Application of Machine Learning and Parametric NURBS Geometry to Mode Shape Identification

Robert M. Porter

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
Christophe Girard-Carrier

Department of Mechanical Engineering

Brigham Young University

September 2013

ABSTRACT

Application of Machine Learning and Parametric NURBS Geometry
to Mode Shape Identification

Robert M. Porter
Department of Mechanical Engineering, BYU
Master of Science

In any design, the dynamic characteristics of a part are dependent on its geometric and material properties. Identifying vibrational mode shapes within an iterative design process becomes difficult and time consuming due to frequently changing part definition. Although research has been done to improve the process, visual inspection of analysis results is still the current means of identifying each vibrational mode determined by a modal analysis. This research investigates the automation of the mode shape identification process through the use of parametric geometry and machine learning.

In the developed method, displacement results from finite element modal analysis are used to create parametric geometry which allows the matching of mode shapes without regards to changing part geometry or mesh coarseness. By automating the mode shape identification process with the use of parametric geometry and machine learning, the designer can gain a more complete view of the part's dynamic properties. It also allows for increased time savings over the current standard of visual inspection

Keywords: NURBS, machine learning, parametric geometry, mode shape identification, automation

ACKNOWLEDGMENTS

I would like to express my appreciation for all those who have helped me to complete this research. I thank Dr. Greg Jensen from Brigham Young University, for his time, support, and guidance. I thank Dr. David T. Fullwood and Dr. Christophe Giraud-Carrier for their guidance and input during this process. I am grateful to Evan Selin and all those 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 immensely grateful. Finally, I thank my wife, Megan, for her constant support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Identifying vibrational mode shapes with their corresponding frequencies becomes important when designing objects or structures that are subjected to dynamic forces. That is, to alleviate structural weakness due to resonant behavior: natural frequencies being excited by operational forces [1]. As a check the designer can perform a modal analysis using the Finite Element Method to easily identify the mode shape of his/her given object. However, if an optimization were implemented, where the design changes on an iterative basis, identifying and comparing these mode shapes becomes a complex problem. In 2011 Selin et al. explored applying parametric NURBS geometry and the Modal Assurance Criterion (MAC) to mode shape identification with the result being the ability to identify mode shapes of parts with differing geometries and mesh densities [2]. Selin's research addressed the problem of automating mode shape identification and it will also be investigated here.

This research applies machine learning to the mode shape identification problem in efforts to improve the task of identification. Machine learning is broadly defined to include any computer program that improves its performance at some task through experience [3]. Machine learning has been used in various classification and regression problems where one may wish to know a type or category that given inputs fall under (classification) or a numerical prediction given inputs with some training data (regression) [4]. This research seeks to leverage machine learning along with parametric NURBS geometries to classify vibrational mode shapes from a finite element analysis. In doing so, a designer can run an iterative optimization with information about the dynamic behavior of the object. While Selin's research was successful within its scope, this topic of automating the identification process utilizing machine learning will show added benefits and decreased disadvantages over utilizing the MAC. This research seeks a method by which machine learning, along with parametric geometries will be used to automatically identify vibrational mode shapes and frequencies from displacement data. The described method will be an important step

towards the development of a complete iterative vibrational mode shape identification tool that could be used for a wide variety of parts or models.

1.1 Problem Overview

Within a design process modeled parts often go through many changes, especially within an iterative design process such as an optimization or design of experiments. During an iterative design process parametric models can be updated and changed in either small or large ways. The purpose of changing model parameters and geometries, especially in a iterative design, is to obtain a design that is superior to an initial, or starting design. Changing the properties of a part via geometry, material properties or other, can have a significant effect on its static and dynamic behavior.

In a modal analysis the natural frequencies are affected by design changes and the vibrational mode shapes excited by these natural frequencies can exhibit themselves in a different manner than in consequent design iterations. These results can be reported in order of increasing natural frequency. An example of the resulting contour plots from a modal analysis can be seen in Figure 1.1 below. These contour plots are two dimensional representations of three dimensional objects. The color dispersion present in Figure 1.1 represent the object's displacement magnitude, where red represents the largest positive displacement(s) and blue represents the largest negative displacement(s).

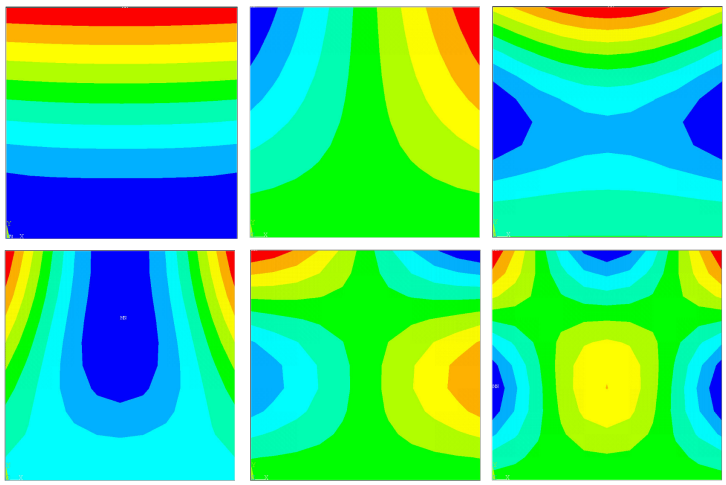

Figure 1.1: Mode shapes produced by a modal analysis

It is important to note that the mode shapes are affected by design changes and the vibrational mode shape excited by the parts natural frequency can exhibit themselves in a different sequence on different design iterations. This makes the task of identifying the specific mode shape associated with each natural frequency more complicated when executing an iterative design.

If a designer were to view these changes after each iteration, they may take measurements of the analysis results or simply view the surface to see how the mode shape has changed. In this way the designer identifies the mode shape like they would any other shape, by looking at the object's features (number of sides, scale, how many peaks and valleys and their locations on the part, the angles contained within the part, etc.). The designer can identify the parts mode shape without regard to its mesh density or overall size. Machine learning has the ability to take in feature attributes, like those described, and given a label can create its own hypothesis as to the correlation between the two [5]. Machine learning has been used in a wide variety of classification problems where the inputs and outputs of a system for some instance are known but how to arrive at the output is unknown or unclear. Applying machine learning to mode shape identification is not a straight forward process, and some conditioning must take place in order to benefit from machine learning capabilities in this research. Once some pre-processing is done, an algorithm must be chosen and that algorithm must be trained given training examples thus allowing it to learn. In the end a properly trained machine learning algorithm is able to classify mode shapes based on previous examples with good accuracy.

1.2 Thesis Objective

The purpose of this research is to develop a general method that will aid in the automation of identifying mode shapes using parametric models and machine learning to improve upon previous attempts. This method will create parametric surfaces from nodal displacements that result from a finite element analysis to represent individual mode shapes. Through automating this process of mode shape identification, the designer will be provided with a more complete understanding of the model's dynamic properties.

This research will also investigate the use of parametric geometry to collect proper feature attributes for machine learning to match and correlate mode shapes. The use of parametric geometry will become important when trying to identify mode shapes from geometries that vary in

part definition and mesh coarseness. The use of parametric geometry would allow the same number of features to be extracted from the surface every time, which will aid the machine learning process [6].

This research will report the time required and matching accuracy for the method and give a comparison to previous research completed. The main benefit of this research will be in determining if a method which utilizes machine learning is capable and robust enough to correctly identify mode shapes of differing geometry to known mode shapes through the use of parametric geometry.

