DEVELOPMENT OF A USER INTERFACE IN MATLAB TO ASSIST
PRECISION ULTRASONIC THICKNESS MAPPING

JILLIAN J. WOOLRIDGE

Spring 2010

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Engineering Science
with honors in Engineering Science

Reviewed and approved* by the following:

Kevin L. Koudela
Senior Research Associate, Applied Research Laboratory
Thesis Supervisor

Christine B. Masters
Assistant Professor of Engineering Science
Honors Advisor

Judith A. Todd
P.B. Breneman Department Head
Department of Engineering Science

* Signatures are on file in the Schreyer Honors College and in the Engineering Science and Mechanics Office.
Abstract

Development of graphical user interfaces for very specific applications is more and more common as engineering becomes more diverse. Specifically, material development and processing is an ever-expanding field as new materials are created or explored for various applications. Composite materials are high up on the list of those being explored, and many old products are now being made out of composite materials due to their strength to weight ratio. For some applications of composite materials, the thickness needs to be very tightly controlled across large areas. To ensure that the components are within tolerances, measurements can be taken conveniently with ultrasonic inspection. However, interpretation of these data require better methods than manual evaluation of the time-of-flight signals recorded by the transducer set up. Inspecting so many points over the area of a component by hand reduces throughput and decreases efficiency in the production of the components. The acquisition of data is significantly less time-consuming than data interpretation.

Therefore, a method of quickly yet accurately reducing the data to determine thickness tolerance deviations in the part must be developed. This, in part, requires good signal processing to ensure accurate translation from a voltage measurement to a time measurement to a thickness measurement. The other part requires the development of a user interface that facilitates convenient and quick viewing of the thickness data. While signal processing requires only a programmer, a successful interface needs both the input of a programmer and end-user to determine the attributes that make a program successful. The research described in this paper explores the development of a graphical user interface in the MATLAB programming language that meets the needs of the user engineers while providing a convenient and practical way to view thickness variation along the surface of advanced composite material components.
Acknowledgements

I am grateful for the help and guidance of my direct supervisor Clark Moose, who tutored and explained the nuances of ultrasonic NDI and helped as go-between with the company end-users, to my thesis supervisor Dr. Kevin Koudela, for his sincere interest in my work and writing, and to Derek Lang for his large part in the signal-processing developed parallel with my work. I would also like to thank Dr. Christine Masters for her enthusiasm and flexibility in advising a person with varied interests and plans outside the usual academic path and Dr. Mark Traband, without whom I would never have known of this honors opportunity at ARL. Lastly, a general thank you to my friends and family who have stood by me in my endeavors and supported me academically, financially, and emotionally throughout my education.
List of Figures and Tables

Figure 1: Old GUI’s – Data Input, Signal Viewer, and Thickness Plotter ........................................ 12
Figure 2: Old GUI #3 - thickness viewing GUI .............................................................................. 12
Figure 3: Diagram of sample component with rounded edges ...................................................... 13
Figure 4: Rectangular mask matrices of 1’s and 0’s ................................................................. 14
Figure 5: Upper left and bottom right corner of the border mask matrix ..................................... 15
Figure 6: Mask matrices with rounded corners added ............................................................... 16
Figure 7: Border and window masking of an array of equal border and thickness value .......... 16
Figure 8: Table of values representing tolerances ..................................................................... 17
Figure 9: Flow of data for opening files and initial viewing of data ............................................ 18
Figure 10: Ultrasonic data collection hardware set-up .............................................................. 20
Figure 11: Working/Simple View ............................................................................................... 23
Figure 12: Advanced/Research View ......................................................................................... 23
Figure 13: Combination of Advanced and Simple GUI’s ............................................................ 28
Chapter 1: Introduction

1.1. Higher demands on materials

In recent years, engineers have been developing novel advanced materials with very specific design specifications to meet rising performance demands of products. New methods and processes are simultaneously developed to create the material compounds, but due to the relative infancy of both product and process, combined with the very strict design and manufacturing specifications, better and more accurate methods of quality control must be implemented. In addition, the methods must be easy to implement and interpret so as not to cause an increase in acquisition costs or time.

Such quality control methods require both new developments in hardware and software. Hardware must be readily available and simple to accurately assemble. Software must be able to handle the data and accurately represent it to the operator for accurate analysis. Software that is not capable of displaying the data may misrepresent the quality of the part to the operator and be detrimental to the quality assessment of the component as faulty hardware. Since parts must be able to function after quality control has been implemented, non-destructive evaluation (NDE) techniques are used, and the NDE technique chosen for this application is ultrasonic inspection.

1.2. Composite Material Monitoring

This section describes the monitoring of the advanced composite materials used during the research.

1.2.1 Description of Process

Currently the task is to monitor thickness of advanced composite materials which undergo many cure cycles. The goal is to monitor thickness between each cure so as to detect
gross variances before the component final fabrication step. Early detection of thickness deviations enables implementation of repair or alternate lay-up methods to prevent the component from becoming completely unusable. The composite components in question are rather thin and have an extremely large surface area (greater than 60 square feet). Due to their large size and small thickness they must remain on the mold plate throughout production. Therefore, measurements must be taken on the plate. Multiple ultrasonic scans must be made after each lay-up and cure process, which means that each scan must be processed and analyzed before the next cure is begun. The most logical thickness measurement therefore is via ultrasonic inspection.

