2012-04-12

Application of Parametric NURBS Geometry to Mode Shape Identification and the Modal Assurance Criterion

Evan D. Selin
Brigham Young University - Provo

Application of Parametric NURBS Geometry to Mode Shape Identification and the Modal Assurance Criterion

Evan Selin

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

C. Greg Jensen, Chair
David T. Fullwood
Brian D. Jensen

Department of Mechanical Engineering

Brigham Young University

June 2012

ABSTRACT

Application of Parametric NURBS Geometry to Mode Shape Identification
and the Modal Assurance Criterion

Evan Selin
Department of Mechanical Engineering, BYU
Master of Science

The dynamic characteristics of a part are highly dependent on geometric and material properties of the part. The identification and tracking of vibrational mode shapes within an iterative design process becomes difficult and time consuming due to the frequently changing part definition. Currently, visual inspection of analysis results is used as the means to identify the shape of each vibrational mode determined by the modal analysis. This thesis investigates the automation of the mode shape identification process through the use of parametric geometry and the Modal Assurance Criterion.

Displacement results from finite element modal analysis are used to create parametric geometry templates which can be compared one to another irrespective of part geometry or finite element mesh density. Automation of the mode shape identification process using parametric geometry and the Modal Assurance Criterion allows for the mode shapes from a baseline design to be matched to modified part designs, giving the designer a more complete view of the part's dynamic properties. It also enables the identification process to be completed much more quickly than by visual inspection.

Keywords: NURBS, modal analysis, MAC, mode shape identification, automation

ACKNOWLEDGEMENTS

I would like to thank Dr. Greg Jensen from Brigham Young University as my graduate committee chair for all of his time, effort, and direction in helping me to pursue this research and the thesis originating from it. I would also like to thank Kurt Heinemann and Ammon Hepworth from Pratt & Whitney who have been instrumental in supporting and contributing to the success of this research project on a weekly basis. Pratt & Whitney is also responsible for the funding of this research for which I am especially grateful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Determining an object's vibrational mode shapes and each mode's corresponding frequency is the aim of performing a modal analysis using the Finite Element Method and is the first step in modeling the dynamic behavior of a structure[1]. Finite Element modal analysis provides a simple check of a part's dynamic properties for designers in the product development process. If an iterative process, such as an optimization, is used to define and modify parameters of the design, the frequencies of the vibrational modes and even the order in which they are excited can change at each design iteration. A Modal Assurance Criterion has been developed and used mostly in comparing and identifying analytical and experimental mode shapes [2], [3]. It can likewise be used to compare results between two analytical eigenvectors so long as the nodal vectors are of the same size. If this is not the case, parametric curve and surface geometries can be used as a means to compare and identify mode shape and frequency data for geometries that are geometrically identical or which are meshed with differing levels of detail. This research seeks a method by which parametric NURBS geometries are used in conjunction with the modal assurance criterion to automatically identify vibrational mode shapes and frequencies from modal analysis displacement data. This will make it simple to obtain detailed information about a part's dynamic behavior when an iterative design process is being used.

1.1 **Problem Overview**

When an iterative design method, such as an optimization or design of experiments, is used in the design process, parametric models can be updated with new geometry, feature dimensions, positions, and material properties among other things. The goal in changing the parameters and geometries is to arrive at a design that is superior to the baseline, or starting design. Iteratively modifying the properties of the part in turn modifies the static and dynamic behavior of the part when compared to the baseline design. Depending on how much the part is changed, these behavioral differences can be large or small.

A modal analysis completed using finite element analysis results in natural frequencies and mode shapes of the part modeled. These results are reported in order of increasing natural frequency. Contour plots from a modal analysis can be seen in Figure 1-1 below.

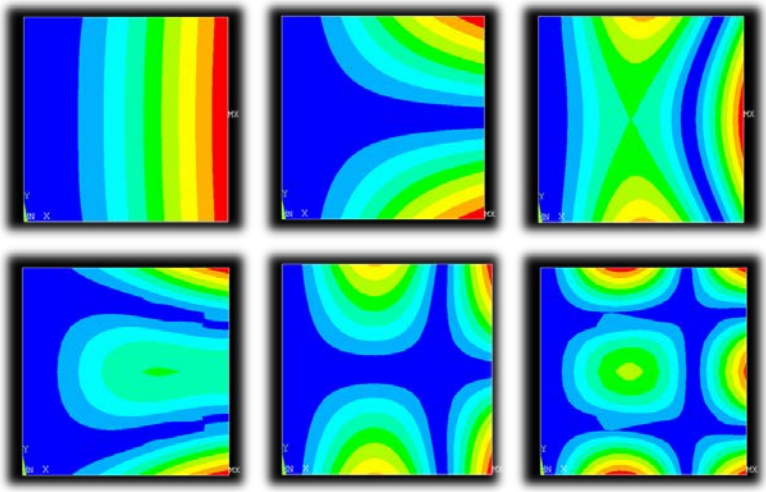

Figure 1-1 Mode shapes produced by a modal analysis

The frequencies of the mode shapes are affected by design changes and the vibrational mode shapes excited by these natural frequencies can exhibit themselves in a different sequence on

different design iterations. The task of identifying the specific mode shape associated with each natural frequency is complicated by this reordering of natural frequencies between iterations.

The Modal Assurance Criterion (MAC) was developed in order to provide a measurement of consistency between modal vector estimates from different sources[4]. A set of modal vectors from a finite element analysis can easily be compared to other measured or estimated modal vectors using the MAC in order to determine if the mode shapes are consistent between the two. The MAC has also been used to compare and identify mode shapes from the displacement results of two differing finite element analyses. This works when comparing two sets of modal vectors of the same size. When modifying and re-meshing geometry within an iterative process it is not always feasible to create consistent finite element meshes at each iteration. This makes it difficult to identify the mode shape at each natural frequency using the MAC. A method must be developed by which this important information can be obtained automatically throughout the entire optimization process.

1.2 **Thesis Objective**

The purpose of this thesis is to develop a method which is capable of automatically matching the mode shape for each natural frequency of a modal analysis to known mode shapes using the displacement results. This automated process will provide the designer to have a more complete understanding of the part's dynamic properties throughout the whole optimization process.

This research will also investigate the feasibility of using parametric geometry to represent modal analysis displacement results and compute correlation between mode shapes using the MAC. The parametric geometry will enable the MAC to match the mode shape to known modes even for models with differing dimensions and/or mesh coarseness. This research

will compare the time required and matching accuracy of two methods, one using parametric curves and one using parametric surfaces, with the standard MAC in order to determine the most effective matching method. The main benefit of this research will be in determining a method by which parts of differing geometry can be correctly matched to known mode shapes due to the unique properties of parametric geometry.

The mode matching application will be embedded into an optimization software package to make it easy for a user to set up and run optimizations that utilize its functionality. This embedded application will allow a designer to create templates of known mode shapes as well as compare and correlate mode shapes of unknown results to saved templates throughout an optimization run.

1.3 **Problem Delimitations**

The purpose in developing an application to accomplish the tasks mentioned above is to ensure that the proposed method is feasible and effective. In addition, as NURBS parametric surface geometry will be used by the application, the development of the method will be limited to matching parts that can reasonably be represented as four-sided surfaces. Modal analysis carried out as a part of this research will be done using the ANSYS analysis software package. Analysis models built and used will be modeled using ANSYS SHELL63 elements and will be limited to two-dimensional representations of parts. All parametric data structures needed in this research will utilize Solid Modeling Solutions' GSNLib libraries which are commercially available. The application will be integrated into SIMULIA's Isight optimization framework.

1.4 **Thesis Organization**

This thesis is organized into six chapters. Chapter 2 is a literature review that introduces the reader to the most relevant literature in relation to this thesis. This will include a brief discussion of modal analysis, a discussion about the Modal Assurance Criterion, and NURBS parametric geometry and mathematics. The third chapter discusses the general methods used to automate the matching of modal analysis results to templates of known mode shapes. The fourth chapter discusses the implementation of those methods using C++ programming and integrating the application into the Isight optimization software. Chapter 5 details the results of the mode matching application and compares these results to results obtained using the basic MAC calculation. The sixth chapter discusses the conclusions and future work of this research.

2 BACKGROUND

The intent of this chapter is to give the reader a foundational understanding of the material in order to understand the significance of this research. A description of modal analysis will be given first, in Section 2.1, in order to introduce the general procedure followed and results obtained from dynamic analysis using the Finite Element Method. Section 2.2 serves as an overview of the purpose, definition, and properties of the Modal Assurance Criterion. Research demonstrating the properties and application of the Modal Assurance Criteria to mode identification and the Finite Element Method will also be presented in this section. Following this, Section 2.3 presents a brief overview of NURBS parametric geometry including the mathematics formulas that define the geometry and some unique properties they possess which demonstrate their application to this research.

2.1 Modal Analysis

In the design process, important dynamic characteristics of a structure or component are determined by means of a modal analysis. The dynamic properties determined in a modal analysis are the natural frequencies and vibration mode shapes of the component[5]. These characteristics are of vital importance for structures designed for dynamic loading conditions, such as turbine blades. In its most basic form, that of free-vibration with no damping in the system, a modal analysis consists of solving the matrix eigenvalue problem

$$[[\boldsymbol{k}] - \omega_n{}^2[\boldsymbol{m}]]\boldsymbol{\phi_n} = \mathbf{0}\,, \qquad \textbf{(2-1)}$$

where **[k]** and **[m]** represent the system stiffness and mass matrices and ω_n and $\boldsymbol{\phi_n}$ are the natural frequencies and mode shape vectors for the system[6]. Boundary conditions, system damping, and stress states may be added resulting in a similar yet more complicated solution process. Finite Element solvers perform modal analysis on meshed models, with the end result being a natural frequency and a vector of nodal displacements (mode shape) for each vibrational mode. Figure 2-1 shows a sample contour plot of nodal displacement results for a specific natural frequency solution from a modal analysis.

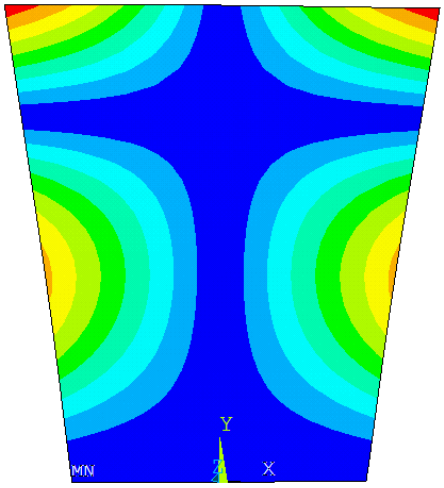

Figure 2-1 Contour plot of nodal displacement magnitude

Components with differing geometry can exhibit the same overall mode shape, although the natural frequency and exact nodal locations and displacements in the model differ. This research develops a method by which matching mode shapes can be identified despite having

differing geometry or nodal vectors. Figure 2-2, below, shows modal analysis results with the same mode shape as the figure above even though the geometry is different.

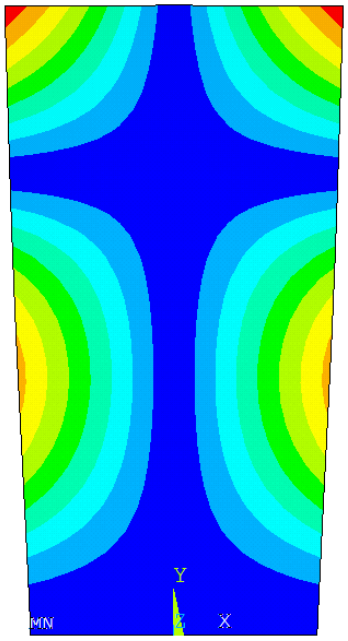

Figure 2-2 Countour plot of mode shape on different geometry

2.2 The Modal Assurance Criterion

The Modal Assurance Criterion (MAC) was developed around thirty years ago and has been used widely in the engineering community to measure the consistency of modal vector estimates, calculations, or experimental data[4], [7]. The MAC calculates the linearity between any modal vector and a reference modal vector. The mathematical definition of this comparison is:

$$MAC = \frac{\left|\sum_{q=1}^{N_0} \phi_{Aq}\phi_{Rq}\right|^2}{\sum_{q=1}^{N_0} \phi_{Aq}\phi_{Aq} \sum_{q=1}^{N_0} \phi_{Rq}\phi_{Rq}}, \quad (2\text{-}2)$$

where $\boldsymbol{\phi}_A$ is one modal vector which is to be compared to a reference modal vector, $\boldsymbol{\phi}_R$. In this definition, each vector contains N_0 locations where displacement is measured and q denotes the specific element in the vector. The result of this calculation is a value between zero and one, which indicates the consistency between the two mode shapes, a result of one being a perfect correlation and zero being no correlation. Comparing modal vectors by means of this calculation allows for similar mode shapes to be identified and matched simply. Figure 2-3 shows a graphical representation of the MAC values resulting from a comparison of two sets of ten mode shapes. The dark elements seen across the diagonal of the grid indicate high correlation between the mode shapes. Lighter elements indicate low correlation between the modes. As demonstrated by mode shapes number seven and eight, it is possible for a single mode shape to correlate well with multiple mode shapes.