This method has been implemented in an optimization framework to correctly match and report the model's natural frequency and mode shape. By utilizing the functionality of an optimization software package, the designer can create training sets of known mode shapes as well as identify unlabeled mode shapes based on previously trained examples.

1.3 Problem Delimitations

The purpose of integrating the method into a optimization framework to accomplish the tasks mentioned above is to show that the developed method is feasible and implementable. This method will be limited to parts that can reasonably be represented as four-edge surfaces due to the fact that NURBS surfaces will be used. Modal analysis performed in this research will be using ANSYS analysis software package, and as such will be modeled using the ANSYS SHELL63 elements. The models will also be limited to two-dimensional representations of parts. The parametric data structures needed herein will utilize the NXOpen libraries which are commercially available. Weka (Waikato Environment for Knowledge Analysis) will be used in this research to run machine learning algorithms from acquired data and for the use of statistical testing of algorithms. There are several applications within Weka such as the Explorer, Experimenter, KnowlegeFlow, and Simple-CLI. For this research the Explorer and Experimenter will be utilized and are described in Chapter 4. Weka's user guide may be consulted for more information about Weka and its applications [6]. Finally the method 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 related to this thesis. This will include a brief discussion of modal analysis, a discussion about machine learning, and parametric geometry. The third chapter discusses the general methods used to automate the matching of modal analysis results to previously trained instances using machine learning. The fourth chapter discusses the implementation of those methods using C++ programming and integration into the Isight optimization software. Chapter 5 details the results of the mode identification method and compares it to previous methods. The sixth chapter discusses the conclusions and future work of this research.

CHAPTER 2. BACKGROUND

2.1 Modal Analysis

In a design process, a modal analysis becomes useful when the designer wishes to characterize the dynamic behavior of a part or structure. The dynamic characteristics that the modal analysis determines are the natural frequencies and the vibrational mode shape of the given part or structure [1]. Finite Element solvers perform modal analysis on meshed models with results including a natural frequency and a vector of nodal displacements (mode shape). Historically the MAC has been the general method for measuring the consistency of modal vector estimates, calculations, or experimental data [7]. The MAC has the ability to calculate the linearity between two modal vectors. After running the MAC calculation, it results in a value between zero and one, which indicates the correlation between the two mode shapes, one being a perfect match and zero being no correlation. One of the downsides to using the MAC is the inability to compare modal solutions from a Finite Element model that has differing meshes or number of elements. The usefulness of the MAC has further been described in Selin's work [2]. In Selin's research he was able to leverage parametric NURBS geometries in conjunction with the modal assurance criterion to automatically identify vibrational mode shapes and frequencies from modal analysis displacement data. The proposed research will be benchmarked against this work with the hopes of creating a more robust method. For more information on the MAC, the following resources may be referred to [1, 8, 9].

2.2 Machine Learning

Machine learning is actively being used today in many instances where it is necessary to turn data into information. Machine learning is a mix of computer science, engineering, and statistics, and often appears in many other disciplines from politics to geosciences [10]. Any field that

needs to interpret and act on data can benefit from machine learning techniques. While machine learning has been successfully applied to many classification problems in many fields, there is no known research that has leveraged machine learning to identify vibrational mode shapes. To properly implement machine learning for this research and to any problem that can benefit from machine learning some design decisions must be made. These decisions include identifying the type of knowledge to be learned, how to represent the target knowledge, and a learning mechanism [3].

For this research the type of knowledge to be learned is information about a model, namely mode shape. Choosing a representation for this target knowledge is an important design decision. It is necessary to choose attributes of the target knowledge that are both descriptive and consistent. For example, if a machine learning task was to learn to identify pencils and the designer chose a pencils shape as its representation, describing a pencil by its color would then be meaningless. Thus choosing a representation for the target knowledge must be consistent in order to maintain a basis for comparison. Learning mechanisms require a set of training examples in order for machine learning to identify the target knowledge. Each example or instance in this research will describe a specific mode shape. How this research obtains consistent attributes that represent mode shapes will be discussed in subsequent chapters. More on designing a machine learning system can be found in several known works [3, 4].

Satpal et al. applied machine learning methods to health monitoring of beam-like structures using vibration-induced modal displacement data [11]. By using the modal properties of the structure, Satpal et al. were able to view changes in the modal displacement data to identify damage in a simulated cantilever beam. Satpal found that the machine learning model known as support vector machine (SVM) was a powerful tool for predicting the intensity and location of the damage. By using the SVM model, Satpal et al. were able to predict any damage intensity or location of the training set with almost zero error. In testing, Satpal observed errors ranging from 2.5% to 22% depending on location along the beam and the level of noise present in the system. This research explores SVM along with other machine learning models in the mode shape identification process.

Liu applied machine learning methods to character recognition proving its flexibility and performance [12]. Liu describes early character recognition being done using template matching and structural analysis, similar to previous work done by Selin et al. with template matching in

mode shape recognition. By switching over to machine learning methods, character recognition has benefited tremendously by releasing the designer from creating and selecting templates along with seeing significant improvements in recognition accuracy due to learning from large sample data. Similar to Liu, this research hopes to reduce mode shape matching error from previous methods by utilizing machine learning methods. A challenge that Liu encountered was the comparing of classifiers or algorithms because many classifiers are flexible in their implementation and can be easily influenced by any processing step.

Hung and Jan describe how machine learning can be implemented in engineering design and specifically applied to a structure optimization problem [13]. An unsupervised fuzzy neural network (UFN) case based learning model was developed to optimize for weight in a simply supported steel beam under LRFD specifications. It was found that their learning model adequately represented their model, and could handle large amounts of instances with reasonable computing time. Furthermore, Hung and Jan discuss the fact that selecting model attributes by trial and error affected the performance of the UFN model. As a result, care must be taken in selecting attributes that adequately describe the model in a consistent manner. Attribute selection will be an important part to the proposed research in order to adequately represent any given model to correctly identify its mode shape. Many resources are available to learn more about machine learning [3, 4, 14, 15].

2.3 Parametric Geometry

Parametric geometries are beneficial in that it is easy and quick to compute (x,y), or (x,y,z) coordinates of points existing on a curve or surface [16]. By generating a parametric surface with a uniform parameterization surfaces of differing size and complexity can be segmented into distinct areas through using a u and v coordinate system separate from the actual x,y,z coordinate system. Non-uniform rational B-spline (NURBS) surfaces are widely used parametric geometry representations. Each surface can be parameterized such that u-values of zero and one correspond to two edges of the surface and v-values of zero and one correspond to the other edges of the surface as shown in Figure 2.1. Since NURBS surfaces are defined this way, they can only be used to represent surfaces that are relatively four-edged, which can be a problem. The benefit of using parametric geometry is that surfaces of varying size and shape can relate to a single set of attributes by means of similar parameterizations that can directly relate the points on each surface.

This becomes important when assigning feature attributes that must remain consistent through a variety of surfaces that can vary in size, shape, and mesh densities.

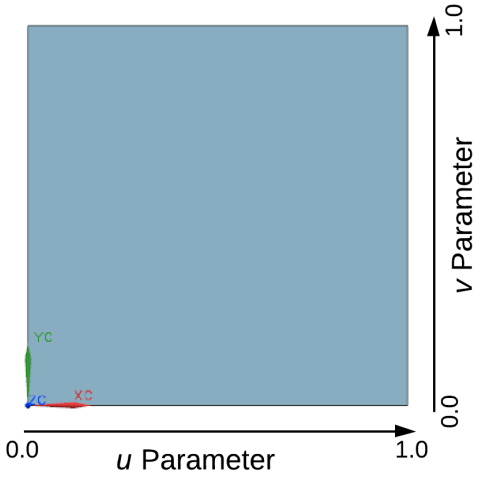

Figure 2.1: A parameterized surface

Interpolation through existing points and extrapolating points from control points are common methods for creating NURBS surfaces [17]. This research will utilize global surface interpolation to create NURBS surfaces from point clouds which approximates a surface through a set of existing points. A brief description of the mathematical definition of NURBS surfaces and global surface interpolation will be presented here. In creating a NURBS surface, 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, influences the surface topology as well. The following equation is the mathematical definition of a NURBS surface of degree p in the u direction and degree q in the v direction.