1.2.2 Ultrasonic Thickness Data Interpretation

To scan and analyze the part, three key elements must be developed. These include 1) a hardware set-up that can accurately and consistently scan a part, 2) a method of processing the ultrasonic signals to give accurate readings of the thickness, and 3) a user interface that brings the data to the operator in a way that is convenient and easy to interpret, and aids in the production and development process. The first two are straightforward and, once developed, are improved or changed when a better or more accurate inspection method is found. However, the third can be considered a more objective element, since a robust user interface depends on the preference of the intended users to truly develop a program that they would consider useful. Currently, user interfaces created by companies such as NDT Automation, Inc. are able to display the data as 1) a peak amplitude from the threshold to the peak amplitude in a given time window, and 2) the related time of flight from various wave forms. However, these display attributes are not adequate for the purposes of this project, and this specific user interface is too time-consuming for individual point analysis.
1.2.3 Usability and Efficiency

Therefore, to complete this task, end-user surveys must be taken at different stages of the user interface development process to determine the appropriate attributes for the intended operators. A user survey would first demonstrate the capabilities of the baseline user interface, and then have the operators decide on the relevance of the user interface existing features, attributes that need to be added, and the ease of learning and operation of the interface. It is recommended to have the end-user input when programming since it is, in the end, these users who will have to utilize the tools and capabilities of the program. If the user interface is intended to increase accuracy of the part to its specifications without loss of production time, it is therefore necessary that the user interface not be a hindrance, costing more time to learn and navigate.

1.3. Thesis Objective

The objective of this thesis is to develop a graphical user interface (GUI) in MATLAB capable of accurately displaying the acquired thickness data of advanced composite materials during production to reduce acquisition costs and increase product accuracy.
Chapter 2: Review of Relevant Literature

2.1. User Interface Design

This section focuses on a brief explanation of usability and its role in user interface design.

2.1.1 Usability

“Form follows function” has been the basis of most requirement engineering since the dawn of time. When a solution was needed, a workable tool was developed, and only later refined as time allowed (Grudin, 1987). User interface development has been no different. Initial programs for interacting with computer intelligence required an intimate knowledge of the command systems and languages in order to display the desired output (Landsale, 1994). As the user base for computing systems expanded, more intuitive and user-friendly interfaces were required.

“Usability” is generally defined as the intuitiveness and ease of learning and navigating a user interface (Barnum, 2002). Programs designed to be used on a large scale typically consider usability a high priority, since failure to provide a usable interface reduces product usage and sales, and opens the door for competition. Take, as example, the most recent Microsoft Windows Vista operating system, which was disliked by such a large percent of users that it was common practice to “upgrade” to the previous system release, XP.

Smaller releases, such as application-specific programs, are much more likely to be “successful” while not entirely adapting to user needs or desires, though product quality and acquisition costs are not fully maximized (Ratchev, 2003). Engineering programs are a common example; such programs as MATLAB, Mathematica, AutoCAD, SolidWorks, and many more very specific programs are designed for an exact application and are not nearly as widely used as an
operating system or word processor. This “product-oriented” development results in a “stand-alone solution” and “assumes a predefined context of use” (Stary, 2002). The knowledge required to develop the program is also specialized, and coming up with an equivalent substitute for such a program is not a high priority. The time used to become familiar with a technical program is considered a marketable skill. The language used by each program is complex in order to achieve the complexity of the program's abilities.

2.1.2 User-Based and Requirements Programming

Every now and then, programmers are able to recognize that the best way to ensure a successful product is by consulting the end users. Analysis is used to acquire the requirements of end-users and used in the design, or structuring and implementation of software (Stary, 2002). Not only are the functions of the software a concern for programmers but also the ability of end-users to accomplish the tasks supported by the program.

Unfortunately, many times this is not a feasible option for one or more reasons (Grudin, 1987). Much of the time, economics are in the way. There is simply no time to consult the end user and incorporate their wishes and suggestions into an interface before a program is scheduled to be released. Sometimes, the management and the programmers have different goals in mind, and in general, the business management will end up having the final say in the content of a program.

Programming to user needs also fails when the programmers fail to acquire the necessary input from the end users (Saiedian, 2000). Asking the users questions that are too direct or too vague can lead to users either feeling pigeon-holed by what they think the programmers are asking, or not having any way to direct their thoughts. Lack of good dialogue is just as unproductive and unhelpful as no dialogue at all, since it does not end up incorporating
the user’s actual requirements into the program. Customers can resist a new program when they do not understand or appreciate the new approaches presented by a program.

Finding the right questions can be problematic, however, when one considers that both programmer and end user, while well-versed in their respective fields, know little about the others’ expertise (Saiedian, 2000). Software developers seldom have experience as a user in the application for which they are programming, and end-users would not be outsourcing if they could program to meet their own requirements. Terminology may not successfully bridge this gap, and therefore questions and answers may be interpreted differently by both parties (Stary, 2002). A common language that transcends the knowledge gap must be developed, and must not be too specialized towards one side or too common and risk becoming ambiguous.

Some aspects of a program are not always obvious requirements to some programmers, but could be useful in the long run. For example, colorblindness affects about 8% of men (and some women), and using color to differentiate between important aspects or objects within a program could cause difficulties for a small but substantial portion of users (Landsale, 1994).

Interestingly, it is not recommended to go above and beyond the call of duty and exceed the expectations of the customer (Saiedian, 2000). Completing projects ahead of schedule or delivering a program that far surpasses the customer’s needs and requirements can lead the customer to expect the same on future projects. Promising something that cannot be delivered is also a failure in the communication link, since programmers and customers must be realistic with the expectations and limitations of the project.

Documentation of the programming process, time permitting, is also a useful way of returning to a certain step and analyze why or how a problem was identified and resolved (Stary, 2002). Simple notes by an engineer, flowcharting parts or all of the data flow, cataloging
modification requests, etc., are all helpful when controlling and monitoring the development process for future reference.