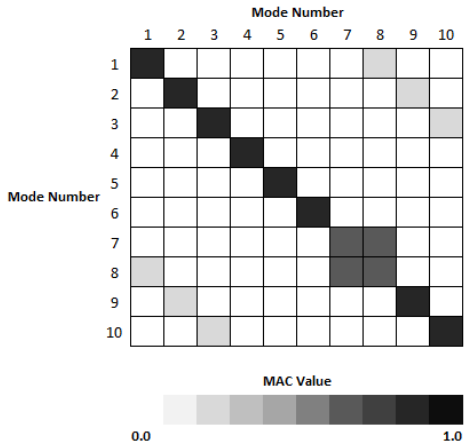

Figure 2-3 Sample MAC evaluation results

As it has been consistently used over several decades, the MAC has served as a foundation for the development of other specialized assurance criteria including a Partial and Spacial Modal Assurance Criterion[8]. One pitfall of the MAC is that because it is "normalized to the highest amplitude response,…a large amplitude local response can mask the global response"[9]. Another is that it also requires that the two vectors being compared have the same number of elements. This means that modal solutions of Finite Element models with differing meshes cannot be compared simply by the MAC because the meshes differ either by the size of the nodal vector or by the locations of the nodes in the model. This shortcoming will be overcome by using NURBS geometry to represent the modal analysis nodal results which then allows the same number of displacement data to be extracted from the parametric geometry that represents the different models.

Burns applied the MAC to identify the characteristic mode shapes of an axisymmetric rotor using Finite Element Analysis results and reference eigenvectors from an actual structure[10]. In doing this, he avoids the necessity of using visual inspection to determine the mode shape, greatly improving the speed and accuracy of the modal identification process. One area of difficulty encountered in his research is the occurrence of false-positive matching of dissimilar mode shapes. By calculating the average displacement magnitude of the eigenvectors, the false-positive identifications were able to be identified and corrected. One of the most important reasons for automation of the mode shape identification process is to eliminate the need to visually inspect each solution to determine the mode shape.

Ewins describes the Modal Assurance Criterion as a scalar least-squares deviation measurement of how much eigenvectors vary from a linear correlation[11]. Excessive variation in the two eigenvectors being compared indicates that the two modes are dissimilar. Ewins

points out that care must be taken in assigning significance to the scalar result calculated by the MAC because the “acceptable” and “non-correlation” result limits will vary based on the data points used in the calculation. In addition, nonlinearities in the model, data noise, poor analysis or experimental data, or inappropriate degree of freedom choice can cause a low MAC correlation value. It should be noted that for Ewins’ research, the MAC was being used as a method for model updating and validation.

An efficient method has been presented by Kim to track modes using the Modal Assurance Criterion during a structural topology optimization[12]. This allows the tracking of the order of excited eigenmodes despite the fact that the structural configuration changes at each iteration step. The method involves assigning reference and desired mode shapes at the initial optimization setup and then comparing estimated mode shapes to these using the MAC throughout the optimization process. This application successfully maximized user-specified natural frequencies and correctly tracked the important mode shapes even as the topology of the structure changed. Only a subset of the model’s nodal data was used in the MAC calculation in order to improve efficiency. Kim’s research demonstrates the applicability of mode shape identification in an optimization process, which will be used in this research.

2.3 **Parametric Geometry**

Curves and surfaces are capable of being represented by several different types of equations including implicit, explicit, and parametric. A parametric definition of a curve or surface allows for a quick and simple computation of the real-space coordinates of a point existing on the geometry based on a parameter value, or values[13], [14]. Non-uniform rational B-spline (NURBS) curves and surfaces are widely used parametric geometry representations. A NURBS surface must be defined by a set of control points which lie topologically in a

rectangular grid. Due to this, their use is mainly limited to represent surfaces that are four-sided. Interpolation through existing points and extrapolation of points from a set of control points are two common methods for creating NURBS curves and surfaces[15]. The interpolation of a surface from a set of points is shown below in Figure 2-4.

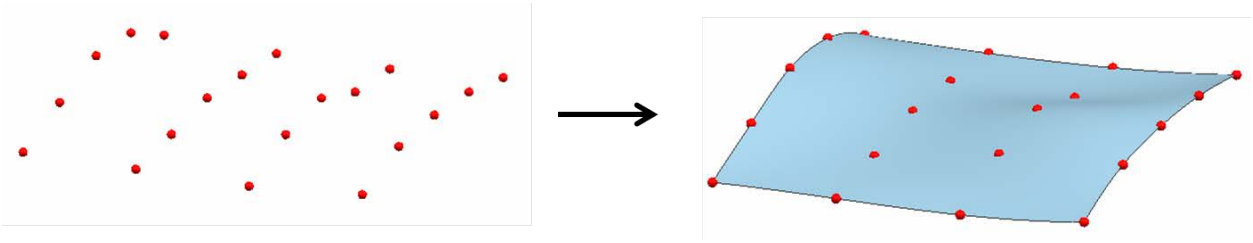

Figure 2-4 Surface interpolation from a set of points

A brief description of the mathematical definition of NURBS surfaces will be presented here with the understanding that the same principles apply to curves by removing one of the parameter dimensions. Each control point defining a surface has a weight and a B-spline basis function which together determine the extent to which the point influences the surface. A knot vector in each parametric direction, *u* and *v*, also influences the surface topology. The following equation is the mathematical definition of a NURBS surface of degree *q* in the *u* direction and degree *r* in the *v* direction.

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)W_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)W_{i,j}}, 0 \leq u, v \leq 1 \qquad (2\text{-}3)$$

In the definition above, $P_{i,j}$ are the control points, $W_{i,j}$ are the weights of each control point, and n and m are the number of control points in the u and v directions respectively. The expressions $N_{i,p}(u)$ and $N_{i,q}(v)$ are the NURBS basis functions which are defined on the knot vectors:

$$U = \{0, \dots, 0, u_{p+1}, \dots, u_{r-p-1}, 1, \dots, 1\}, u_i \leq u_{i+1} \quad (2\text{-}4)$$

$$V = \{0, \dots, 0, v_{q+1}, \dots, u_{s-q-1}, 1, \dots, 1\}, v_i \leq v_{i+1} \quad (2\text{-}5)$$

where $r = n + p + 1$ and $s = m + q + 1$. At the beginning and end of each knot vector, there are p, or q, repeated zeros and ones, respectively, which terminate the surface's control curves and indicate that the surface is parameterized between zero and one. More information on the mathematical and geometric properties of NURBS curves and surfaces can be found in numerous sources on the topic[13–16].

The real power in using parametric representations of geometry in this research is that two dissimilar curves or surfaces can be related to one another through their parameterizations. The parametric definition allows for two separate geometric entities to be easily compared in parameter-space even though the two entities may be of differing size, configuration, and complexity in real-space. Figure 2-5 shows how this comparison can be made with a set of points at given parameter values on two different parametric curves.

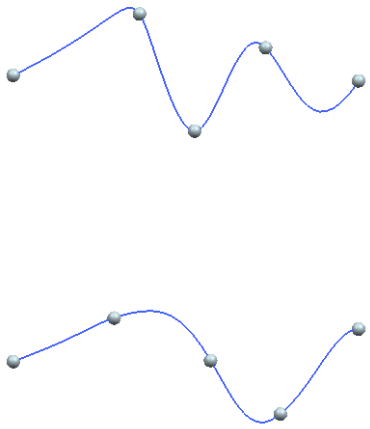

Figure 2-5 Corresponding parametric points on different NURBS curves

Countless algorithms for creating, editing, and storing NURBS geometry has been created and is available in the form of published geometry libraries. One of these powerful libraries, named GSNLib, has been developed for use in C++ programming. Additionally, many CAD systems, including Siemen's NX and Dassault Systems' CATIA, have excellent NURBS algorithms available for use by the means of their software's Application Programming Interface (API). The benefits of using parametric geometry in this form while working with product design and Finite Element modeling have been demonstrated by Hepworth[17] and Astle[18].

3 METHOD

This chapter discusses the methods developed to automate the identification of mode shapes represented by the displacement results of a modal analysis. The following steps are involved:

1. Read the modal analysis displacement results and develop an understanding of the modeled part's geometry and boundary node sequence.
2. Transform the nodal displacement data into a mathematical NURBS representation of the results for each modal solution and save the data as a template for future use.
3. Compare one NURBS representation of a modal solution to a stored template of known mode shape using the Modal Assurance Criterion and determine if results match the template's mode shape.
4. Gather and store information required to automatically create, store, compare, and report modal analysis results and identified mode shapes from a user familiar source (i.e. an optimization program).

A visual representation of these individual steps is shown in Figure 3-1.

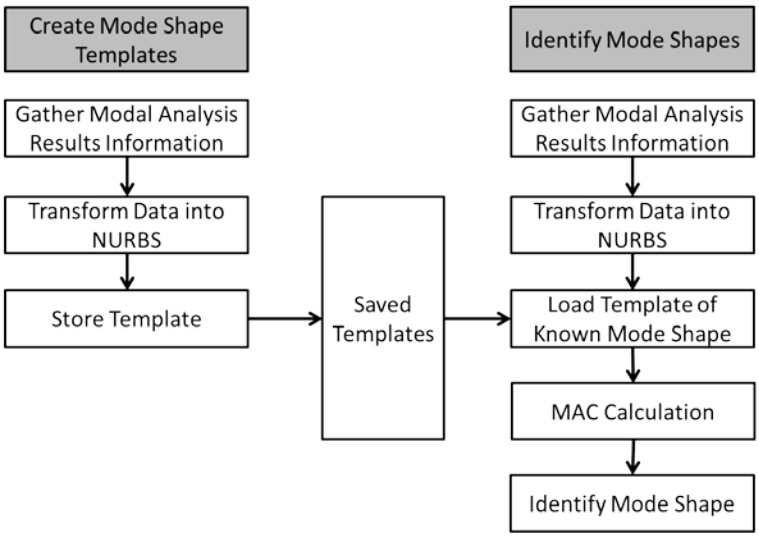

Figure 3-1 Flow chart of the mode identification method

3.1 Gather Information From User

In order for the full functionality of this application to be easily accessed and used, a simple front end is created that allows the user to set up and run the application. A Java application is set up to collect all required information from the user in order to create template surfaces or to run the mode shape identification application within an optimization.

3.2 Transform Data into NURBS

The transformation of the nodal displacement results into a mathematical NURBS representation is handled differently depending on the type of NURBS geometry being created. The method is different depending on whether NURBS curves or surfaces are created. Each of the tasks described below contributes to the completion of this process.

3.2.1 Read Displacement File and Determine Node Sequence

Modal analysis is carried out on a Finite Element model of a component and the nodal displacements are written to a file. Included in this file is a list of each node that exists on the four edges of the component and the position and displacements of all nodes in the model for each natural frequency solution determined in the analysis. After all of this information is stored, the node sequence around the component boundary must be determined.

It is important to determine the node sequence as well so that each node's location along the boundary in relation to the other nodes is known. First, the corner nodes are found by determining which four nodes exist on more than one edge. The sequence of nodes along each edge can then be determined based on the distances between each edge node and the corner nodes of that edge.

3.2.2 Normalize Nodal Displacements

The normalized nodal displacement for each node in the solution is required to create the NURBS templates. To accomplish this, each node's displacement is normalized by the maximum nodal displacement found in the solution set. If U_{all} denotes the set of all nodal displacements in the model solution, the normalized displacements, U_{norm}, are found by:

$$U_{norm} = \frac{U_{all}}{\max(|U_{all}|)}, \tag{3-1}$$

which results in all normalized nodal displacements falling between negative and positive one, with a value of one representing the maximum positive displacement.