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)W_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)W_{i,j}}, 0 \leq u,v \leq 1 \tag{2.1}$$

In Equation 2.1 $P_{i,j}$ are the control points. $W_{i,j}$ are the weights of each point, and n and m are the number of control points in the u and v directions, respectively. The terms $N_{i,p}$ and $N_{j,q}$ are the basis functions that are defined on the knot vectors shown in Equations 2.2 and 2.3.

$$U = \{0,...,0,u_{p+1},...,u_{r-p-1},...,1\}, u_i \leq u_{i+1} \tag{2.2}$$

$$V = \{0,...,0,v_{q+1},...,u_{s-q-1},...,1\}, v_i \leq v_{i+1} \tag{2.3}$$

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, which terminate the surfaces control curves and indicate that the surface is parameterized between zero and one.

The following equation 2.4 is the mathematical definition for finding a B-spline surface $\mathbf{P}_{i,j}$ of degree (p,q) that is interpolated from supplied $(m+1)(n+1)$ data points $\{\mathbf{Q}_{k,l}\}, k = 0,...,n$ and $l = 0,...,m$.

$$\mathbf{Q}_{k,l} = \mathbf{S}(\overline{u}_k, \overline{v}_l) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(\overline{u}_k) N_{j,q}(\overline{v}_l) \mathbf{P}_{i,j} \tag{2.4}$$

Here terms $\overline{u}_k$ and $\overline{v}_l$ are parameter values that can be chosen using various methods that allow for the solving of Equation 2.4 for the approximated surface $\mathbf{P}_{i,j}$. More information on the mathematical and geometric properties of NURBS surfaces can be found in numerous sources on the topic [16] [17] [18].

By using parametric geometry this research will be able to relate dissimilar surfaces through their parameterizations. This will become important when trying to identify mode shapes that result from geometries that have not been previously encountered due to changes in size, configuration, or complexity. By moving surfaces in to parameter-space as opposed to real-space, a basis for comparison can be made and mode shapes can be easily matched to previously identified modes.

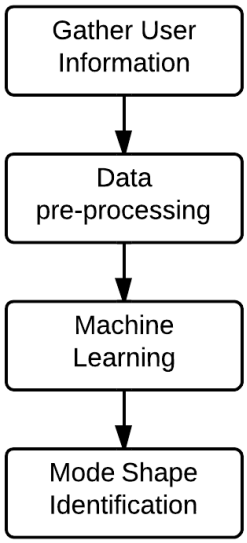

Figure 3.1: Overall method

This section presents a method of automatically identifying the mode shape of an object represented by a four-edge NURBS surface resulting from a modal analysis by using machine learning. To achieve our goal a classifier must be obtained. A classifier is a function that can identify to which category a new observation belongs. For example, a classifier could be trained to identify between spam and non-spam email messages. After learning, it can then identify if an incoming email is spam or not and take an appropriate action. Once trained, our mode shape classifier can be utilized in a stand-alone or in an automated approach to mode shape identification. The following steps outline this method:

- Gather information from the designer in regards to a base line design and the design space.
- Normalize, transform, and label nodal displacements resulting from a modal analysis into a NURBS surface representation in preparation for machine learning.

- Obtain and evaluate a classifier that will be used in this research to identify the mode shape for a given part.
- Utilization of the classifier to automatically match and report vibrational mode shapes within an optimization framework or as a standalone operation.

A visual representation of these steps is shown in Figure 3.1.

3.1 Gather User Information

Some information from the designer is required before training or testing the method. A designer must specify the model parameters that will act as a baseline design. These parameters represent the dimensions or properties of the part such as length, width, thickness, mesh coarseness, etc. The parameters of the baseline design that will be changed or iterated upon would be those that would either optimize some aspect of the model or fulfill some requirement of an overall design. As model parameters change new designs will arise that may not have been observed previously and may result in unexpected mode shapes.

3.2 Data Pre-processing

Below describes the method used to prepare data resulting from a modal analysis by normalizing and transforming the resulting nodal displacements into a NURBS surface representation. Figure 3.2 is an illustration of the work sequence.

3.2.1 Read Displacement File

A modal analysis is done on a Finite Element model of a part or component and the results are written to a file. This file contains the position and displacements of every node in the model corresponding to its natural frequency solution. This method parses through this file to obtain desired information.

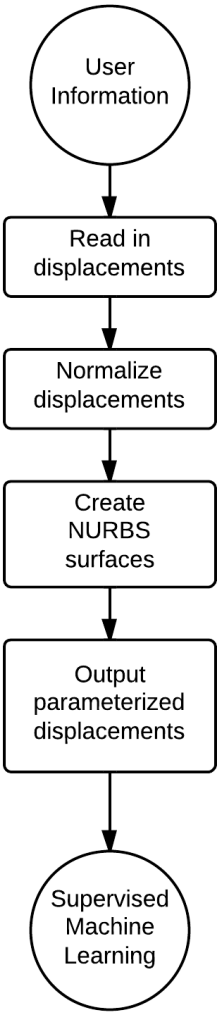

Figure 3.2: Pre-processing method

3.2.2 Normalize Nodal Locations and Displacements

In order for the method to obtain consistent comparable attributes on which to consistently identify a given mode shape, all surfaces are normalized by the maximum nodal displacement found in the solution set. For the normalizing equation below, U_{all} denotes the set of all nodal displacements in the modal solution, therefore the normalized displacements, U_{norm}, are found by:

$$U_{norm} = \frac{U_{all}}{max(|U_{all}|)} \tag{3.1}$$

which results in normalizing all nodal displacements to fall between numerical values of negative one and positive one. In Equation 3.1 the *max* function finds the largest absolute value contained in U_{all}.

3.2.3 Transform Data into NURBS Surface

When creating a NURBS surface in this method, the normalized positions and displacements for each node is used. In this research we use a CAD API to perform global surface interpolation and fit a preliminary surface through a set of points containing the normalized node locations from the original model. This surface will then be projected onto a working plane which was chosen to be the xy plane for this research. While it is possible for this method to take into account displacements in all three directions (x, y, z) in this research the projection of the surface will lie in the xy plane and displaced in the z direction. Figure 3.3 illustrates this process of projecting a preliminary normalized parametric surface onto a working plane in preparation for the next step. The red, green, and blue dots in Figure 3.3 represent points on the surface as they are projected onto the working plane.