2.1.3 Prototyping

A picture is worth a thousand words, and in this case, sometimes more. Discussing hypothetical or theoretical functionalities of a program with a customer can vastly improve the ultimate success of the program, but the ability to prototype and present a mini-version of the current abilities of a program is a more tangible way of evaluating its features and capabilities (Saiedian, 2000). A prototype is not a perfect version of a program, free of bugs and with all capabilities running smoothly; instead, it is a quick mock-up of the program, meant to work in a general case and be tested by end-users. It most likely will still have errors for exceptional cases, crash when memory runs out, or have some more advanced features offline at the time of testing. The idea is simply to get a working example into the hands of the end-users and allow them to “feel” their way through the program so that they can more accurately articulate what is acceptable and unacceptable with the current version. However, while prototyping is widely considered a helpful mode of user-requirements programming, dangers exist (Grudin, 1987). Two major issues to be considered are that, in general, a prototype will become the end program, and therefore code cannot be too hastily thrown together in such a way that a programmer cannot go back and quickly streamline it to remove bugs and enhance run-time. Secondly, adding “bells and whistles” may happen between a prototype and end product, since the infrastructure of the prototype is already in place and much more easily modified and updated than a program built from scratch. Users may not want or need these, or they may be pleased with the extra features and expect the same in the future (a problem mentioned previously).
A useful application of a prototype is a “think-aloud” run through, where one program-familiar user runs through the operation of features while explaining the process to a new user (Grudin, 1987). This can help to articulate potentially tricky or confusing areas of program use, because either the familiar user will have trouble describing a process or an unfamiliar user will not understand the instructions and request an explanation.

Testing a product on more than one user is generally recommended, whether the product is a computer program or a new pie recipe. When searching for programming bugs, the key number tends to be about 5 or 6 testers before the majority of problems are found (Spolsky, 2001). Therefore, testing the program on more users does not mean significantly fewer bugs. If the program has obvious errors it will only take a few users to find them, and there are only a few bugs to find so bringing in more users won’t find more bugs. Also, rigidly structured usability tests tend to bias results, as users are aware that they are being “tested” and consequently pay more attention to instructions or put more effort into understanding a program. Simple “hallway tests” can be useful for determining usability; using a colleague to answer a brief “what do you think of this lay-out” gives a quick, honest opinion without the need to monitor a whole usability test.

2.2. Using MATLAB

This section highlights the graphics user interface development capabilities of MATLAB.

2.2.1 Graphic User Interfaces

MATLAB, a computing language from the MathWorks company, is “a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran.” (The MathWorks, Inc., 2009). Used to process data in the form of matrices, plot functions or data, and develop
and implement algorithms, among other applications, it is highly useful for the manipulation and processing of large volumes of data. It can also interface with other well known programs, such as Microsoft Excel. For such reasons, it was natural to use MATLAB to process the large volumes of ultrasonic data that would be collected for each composite material component evaluated as part of this study.

In addition to its computational abilities, MATLAB offers a GUI programming interface, called GUIDE (DeMoyer, 1999). The programmer lays out various implements useful for the interpretation of data, such as axes, blank edit boxes, buttons, drop-down menus, etc, in the GUIDE interface, and MATLAB takes care of making sure each element becomes part of the final GUI. (The MathWorks, Inc., 2009). Once the outer appearance has been laid out, the MATLAB user programs the functions that are called when each element is chosen. For example, a button push does nothing until the programmer opens the GUI's accompanying function (*.m) file, and uses the MATLAB language to program the button to plot the function from the edit box into the axis. In such an interface, the interpretation and viewing of data requires much less knowledge of the MATLAB language, once the initial programming is complete, since it can act and even become its own independent user interface.

MATLAB GUI’s are able to perform much of the functions of the original program, while enhancing the usability for users less able to produce the same data processing capabilities through just using MATLAB (DeMoyer, 1999). In addition, programming all functions to happen at a “simple” button click saves experienced users the hassle of inputting each command. However, it does require a user familiar with the MATLAB coding language to program the initial functions into the GUI. The MathWorks site, as well as numerous online forums, provide function coding help, tips, and pointers. Sometimes the combination of two or more of the
MATLAB function calls produces the desired outcome, but the MathWorks site itself was, in general, theoretical about what each individual function did and one had to be familiar with the terminology in order to find functions that would operate together.
Chapter 3: Methodology

In this chapter the computational processes and analysis methods behind the implementation of the GUI developed for ultrasonic thickness inspection data interpretation are discussed.

3.1. Current Methods

Since a workable interface and algorithm existed for the translation of the thickness data into something marginally graphically viewable, the objective of this research is more to interpret what the interface is showing, and how to make a more intuitive and usable GUI. The initial method consists of two interfaces: one to convert files to another data form, and a second to plot these data files. Unless the user is familiar with the file structure and order in which to run the files (and indeed, which files and what information to enter into which fields), the program is essentially useless, because it does not give understandable feedback on user input errors. In addition, no guide existed to point the way through the interface. Hence the first task in this research is to create a single GUI that does the job from start to finish, guides the user through the data input, and does it quickly, effectively, and in an easily interpretable manner. Figures 1 and 2 show examples of the original GUI’s used for data processing and viewing.
3.2. Development of New Methods

This section describes the methods used to develop a new GUI.

3.2.1 Analysis and adaption of previous methods

The easiest way to begin this process of designing the user interface was to simply put the required input and output fields in the most logical and aesthetically pleasing arrangement. As the user interface was tested, certain arrangements were considered more awkward or unlikeable and were changed. To further help this development process, general surveys were given to the end-user, asking questions about current data input options, potential additions, and any remarks or required features that they would like to see implemented. The new user interface followed the old data input-style, since no indication of a radical change in the data type that would be processed was indicated. At first, some of the inputs that had been manual
were programmed into the hidden data processing, but once the program was more advanced, they were reincorporated as manual input fields to ensure correct data reading.

3.2.2 Matrix Logic

More behind-the-scenes programming used matrices to represent the component data in two-dimensional format. Since some of the composite components included rounded corners and two regions of different thicknesses that needed to be monitored independently, the data from a single ultrasonic scan had to be reliably divided into the two different sections and analyzed separately. Figure 3 shows a simple example of a potential part shape. The dimensions of the part, the number of data points averaged, and the geometry of the two inspection regions not only had to be input by the user but then used by the program to calculate the correct separation of the data so that the correct output data was displayed.