3.2.3 Normalize Node Locations

The normalized node location of each node along the border of the model is needed in order to create the NURBS templates. In order to calculate the normalized node location for each border node, a particular edge and corner node are selected to define the starting point and direction that the template curve will follow. A normalized position value, S_{edge}, is calculated for each node along the starting edge based on its distance from the starting corner node and the total length of the starting edge according to the equation

$$S_{edge,i} = \frac{|P_i - P_0|}{|P_{n-1} - P_0|}, \quad i = 0,1,\dots,n-2,n-1 . \tag{3-2}$$

In this equation, i is the number of the node in sequence and n is the total number of nodes along the edge. After this has been done for the starting edge, a normalized position value is likewise calculated for all nodes on the other edges, continuing in the same direction around the model as the starting edge. The normalized position values begin at zero and increase to one along each edge. A total position value, S_{tot}, is obtained for each node by the equation

$$S_{tot} = S_{edge} + i - 1, \quad i = 1,2,3,4 , \tag{3-3}$$

where S_{edge} is the node's normalized position on its own edge and i indicates the sequence of the edge on which the particular node resides around the boundary, beginning at one and proceeding around the model to four. Thus, the normalized position values for the nodes around the edge of the component begin at zero and increase to four.

3.2.4 Transform Data into NURBS Curve

It is desired to use the node location and displacement for a particular modal solution to create a parametric NURBS curve. The normalized node positions around the border of the model and the normalized nodal displacements are used to do this.

A new point set is created from the correctly ordered nodes, beginning at the starting corner point and moving around the border of the model. Each point in this set has only two dimensions, one coming from each original node's normalized position value, S_{tot}, and one from each original node's normalized displacement value, U_{norm}. The third dimension for each point is simply set to zero so that the curve only uses two dimensions. An algorithm existing in the NURBS libraries used interpolates the set of points, creating a NURBS curve that represents the mode shape of the solution.

3.2.5 Transform Data into NURBS Surface

It is desired to use the node location and displacement for a particular modal solution to create a parametric NURBS surface. Similarly to the method for creating a NURBS curve from the data, this method to create a NURBS surface uses the normalized displacement data for each node. This method also uses the position of all nodes in the model to create the template surface.

A CAD API function fits a preliminary parametric surface through a set of points containing each node location in the model. This surface, which exists in the CAD system is transformed into a CAD independent form which allows easy storage and programmatic manipulation of the surface. This transformation of the CAD control points and knot vectors into the CAD independent data structure is shown in the equations below.

$$\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}P_{i,j}{}_{CADIndep} = \sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}P_{i,j}{}_{CAD} \quad (3\text{-}4)$$

$$U_{CADIndep} = U_{CAD} \quad (3\text{-}5)$$

$$V_{CADIndep} = V_{CAD} \quad (3\text{-}6)$$

As a parametric surface, each node in the model has a U and a V parameter which define that node's approximate position on the surface. The parameter values associated with every node in the model are determined and stored using a function existing in the geometry libraries. Since all nodes do not lie exactly on the surface, this is done by adjusting the parameter values to minimize the magnitude of

$$P(x,y,z) - S(u,v) = P(x,y,z) - \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)W_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)W_{i,j}}. \quad (3\text{-}7)$$

A new point set is created with data from each node. Points in this set are defined in three dimensions by the U parameter of the node on the preliminary surface, the V parameter of the node on the preliminary surface, and the normalized displacement of the node. The surface fitting function from the API is once again used with this new set of points, creating a new surface representing the mode shape within CAD. The resulting NURBS surface is translated once again from CAD into a CAD-independent data structure. The NURBS surface is also reparameterized in order to ensure that the parameterization is uniform in both directions.

3.3 **Store Template**

After the NURBS template has been created, it is stored for future use. The user defines where the mode shape template for the model is to be saved. The unique user-defined name for each mode shape represented within the template is also stored in the template alongside the geometric definition of the curve or surface. The storage of the templates as files allows the application to use the template's stored mode shape definitions immediately after creation, or in the future as the need arises.

3.4 **Load Template of Known Mode Shape**

The application must load a template file prior to completing any mode shape identification to current results. A mode shapes template file that exists in a user-specified location, as described in Sections 3.2 and 3.3, is opened and read by the application. Each geometric curve or surface definition, representing a mode shape, in the file is read in and precisely reproduced and the specific names associated with each mode shape are stored for future use by the application.

3.5 **MAC Calculation**

The Modal Assurance Criterion will be used to compare saved templates of known mode shape to results obtained from a modal analysis in order to classify the mode shapes of the modal analysis results within an optimization loop. The modal analysis is first carried out and the results written to file. The data in the results file are transformed into a normalized NURBS geometric representation as has been described in the previous sections. After this NURBS geometry has been created, a template of known mode shape is loaded from the database of saved templates. When both the NURBS geometry from the current results file and the template

of known mode shape are loaded, or created, the two geometries can be compared and the results reported back to the optimization loop. How this is done for NURBS curves and surfaces is detailed in the following sections.

3.5.1 Comparing Curves

When working with NURBS curves, the comparison is very simple. Since the NURBS curve created from the current modal analysis results is normalized and parameterized the same as the saved template of known mode shape all that must be done is to discretize both curves at user-defined intervals and calculate the MAC correlation value. Each NURBS curve is divided into *n*-1 equal parameter intervals, resulting in *n* equally-spaced parameter values along each curve where *n* is defined as the number of points the user has specified to use in the MAC calculation. The template curve is evaluated for each of the *n* parameter values, where the set of all evaluated points makes the reference vector, $\boldsymbol{\phi}_R$. The curve created from the current analysis results is likewise evaluated at each parameter value, defining the $\boldsymbol{\phi}_A$ vector. With these two vectors defined and substituting the number of calculation points used, *n*, for N_0 in the equation below, the MAC correlation value between the two curves is evaluated.

$$MAC = \frac{\left|\sum_{q=1}^{N_0} \phi_{Aq}\phi_{Rq}\right|^2}{\sum_{q=1}^{N_0} \phi_{Aq}\phi_{Aq} \sum_{q=1}^{N_0} \phi_{Rq}\phi_{Rq}} \tag{3-8}$$

3.5.2 Comparing Surfaces

When working with NURBS surfaces, the comparison is much like that for curves, only it is performed using two parameter values as opposed to a single parameter value. Both the NURBS template surface and current results surface are created using the same method

described in Section 3.2.5. Both surfaces are discretized at user-defined intervals in both the U and V parameter directions resulting in a grid of points across the surfaces spaced at equal parameter intervals. Each NURBS surface is divided into *n*-1 equal parameter intervals in the U direction and *m*-1 equal parameter intervals in the V direction, resulting in *r* total points across the surface, where *r* is defined by the product of *n* and *m*. The template surface is evaluated for each of the *r* parameter values, where the set of all evaluated points makes the reference vector, $\boldsymbol{\phi}_R$. The curve created from the current analysis results is likewise evaluated at the same parameter values, defining the $\boldsymbol{\phi}_A$ vector. With these two vectors defined and substituting the number of calculation points over the surface, *r*, for N_0 in the MAC, a correlation value between the two surfaces is evaluated.

3.6 **Identify Mode Shape**

Section 3.5 has described how the MAC correlation value is calculated between various types of NURBS geometries. The NURBS geometry created from the current analysis results is compared to a set of *s* user-defined templates of known mode shape, where *s* is the number of number of templated mode shapes that the current results will be compared to. This results in a set of s correlation values. The maximum correlation value of set indicates which template matches the current results most closely. If this maximum value is above the match threshold then the matching mode shape is reported back to the optimization loop; otherwise a value is reported back to the optimization loop indicating that no matching mode shape was found among the user-specified templates. The matching threshold should be high enough that false-positive matches do not occur, but low enough to allow for slight discrepancies in the data representing the same mode. An exact threshold to use in any given case is not easy to define for the reasons described in [10], [11].

4 IMPLEMENTATION

The methods which have been described in Chapter 3 were implemented in computer programs for the automated mode identification application. A program to create templates for results of known mode shape and a program to compare templates to other modal results were created. Because this research investigates using both NURBS curves and surfaces as a means of identifying the mode shape of modal analysis results, each of the programs was implemented once for curves and once for surfaces, making a total of four computer programs.

The applications written for this research integrate the Siemens NX software API and the CAD independent General Surface NURBS Library (GSNLib) which is distributed by Solid Modeling Solutions Inc. Both of these commercially available products are based in the C/C++ programming language which allows their functionality to be easily combined and leveraged in the applications developed. An additional programming language used was the ANSYS Parametric Design Language which was used to automate the modeling, meshing, analysis, and results reporting for testing. All of the modal analysis completed for this research was done using ANSYS 11.0.

The applications are also integrated into optimization software to enable a user to easily use their functionality. SIMULIA's Isight optimization software is used in this research because custom components can be easily created using the Java programming language. The custom

components developed are used alongside native Isight components and parameters to set up and run the optimization loops.

4.1 Gather Information From User

A Java application was developed to allow the user to easily configure and run the application within the Isight optimization software. The application creates a graphical user interface (GUI) that allows the user to configure the mode identification application as well as create templates. Both of these GUI's are described in this section.

4.1.1 Mode Identification Application GUI

The mode identification application GUI provides input fields, buttons, and scroll boxes for the user to specify all files and other items required to run the application. The inputs required by this application are a template file name, the type of geometry used as templates, and the number of points to use in the MAC calculation in each parameter direction. The GUI developed for this application is displayed in Figure 4-1.

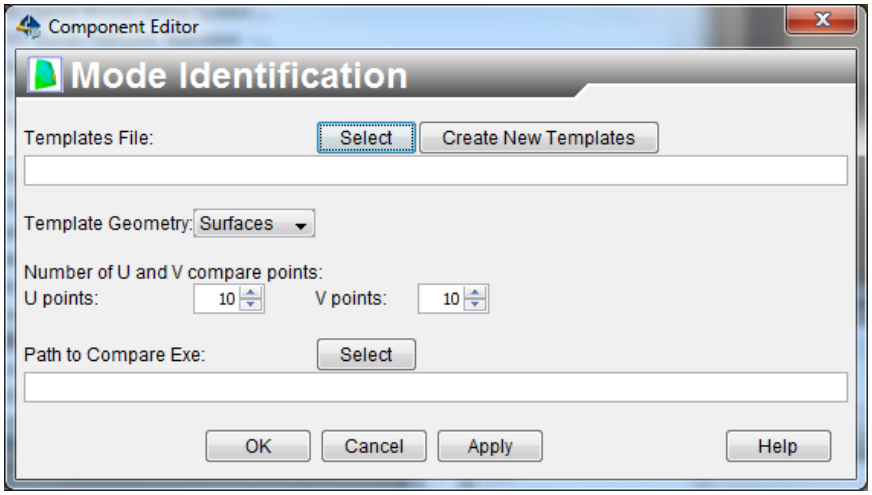

Figure 4-1 The mode identification application GUI

The template file name can be specified in the “Templates File” text field by typing the name in manually, accessing a file selection dialog through the “Select” button, or a new template can be created and input into the text field through the “Create New Templates” button. In addition to setting the template file name, the “Create New Templates” button also launched the Create Templates GUI which is described later. If curves are specified as the geometry to be used, only the number of points in the U parameter direction is input. If surfaces are specified as in the previous figure, then a number of points in both the U and V parameter directions are required. Selection of the template geometry type controls whether only the U parameter scroll box or both the U and V parameter control points scroll boxes are visible. The path to the mode identification executable is specified in the “Path to Compare Exe” text field manually or with the “Select” button. The “Help” button displays information about the application.

4.1.2 Create Templates GUI

The create templates GUI provides input fields for all information required to create new templates for known mode shapes. The inputs required for this application to execute are a name for the new templates file, the displacement file representing the results of the known mode shapes, a directory containing images of the mode shapes, and a path to the executable for the create templates application. Figure 4-2 shows the GUI for the create templates application.