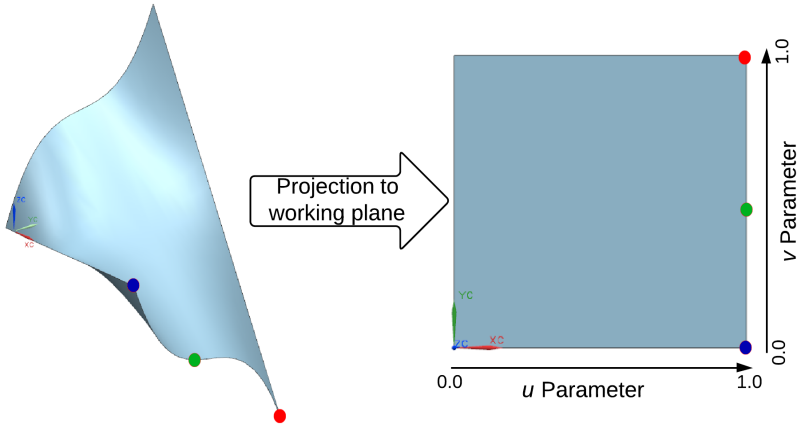

Figure 3.3: Normalized surface parameterized and projected onto working plane

The parameter values associated with every node in the model are determined by querying the surface using an API function. After the parameter values for each node are found, those values are then stored for future use. A new point set is then created with data from each node. Points in this set are defined in three dimensions by the u and v parameters of the node on the preliminary

surface and the normalized displacement of the node that is perpendicular to the working plane or in the case of this research, in the z direction. The surface fitting function from the API is once again used with this new point set, creating a new surface representing the mode shape. This process is illustrated in Figure 3.4.

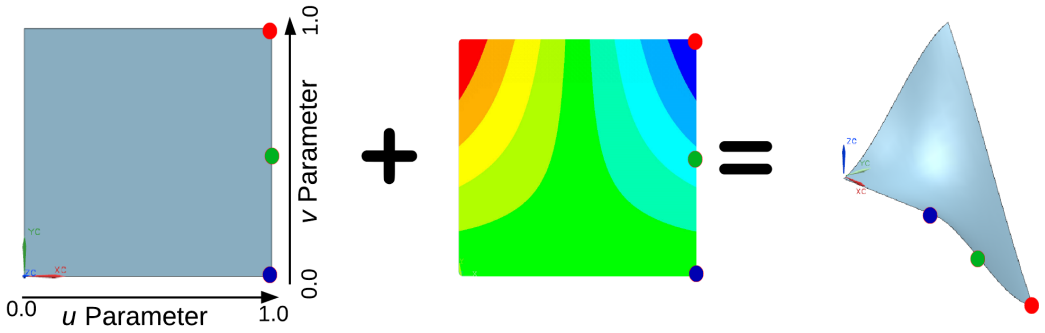

Figure 3.4: The preliminary surface and displacement data create the mode shape surface

3.2.4 Output Parameterized Displacements

After a mode shape surface is created we then can make comparisons with other surfaces in the presence of differing part definitions such as length, width, thickness, mesh coarseness, etc. An example of this can be seen visually in Figure 3.5 where the top level of three surfaces have the same mode shape but result from differing geometries. We can see visually how close these surfaces relate despite differences in their original part definitions.

Once a surface has been created to represent the model's mode shape, attributes are then pulled from that surface. Displacements are chosen to be feature attributes for simplicity in lieu of a more informative attribute. The parametric surface displacements in the z direction are grouped together in a grid like fashion. Each section of the grid represents the average displacement found over a given number of nodes. How these displacements are grouped is shown below in equation 3.2.

$$\bar{x}_z = \frac{\sum_{u=0}^{M} \sum_{v=0}^{N} Q_{u,v}(z)}{n} \tag{3.2}$$

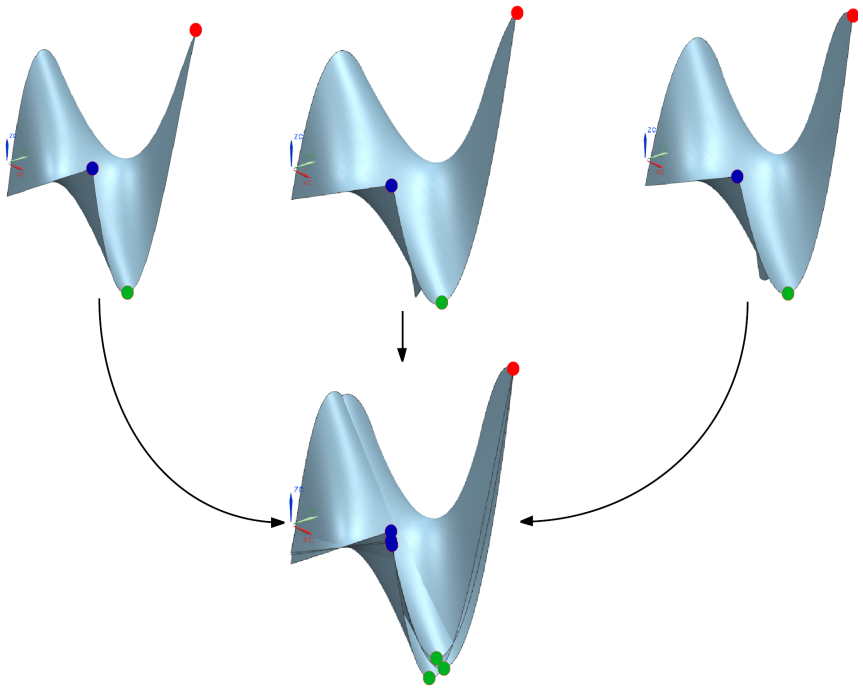

Figure 3.5: Normalization of geometry

In Equation 3.2 $Q_{u,v}$ is the vector containing all nodal displacements. $\bar{x}_z$ is the averaged displacement found over the number of nodes per grouping n. Each average displacement is considered a feature attribute of the surface. All average displacements are then written as a geometric definition of the surface to a file in preparation for machine learning. For machine learning a set of geometric definitions that describe a surface are called instances, and each instance contain feature attributes. Again, it is important to note that the number of feature attributes must remain consistent throughout this process for a basis of comparison as mentioned previously.

An example of how these geometric representations of mode shape surfaces are transformed into instances for machine learning can be seen in Figure 3.6. Figure 3.6 illustrates points that have been identified, grouped, and averaged into sections to act as features of the surface. The number of points and thus the number of sections required for this method is dependent on the complexity of the part. More complex geometry and/or higher order mode shapes would require more features to adequately describe the mode shape. Whereas if a designer will only be observing simple geometries with lower order mode shapes, then fewer features would be sufficient.

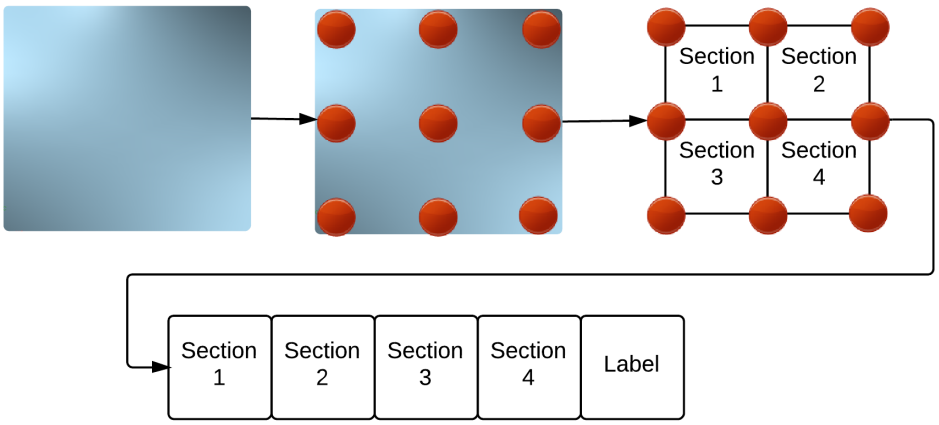

Figure 3.6: Grouping average displacements into instances

These sections or features are then organized into a vector with the last entry being a label that identifies the mode shape that the vector represents. These feature vectors are then formatted and output to an Attribute-Relation File Format (ARFF) for the purpose of using Weka to handle the machine learning calculations. ARFF files have two distinct sections: the Header information and the Data information. The header of the ARFF file contains the name of the relation, a list of the attributes, and their types. This section remains consistent throughout the method to ensure proper learning. The data section is where the surface attributes (displacements) are stored as instances. One instance is equivalent to one surface that has been discretized as previously described. It is in the data section that, when preparing a training set or a test set, each instance is labeled.