![Figure 3: Diagram of sample component with rounded edges](image)

The number of data points collected during scanning was given as one per square inch of the part, but before division of the two regions, the signal processing had been set up to average data points based on user specifications. Therefore, the scaling had to be taken into account. During the signal processing, a specified number of data points had been averaged, and so the data stored in the program no longer represented one point per square inch; rather, one data point represented, for example, 16 square inches, if the number of points to be averaged
had been chosen as 4. Thus, though dimensions of the part had been given by the user, the processing had to take into account that the data points were no longer at one per square inch. That is, at this point, the data existed as a matrix of thicknesses with dimensions of the length or width divided by the number points averaged.

The dimensions of the entire component, including the radius of curvature of the corners and the widths of the outer region and the space between the outer and inner region (called the “transition”), were specified by the user. The geometric data were then converted, using the number of points averaged, into a “width in data points.” Two empty matrices were then created, one for the outer region and one for the inner, to the exact dimensions of the thickness matrix. These “mask” matrices, which “masked” out points, used the “width in data points” of the different regions to create rectangles, one with a border of “1” and “0” elsewhere, and one with an inner region of “1”, “0” elsewhere. Figures 4a and 4b show two simple rectangular mask matrices of dimensions 20 by 20, border 2 and transition width 2.

![Figure 4: Rectangular mask matrices of 1's and 0's](image)

Once these rectangular regions were created, a new temporary square matrix with sides of the radius of curvature, scaled to the number of samples averaged, was initiated. The
geometry of the part was used to determine if a point was within a certain location of a corner, i.e. if a point was within the radius of curvature and along the border width in points. If it was too far or too close to the chosen corner of the matrix, it was left as “0”, but if the point existed along the radius of curvature and was contained in the border region, the value was changed to “1.” The same was done for a square matrix for the inner region, where if the point existed along the radius of curvature from a given corner and was within the border and transition widths, it was changed to a value of “1.” Each rectangular matrix had its respective “corner matrix” replace each of its corners, rotating the corner matrices to match the geometry of the part. The border or inner region of “1’s” that had existed as rectangles now has smooth, rounded edges. Figure 5 shows the small rounded corner matrices. The other corners were made by rotating the first 90 degrees, and then these matrices replaced the corners of the rectangular border matrix in figure 4a. Similar corner matrices were made for the window matrix. Figures 6a and 6b show the entire mask matrices with the corners rounded.

Figure 5: Upper left and bottom right corner of the border mask matrix
Next the matrix of thicknesses was multiplied point-by-point to each created mask matrix; that is, the values of 1 and 0 were multiplied by the thickness value corresponding to the identical position in the thickness value array. Instead of “1,” the values of each mask matrix are now the value at that given point, but the values of 0 remained 0. A border and inner region of thicknesses now existed separately. These two matrices were then used to evaluate the statistics of the region, such as using these points to find the mean and standard deviation of a single region. Figure 7 shows a thickness matrix of value “22” for all regions, and the result of the point-by-point multiplication of this matrix by both the border and window mask matrices.

![Figure 6: Mask matrices with rounded corners added](image)

![Figure 7: border and window masking of an array of equal border and thickness value](image)
Once these statistics were obtained, the “target” thickness and tolerance for each region was then compared to each data point. “0” points were ignored, but points that were outside of tolerance were given a “1” for too thin and “5” for too thick. Points that were between 75% and 100% of this tolerance were given a “2” and “4” for almost too thin or almost too thick, respectively, and points that were within 75% tolerance in either direction were given a “3.” Now the two matrices of values 0-5, one representing the border and one the inner region, were added together. Since where one existed the other didn’t, adding the value of a thickness range designation was not an issue that had to be considered. Once again, one matrix represented the entire part, but now it specified the range of thicknesses for the two regions in a format (1-5) that is easy to interpret. Figure 8 below shows the guidelines for the values shown by the final output matrix.

<table>
<thead>
<tr>
<th>Value</th>
<th>Thickness Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not included in calculations</td>
</tr>
<tr>
<td>1</td>
<td>Too thin, outside of tolerance limit</td>
</tr>
<tr>
<td>2</td>
<td>75% to 100% of tolerance limit thinner than the target value</td>
</tr>
<tr>
<td>3</td>
<td>Within 75% of tolerance limit from target value, either thick or thin</td>
</tr>
<tr>
<td>4</td>
<td>75% to 100% of tolerance limit thicker than the target value</td>
</tr>
<tr>
<td>5</td>
<td>Too thick, outside of tolerance limit</td>
</tr>
</tbody>
</table>

Figure 8: Table of values representing tolerances

3.2.3 Flowcharting

Flowcharting was useful for the organization and forward planning of the GUI, or to tell where the development stood at any given point. Basic data flow charts were made for various operations of the program. Figure 9 shows the flow of data for the entire GUI, to have a basic idea of where the data was transferred from or to.
3.3. Signal Processing

The signal processing used in the MATLAB code, developed by Derek Lang from the Applied Research Lab at Penn State, was included in the program to convert the time-of-flight ultrasonic A-scan data to a thickness measurement. The data was received in the form of *.csv files which consisted of each line of data and the corresponding A-Scan values for each data point along the given scan line. The files were opened, line by line, the signal processing code was used to operate on each file and find the thickness at each point along the surface of the component. The thickness was calculated using an effective velocity for the given material, which was found by using calibration blocks to measure the absolute thickness and relate that to the corresponding ultrasound data. The data for each line was then stored in a thickness matrix, with each row of the matrix corresponding to a row of scan data from the part.
Chapter 4: Procedure

This chapter focuses on the hardware and software implementation that was utilized during the development of the user interface.