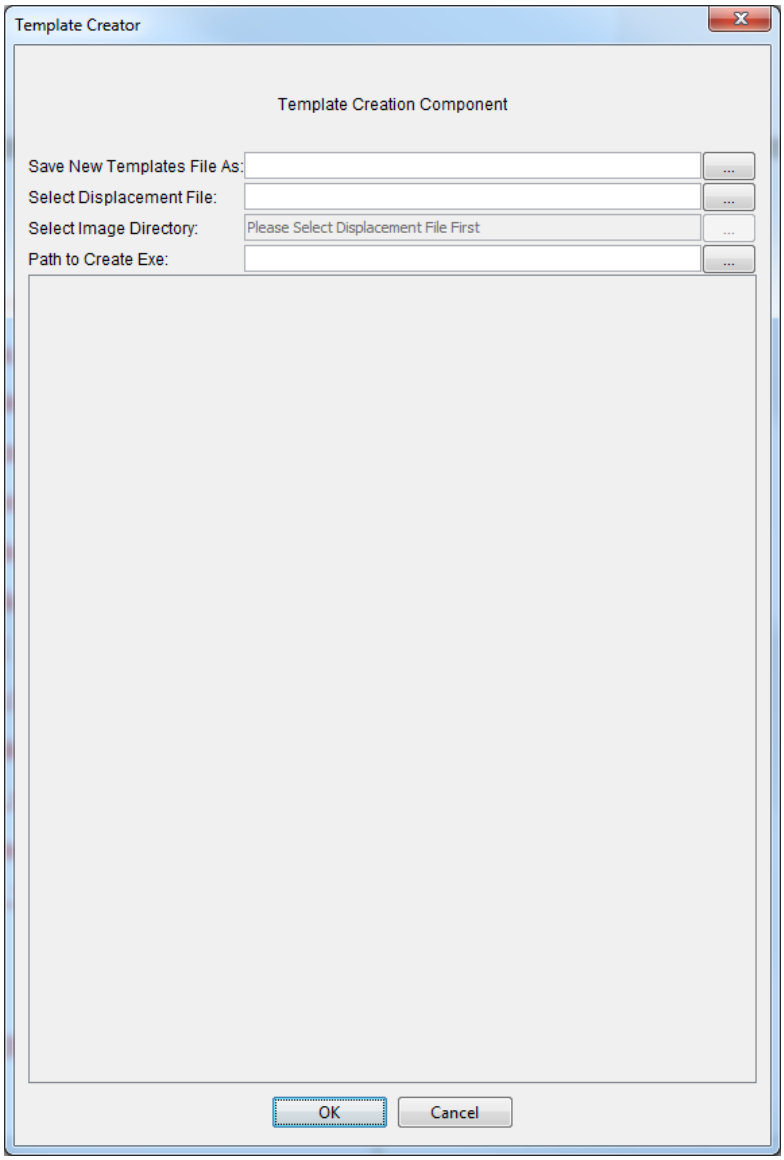

Figure 4-2 The initial create templates GUI

The new templates file name is input in the “Save New templates File As” text field by typing in the field or selecting the “…” box, which initializes a file selection dialog. The “Select Displacement File” text field stores the file name of the modal analysis results file. After a displacement file has been selected, “Select Image Directory” field and button are enabled and the user must select or specify a directory containing images of the mode shapes contained in the

displacement file which has been specified. Once this directory is specified, the images are loaded into the GUI as shown by Figure 4-3.

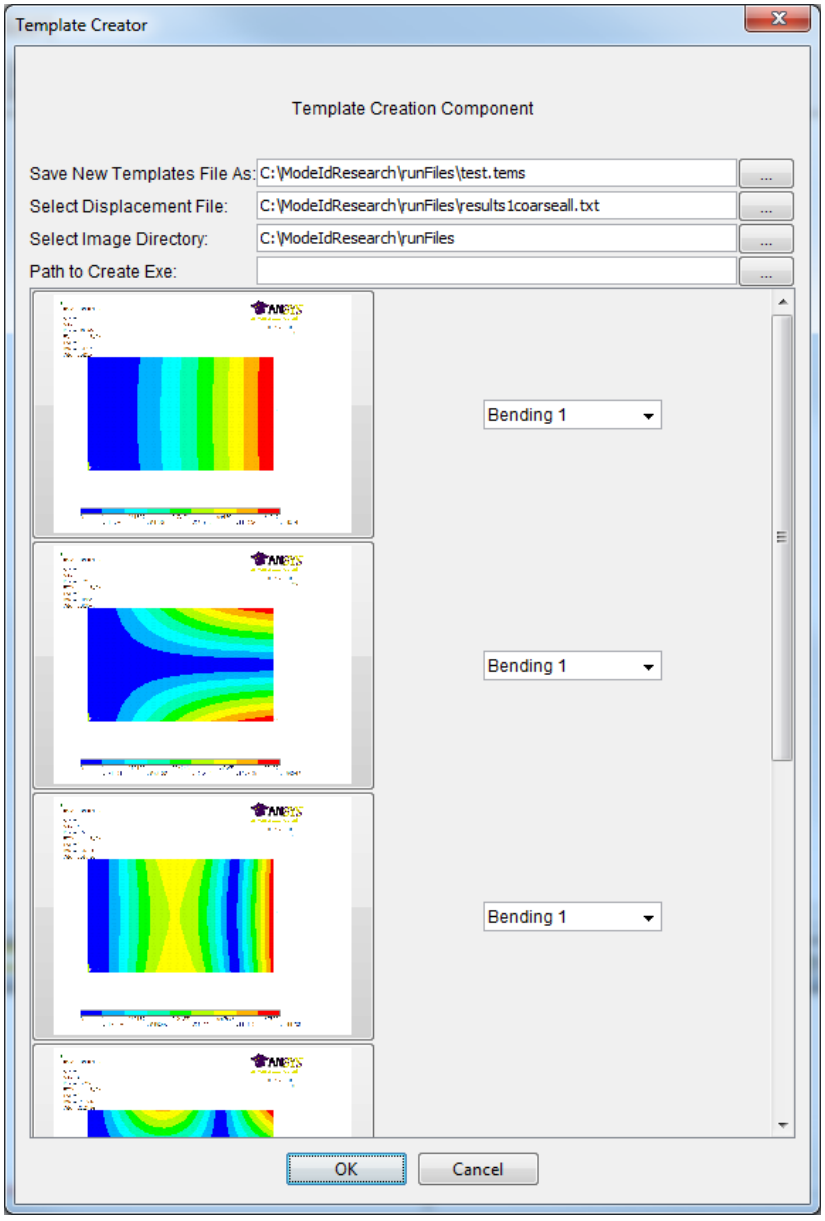

Figure 4-3 The create templates GUI with mode shape images

Each image in the directory is loaded in alphabetical order and displayed next to a drop down box for selecting or entering the mode shape that the image represents. Since they are

loaded alphabetically, the images in the directory must be named in alphabetical order beginning with the first mode shape and proceeding in the order of the solution results. This ensures that the mode shapes represented in the displacement file are accurately displayed in the GUI. Clicking the mouse on the image toggles it between a small and large image. The user must select the mode shape represented by the image in the drop down menu beside the image. If the mode shape is not in the menu, then the user may input the mode name manually into the menu. After this is completed, the user selects the "OK" button, which creates the new templates file, exits out of the GUI, and automatically puts the new templates file name into the "Templates File" field of the mode identification GUI.

4.2 **Transform Data into NURBS**

Transforming the displacement data contained in the modal analysis results file into NURBS geometry has several steps which are detailed below. Section 4.2.1 describes the reading of the displacements file and the sequencing of the nodes in the model. The normalization of nodal displacement data is described in Section 4.2.2. Section 4.2.3 describes the node location normalization which is required only when working with NURBS curves. The transformation of the data into NURBS curves and surfaces is discussed in Sections 4.2.4 and 4.2.5 respectively.

4.2.1 Read Displacement File and Determine Node Sequence

The displacement file specified by the user must be read and stored by the program. The reading of the modal analysis displacement results is done simply by storing all lines of the file in a std::vector of std::strings, then a basic tokenizer is used to extract the node positions and displacements. Since this is a relatively simple process, the code used to do this is not presented here.

Once the data is stored in the application, the node sequence is determined. The node sequence and corner nodes are required for creating a NURBS curve template since the template represents the displacement around the model boundary and the nodal solutions are reported in ascending numerical order, not necessarily in geometric order. In order to determine a corner node between two edges, a nested loop is created. Both loops iterate through all of the nodes on one of two adjacent edges. Two edges will contain a single similar node number, so Looping through each edge and finding the similar node number identifies the common node at the corner of the two edges. The same process is used to determine all corner nodes. The process for finding the corner node between the base and leading edge of the part is shown below.

```
for(int i=0; i<(int)baseNodes.size(); i++)
{
	for(int j=0; j<(int)leadingEdgeNodes.size(); j++)
	{
		if(baseNodes[i].number==leadingEdgeNodes[j].number)
			baseLeadNode = baseNodes[i];
	}
}
```

Once the corner nodes are identified, the node sequence around the edge is determined. This is done by beginning at one corner node on a single edge. Once the starting corner node is set as the first node in the sequence, the next node in the sequence is found by determining the node that is farthest away from the second corner node on the same edge. If this node has not yet been used, it is added to the ordered nodes std::vector. This process is repeated for each edge of the part. Again, nested loops are used, but each loop iterates through the same set of edge nodes, calculating the distance between the current node and the second corner node per iteration. After the distance is found, the node is added to the ordered nodes set if it is the next node in the

sequence. The process of ordering the nodes of a single edge is shown in the following sample of code.

```
leadingEdgeNodesOrdered.push_back(baseLeadNode);
for(int i=1; i<(int)leadingEdgeNodes.size(); i++)
{
	maxDist = -1.0;
	maxNodeNum = 0;
	for(int k=0; k<(int)leadingEdgeNodes.size(); k++)
	{
		double dist = leadTipNode.GetDistance(leadingEdgeNodes[k]);
		if(dist>maxDist)
		{
			int used = 0;
			for(int j=0; j<(int)leadingEdgeNodesOrdered.size(); j++)
			{
				if(leadingEdgeNodes[k].number==
					leadingEdgeNodesOrdered[j].number
				)
					used = 1;
			}
			if(used==0)
			{
				maxDist = dist;
				maxNodeNum = k;
			}
		}
	}
	leadingEdgeNodesOrdered.push_back(leadingEdgeNodes[maxNodeNum]);
}
```

After the sequence on each edge has been determined, the sequence around the entire part is known and can be used to help define the NURBS geometry. Figure 4-4 shows the sequence of the nodes around the model's perimeter, beginning at the common node between the base node set and leading edge node set.

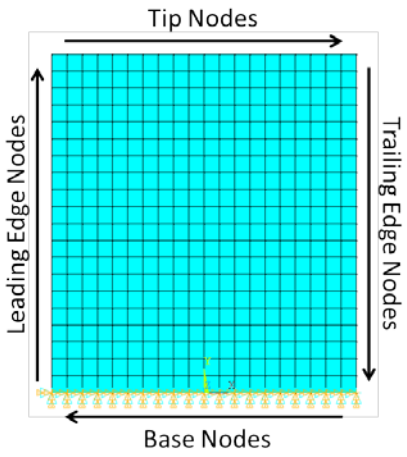

Figure 4-4 Node sequencing, beginning at the Base-Leading Edge common node

4.2.2 Normalize Nodal Displacements

Since parts exhibiting the same vibrational mode shape will have different actual displacements at each node based on the part's geometry, the actual nodal displacement at each node must not be used. The magnitudes of the nodal displacements are normalized based on the maximum nodal displacement magnitude so that the normalized nodal displacements lie between negative and positive one. The normalization of nodal displacements is done as described in Section 3.2.2 After determining the largest displacement magnitude in the results set, all displacements are normalized as shown below:

```
// Normalize the Displacements of Nodes Based on Maximum Displacement
if(usumMax!=0.0)
{
	// Normalize the Displacements of Nodes Based on Maximum Displacement
	for(int i=0; i<(int)surfaceNodes.size(); i++)
	{
		surfaceNodes[i].usum = surfaceNodes[i].usum/usumMax;
	}

}
```

4.2.3 Normalize Node Locations

When converting the modal analysis data into NURBS curves, additional node location normalization must take place. This location normalization computes a value between zero and one for each node along an edge. Since there are four edges, this process is done four times, once of each edge, and every node on each edge is given a normalized location value. This value is equal to the length from the first corner node to the current node divided by the total distance between the edge's corner nodes. The C++ implementation is as shown below, where the node's "sval" is the normalized location along the edge. After the total edge length is determined, the edge's nodes are looped through, executing the normalization at each node in the set.

```
double sval;
double length = 0.0;
double totlength = leadingEdgeNodes.front().GetDistance(leadingEdgeNodes.back());
for(int i=0; i<(int)leadingEdgeNodes.size(); i++)
{
	length = leadingEdgeNodes[0].GetDistance(leadingEdgeNodes[i]);
	sval = length / totlength;
	leadingEdgeNodes[i].sval = sval;
}
```

