revision Date: December 1, 2010
% Revised by: Nikolas Geiselman

clear all
clc

image(imread('3D_FS_I', 'bmp'))

% This finds the image file and stores it into matrix 'map_b' as a
% colormap
[B, map_b] = imread('3D_FS_I', 'bmp');

%Only Read Green layer
B=B(:,:,2);

% This sums up the row or columns of the array and transpose them
cr = sum(B);

%To add or substract (%) to sums the rows or columns accordingly
ccrr= cr(1,:);

% Get the numbers of rows 'm' and columns 'n'
[m,n] = size(ccrr)

% Find the maximum value value in the matrix
g= max(ccrr)

% Normalizes the matrix
gg=ccrr/g;

% Plotting data
% Format
%       - first image is the image file
%       - second image is a scatter graph of the sum of the columns

subplot(2,1,1); image (B)
title ('Image of the Test Specimen')

subplot(2,1,2); plot(gg)
```

```
title ('Image')
xlabel('Position')
ylabel('AMP')

% Write to a text file
dlmwrite ('3D_FS_I.txt', gg,'\t');
```

MATLAB PROGRAM TO FILTER FAST FOURIER TRANSFORM

```
%This is a MATLAB Filter C-Scan images using Square filter of user
%defined size
clc
clear all

%Open the image file
img=imread('27D_FS_I', 'bmp');

%Only Read Green layer
img=img(:,:,2);

%Display Original Image
figure, imshow(img, [])
title('Original image of damaged coupon');

%Taking the fft2 (2-D fft)
freq_img=fft2(img);

%Shifting the fft image
Shifted=fftshift(freq_img);

%**********************************************************************%
%**************              Rectangular Filter            ************%
%**********************************************************************%

%Apply Rectangular filter to shifted FFT
[row,col] = size(Shifted);%Find size of image
centrow=row/2;            %Find center row number
centcol=col/2;            %Find center col number
fsize=20;                 %Filter size, counting pixels out from center

%Filtering image
for i=1:centrow-fsize-1, %Filter rows (top)
    Shifted(i,:)=0;
end

for i=centrow+fsize+1:row %Filter rows (Bottom)
    Shifted(i,:)=0;
end

for j=1:centcol-fsize-1, %Filter Cols (left)
    Shifted(:,j)=0;
end

for j=centcol+fsize+1:col,  %filter cols (right)
    Shifted(:,j)=0;
end

Logshift=log(Shifted); %take log of filtered image to increase
brightness
```

```
unshifted=ifftshift(Shifted); %Unshift the FFT before inversing the FFT
inverse_FFT=ifft2(unshifted); %Inverse the FFT to bring image back to
spacial

%*****************************************************************%
%**************              Chris Hess Code FFT         ************%
%*****************************************************************%

% This sums up the row or columns of the array and transpose them
cr = sum(inverse_FFT);

%To add or substract (%) to sums the rows or columns accordingly
ccrr= cr(1,:);

% Get the numbers of rows 'm' and columns 'n'
[m,n] = size(ccrr);

% Find the maximum value value in the matrix
g= max(ccrr);

% Normalizes the matrix
gg=ccrr/g;

%gg=abs(gg);
inverse_FFT=abs(inverse_FFT);

subplot(4,1,1); image (inverse_FFT)
title ('Image of Test Specimen 27D')

subplot(4,1,2); plot(gg)
title ('Image')
xlabel('Position')
ylabel('AMP')

% Write to a text file
dlmwrite ('27D_FS_I.txt', gg,'\t');

%*****************************************************************%
%**************       End of Chris Hess Code FFT         ************%
%*****************************************************************%

%*****************************************************************%
%**************             Chris Hess Code  IMAGE       ************%
%*****************************************************************%

% This sums up the row or columns of the array and transpose them
coro = sum(img);

%To add or substract (%) to sums the rows or columns accordingly
ccorro= coro(1,:);