4.1. Materials for research

For the basis of this thesis, the requirements in terms of materials were very minimal. Since the work and software development is entirely computer-based, the program was developed with use of a computer, a copy of MATLAB, and reference tools.

4.2. Ultrasonic NDI Data Collection

This section describes the methods of data acquisition.

4.2.1 Data Collection Set-Up

The data collection hardware set up, designed by Clark Moose of the Penn State Applied research lab, is configured to accommodate a large, flat, thin composite part lying on a flat mold plate. The part, which would undergo multiple cure cycles, could not be removed from the plate for a couple reasons. The part must remain in the same position for correct alignment of subsequent layers, and the precision required for the manufacturing made it most logical to refrain from removing the part from the mold. Secondly, the length to thickness ratio was so large that the part would bend and act similar to a sheet of paper after the first cure. Removing the part from the mold plate to manually inspect thickness would be impractical and counter-productive, as well as possibly damaging the component. Therefore, the design of the hardware set up took this into consideration. Commercially available transducers, bridges, motors and tracking devices were used to reduce the capital investment for the new inspection method. The final hardware set up, shown in figure 10, includes two magnetized tracks with a “bridge”
suspended between the tracks, which carries the transducer. Precision motors drive the bridge across the length of the part, and the transducer head scans across the width of the part by traversing the length of the bridge. Figure 10 shows the actual hardware set-up.

![Figure 10: Ultrasonic data collection hardware set-up](image)

In addition to the precision motors and programming implemented to track their precise location, laser tracking was used to measure the absolute distance of the transducer head from the laser point and provide a second set of complementary data. This set of data provides the absolute distance above the mold plate, whereas the data generated by the ultrasonic transducer is translated into a thickness between two points.

### 4.2.2 Scan process

Once the bridge-track with the precision motors as well as the laser tracking was set up, a data acquisition program was used to gather the data. Scans started at a chosen “zero” point, usually about an inch into each corner of a part. Scans were set to take one data point per inch, and told to run a certain x distance across and y distance along the part. Once the scan was started, the transducer head traveled along the bridge for the given distance, taking data each inch. Once one pass was made, the magnetic feet advanced one inch in the y direction, and the
head would then scan along the next x width. The data acquisition program then created the
*.csv files that were used by the MATLAB GUI.

4.3. GUI Program developments

This section describes the steps taken to develop the MATLAB GUI.

4.3.1 Analysis of GUI’s In Place at Beginning of Project

To find a basis on which to begin the development of the GUI, the existing inspection
data reduction method was reviewed. At the time, the data were being taken from the
ultrasound binary data and translated into *.csv files, one for each scan-line lengthwise across
the part. A file of *.log extension was manually created with the necessary information for
processing this data, and then the location of the file was entered into the first GUI. This GUI
processed the data into a new form, stored the new files, and then the second GUI was opened.
The location of the files, as well as a new manually-compiled information *.log file, were
entered into the fields, and the data was again processed and opened a MATLAB figure with a
graphic representation of the thickness distribution. The data did not correspond with the
correct scan x-y coordinates, there were missing data (missing from the graphical display)
around the edges corresponding to the last x-y coordinates, there was no labeling on the axis for
a user to tell which orientation the part was in, and the color scale was not logical or explained.
Furthermore, there were no directions on what order to proceed with which GUI first, there was
no user's manual to determine how to enter the data, or what files were required. Trial and
error was very ineffective since the file formats were unfamiliar and there were no specific error
messages; all error messages came up in MATLAB and were unhelpful or not descriptive enough
to determine where the fault was. Only once a trial-run-through were conducted of the GUI with
the initial developer could the files and program be used independently by the author.
4.3.2 Initial Developments for a New User Interface by the Author

From this personal survey, ideas were developed about what to incorporate into the GUI. For example, the format of the files received was set, but the design and implementation of the user interface was open to interpretation. One interface that integrated all processing behind the scenes was most logical, and a viewable screen with dimensional orientation available was a must. A way of mapping the data back to the correct x-y location on the part surface had to be implemented, since MATLAB’s handling of the matrices being used took the indices of each matrix as the actual dimensions of the part. The use of the incorrect dimensions was especially problematic when multiple points were averaged, and then the data had to be mapped back to the part dimensions for correct viewing and interpretation.

Some of the biggest questions arose when trying to decide how much flexibility and user input to allow. From the analysis of the original interfaces, it was decided that the more “random” numbers and information that must be entered into the GUI about the part, the more confusing the program and greater likelihood of a faulty analysis. Before any end-user input was obtained on which features and preferences to include in the GUI, baseline layouts of the GUI, and automation of some of the more random inputs was planned. Two directions presented themselves: a more rigid, working application that did not allow users many decisions, and a research-version that would allow for users more familiar with the application to have flexibility in some of the inputs. Figures 11 and 12 show the two different designs of the initial layouts: a simple working view and a more advanced research view respectively.
4.3.3 Surveying Users

The next step was to take the proposed GUI design layouts and present them to the end users. The layouts were presented with an in-depth survey of the features and functionality that required feedback from the end-users. Very direct questions were asked about one feature versus another and also more open-ended questions for the users to develop some of their own ideas for program features and GUI design implementation.
Once the end-user responses were received it was a matter of determining the best way to incorporate all suggestions in the program. Some applications and options were implemented in theory without a working code (such as the option to choose different frequencies). The code was developed to support all features and decisions were made as to the best layout, organization, and implementation of the features and specific MATLAB code.

4.4. Debugging and MATLAB’s “Features”

One of the many challenges outside of design and incorporation of user needs was the implementation of certain features and use of the MATLAB language and program. MATLAB, at times, is one of those programs that has very specific uses and therefore some of the usability is, at times compromised. One has to be patient and, at times, creative, if one wants to successfully and advantageously use the program. What is automatically generated is possibly not what the user is wishing for, and care must be taken to ensure that the results are indeed correct.