3.3 Machine Learning

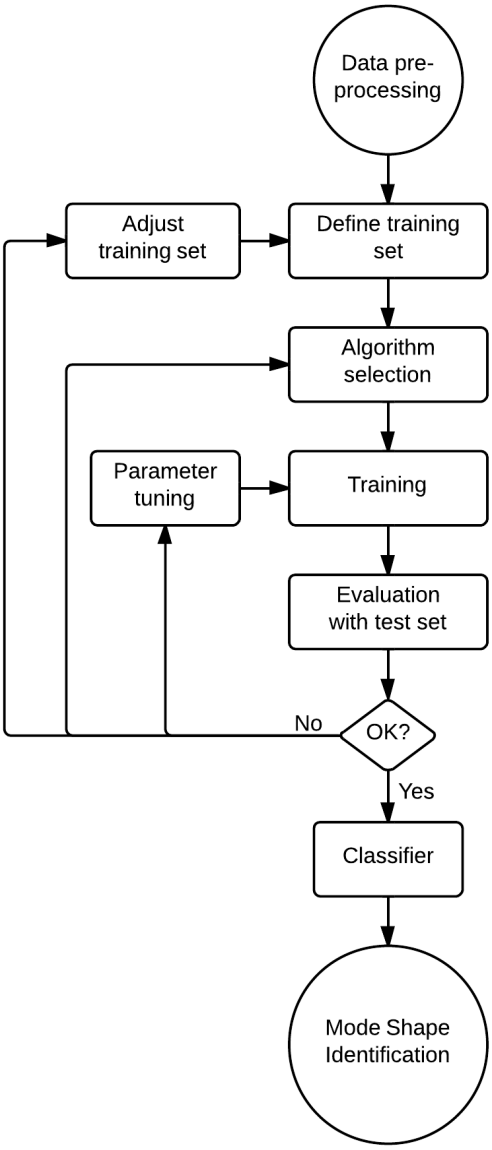

Figure 3.7: Machine learning method

This section reviews the method used for obtaining a classifier that will be used in this research to identify the mode shape for a given part. The sequence of obtaining a classifier is shown in Figure 3.7.

3.3.1 Define Training Data

The first step toward obtaining a classifier is collecting and defining a training dataset. Every geometric definition of a surface, or in other words, every instance from the previous data pre-processing step must be stored with a unique user-defined label for its mode shape. Unfortunately this step must be done manually. Upon creating these training instances, they are stored for further training and testing where they may be adjusted. This training set may be adjusted to contain more examples or different feature attributes if it is found that the attributes chosen are insufficient.

3.3.2 Selection of Machine Learning Algorithm

Choosing a specific learning algorithm to use in this classification problem is a vital step. It is known in machine learning that there is no algorithm that is uniformly superior over all possible problems [19]. The category of machine learning that this research falls into is supervised learning where the algorithms reason from externally supplied instances to produce general hypothesis which then make predictions about future instances [20]. There are several supervised machine learning algorithms to choose from and which algorithm we choose will be determined by cross validation to ascertain how well the algorithm can learn the training data and by a paired t-test. Cross validation is a statistical method of evaluating and comparing learning algorithms by dividing data into segments where one is used to learn or train a model and the other used to validate the model. The basic form of cross-validation is k-fold cross validation where the data may be divided in a specific number of segments [3, 4]. A t-test is a common method that assesses whether the algorithms chosen are significantly different from each other in the presence of a supplied training set. Given two paired sets (classification results) of n measured values, the paired t-test determines whether they differ from each other in a significant way under the assumptions that the paired differences are independent and identically normally distributed [21]. In so doing we can find the algorithm that best fits our data. If it is found that one classifier is not statistically different from another a classifier may be chosen based on performance. After an appropriate classification algorithm is selected we can then use it to run the method. While there are a number of algorithms to choose from, this research implements k-Nearest Neighbors (k-NN), Decision Trees, and Support

Vector Machines (SVM). These three will give the reader a good idea of how this method performs and how well we chose our feature attributes. For this thesis brief descriptions of each algorithm will be presented to give the reader a general understanding of the variation in approaches. More information on these learners can be found through numerous publications [4, 22, 23].

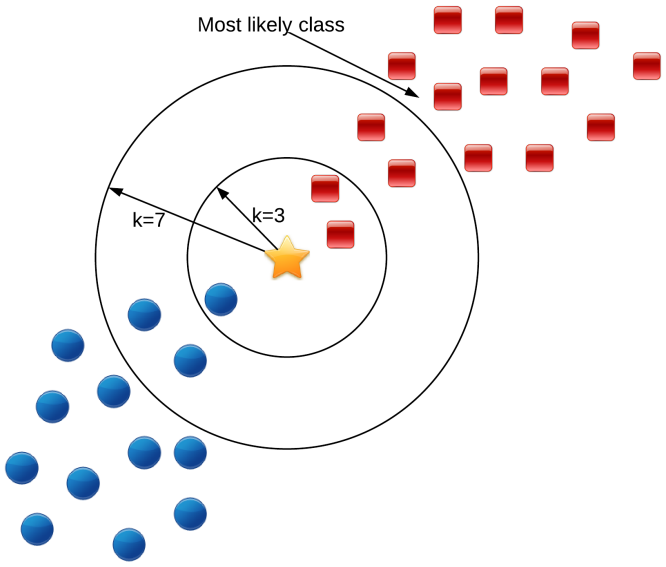

Figure 3.8: Principle of k-Nearest Neighbors (k-NN)

The simplest, most used instance-based learning algorithm is the k-NN algorithm. Once trained, a k-NN classifier assigns a new query instance to a class having the most examples among the k neighbors of the input [4]. k-NN assumes that all instances are points in some n-dimensional space and defines neighbors in terms of distance from the new query instance, usually the Euclidean distance [24]. k is the number of neighbors considered as exampled in Figure 3.8 where values of $k = 3$ and $k = 7$ are chosen.

The classification of the new query instance is simply the most common class among the k selected examples. It can be seen that for the example in Figure 3.8, for both $k = 3$ and $k = 7$, the most likely class for the new query instance would be a red square. While both red squares and blue circles are present, red squares are the most common and the query instance is classified as a red square.