4.2.4 Transform Data into NURBS Curve

Once the items in Section 4.2.1-4.2.3 have been done, the data may be transformed into a NURBS curve. The transformation is done by creating a point set from the normalized location and displacement for each node in sequence around the border of the part. The GSNLib library being used requires three-dimensional points to be used. Since each node has only two dimensional information (a normalized location value and a normalized displacement) the third dimension of each point is set to zero. Additionally, since the nodes along each edge have a normalized location value between zero and one, a total normalized border value is calculated

and used. Each edge of the part constitutes one fourth of the border. The normalized boundary value for all nodes on the first edge of the part is the same as the node edge location value (described in Section 4.2.3). The normalized boundary value for all nodes on the second edge of the part is equal to one plus the node edge location value. The normalized boundary value for all nodes on the third edge of the part is equal to two plus the node edge location value, and so forth. The lowest normalized boundary value will be 0.0 at the corner of the starting edge and the highest will be 4.0 at the final corner of the final edge. The normalized boundary value is denoted "svalNorm" in the sample below. The process of adding the first edge's normalized node location and displacement data to a point set is shown below.

```
double sval = 0.0;
double svalTot = leadingEdgeNodes.back().sval+tipNodes.back().sval
       +trailingEdgeNodes.back().sval+baseNodes.back().sval;
double svalNorm = 0.0;
double currentSval = 0.0;

// Add points from leading edge to the point set
for(int i=0; i<(int)leadingEdgeNodes.size(); i++)
{
       sval = currentSval + leadingEdgeNodes[i].sval;
       svalNorm = sval;// / svalTot;
       IwPoint3d myPoint;
       myPoint.x = svalNorm;
       myPoint.y = leadingEdgeNodes[i].usum;
       myPoint.z = 0.0;
       pointSetUsum.Add(myPoint);
}
leadingEdgeNodes.clear();
currentSval = sval;
```

Once the point data has been gathered for all edges, a GSNLib library function called IwBSplineCurve::InterpolatePoints creates the actual NURBS curve. This process is simple and is shown in the following code sample. It is also depicted in Figure 4-5.

```
// Create the Usum NURBS curve (degree 3) interpolating the points
IwBSplineCurve::InterpolatePoints(crContext, pointSetUsum, NULL, 3, NULL, NULL, FALSE,
IW_IT_UNIFORM, profileCurveUsum);
```

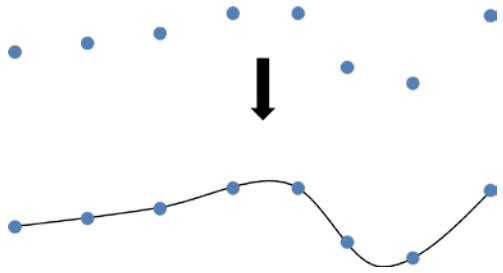

Figure 4-5 Transformation of node data into a NURBS curve

4.2.5 Transform Data into NURBS Surface

In practice, transforming data into a NURBS surface is significantly more difficult than transforming data into a curve. The first step in this process is to create a point set representing the original nodal positions in the model analyzed. This point set is then used to create a parametric surface representing the original (not displaced) model geometry, which is referred to as "Surface 1". This is done using NX API and GSNLib function calls in the following manner:

```
std::vector <IwPoint3d> set1;
IwPoint3d point;
for(int i=0; i<(int)surfaceNodes.size(); i++)
{
	point.x = surfaceNodes[i].xPos;
	point.y = surfaceNodes[i].yPos;
	point.z = surfaceNodes[i].zPos;
	set1.push_back(point);
}

tag_t surf1 = fitSurfaceNX(set1,0);
IwBSplineSurface *surface1;
surface1 = Converter::ConvertNURBSSurface(surf1,crContext);
surface1 = reparameterizeSurface(surface1);
```

The call to the function fitSurfaceNX() simply transfers the data stored in a std::vector into an array and, using the corner points found in Section 4.2.1, calls an NX Open API function

that creates a parameterized surface inside of NX from a point cloud. Figure 4-6 shows a representation of this surface fitting process.

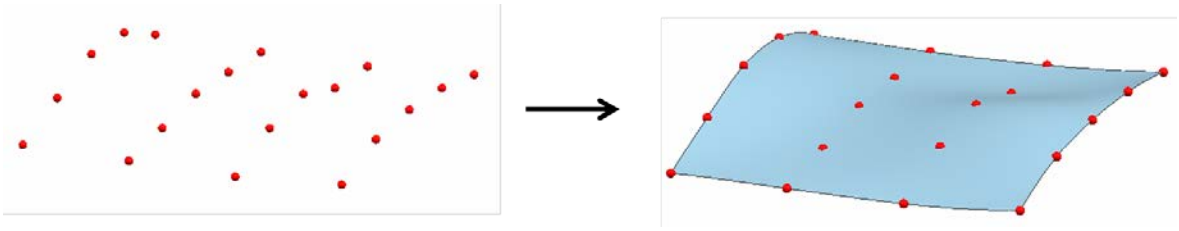

Figure 4-6 Transformation of node data into a NURBS surface

The created surface is then reparameterized in order to insure that the parameterization is uniform over the surface in both directions. The surface created in NX exists in the CAD program, so in order to extract the surface and store it using GSNLib data structures, a conversion function is used. The conversion function is not shown here but it is done as described in Section 3.2.5 where the knot vectors and control points in each parameter direction are copied from the CAD data structure into the CAD independent (GSNLib) data structure and an identical surface external to CAD is created from the data.

Once this has occurred, Surface 1 has been created, representing the original geometry of the analyzed model. Since Surface 1 was interpolated from the nodes in the model, each node exists on or near Surface 1. The next step is to determine the U-parameter and V-parameter of each node from the model. This is done with a call to the GSNLib library function GlobalPointSolve(). The U and V surface parameters perfectly define the position of the node in parameter space between zero and one in each case, forming an intermediate two dimensional square plate in parameter space, as illustrated in Figure 4-7.

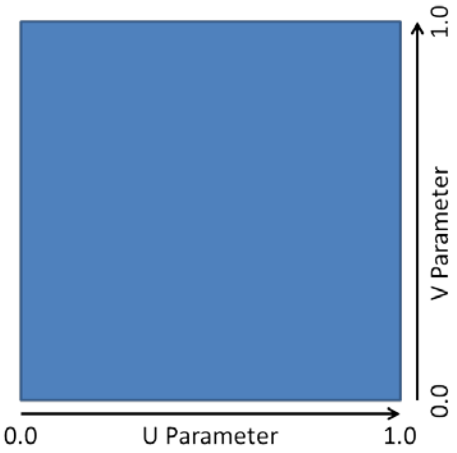

Figure 4-7 Intermediate square template in parameter space

A new point set is created next, describing the template surface. Instead of using nodal X, Y, and Z locations like Surface 1 does, the U-parameter, V-parameter, and normalized nodal displacement are used for the three dimensions of each point in the set. Each U and V parameter is scaled by a factor of ten in order to make the template surface larger and easier to fit with a NURBS surface. The process of finding each node's scaled U and V parameter, creating a new point using the scaled U and V parameters and normalized displacement, and adding the point to the point set is shown in the code sample below.

```
std::vector <IwPoint3d> setusum;
for(int i=0; i<(int)surfaceNodes.size(); i++)
{
	IwPoint3d myPoint;
	myPoint.x = surfaceNodes[i].xPos;
	myPoint.y = surfaceNodes[i].yPos;
	myPoint.z = surfaceNodes[i].zPos;

	IwPoint2d uvGuess;
	uvGuess.x = 0.0;
	uvGuess.y = 0.0;

	IwSolution solution;
	IwSolutionArray solutions;
	IwExtent2d domain = surface1->GetNaturalUVDomain();
	surface1->GlobalPointSolve(domain,IW_SO_MINIMIZE,myPoint,
		0.001,NULL,IW_SR_SINGLE,solutions);
```

```
        uvGuess.Set(solutions.GetAt(0).m_vStart.m_adParameters[0],
               solutions.GetAt(0).m_vStart.m_adParameters[1]);
        myPoint.Set(10.0*uvGuess.x,10.0*uvGuess.y,surfaceNodes[i].usum);
        setusum.push_back(myPoint);
}
```

When this point set is fully constructed, the fitSurfaceNX(), ConvertNURBSSurface(), and reparameterizeSurface() functions, which have been described previously in this section, are called again to fit a surface to the new point set. The new surface is the template surface and it is stored in an array for future use.

This process is repeated for each modal solution in the displacement results file supplied by the user. Each time, the square parametric plate described previously has nodal displacement data added for each node in the model. These two items allow the template surface to be created. This process is depicted simply in Figure 4-8. Each template surface is then temporarily stored in an array for future use.

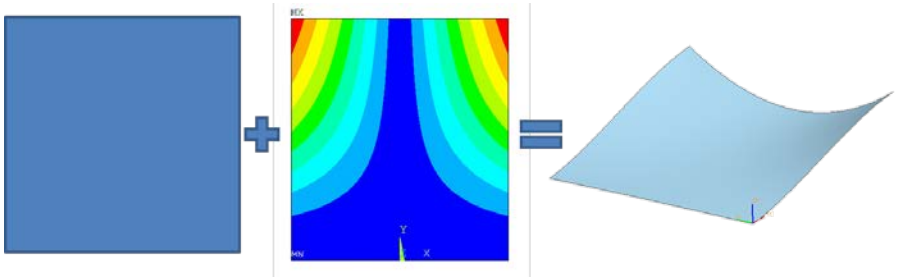

Figure 4-8 The intermediate template and displacement data create the final template

4.3 Store Template

After the template surface has been created for all known mode shapes in the displacement results file, they may be stored in a library or database to be used at a later date. This is done programmatically using another GSNLib library function call. The WriteToFile()

function is defined for both NURBS curves and surfaces in the GSNLib library. This function takes an array of NURBS objects and writes them to a file with the path that is specified in the function call. For surfaces, the function call is shown below:

```
// Write Each Surface Template in the Templates Array
IwBSplineSurface::WriteToFile(path,templatesArray);
```

The surfaces in the array are saved in the user-specified file and the final step is to write the mode shape names supplied by the user in the Create Templates GUI (Section 4.1.2) into the bottom of the same templates file. The mode names are written at the bottom of the file because in order for GSNLib to properly read in the NURBS geometry at a future time, the data in the file must not be disturbed in any way. The easiest way to avoid disturbing the data is to simply write the names of the mode shapes in the order the user input them at the end of the templates file created.

4.4 Load Template of Known Mode Shape

Loading the templates of known mode shape takes place in two steps. In the first step, the template surfaces are read from the file in which they exist and are stored in GSNLib data structures. This is relatively straight forward and is demonstrated in the readTemplateSurfaces() function.

```
// Function to Read NURBS Template Surfaces From Templates File
int surfaceTemplate::readTemplateSurfaces(char* file)
{
	IwContext crContext;
	if(IW_SUCCESS != IwBSplineSurface::ReadFromFile(crContext,file,templatesArray))
	{
		std::cerr<<"NURBS Template Surfaces Not Read Successfully.";
		return 1;
	}
	numTemplates = templatesArray.GetSize();
```

```
	return 0;
}
```

The second step is that the same file is parsed in order to determine the names of the mode shapes that each surface represents. The following readModeNames() function shows this very simply.

```
// Function to Read the Mode Shape Names From the Bottom of the Templates File
int surfaceTemplate::readModeNames(char* file)
{
	std::vector <std::string> contents;
	infile.open(file);
	readDataFile(infile,contents);
	infile.close();
	infile.clear();

	int position = 0;
	int store = 0;
	for(int i=0; i<(int)contents.size(); i++)
	{
		if(store==1 && position<numModes)
		{
			modeNames.push_back(contents[i]);
			position++;
		}
		if(contents[i]=="Mode Names")
			store = 1;
	}
	return 0;
}
```

4.5 **MAC Calculation**

The MAC calculation can be carried out programmatically after two NURBS geometries exist in the computer's memory. The calculation is similar whether using NURBS curves or surfaces, and the implementation for surfaces will be detailed here.

As discussed in Section 2.2, the MAC calculation is as follows:

$$MAC = \frac{\left|\sum_{q=1}^{N_0} \phi_{Aq}\phi_{Rq}\right|^2}{\sum_{q=1}^{N_0} \phi_{Aq}\phi_{Aq} \sum_{q=1}^{N_0} \phi_{Rq}\phi_{Rq}} . \quad (4\text{-}1)$$

.

When working with surfaces, each nodal vector ($\boldsymbol{\phi}_{Aq}$, $\boldsymbol{\phi}_{Rq}$) have N_0 elements composing them, where N_0 is equal to the number of points in the U-direction multiplied by the number of points in the V-direction. The number of points in each parametric direction is specified by the user as described in Section 4.1.1. The reference nodal vector, $\boldsymbol{\phi}_R$, is the set of all points evaluated on the reference surface of known mode shape, which is loaded as described in Section 4.4. The actual nodal vector consists of all points evaluated on the surface created from the current displacement results file, which creation is described in Section 4.2.5.