Many times the code for processing and displaying the data had to be checked for accuracy using actual data, and solutions had to be found when the viewable data did not match the expected results. One of the most prominent examples of this was the dimensions of the part versus the thickness data matrix indices. The dimensions of the part were given from zero to the length or width, while the matrix dimensions in MATLAB began at 1. Therefore, the data point that corresponded to the square inch located with vertices at (0,0), (1,0), (0,1) and (1,1) was actually displayed as though it represented the square with vertices (1,1), (1,2), (2,1), (2,2). Also, when data points were averaged, e.g. “samples averaged” was chosen to be 4, which meant that 16 data points were represented by a single point with a single matrix index, which was graphed as only that single point. The dimensions on the graph were then the matrix indices and not the dimensions of the part. To fix this display, transformations of the matrix data during
plotting were used to map the data to the correct placement and then to show the part
dimensions as opposed to the matrix indices as the X and Y labels. This then led to more
complications, when the “Select Points” option used the new labels as the indices to go back
and do the zoom data. At this point, the dimensional location of a point had to be mapped back
into its matrix index.

Still to date it cannot be determined why, but one of the biggest problems came with
MATLAB refusing to show the entire data set. When graphing, the entire final row and column
of the matrix were ignored, even before the data mapping and transformations were
performed. The reason behind this loss of data was never determined. When simply printed out
in the MATLAB command window, all data points existed. The fix to this problem was to add a
final row and column just before graphing of any random value. This value was never displayed,
as it became the new “ignored” row and column, and was added after all analysis of the part
was complete.

Another minor MATLAB coding challenge came with the development of the matrix
graph geometry processing, especially regarding the scaling of dimensions as it pertained to the
number of samples averaged. As the radius of curvature, border and transition zone thickness
were all given in part dimensions before the samples were averaged, they had to be converted
into number of matrix indices. If samples were averaged, i.e. “samples averaged” was greater
than 1, the dimensions of the different regions had to be altered by this scaling factor to ensure
the correct display.

Countless similar bugs were faced, and many creative yet robust solutions were found.
Solutions such as the aforementioned addition of an extra row and column for plotting purposes
were not necessarily logical, and had to be tested with multiple data sets to be assured that it
was not a “quick fix” for a problem with one data set. Similar problems included the retranslation of the zoom data plot so that the data points shown matched the dimensions labeled.
Chapter 5: Results

The “Results” chapter focuses on the integration of the user preferences into the user interface.

5.1. Integration of Survey Results

The first results considered were the survey questions posed to the end users. The survey and baseline GUI designs were presented to the engineers who would eventually be the primary users of the GUI. Their responses to the questions, both directed and open-ended, drove the later development of certain features of the GUI.

5.1.1 Black and White Option

The survey included specific directed questions about features already planned for the program, as well as open-ended questions about what features or preferences had not been considered that they would deem useful. The responses to the questions were helpful for directing the modification and improvement of certain features, and brought about the creation of some new ones.

An example of the impact of the survey on the program was the implementation of a black and white option. Since the different thicknesses were shown by color (with ranges for too thick, a warning for a thicker part, a color for close to the target thickness, a warning for a thinner part, and a too thin color), it was considered that users with color vision impairments could not easily or successfully use the program that was developed to make their analysis much easier. The survey asked whether or not they would like an option for “textures” to make the data interpretation easier. The end-users were very interested in having this option, so a different color scale was used (instead of textures) that MATLAB made available for shading
between black and white with a blue tinge. On the same topic, they said they preferred a 5 color scheme (as mentioned), so the two color and three color scheme options were abandoned.

5.1.2 Combination of different interfaces

The end-users requested that many of the “advanced” options be included in the final GUI. Therefore, the advanced research and more rigid working GUI’s were combined and developed into one GUI. The end-users also liked the idea of a progress bar to show time remaining for processing, as well as the ability to reload parameters from a saved file without typing them all back in. Features that they weren’t as excited about or really had no strong opinion on were the ability to change the size of the output image and specification of the type of image file. Users liked the idea of a “zoom” feature but had no opinion where it was displayed on screen. Figure 13 shows an in-progress version of the combination of the two GUI’s into a single user interface.

Figure 13: Combination of Advanced and Simple GUI’s
5.1.3 Addition of “notes” section

To the open-ended question of “what other features would you like to see?” the engineers asked for a notes section to add their personal notes associated with a part. Since they scanned the part and knew the physical part better than the program, they could identify faulty data points resulting from scan issues in the “notes” section. All the data, including the notes and other data gathered in this program, follow the part. When the data indicates there is a problem in a certain area, the notes on the part entered by the user could identify that there were other problems than one with the part, e.g. dirt particles on the part that manifest as an increase in measured thickness.

5.1.4 Tracking Record

The users also requested that a tracking file be generated for each part. Since the input files were already *.log files, it was easy to turn around and print all fields out into a similar file. Eventually this tracking “log” file became the input file for re-analysis of a part, i.e. it was the *.log file that was loaded at the beginning of a session when all parameters were known. Once the “Notes” section was added, this was continuously appended to the end of the *.log file and included in the archived records.

5.1.5 Problems with Request Interpretation and Implementation

Obviously some requests were either easier, of greater interest to the end users, or more interesting when it came to design and implementation, and therefore were more likely to be given first priority. Some of the setbacks with implementation of ideas came from a lack of specific request from the end users, or from a lack of direction as to how to implement an idea. An example of the former is the ability to open any *.log file and corresponding *.csv files, as opposed to those within the same MATLAB directory as the program files themselves. In the
early development stages this was tried and failed, and since a specific request to incorporate this option was not made, a solution to the failed code didn't come until many months later. Some minor requests, such as showing the absolute or relative error, were not implemented, since they were rather vague and not highly emphasized by the users. Did the users mean for each point's error to be shown? Or the mean? Instead, the standard deviation of the data was chosen. This, in turn led to requests for other statistical data, as seen in the second survey results.