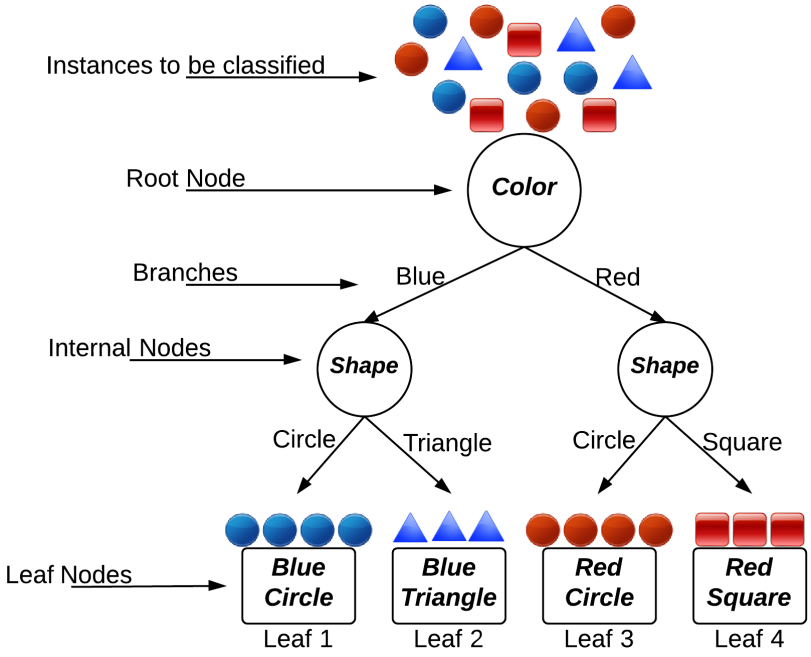

Figure 3.9: Principle of Decision Trees

Decision trees are trees that classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume [4]. As indicated in Figure 3.9, a decision tree consists of a root node, branches, internal nodes, and leaf nodes. The root node and internal nodes test some property with discrete outcomes labeling the branches. This process of testing a property at the root node and choosing a branch from the outcome is repeated through internal nodes until a leaf node is hit. The leaf node provides an appropriate classification for the given instance. An example of a decision tree can be seen in Figure 3.9. In this example we start with some instances to be classified, and test them for color at the root node. After separating the instances by color, the decision tree then tests for shape for both cases red and blue. Once this is done each instance arrives at a leaf node where the most appropriate class in assigned.

A common method for training decision trees is the process of *top-down induction of decision trees* (TDIDT). TDIDT works by splitting a provided data set into subsets by testing some attribute value. This process is repeated at each derived subset in a recursive manner until a subset is reached that has the same value as a target leaf node. More on decision tree learning can be found in many publications [3–6].

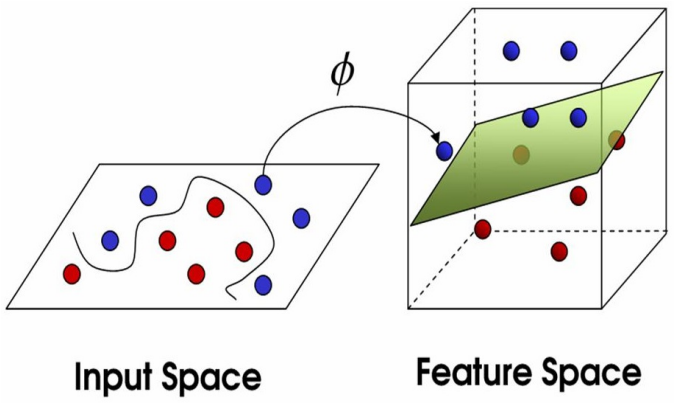

Figure 3.10: Principle of Support Vector Machines (SVM)
http://www.imtech.res.in/raghava/rbpred/svm.jpg

Lastly, we will discuss the *support vector machine* (SVM) approach. SVM is a linear classification model that seeks to separate classes via a hyperplane. For example, if a data set has two distinct classes that can be linearly separated by a line then one class would be lie on one side of the line and the second class on the other. The distance from the line to the instance closest to it on either side is called the margin. Maximizing the margin will produce best results due to the fact that points near the line represent very uncertain classification decisions. For example, if a point were to lie on or next to the line, there would be a near 50% chance of the classifier choosing either side. When a data set is not linearly separable, the data set can be mapped to a new space using a nonlinear transformation. This is known as the *kernel trick* and more information about it and the SVM method can be found in numerous publications [3–5, 25]. Figure 3.10 is an example of instances in an input space being mapped to a feature space where a plane is identified that separates the two classes shown. By moving from the input space into a possibly high-dimensional feature space more features can be identified to separate individual instances.

3.3.3 Training

Exposing classification algorithms to a training set allows it to learn to do its task. A training data set contains examples that are characteristic of the problem to be solved. It learns by looking at the provided examples of attributes and given a label or class can create its own hypothesis as to the correlation between them [5]. By training our classifier we will be evaluating how well the learning algorithm is able to learn the sample training data. This evaluation is done

using cross validation as mentioned previously. Many classification algorithms allow for some tuning of parameters which can enhance the classifiers ability to interpret given data. If it is found that an algorithm is not performing well some parameter tuning of that algorithm may increase its ability to learn.

3.3.4 Evaluation with Test Set

When applying this method to a new design space, a designer may evaluate the algorithm chosen to ascertain how accurately that algorithm is at predicting mode shapes. To evaluate our chosen classifier, we will use test sets where every instance has been properly labeled with its corresponding mode shape. These test sets contain instances of mode shapes that result from parts or models that may or may not have been observed when creating the training data. For example, training data may be compiled from a model that has varied in its parameters by $\pm 10\%$ and testing data may be compiled from the same model that has varied in its parameters from $\pm 10\%$ to $\pm 100\%$. While the change in parameters are randomized every iteration, there is a chance that within $\pm 10\%$ some test instance may be identical to some training instance.

By testing on data that has not been previously observed we can evaluate how well the classifier and this method can extrapolate a mode shape from geometry with which it has no previous experience. If the training or evaluation of a classifier is unsatisfactory the designer can return to a previous stage of the supervised machine learning process (see Figure 3.7). For the problem of mode shape identification a number of factors can be investigated: the most relevant features may not be taken into account, a larger training set may be needed, utilizing an inappropriate algorithm or some parameters need tuning.

3.4 Mode Shape Identification

Once a classifier has been obtained we can now use it to assign class labels (mode shape names) to instances where feature values are known but the class is unknown. This can be done both within and without an optimization workflow. Upon obtaining a classifier a new sequence is needed for implementation of this method. While the first 2 steps remain the same, gathering

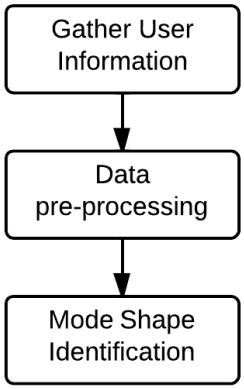

Figure 3.11: Mode shape identification method

information from the user and data pre-processing, we will now use the classifier directly to label mode shapes with the accuracy of the selected classifier. This process is shown in Figure 3.11.

CHAPTER 4. IMPLEMENTATION

The methods described in Chapter 3 were implemented in computer programs in order to be utilized in an automated approach where a design can be iterated upon, or as a standalone process where the designer can examine a specific design. A program was developed to create NURBS surfaces to represent mode shapes for a set of modal analysis results. This program outputs discretized displacements of surfaces to be used as feature attributes in the mode shape identification process. The program written for this research utilized Siemens NX C/C++ API. Another programming language that was used was the ANSYS Parametric Design Language to automate the modeling, meshing, analysis, and results reporting for the training and testing of this method. All of the modal analysis completed for this research was done using ANSYS 13.0. Weka 3.6.9 was used for its collection of machine learning algorithms and ability to perform statistical analysis.