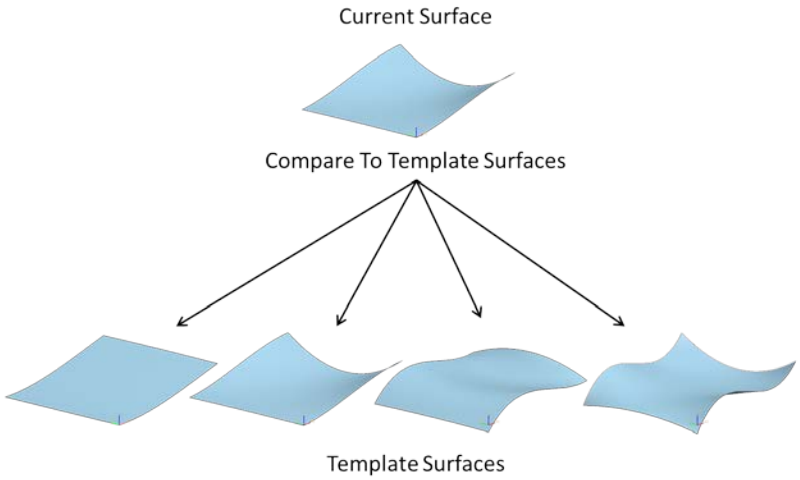

Figure 4-9 The current results surface is compared to all loaded template surfaces

Each surface created from the current results file is compared with all of the template surfaces of known mode shape specified in the templates file as shown in Figure 4-9. The result of each comparison is an integer specifying which template surface most closely matches the

current results surface. A final result where the variable "matchNum" is equal to 9999 indicates that no matching mode shape was found. This process is seen in the code sample below:

```
// Function to Compare Surfaces to a Set of Template Surfaces Using the MAC
int surfaceTemplate::compareSurfaceSet(int uPts, int vPts)
{
	double rMax = 0;
	int matchNum = NULL;
	// Loop Through all Templates (4 at a time)
	for(int i=0; i<numTemplates; i++)
	{
		int t1(4*i), t2(4*i+1), t3(4*i+2), t4(4*i+3);
		double r1(0);

		// Compare the Usum Profile and Template Surfaces
		r1 = compareSurfaces(templatesArray[t1],profileSurfaceUsum,uPts,vPts);

		// Average the Results From All 4 Calculations
		double rAvg = r1;
		// Save the Maximum MAC Result (Profile to Template)
		if(i==0)
		{
			matchNum = 9999;
			rMax = 0.0;
		}
		if(rAvg>0.65 && rAvg>rMax)
		{
			matchNum = i;
			rMax = rAvg;
		}
	}
	matchValue = rMax;
	return matchNum;
}
```

The compareSurfaces() function in the sample above carries out the actual MAC calculation. The function takes a template surface, a current results surface, and the number of points in each parameter direction to use when evaluating the MAC as inputs. The result returned by this function is the calculated MAC correlation value. This function is shown below.

```
// Function to Compute MAC Calculation Between Two Surfaces
double surfaceTemplate::compareSurfaces(IwBSplineSurface* tem, IwBSplineSurface* pro,
int numU, int numV)
{
	double numerator(0), denom1(0), denom2(0);
	double numeratorN(0), denomN1(0), denomN2(0);
	// Loop Controlling U Parameter
```

```
	for(int i=0; i<numU; i++)
	{
		IwPoint3d temPoint, proPoint, proPointNeg;
		IwVector2d uv;
		uv.x = (double)i/(double)(numU-1);
		// Loop Controlling V Parameter
		for(int j=0; j<numV; j++)
		{
			uv.y = (double)j/(double)(numV-1);
			// Get the Displacements From the Template and Profile Surfaces
			tem->EvaluatePoint(uv,temPoint);
			pro->EvaluatePoint(uv,proPoint);
			proPointNeg.Set(proPoint.x,proPoint.y,-proPoint.z);
			// Sum of disp1*disp2
			numerator = numerator + (temPoint.z*proPoint.z);
			numeratorN = numeratorN + (temPoint.z*proPointNeg.z);
			// Sum of disp1*disp1
			denom1 = denom1 + (temPoint.z*temPoint.z);
			denomN1 = denomN1 + (temPoint.z*temPoint.z);
			// Sum of disp2*disp2
			denom2 = denom2 + (proPoint.z*proPoint.z);
			denomN2 = denomN2 + (proPointNeg.z*proPointNeg.z);
		}
	}
	// MAC Calculation
	double result1 = (numerator*numerator) / (denom1*denom2);
	double result2 = (numeratorN*numeratorN) / (denomN1*denomN2);

	if(result1>result2)
		return result1;
	else
		return result2;
}
```

4.6 **Identify Mode Shape**

The mode shape identification has been partially described in Section 4.5. As described, the compareSurfaceSet() function returns an integer value indicating the closest matching template mode shape. After this integer value has been identified, the corresponding mode shape name and the frequency associated with the shape are stored in a std::vector of std::pairs called "matchedPairs". The MAC value associated with the match is also stored in a std::vector for later use. This is shown below:

```
void surfaceTemplate::setResultSet(int modeNum, double freq)
{
	std::pair <std::string,double> newPair;
```

```
	if(modeNum!=9999)
		newPair.first = modeNames[modeNum];
	else
		newPair.first = "No Match";
	newPair.second = freq;
	matchedPairs.push_back(newPair);
	matchedValues.push_back(matchValue);
}
```

After all of the matches have been identified, the results are written to a results file where they can be transferred back into Isight for the optimization to use.

```
int surfaceTemplate::writeModeResults(char* file)
{
	outfile.open(file);
	for(int i=0; i<(int)matchedPairs.size(); i++)
	{
		outfile << matchedPairs[i].second << "," << matchedPairs[i].first
			<< "," << matchedValues[i];
		outfile << std::endl;
	}

	outfile.close();
	outfile.clear();

	return 0;
}
```

The results are read back into Isight using the Java application that defines the GUIs described previously. The results file is read and an Isight parameter corresponding to each known mode shape is created. The frequency associated with each known mode shape is stored in these parameters. Basic Java file reading capabilities read the mode shape, frequency, and match value from the results file. The frequency can then be stored in the Isight parameter using the following line:

```
a.getScalarByTag(matchNames[j]).getValueObj().setValue(matchFreqs[j]);
```

After the parameters are set in Isight, the optimization can use them however the user has specified they be used. For every iteration of the optimization, if the mode shape is matched to the results, its frequency is assigned into the parameter.

5 RESULTS

The purpose of this research, as discussed in Chapter 1, is to develop a method by which the mode shapes of a finite element modal analysis may be automatically identified based on the nodal displacements and previously stored templates of known mode shapes. The method uses parameterized geometric representations of the vibrational mode shapes which allow two parts of varied geometry or mesh density to be compared through the Modal Assurance Criterion. Specifically, this method makes it possible for the frequencies of specific mode shapes to be identified at every step of an iterative design process, such as an optimization or design of experiments.

Section 5.1 shows the results of using the method to compare the mode shapes of two geometrically identical models which have been meshed with different levels of detail. Section 5.2 details the efficacy of the method in comparing parts with different geometrical definition, but still of the same general size and shape. Section 5.3 shows the results of embedding the method into an iterative design process.

5.1 Mode Identification – Differing Mesh Densities

Vibrational mode shapes for models which are identical in geometric definition but with differing mesh densities have been successfully identified using the method described in the previous chapters. This has been accomplished using both parametric curve template

representations and parametric surface template representations. Two models which have differing mesh densities, and thus have a different number of nodal displacements in their modal shape vectors, and which have been compared using the method developed are shown in Figure 5-1. Although it appears that these parts are simply two dimensional tapered plates, they actually vary in all three dimensions.

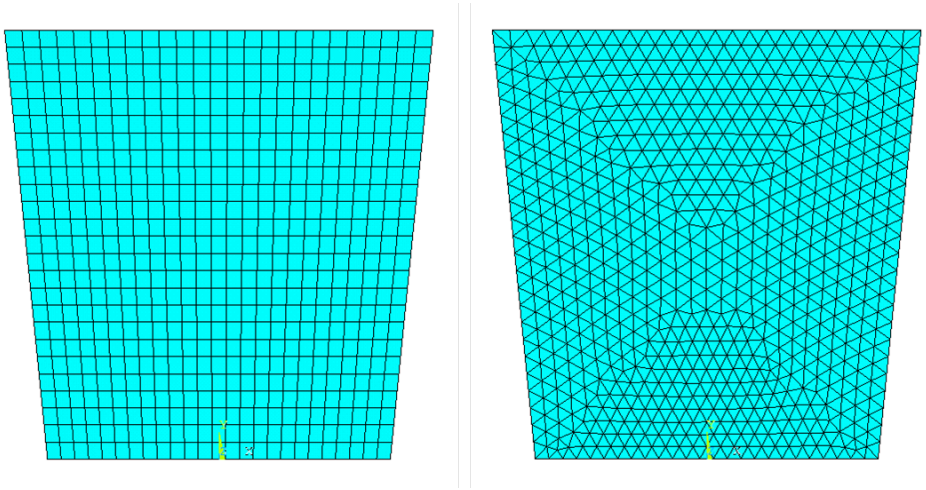

Figure 5-1 Two geometrically identical models meshed differently

As expected, because the models shown in Figure 5-1 are identical except for the mesh, the mode shapes determined from a modal analysis are identical. The modal analysis results from the model on the left of Figure 5-1 were used to create two mode shape templates, one using curves and one using surfaces. The modal analysis results from the model on the right were then compared to those templates using the method described previously. This research investigated the method's capabilities using three different model definitions. The first is a simple rectangular plate that is 10 inches long, 8 inches wide, and 0.5 inches thick. The second is a tapered and twisted linear plate which is 10 inches long, 8 inches wide at the base, 10 inches

wide at the tip, 0.5 inches thick, and has corners which are displaced an inch out of plane in opposite directions at the tip. The corner nodes at the base and tip are connected by a line of nodes. The third test geometry is basically the same as the tapered twisted linear plate just described, except that the corner nodes on the base and tip are connected via nodes along a non-linear spline curve. All models are meshed using two-dimensional elements. Each of the test geometries described above is shown in Figure 5-2.

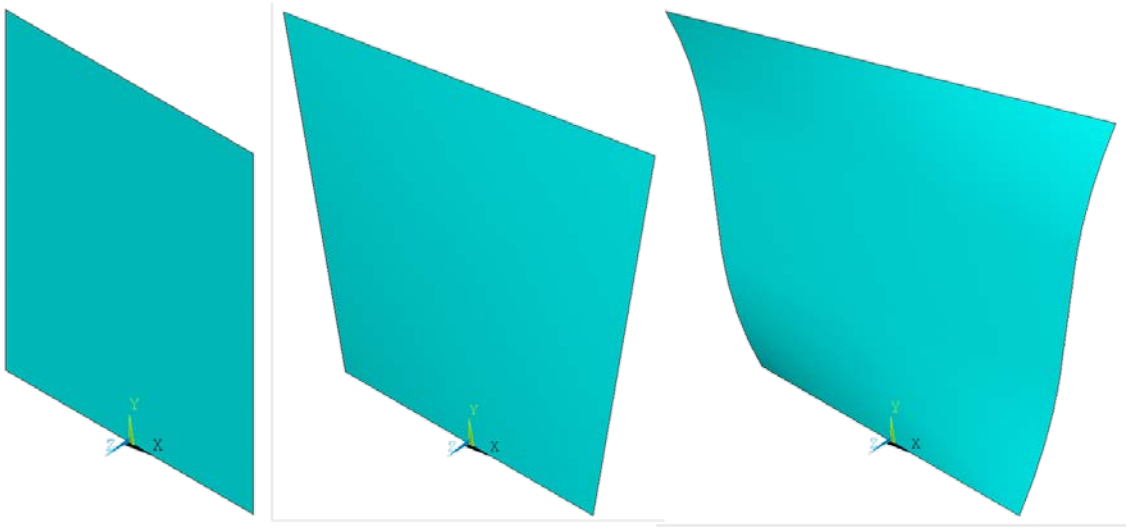

Figure 5-2 Isometric views of each test geometry

Each of these model definitions was meshed twice, once using quad elements and once using tria elements. Curve and surface templates were created from the model meshed with quad elements. Then the model with tria elements was compared to the template. This testing resulted in the correct identification of all ten mode shapes determined by the modal analysis for all of the test geometries. The MAC values calculated in the identification process using surface templates for the tapered twisted non-linear plate definition are displayed in Table 5-1.