5.2. Second Survey Results

A few months after the initial survey and intense development of the single user interface, a second survey was conducted to ensure that the interface was indeed meeting the user's needs. Some similar questions to those in the first survey were asked, as well as questions targeting the new developments that had not been addressed or anticipated as part of the initial GUI development. The engineers were able to answer some questions, but were also confident enough to say when they did not have enough experience or knowledge to give helpful feedback. It was helpful and appreciated that the engineers didn't try to make up answers or change the question into one they could answer.

5.2.1 Specific Component Choice

When asked if the engineers would find it more helpful to simply select part A, B, C, etc, which each had its own dimensions and tolerances, as opposed to having to manually put in all dimensions and tolerances, they preferred the latter. The end users said they did not actually have an itemized list of each part they would be analyzing, and therefore found the flexibility offered in the current option more desirable.
5.2.2 Various Image File Types

The user can chose to store the full or zoom images and all images are stored in JPEG format. In contrast to their initial lack of preference given in the first survey, when asked if the engineers wished for another format, they requested a PNG format, due to its smaller file size.

5.2.3 Statistics and Threshold numbers

The engineers seem to approve of how the current statistical data is displayed, but also requested to see absolute maximum, minimum, and a corresponding range. In addition, the color scale was appreciated, but the labels were deemed a bit confusing. The engineers offered no clearer naming convention for the varying levels of thicknesses, but suggested that another one be tried. They also requested exact numbers for where each thickness region was defined, i.e. what thicknesses were encompassed by “acceptable,” “too thick,” etc.

5.2.4 Unanswerable Questions

Since the engineers did not extensively exhaust the capabilities of the program they felt unable to answer such questions as “What other features would you like to see?” Here was an example of a tight turnaround time for user input, where, had there been more opportunity for the users to test the program, the questions could have been answered and more thorough input could have been generated to further improve the “usability” of the GUI.

Development of a user’s manual by the programmer is generally frowned upon; it is believed that the programmer is too close to the software to objectively or concisely describe the use of the program. Due to the small number of researchers available, however, the author was given the task of writing the manual for the GUI.

To develop the user’s manual, the GUI was initialized and familiar data was used to test each feature. The flowchart was followed for each data-opening option, as well as later features, such as point selection, zoom-view creation, image saving, and statistical data acquisition.

At the time of this research the users have been unable to evaluate the effectiveness of the user’s manual; however, it shall be included in future developments and surveys to evaluate its performance similar to that of the GUI.

The user’s manual can be found in Appendix A.
Chapter 7: Summary and Conclusions

7.1. Summary

This paper describes the successful development of a GUI designed to assist engineers in accurately displaying the acquired thickness data of advanced composite materials during production to reduce acquisition costs and increase product accuracy.

User input in the user interface design and development process is a useful and, at times, necessary tool to determine the needs and requirements of a program, especially a program with a very specific and small end application. Using multiple user surveys to have continuous feedback and communication during the GUI development process saves time and is more productive in the long run to create an efficient, user-friendly GUI for end users.

7.2. Conclusions

- User input and feedback must be considered for GUI programming
- Interface programmers must keep in mind that they are not as familiar with the application of their product as the end user may be
- Open dialog and common language must be used when communicating between programmers and users
- Both end-users and programmers must be open to suggestions and developments that they may not initially see the practicality and efficiency of
Chapter 8: Future Work

Two directions of future work exist for this project. The first is to continue to develop the existing program for its current application, for thickness viewing. Further developments can be made for recognizing and storing abnormal points and regions, as well as monitoring “hot spots” or areas likely to be out of tolerance.

The second direction requires more modifications of the current MATLAB code, and would alter the program from a thickness viewer to a defect detection tool. Defect detection requires a much higher scanning resolution to capture the defects of interest (e.g. as small as ¼” diameter). Currently the GUI only handles one data point per square inch. However, with careful recoding of some functions and implementation of additional options, the GUI could handle defect viewing as well, for examining such things as dirt, debris, delaminations, etc. that occur within the part during fabrication. The development of this GUI requires ultrasonic test data and end-user feedback to ensure an efficient, user-friendly result.
Works Cited


http://www.mathworks.com
Appendix A: User’s Manual

Advanced Composite Materials Precision Thickness Viewer User’s Manual

Opening a file

Before the data can be viewed, information must be entered on the part to assure accurate data conversion.

Information that must be known about the part:
• Dimensions of part
• Radius of curvature
• Transition width
• Border region width
• Space on outside of part that was included in scan
• Number of *.csv files
• Number of header lines on *.csv files
• Title of raw data *.csv files
• Cure number, internal feature

Data about the dimensions of the part must all be in the same units, either inches or feet. These should be obtained during or before the scan of the part.

Once the data has been converted into *.csv files, information about the *.csv files can be obtained by opening the file and counting the number of lines before the data starts, if the number was not previously known. The number of *.csv files can either be known or counted, and the titles should all be the same minus the final scan line number, as seen below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>shared_complex_data</td>
<td>7/12/09 7:25 AM</td>
</tr>
<tr>
<td>thickness_data</td>
<td>7/12/09 8:38 AM</td>
</tr>
<tr>
<td>Aluminum-jinshin-025step-Sus ColtB intricate 2.csv</td>
<td>4/28/09 11:07 AM</td>
</tr>
<tr>
<td>Aluminum-jinshin-025step-Sus ColtB intricate 5.csv</td>
<td>4/28/09 11:07 AM</td>
</tr>
</tbody>
</table>

Once this title is known, it is entered into the program, as seen below, without the final number.
The first and last line numbers are entered into the *.csv “Start File” and “End File” fields:

Once all data has been entered into the “Part Information” area, the file can be opened. If the file has not been opened previously, a dialog box will be opened to save the data to a *.log file.