% Get the numbers of rows 'm' and columns 'n'
[mo,no] = size(ccorro);
```

```
% Find the maximum value value in the matrix
go= max(ccorro);

% Normalizes the matrix
ggo=ccorro/go;

subplot(4,1,3); image (img)
title ('Image of Test Specimen 27D')

subplot(4,1,4); plot(ggo)
title ('Image')
xlabel('Position')
ylabel('AMP')

%*********************************************************************%
%**************         End of Chris Hess Code            ************%
%*********************************************************************%
```

TEST AND TRAINING FILE CREATION MATLAB CODE

```
%To assemble test and training files
clc
clear all

%-----------------------------------------------------------------------
%       Read in the Text files
%-----------------------------------------------------------------------

s2A=dlmread('2A_FS_I.txt','\t');
s2B=dlmread('2B_FS_I.txt','\t');
s2C=dlmread('2C_FS_I.txt','\t');
s3A=dlmread('3A_FS_I.txt','\t');
s3B=dlmread('3B_FS_I.txt','\t');
s3D=dlmread('3D_FS_I.txt','\t');
s24A=dlmread('24A_FS_I.txt','\t');
s24B=dlmread('24B_FS_I.txt','\t');
s24C=dlmread('24C_FS_I.txt','\t');
s24D=dlmread('24D_FS_I.txt','\t');
s25A=dlmread('25A_FS_I.txt','\t');
s25B=dlmread('25B_FS_I.txt','\t');
s25C=dlmread('25C_FS_I.txt','\t');
s25D=dlmread('25D_FS_I.txt','\t');
s26A=dlmread('26A_FS_I.txt','\t');
s26C=dlmread('26C_FS_I.txt','\t');
s26D=dlmread('26D_FS_I.txt','\t');
s27A=dlmread('27A_FS_I.txt','\t');
s27B=dlmread('27B_FS_I.txt','\t');
s27C=dlmread('27C_FS_I.txt','\t');
s27D=dlmread('27D_FS_I.txt','\t');

%-----------------------------------------------------------------------
%       Assign coupons index numbers for Training File
%-----------------------------------------------------------------------
index1=s27D;
index2=s25C;
index3=s2B;
index4=s2C;
index5=s24C;
index6=s26D;
index7=s27D;
index8=s3B;
index9=s24D;
index10=s24A;

index11=s2C;
index12=s25A;
index13=s27C;
index14=s24C;
index15=s25A;
index16=s27A;
index17=s25B;
index18=s27D;
index19=s3D;
```

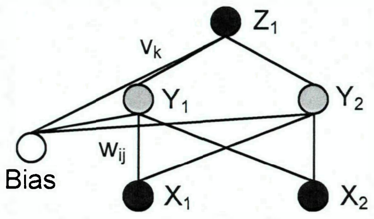
```
index20=s27A;

index21=s25B;
index22=s25C;
index23=s2B;
index24=s3B;
index25=s25C;
index26=s27C;
index27=s24D;
index28=s25A;
index29=s26D;
index30=s3D;

index31=s2C;
index32=s2B;
index33=s24A;
index34=s24D;
index35=s24A;
index36=s3B;
index37=s27C;
index38=s27A;
index39=s26D;
index40=s25B;

index41=s3D;
index42=s24C;
index43=s2A;
index44=s2A;
index45=s2A;

%Generate Training File
training=[index1;index2;index3;index4;index5;index6;index7;index8;
          index9;index10;index11;index12;index13;index14;index15;
          index16;index17;index18;index19;index20;index21;index22;
          index23;index24;index25;index26;index27;index28;index29;
          index30;index31;index32;index33;index34;index35;index36;
          index37;index38;index39;index40;index41;index42;index43;
          index44;index45];
dlmwrite ('Trainer.txt', training,'\t');
%----------------------------------------------------------------------
%       Assign coupons index numbers for Test File
%----------------------------------------------------------------------

testind1=s27B;
testind2=s25D;
testind3=s26C;
testind4=s3A;
testind5=s24B;
testind6=s26A;