Table 5-1 Modal Assurance Criterion values for tapered twisted non-linear plate

		Mode Number									
		1	2	3	4	5	6	7	8	9	10
Mode Number	1	1.0	0.7457	0.6948	0.9363	0.5287	0.9649	0.6380	0.5972	0.7984	0.6521
	2	0.7481	0.9999	0.5746	0.8742	0.7240	0.7964	0.6110	0.5074	0.7230	0.7064
	3	0.6913	0.5760	0.9975	0.6777	0.7890	0.7924	0.9024	0.9131	0.8750	0.6458
	4	0.9294	0.8837	0.6770	0.9995	0.6231	0.9446	0.7247	0.5966	0.7712	0.7166
	5	0.5318	0.7267	0.7973	0.6188	0.9978	0.6544	0.8142	0.8046	0.7550	0.8017
	6	0.9635	0.7958	0.7990	0.9470	0.6556	0.9999	0.7455	0.7295	0.8405	0.7647
	7	0.6303	0.6180	0.8893	0.7231	0.8070	0.7359	0.9978	0.8645	0.7948	0.6928
	8	0.5979	0.5120	0.9274	0.5960	0.8158	0.7313	0.8778	0.9996	0.8099	0.7706
	9	0.7766	0.7099	0.8869	0.7459	0.7501	0.8227	0.7948	0.8047	0.9985	0.6205
	10	0.6494	0.7051	0.6401	0.7107	0.8073	0.7588	0.6823	0.7493	0.6320	0.9993

As is shown by the high MAC values in the diagonal elements of the table, each mode is correctly identified by the method. This demonstrates that parametric surfaces and curves used to represent the modal analysis data enable an accurate comparison to be made between models which are meshed differently. This attribute of the method lends itself to use in an iterative design process because the mode shapes are still able to be identified even though each iteration of a design may yield a new mesh scheme for the model.

5.2 Mode Identification – Differing Geometric Definitions

An iterative design process modifies a model's geometry, usually by changing the parameter values that define a parametric model. The mode shapes for models which are similar

in geometric definition to a baseline design but have slightly modified geometry, due to design changes, have been successfully identified by this method. The two models being compared may have the same or a different meshing scheme. Two models, a baseline design and a modified design are shown in Figure 5-3.

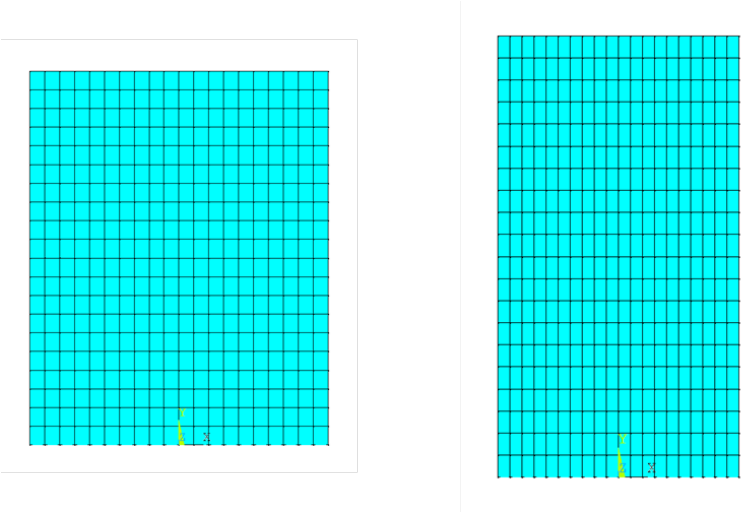

Figure 5-3 A baseline model and modified design

The same three geometric models described in Section 5.2 were used as the baseline designs to test the mode identification of models with differing geometric definition. Curve and surface templates were generated from the baseline design modal analysis results and those templates were compared to modal analysis results completed within an iterative design process. An example of a set of ten mode shapes from the modal analysis of a baseline design is shown below in Figure 5-4.

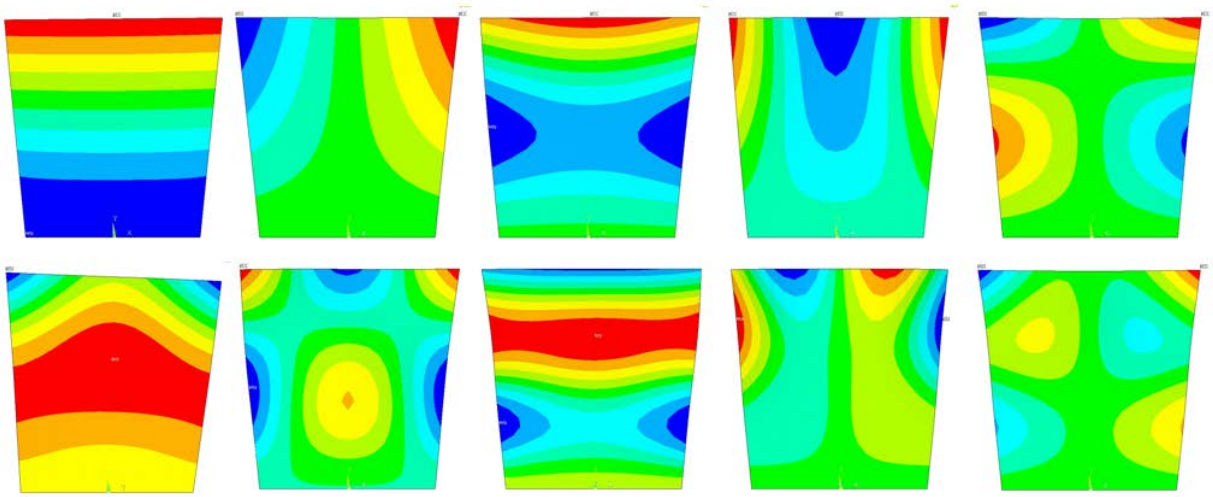

Figure 5-4 Ten sample mode shapes from the modal analysis of a baseline design

The iterative process used in this research was a Design of Experiments (DOE), which allowed the degree of design variation to be set in order to investigate the mode identification capabilities of the method through the entire design space as defined. Four DOE processes were completed for each template type and geometric definition, where each DOE run was set to vary the model parameters to a different level. The levels to which the parameters defining the geometry were varied were plus or minus 10 percent, 20 percent, 30 percent, and 40 percent.

When evaluating the accuracy of the method, the following rules were used to determine if the method was successful or not.

1. If the mode shape in the modified design matches a mode shape in the baseline design and the method correctly identifies the shape, the method is correct.
2. If the mode shape in the modified design does not match any shape in the baseline design and the method does not match it with a mode shape, the method is correct.
3. If the mode shape in the modified design matches a mode shape in the baseline design and the method does not match it with the mode shape, the method is incorrect.

4. If the mode shape in the modified design does not match any shape in the baseline design and the method matches it to a mode shape, the method is incorrect.

The design matrix for each DOE to be tested was generated by Isight and the DOE's were executed using the automated mode identification method to match any mode shapes in the modified designs to mode shapes in the templates generated from the baseline design. Each results file was visually inspected within a post processor to determine the mode shape and the automated mode shape result was compared to the visually determined shape via the rules listed above. This process produced the accuracy of the mode identification method.

The rectangular plate DOE was run and it was determined that the mode identification application accuracy is related to the amount which the model is allowed to be modified. When the model geometry parameters were allowed to change plus or minus ten percent of the baseline, the mode identification application was accurate 98 percent of the time when using curve templates and 96 percent of the time using surface templates. At plus or minus 20 percent modification, the application was correct 95 percent of the time for curves and 87 percent of the time for surfaces. When allowed to modify by plus or minus 30 percent, the method's accuracy fell to 92 and 82 percent for curves and surfaces respectively. The accuracy at plus or minus 40 percent modification of the parameters was 89 percent for curve templates and 87 percent for surface template representations.

The DOEs executed using the tapered twisted linear plate resulted in a similar relationship between the amount of variation and the method accuracy. Using NURBS curve templates, the ten percent variation DOE resulted in a method accuracy of 92 percent. With 20 percent variation in the parameters, the matches were accurate 87 percent of the time. A DOE

with 30 percent variation in the parameters resulted in 80 percent accuracy. The mode identification application was 78 percent accurate when allowing 40 percent variation in the parameters. Using surface template representation, demonstrates the same general trend of decreasing accuracy with increased parameter variation. When using NURBS surface templates, the accuracy with 10, 20, 30, and 40 percent variation in the DOE was 93, 90, 82, and 84 percent respectively.

Next, the tapered twisted non-linear plate model was used as the baseline design for the DOE. The same variation levels of plus or minus 10, 20, 30, and 40 percent were used to set up the design matrix. Upon examination of the results for curve templates, the method was 88 percent, 83 percent, 83 percent, and 80 percent accurate respectively. With surface templates, the mode identification accuracies were 91, 88, 87, and 82 percent, respectively.

Each of the template types and Design of Experiment parameter variation levels demonstrates the same general pattern that as the amount of variation from the baseline increases, the accuracy of the mode matching application decreases. These trends are shown in the results presented in Table 5-2 and Table 5-3, below. In each case, the match accuracy is above 90 percent when the variation is limited to plus or minus 10 percent. The accuracy percentage decreases in all cases into the mid to low 80's as the DOE parameter variation increases to plus or minus 30 percent. When the variation in the parameters was plus or minus 40 percent the matching accuracy increased from the 30 percent level in two cases and decreased in the last case.

Table 5-2 Accuracy of the method using curve templates

Mode Identification Method Accuracy - Curves				
	DOE Parameter Variation			
	± 10%	± 20%	± 30%	± 40%
Rectangular Plate	98%	95%	92%	89%
Tapered Twisted Linear Plate	92%	87%	80%	78%
Tapered Twisted Non-linear Plate	88%	83%	83%	78%

Table 5-3 Accuracy of the method using surface templates

Mode Identification Method Accuracy - Surfaces				
	DOE Parameter Variation			
	± 10%	± 20%	± 30%	± 40%
Rectangular Plate	96%	87%	82%	87%
Tapered Twisted Linear Plate	93%	90%	82%	84%
Tapered Twisted Non-linear Plate	91%	88%	87%	82%

In the testing, each modal analysis produced ten mode shapes and natural frequencies. By investigating which mode shapes were incorrectly matched by the application, it was also determined that the majority of the mismatches were in the higher order modes. Of all incorrect matches made by the application, over 83 percent, for curve templates, and 75 percent, for surfaces, occurred in the final four modes studied. This is due to a number of factors.

As the mode number increases, the mode shapes become increasingly complex. The function which is used to create a NURBS surface from the nodal displacements determines an approximate surface from a cloud of points. This is done using a least-squares method to find a best fit surface for the point data. This results in a surface that does not pass through each point,

and thus does not represent the mode shape as accurately as possible. It also produces template surfaces which appear to be more similar to one another than they actually are.

The incorrect matches are also partially due to the fact that only the nodal displacement magnitude was used to match the mode shapes. Because only the displacement magnitude is used instead of the magnitude and direction, dissimilar modes are more easily seen as correlated modes. When this displacement magnitude effect is combined with the NURBS surface approximation function discussed above, the effect is amplified. An example of two modes that may be easily mismatched due to this combined effect, and their respective template surface representations, are shown in Figure 5-5 and Figure 5-6, below. These effects are correctable through further refinement of the identification method which will be further discussed in Section 6.1.