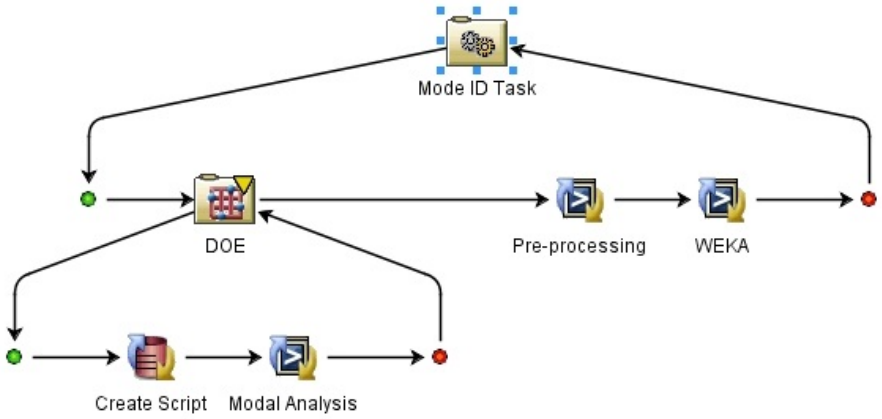

Figure 4.1: Sample Isight task

The applications used in this research have been integrated into optimization software to enable a designer to identify mode shapes iteratively or on a specific design. SIMULIA's Isight optimization software was chosen for this research to allow a user to take advantage of the methods

functionality. A sample of how the programs mentioned above can be used to implement the method presented in this research can be seen in Figure 4.1.

The task in Figure 4.1 can be run to gather data to be used in training and testing, or for identification if a classifier has been obtained. A design of experiments (DOE) is the first component in the task. The DOE initializes selected parameter values and passes them into the subflow which creates a new modal analysis script and solves the modal analysis in ANSYS. The results of the modal analysis are then stored to be used by the next step. The results of the modal analysis are then read by the pre-processing component and used to create parametric surfaces that represent mode shapes. The pre-processing component then outputs attributes that describe each mode shape. These attributes can then be used by the Weka component to identify modes shapes if a classifier is present. How a classifier is obtained will be described in the subsequent sections.

4.1 Gather User Information

The task in Figure 4.1 allows users to identify which parameters of a baseline design they wish to change, by how much, and how many times within the DOE component. An example of how the user can choose their design space is shown in Figure 4.2 below.

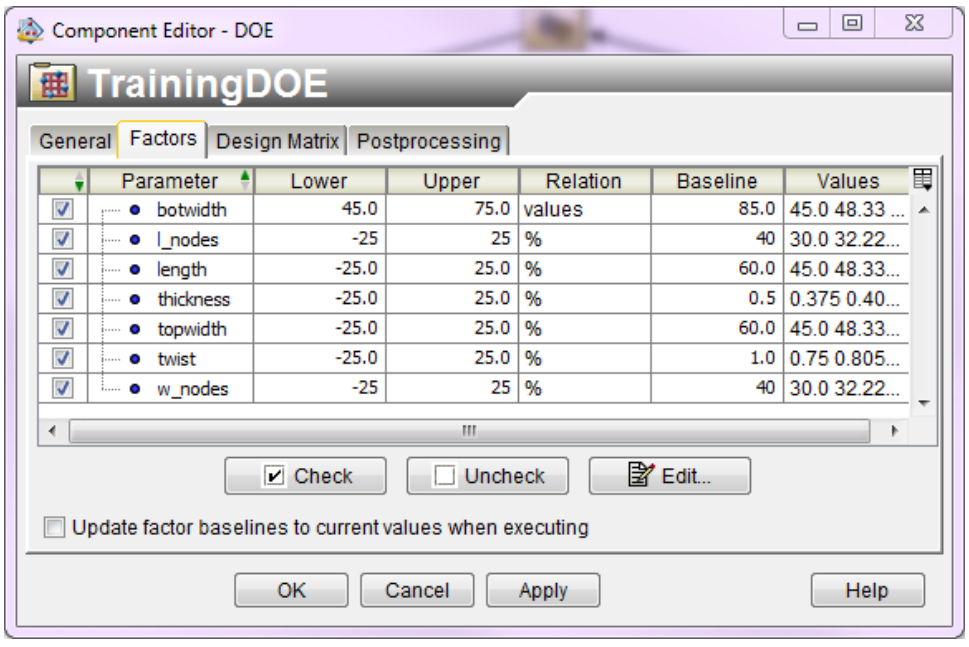

Figure 4.2: Modifying the design space in the DOE editor

4.2 Data pre-processing

Preparing the data contained in the modal analysis results file to become feature attributes of the model has several steps which are detailed below. The purpose of these steps, as described previously, is to provide a basis for comparison in the presence of differing geometries and mesh coarseness of four-edge surfaces. For a reference of this sequence see Figure 3.2.

4.2.1 Read in Displacements

The modal analysis results contain node locations and displacements in directions x, y, z as well as the magnitude of each node's displacement. The reading of the modal analysis results that is to be stored by the program is done simply by storing all lines of the file in a std::vector of std::strings. A basic tokenizer is used to extract the node positions and displacements that lay in xyz space. This is a relatively simple process and so the code used for this step is not presented here.

4.2.2 Normalize Nodal Locations and Displacements

In order to properly compare and contrast models with differing geometries, the magnitudes of the nodal locations and displacements must be normalized. In this way parts with differing geometries but the same mode shape can be easily identified through this method. Once the largest displacement is determined in the results, all displacements are normalized. The normalization of nodal displacements is done as described above in section 3.2.2 and is shown in the code below.

It can be seen from the code below that all nodal positions ($xPos, yPos, zPos$) are normalized as well as the nodal displacement magnitude ($usum$). Because the displacement magnitude is reported from a modal analysis instead of the magnitude and direction, a direction is imposed relative to displacements in the z direction. This is necessary to capture negative displacements as well as positive displacements due to the fact that the reported displacement magnitudes are all positive. Line 16 in the code says that if the model is displaced in the negative z direction by the modal analysis, make the displacement magnitude negative as well.

```
// Determine Maximum Node Locations and Displacement
for(int i=0; i<(int)surfaceNodes.size(); i++)
{
if(fabs(surfaceNodes[i].xPos)>xPosMax) xPosMax = surfaceNodes[i].xPos;
if(fabs(surfaceNodes[i].yPos)>yPosMax) yPosMax = surfaceNodes[i].yPos;
if(fabs(surfaceNodes[i].zPos)>zPosMax) zPosMax = surfaceNodes[i].zPos;
if(fabs(surfaceNodes[i].usum)>usumMax) usumMax = surfaceNodes[i].usum;
}
//Normailize the Node Locations and Displacements & determine the direction of
    the displacement magnitude
for(int i=0; i<(int)surfaceNodes.size(); i++)
{
if(fabs(xPosMax)>0.0) surfaceNodes[i].xPos = surfaceNodes[i].xPos/xPosMax;
if(fabs(yPosMax)>0.0) surfaceNodes[i].yPos = surfaceNodes[i].yPos/yPosMax;
if(fabs(zPosMax)>0.0) surfaceNodes[i].zPos = surfaceNodes[i].zPos/zPosMax;
if(fabs(usumMax)>0.0) surfaceNodes[i].usum = surfaceNodes[i].usum/usumMax;
if(surfaceNodes[i].uz<0.0) surfaceNodes[i].usum = surfaceNodes[i].usum*(-1);
}
```

4.2.3 Transform Data into NURBS Surface

From the normalized nodal positions we interpolate a parametric surface representing the original geometry of the model created and analyzed by the modal analysis. This is done using the NX API function call `UF_MODL_create_surf_from_cloud` as demonstrated in the code below.

```
//Populate 'cloud' with normalized node locations
for(int i=0; i<(int)surfaceNodes.size(); i++)
{
cloud[i][0] = surfaceNodes[i].xPos;
cloud[i][1] = surfaceNodes[i].yPos;
cloud[i][2] = surfaceNodes[i].zPos;
}
//Create a NURBS surface that represents the original model normalized
UF_MODL_create_surf_from_cloud(point_cnt,cloud,NULL,NULL,U_degree,
V_degree,U_patches,V_patches,0,&avg_error,&max_error,&max_error_index,&
    surface_tag);
```

Since this surface was interpolated from the nodes in the original model, each node exists either on or near the surface. Figure 4.3 shows a representation of this surface fitting process.