%Generate Test file
test=[testind1;testind2;testind3;testind4;testind5;testind6];
dlmwrite ('Test.txt', test,'\t');
```

APPENDIX D: SAMPLE BACKPROPAGATION NEURAL NETWORK [DORFMAN (11)]

```
STAGE 1: Forward propagation of input vector
Step 1: Initialize weights to small random values
Step 2: Do while stopping condition is false
        Step 3: Compute input sum and apply activation function for each middle PE:
                $y_j = f(w_{ij} * x_i)$
        Step 4: Compute input sum and apply activation function for each output PE:
                $z_k = f(v_{ij} * y_i)$
STAGE 2: Back propagation of error
        Step 5: Compute error: $\delta_k = (t_k - z_k) * f'(w_{jk} * y_j)$
        Step 6: Compute delta weights: $\Delta v_{jk} = (\alpha)(\delta_k)(y_j) + \{Momentum * \Delta v_{ij}(old)\}$
        Step 7: Compute error contribution for each middle layer PE:
                $\delta_j = \delta_k * w_{jk} * f'(w_{ij} * x_i)$
        Step 8: Compute delta weights: $\Delta w_{ij} = (\alpha)(\delta_j)(x_i) + \{Momentum * \Delta w_{ij}(old)\}$
        Step 9: Update weights: $Q_{rs}(new) = Q_{rs}(old) + \Delta Q_{rs}$
Step 10: Test stopping condition
```

EXAMPLE

Consider a backpropagation network with 2 inputs and 2 hidden or middle layer PEs and a single output [9]. Find the new weights when the network is presented with an input vector $X_i = [0.0, 1.0]$ and target vector $Z_1 = 1.0$ using a learning coefficient of 0.25 and a sigmoid activation function.



The initial weights are given as:

$$w_{ij} = \left| \begin{array}{cc:c} 0.7 & -0.4 & 0.4 \\ -0.2 & 0.3 & 0.6 \end{array} \right|$$

$$v_k = \left| \begin{array}{cc:c} 0.5 & 0.1 & -0.3 \end{array} \right|$$

First compute the middle layer output using the relationship: $y_j = w_{ij}\, x_i$