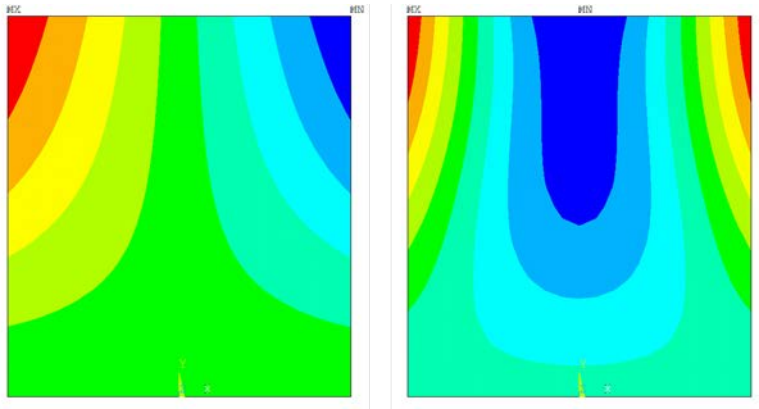

Figure 5-5 Contour plots for two easily misidentified mode shapes

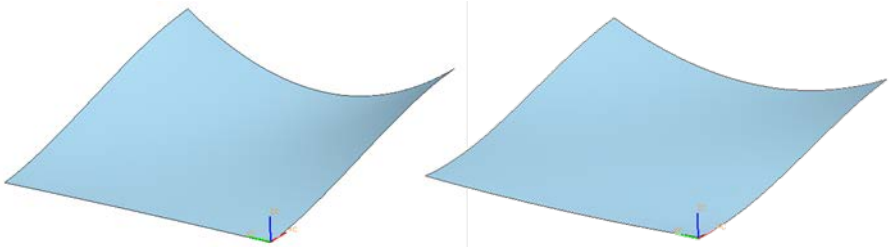

Figure 5-6 Template geometry for two easily misidentified mode shapes

Another important piece of information is that the results indicate that the surface templates are more accurately used to identify mode shapes than the curve templates are. The results indicate that this is the case for all models used except for the flat rectangular plate. The increase in match accuracy when using surface templates is because the surface templates contain shape information internal to the model as opposed to the curve templates which only stores information about the outside edge of the model. Surface creation and evaluation increases the amount of time taken to run the application, but results in greater accuracy, especially when using models with a more complex geometric definition.

Despite these inefficiencies in the method, the results show that the identification of mode shapes using parametric NURBS curves and surfaces created from the modal analysis displacement data is feasible. The results also demonstrate that the accuracy of the method when using NURBS curve templates is comparable to the accuracy when using NURBS surface templates. The advantage of using surface templates in the method is that the part's interior data factors into the comparison while curve templates simply use the part's boundary data.

5.3 **Mode Identification in Iterative Design**

Section 5.2 discussed the accuracy of the mode identification method when comparing two models of differing geometric definition. This was accomplished by executing the application within an iterative design process and inspecting the results. Figure 5-7, below, shows a sample DOE set up that was run to test the method. The DOE initializes the parameter values and passes them into the subflow which calculates other values required, creates a new modal analysis script, solves the modal analysis in ANSYS, and finally identifies the modes from the modal analysis results.

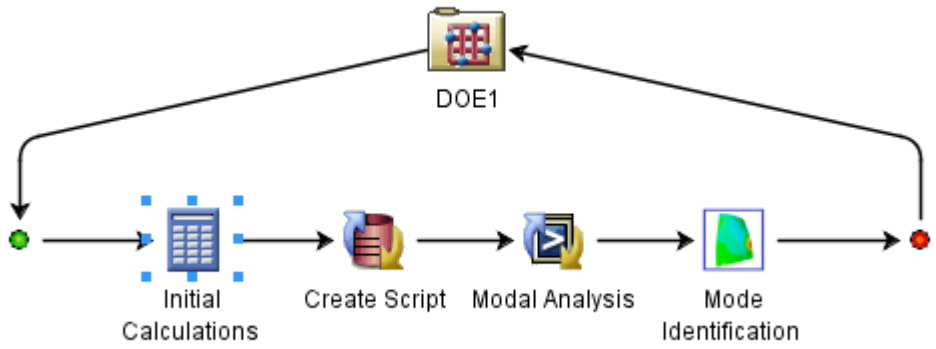

Figure 5-7 Isight Design of Experiment workflow

The capability to automatically identify the mode shapes throughout the iterative process is advantageous because it provides significant time savings to identifying the mode shapes by visual inspection of the analysis results which is the current process for mode shape identification[10].

In generating the results presented in this section, each mode shape inspection method, automated and visual, was completed 20 times. The automated mode shape identification runs identified 10 mode shapes. Visual inspections were completed by having a user view and identify the mode shape and natural frequency for ten modal solutions within the ANSYS post processor, having already loaded a previously solved modal analysis results files.

When curve templates were used by the automated application, it took an average of 0.891 seconds to match 10 mode shapes for a speed of 0.0891 seconds per match. The automated application took an average of 15.55 seconds to execute the matching application 10 times when using surface templates. This means that the algorithm was able to execute at a rate of 1.555 seconds per match. When the same modal analysis results files were investigated visually to

determine the mode shapes and natural frequencies, it was found that on average it took 121.66 seconds to identify the mode shapes for 10 modes. The rate of identification for this visual inspection is 12.166 seconds per match, which is more than 136 times the speed of the automated method when using curve templates and almost eight times more than when using surface templates. These results are presented side by side in Table 5-4.

Table 5-4 Time required by mode shape identification methods

Identification Method	Total Time (sec)	# of Matches	Time per Match (sec)
Automatic - Curves	17.82	200	0.0891
Automatic - Surfaces	311.07	200	1.55535
Visual	2433.28	200	12.1664

When visual inspection is used to identify the mode shape, the designer can be sure of what each mode shape is, while using the mode identification method introduces the possibility of incorrect matches occurring as discussed in Section 5.2. However, the significant time savings made possible through the method present a compelling argument for automation even if the identification is not always perfectly accurate. This is especially true as the designer increases the number of mode shapes that he desires to identify at each step of the iterative process which increases the amount of time that would be invested into mode shape identification.

6 CONCLUSIONS

The Modal Assurance Criterion, which has been developed over the past three decades to determine how a set of modal vectors are correlated, can be effectively used to compare and identify the mode shapes of dissimilar finite element models. This is made possible through the representation of the models' modal analysis results as parametric NURBS curves and surfaces which allows parts with different mesh attributes and geometrical definitions to be compared against previously known mode shapes in parameter space as opposed to real space. The use of surface templates is more time intensive than using curve templates. The accuracy of the method using curve templates is comparable to when using surface templates.

The interpolate points function in GSNLib and the surface approximation function in the NX Open C API make it possible to use the nodal position and displacement data to create a parameterized curve or surface representing a specific mode shape which can then be stored to a database using a GSNLib data structure. Once the finite element model and results data from two models is transferred into parameterized geometric form, it is simple to compare them one to another at user-defined intervals using the Modal Assurance Criterion. The MAC calculation is then able to provide the correlation value that identifies matching mode shapes between models of different mesh density and geometric definition.

The automated mode identification method provides a fast and simple way to gain a more detailed understanding of a part's dynamic properties without having to visually inspect each

modal solution within a finite element post processor. The real benefit of this is when iterative design processes, such as optimization or design of experiment, are being utilized. Since the iterative design methods modify the geometric parameters of the model with each iteration, the NURBS template representations allow the differing models to be successfully compared irrespective of their size or mesh properties. Using NURBS surface templates offers an advantage in that data from a part's interior regions are used while curve templates simply use boundary data. Using NURBS curve templates offers a distinct time advantage over surfaces. Although the method is not 100 percent accurate at identifying the modes present in the analysis results, it is accurate enough to provide an understanding of the part's properties especially in the early steps of the design process.

The similar accuracies of the method, whether using NURBS curve or surface templates, indicate that both geometries are useful in the identification process. If time is of major concern, then curve templates should be utilized in order to leverage their minimal time cost. However, doing so means that internal part data will not factor into the identification process. Surface templates should be used when it is desired to capture the behavior of the entire part, as opposed to curve templates which only use the boundary data. On another note, because NURBS surfaces can only be used to represent models that are four sided, NURBS curve templates are more applicable for use with a more generalized shape. Implementing the automated method results in large time savings over the visual inspection of results to determine mode shapes, which is the currently used method. The automated identification method is almost eight times faster for surface templates and 136 times faster for curve templates than visual inspection of results within a post processing environment and the method's time savings add up significantly when a large

number of modes are identified within an iterative process. The mode identification application also allows for more objectivity in the identification process.

6.1 **Recommendations**

One current limitation to the automated mode identification method is that only the nodal displacement magnitudes are being used to define the NURBS representations of the mode shapes. An improvement would be realized if the method were to use the nodal displacement in each of the three (X,Y,Z) directions including the sign. This would increase the level of detail captured in the NURBS surface representations of the data and allow for a greater number of unique mode shapes to be more accurately identified. This is due to the fact that positive and negative displacements could be captured as well as displacements in each of the three principle directions, avoiding the confusion between two different mode shapes which have different mode shapes but similar displacement magnitudes.

Another limitation in the current method is the surface fitting algorithm within the NX Open C API that is currently being used. The method is not able to consistently fit a surface through a set of points. A major improvement to the method would be to use another method that is more capable of fitting a surface through each point individually in a point cloud as opposed to using a simple least squares approximation. This would also improve the quality of the NURBS surface representations, allowing the method to more accurately identify the mode shapes of a set of results. A function exists within the GSNLib function library which may be more effective at producing an accurate surface representation from the displacement data.

Another benefit of switching from the NX API function to another method would be that the application would be independent of the CAD environment completely. Currently, a CAD session must be started each time the application is run in order to call the function from the API.

This accounts for a large portion of the execution time within the iterative process. If the function could be switched to a CAD independent algorithm, more drastic time savings would be realized.

Identification of mode shapes as done in this research, using the MAC correlation value and a threshold value, introduces uncertainty into the identification process. The uncertainty in the identification is an important issue that, when quantified, can be used to aid the designer in understanding the results of the method. A means to leverage this uncertainty, which is currently not used, to improve the method should be developed. Uncertainty is indicated by MAC values which are near the threshold value. Multiple MAC values above the threshold also indicate uncertainty as to which mode shape the results actually represent. Through quantifying the uncertainty in the mode shape identification, the method will be improved. One way in which the uncertainty can be used to improve the method is by using NURBS curve templates for a very fast initial comparison based on the part's boundary. For match results with high uncertainty, NURBS surfaces could be used to identify the mode shape using data from the part's interior. Finally, if the uncertainty is still high, a flagging process could identify the result as a high uncertainty match for the designer to investigate. Flagging these uncertain results will allow the designer to note the less confident matches, inspect the results to ensure the design has not varied too far from the baseline geometry, and modify the design process in order to reduce the uncertainty and improve the accuracy of the method.

The current method also only applies to two dimensional finite element models. This is adequate for basic analyses, but a modification to the method that allows for three dimensional models to use the method would allow for mode identification of more detailed models. Three

dimensional finite element results may also provide a means for more accurate mode matching capabilities.

REFERENCES

[1] S. Gade, H. Herlufsen, and H. Konstantin-hansen, “How to Determine the Modal Parameters of Simple Structures,” 2001.

[2] D. J. Ewins, *Modal Testing: Theory and Practice*, 1st ed. New York: Research Studies Press Ltd., 1984.

[3] M. Rades, *Mechanical Vibrations II*. 2010.

[4] R. J. Allemang, “The Modal Assurance Criterion – Twenty Years of Use and Abuse,” *Sound and Vibration*, vol. 1, no. August, pp. 14-21, 2003.

[5] ANSYS Inc., “ANSYS Documentation.” ANSYS Inc, 2007.

[6] A. K. Chopra, *Dynamics of Structures*, 3rd ed. Upper Saddle River: Pearson Prentice Hall, 2007, p. 876.

[7] R. J. Allemang, “Vibrations: Experimental Modal Analysis,” University of Cincinnati, Cincinnati, 1999.

[8] W. Heylen and T. Janter, “Extensions of the Modal Assurance Criterion,” *Transactions of the ASME*, vol. 112, no. October, pp. 468-472, 1990.

[9] M. E. McNelis, T. W. Goodnight, K. S. Carney, K. D. Otten, A. Corporation, and B. Park, “Lessons Learned From CM-2 Modal Testing and Analysis,” Cleveland, 2002.

[10] L. V. Burns, “MAC Evaluations Utilized in FEA Analysis for Mode Identification,” South Bend, 2004.

[11] D. J. Ewins, “Model validation: Correlation for updating,” *Sadhana*, vol. 25, no. 3, pp. 221-234, Jun. 2000.

[12] T. S. Kim and Y. Y. Kim, “Mac-based mode-tracking in structural topology optimization,” *Computers & Structures*, vol. 74, no. 3, pp. 375-383, Jan. 2000.

[13] T. W. Sederberg, “Computer Aided Geometric Design - Course Notes,” 2011.

[14] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. New York: Springer-Verlag Berlin Heidelberg, 1997, p. 646.

[15] I. Zeid, *Mastering CAD/CAM*. Boston: McGraw-Hill, 2005.

[16] J. Gallier, *Curves and Surfaces in Geometric Modeling*. San Francisco: Morgan Kaufmann, 2000, p. 491.

[17] A. Hepworth, "Methods to Streamline Laminate Composite Design, Analysis, and Optimization," *Journal of the Electrochemical Society*, vol. 129, no. April. 2010.

[18] T. Astle, "System Architecture and Development of CAD Independent Algorithms for Integration with Commercial CAD Software," Brigham Young University, 2003.