Navigate to the folder containing the *.csv files, and save the log information. It is easiest to save as the provided title for later recollection. The dialog box will only show *.log files, but to make sure the right folder is selected, “Save as type” can be changed to “All Files” and the *.csv files will be shown. Save the data as a *.log file.
Once the file has been saved, the program will ask which frequency has been used. This, along with the input of the cure number and internal feature being investigated, will help choose the correct effective velocity of sound in the material to use for mapping of the ultrasound data into the corresponding thickness. The program will then begin to process the files and plot them into a range of thickness tolerances. For smaller parts this may only take a minute or two, but larger parts will take longer. A progress bar shows the approximate time left.

The data will appear on the screen, with a color scale indicating the acceptable areas and the areas that are too thick or too thin.
If a part has been previously opened and has a saved log file, the “Load Parameters” option can be selected. Navigate to the folder containing the *.csv and *.log files and select the *.log file that contains the information on the part. Select “Open” and the information from the last save will be entered into the fields.

The “Open File” button can be selected once the data fields have been filled, and the data will be opened.
The scale has yellow as acceptable, orange is between 75% and 100% of the tolerance value thicker than the target value, and red is too thick (100% of the tolerance or outside of it). Likewise, aqua is between 75% and 100% of the tolerance value too thin and blue is 100% of the tolerance thinner than the target value. Navy points are part of an outer scrap region or transition zone and are excluded.
**Closer Examination**

If a certain area should be examined in greater detail, use the “Select Points” option under the “Graphics Options” section. Use the cursor to select the point on the entire view.

![Image of Local Mean Thickness](image)

Once a point is selected, the coordinates will be entered into the “Notes” section. This section is also used for typing notes on the part or selected points, which is then saved to the notes section in the *.log file.

![Image of Notes](image)

Many points can be selected; however, when the “Plot Zoom” is selected, the last point chosen is the point that is seen in more detail, and is shown in the “Selected Points” section. No matter how many points have been averaged, in the zoom axis each point is shown.

![Image of Zoom View](image)

A one-foot-square area centered on the chosen point is displayed on the zoom axis, with each data point shown. The data is then shown as one data point per square inch, as the data has been scanned. If a point near the transition zone is chosen, only the thickness for the window or border will be used and the transition zone will be shown as either thick or thin.
The mean and standard deviation are shown in the statistics section for the process engineers to analyze and use for improvement.

The “Line-By-Line Statistics” option will give more detailed statistics.

The “Mean Along X Direction” shows each x coordinate of the entire view window region and the mean of all y coordinates along the given x coordinate. Likewise, the “Standard Deviation Along X Direction” gives the standard deviation of all y coordinates along an x coordinate. For the other axes, the same analysis is done for all x coordinates along a given y coordinate.
Academic Vita of Jillian J. Woolridge

Jillian J. Woolridge
120 Haymaker Circle
State College, PA 16801
H: (814) 238-9428
C: (814) 441-0942
jjw5101@psu.edu
jillian.j.woolridge@gmail.com

Education

Bachelor of Science Degree in Engineering Science with Honors and High Distinction
Minors in Nanotechnology and German
Penn State University, Spring 2010
Thesis title: Development of a User Interface in MATLAB to Assist Precision Ultrasonic Thickness Mapping
Thesis Advisor: Kevin L. Koudela

Related Research Experience

Penn State Applied Research Lab          Spring- Fall 2009
Designed and implemented a graphical user interface in MATLAB to aid in-situ thickness monitoring of advanced composite materials during production.
Supervisor: Dr. Kevin L. Koudela

Ruhr-Universität Bochum, Chair of Materials Technology  Summer 2008
Developed and studied dissimilar weld joints between Nickel-Titanium shape-memory alloy and steel plates for observation of shape-memory properties within the welded region.
Supervisor: Hajo Gugel, Dipl. Ing.

Penn State Engineering Science and Mechanics Dept. Spring 2008
Assisted in the acquisition of data using a prototype phased array for ultrasonic testing of fuel rods in high temperatures.
Supervisor: Dr. Bernhard Tittmann

Penn State Applied Research Lab Summer 2007
Designed NSF-funded project web site; worked with a team of researchers and graduate students developing a program to explore trade-spaces as an alternative to optimization.
Supervisor: Dr. David Spencer

Campus Involvement

Collegiate Horsemen's Association at Penn State – Treasurer, Spring/Fall 2008; Vice-President, Spring/Fall 2009, member since Fall 2007
Penn State Society of Engineering Science – Secretary, Fall 2009, member since Fall 2008
Essence of Joy Choir, 45 men and women, special audition – Member since Fall 2008
Past Member:
- Oriana Singers Choir, 50 women, audition, Fall 2006 until Fall 2007
- Concert Choir, 45 men and women, special audition, Fall 2007 until Spring 2008
- Penn State Society of Women Engineers, Fall 2006 until Spring 2008
- Penn State Archery Club, Fall 2006 until Spring 2007

National Involvement
- American Collegiate Horsemen's Association – Treasurer, March 2009-Present
- United States Equestrian Federation, United States Hunter/Jumper Association – Member

Honors and Awards
- Dean’s List – All Semesters
- Stanley Townsend Award, Rising Junior in German ($450 one-time scholarship)
- Andrew Pytel Scholarship, Engineering Science ($490 one-time scholarship)
- Robert C. Byrd Scholarship Recipient (4-year scholarship, total $6,000)
- Penn State Academic Excellence Scholarship (4-year scholarship, total $14,500)
- PSAT National Merit Finalist 2006
- Penn State National Merit Scholarship ($1,000 one-time scholarship)