$y_1 = w_{11}\, x_1 + w_{21}\, x_2 + w_{1B} = (0.7)(0) + (-0.2)(1.0) + 0.4 = 0.2$

$y_2 = w_{12}\, x_1 + w_{22}\, x_2 + w_{2B} = (-0.4)(0) + (0.3)(1.0) + 0.6 = 0.9$

$y_{1(OUT)} = f(y_1) = 1 / (1 + e^{-y1}) = 0.55$

$y_{2(OUT)} = f(y_2) = 1 / (1 + e^{-y2}) = 0.71$

Next, compute the network output and associated error using the relationship: $z_k = v_{ij}\, y_i$

$z_1 = v_{11}\, y_1 + v_{12}\, y_2 + v_{1B} = (0.5)(0.55) + (0.1)(0.71) - 0.3 = 0.046$

$z_{1(OUT)} = f(z_1) = 1 / (1 + e^{-z1}) = 0.51$

$\delta_k = (T_k - z_{k(OUT)})\, f\,'(z_{k(OUT)})$

$\delta_{z1} = (T_1 - z_{1(OUT)})\, f(z_1)(1 - f(z_1)) = (1.0 - 0.51)(0.51)(1 - 0.51) = 0.12$

The middle to output layer weights can now be updates using: $\Delta v_{jk} = \alpha\, \delta_k\, y_{j(OUT)}$

$\Delta v_{11} = \alpha\, \delta_{z1}\, y_{1(OUT)} = (0.25)(0.12)(0.55) = 0.017$

$\Delta v_{12} = \alpha\, \delta_{z1}\, y_{2(OUT)} = (0.25)(0.12)(0.71) = 0.021$

$\Delta v_{1B} = \alpha\, \delta_{z1}\, \text{Bias} = (0.25)(0.12)(1) = 0.030$

$v_k = |\, 0.517 \quad 0.121 \;\vdots\; -0.270 \,|$

The second stage begins by computing the middle layer error as: $\delta_j = \delta_k v_{kj} f'(y_{j(OUT)})$

$$\delta_{y1} = \delta_{z1}\, v_{11}\, f(y_1)(1 - f(y_1)) = (0.12)(0.5)(0.55)(1 - 0.55) = 0.015$$

$$\delta_{y2} = \delta_{z1}\, v_{12}\, f(y_2)(1 - f(y_2)) = (0.12)(0.1)(0.71)(1 - 0.71) = 0.0025$$

The input to middle layer weights are then updated using: $\Delta w_{ij} = \alpha\delta_i x_j$

$$\Delta w_{11} = \alpha\, \delta_{y1}\, x_1 = (0.25)(0.015)(0) = 0$$

$$\Delta w_{12} = \alpha\, \delta_{y1}\, x_2 = (0.25)(0.015)(1.0) = 0.0038$$

$$\Delta w_{21} = \alpha\, \delta_{y2}\, x_1 = (0.25)(0.0025)(0) = 0$$

$$\Delta w_{22} = \alpha\, \delta_{y2}\, x_2 = (0.25)(0.0025)(1.0) = 0.0006$$

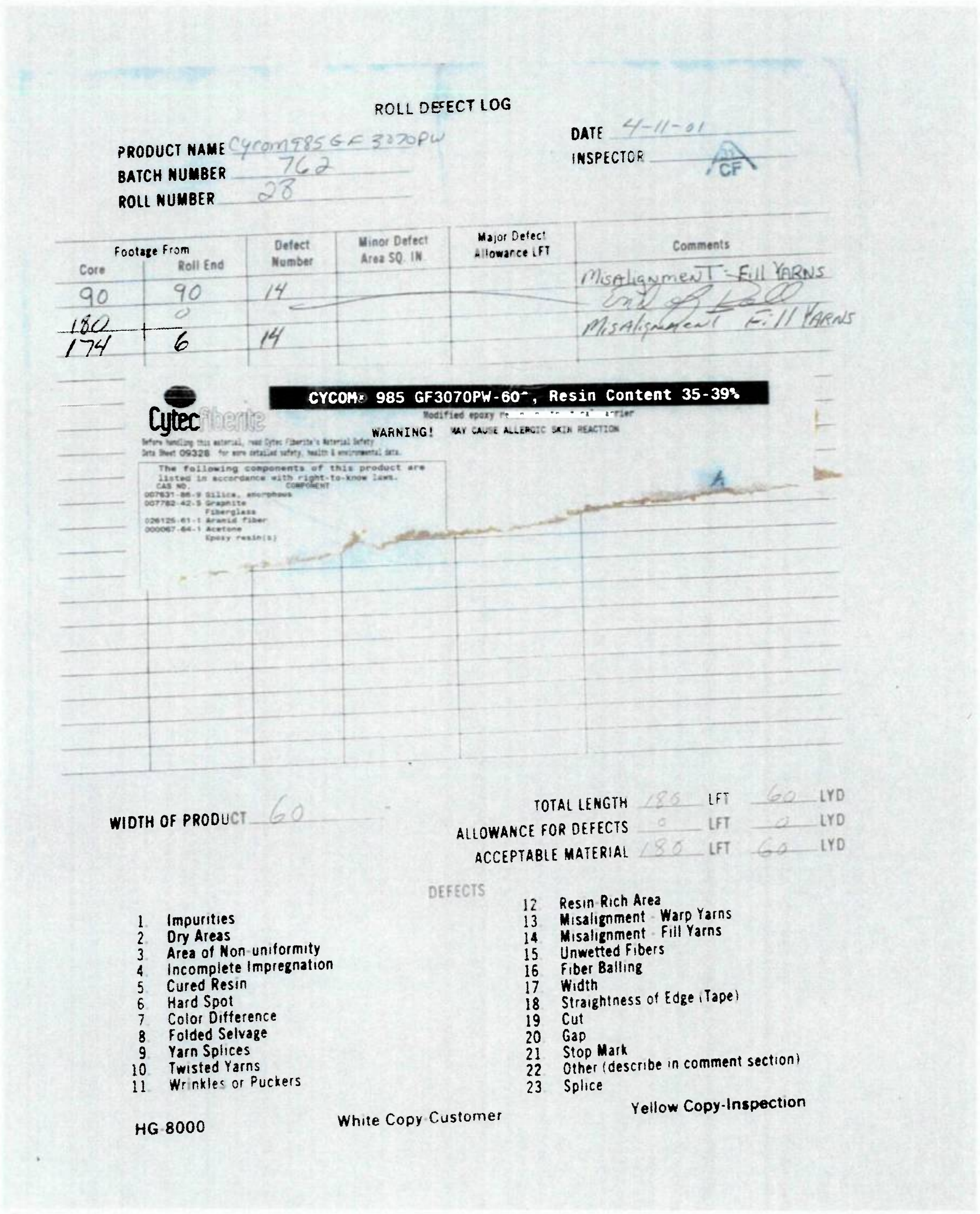
$$\Delta w_{1B} = \alpha\, \delta_{y1}\, \text{Bias} = (0.25)(0.015)(1.0) = 0.0038$$

$$\Delta w_{2B} = \alpha\, \delta_{y2}\, \text{Bias} = (0.25)(0.0025)(1.0) = 0.0006$$

Finally, the new updated weights are given as:

$$w_{ij(NEW)} = \left|\begin{array}{cc:c} 0.7 & -0.3962 & 0.4038 \\ -0.2 & 0.3006 & 0.6006 \end{array}\right|.$$

APPENDIX E: COMPOSITE INFORMATION SHEET



ROLL DEFECT LOG

PRODUCT NAME Cycom 985 GF 3070PW

BATCH NUMBER 762

ROLL NUMBER 28

DATE 4-11-01

INSPECTOR CF

Footage From		Defect Number	Minor Defect Area SQ. IN.	Major Defect Allowance LFT	Comments
Core	Roll End				
90	90	14			MisAlignment - Fill YARNS
	0				End of Roll
~~180~~ 174	6	14			MisAlignment Fill YARNS

CYCOM® 985 GF3070PW-60", Resin Content 35-39%

Modified epoxy [illegible] arrier

Cytec Fiberite

WARNING! MAY CAUSE ALLERGIC SKIN REACTION

Before handling this material, read Cytec Fiberite's Material Safety Data Sheet 09328 for more detailed safety, health & environmental data.

The following components of this product are listed in accordance with right-to-know laws.

CAS NO.	COMPONENT
007631-86-9	Silica, amorphous
007782-42-5	Graphite
	Fiberglass
026125-61-1	Aramid fiber
000067-64-1	Acetone
	Epoxy resin(s)

WIDTH OF PRODUCT 60

TOTAL LENGTH	180	LFT	60	LYD
ALLOWANCE FOR DEFECTS	0	LFT	0	LYD
ACCEPTABLE MATERIAL	180	LFT	60	LYD

DEFECTS

1	Impurities	12	Resin Rich Area
2	Dry Areas	13	Misalignment - Warp Yarns
3	Area of Non-uniformity	14	Misalignment - Fill Yarns
4	Incomplete Impregnation	15	Unwetted Fibers
5	Cured Resin	16	Fiber Balling
6	Hard Spot	17	Width
7	Color Difference	18	Straightness of Edge (Tape)
8	Folded Selvage	19	Cut
9	Yarn Splices	20	Gap
10	Twisted Yarns	21	Stop Mark
11	Wrinkles or Puckers	22	Other (describe in comment section)
		23	Splice

HG-8000 White Copy-Customer Yellow Copy-Inspection