We then determine the u and v parameters for each node using the NX API function call `UF_MODL_ask_face_parm`. Given a reference point `UF_MODL_ask_face_parm` returns the parameter (u,v) on the face that corresponds to that reference point. A new point set is then created that will be used to create a parametric surface that describes its mode shape. This mode shape surface is

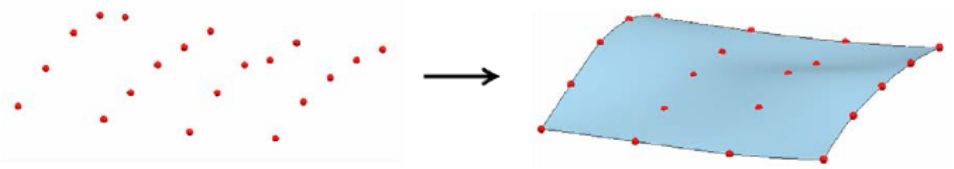

Figure 4.3: Interpolation of node data into a NURBS surface

created using the parameters u and v and the nodal displacements collected earlier for the three dimensions of each point in the set. The code below shows how this step was implemented.

```
for(int i=0; i<(int)surfaceNodes.size(); i++)
{ //Query each node for its u and v parameters
myPoint[0] = surfaceNodes[i].xPos;
myPoint[1] = surfaceNodes[i].yPos;
myPoint[2] = surfaceNodes[i].zPos;
UF_MODL_ask_face_parm(surface_tag,myPoint,parm,face_pnt);
//Populate 'cloud2' with the u and v parameters and displacements
cloud2[i][0] = parm[0];
cloud2[i][1] = parm[1];
cloud2[i][2] = surfaceNodes[i].usum;
}
//Create a NURBS surface that represents a mode shape
UF_MODL_create_surf_from_cloud(point_cnt,cloud2,NULL,NULL,U_degree,
V_degree,U_patches,V_patches,0,&avg_error,&max_error,&max_error_index,&
    surface_tag2);
```

The resulting parametric surface can then be used to obtain information about the mode shape without regards to changing parameters in the design space by conforming all possible geometries to a uniform basis for comparison. That is, all possible geometries within the design space will be parameterized in terms of u and v parameter values as discussed in section 2.3.

4.2.4 Output Parameterized Displacements

As mentioned previously, displacements were chosen to describe a surface that represents a mode shape. A designer can identify vibrational mode shapes visually by observing how the surface is displaced; for this reason displacements were chosen to be feature attributes. How well the displacements describe the surface will be assessed when evaluating the classifier's ability to correctly identify its mode shape in Chapter 5.

After creating the parametric surface that represents a mode shape, points in real space are retrieved from that surface using the NX API function call `UF_MODL_ask_face_props`. The function call `UF_MODL_ask_face_props` returns a point (x,y,z) on the surface face that corresponds to a given parameter (u,v). This can be seen being implemented in the code below where 121 (11 by 11) points were chosen to be collected from the surface to describe its shape. As mentioned in Chapter 3, the number of points collected from a surface may be chosen to be different if desired or if it is found that a different number of points better describes the mode shape surface.

```
1  for (int i=0;i<=10;i++)
2  {
3  P.clear();
4  for (int j=0;j<=10;j++)
5  {
6  paramUV[0] = uParm;
7  paramUV[1] = vParm;
8  UF_MODL_ask_face_props(surface_tag2, paramUV, point, u1, v1, u2, v2, unit_norm
       , radii);
9  xyz[0] = point[0];
10 xyz[1] = point[1];
11 xyz[2] = point[2];
12 P.push_back(xyz);
13 uParm += 0.1;
14 }
15 uParm = 0;
16 vParm += 0.1;
17 Q.push_back(P);
18 }
```

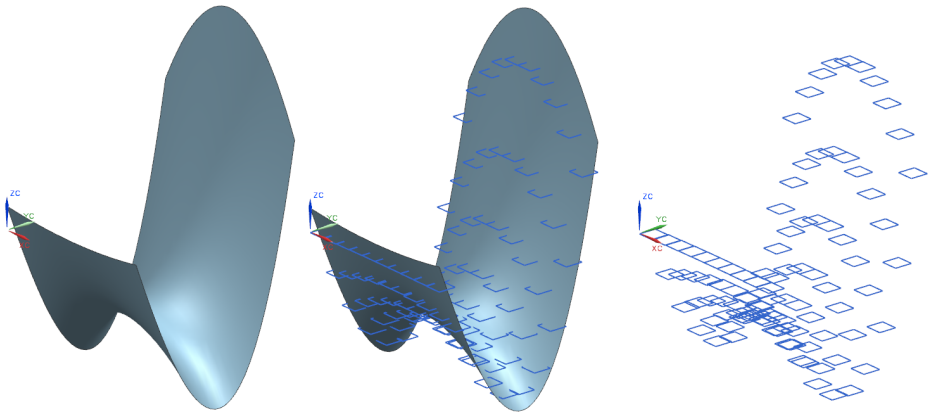

Figure 4.4: Discretization of a mode shape surface into sections

Upon collecting the desired amount of points from the mode shape surface, the displacements in the z direction are then placed in discrete groups as described in section 3.2.4 and as visualized in Figure 4.4. Each square in Figure 4.4 represent the average displacement of the surface for that region. By pulling 121 (11 by 11) points from the surface and grouping them as illustrated in Figure 3.6, the code below is able to group these displacements into precisely 100 discrete sections in preparation to be written to a file and act as attributes of the mode shape surface.

```
int incrV = 0;
for (int p=0;p<10;p++)
{
int incrU = 0;
for (int k=0;k<10;k++)
{
double sumP = 0;
for (int i=incrU;i<=incrU+1;i++)
{
for (int j=incrV;j<=incrV+1;j++) sumP += Q[i][j][2];
}
incrU++;
double aveP = 0;
aveP = sumP/4;
DispArray.push_back(aveP);
}
incrV++;
}
```

As discussed in Chapter 3, these attributes are then formatted and output to an ARFF file. The code below is used to open and format an ARFF file. This file may be appended to if the file already exists.

The code below shows how the discretized groups of displacements contained in `DispArray` are written as a string of features separated by commas. A "?" label is given at the end of each string of features to signify that it is of an unknown class. The class label can then be changed manually to the correct class if needed to act as a training or test set.

```
//If an attribute file does not exists then create
if( !file )
{ attributeFile.open ("C:\\ModeResearch\\attributeFile.arff");
attributeFile << "@relation Mode_Shape\n\n";
for (int i=0;i<100;i++) attributeFile << "@attribute Section" << i+1 << " real
    \n";
attributeFile << "@attribute class {Mode1, Mode2, Mode3, Mode4, Mode5, Mode6,
    Mode7, Mode8, Junk}\n";
attributeFile << "\n@data\n";
}
//If an attribute file exists then open and append to it
else attributeFile.open ("C:\\ModeResearch\\attributeFile.arff",ios::app);
```

```
for (int i=0;i<100;i++) attributeFile << DispArray[i] << ",";
attributeFile << "?\n";
attributeFile.close();
```

4.3 Machine Learning

The training and testing of algorithms to be used as classifiers was done in Weka for this research. Again, a classifier is a function that can identify which class a new instance belongs to. In order to obtain a classifier that will properly identify the mode shape for a given part we first must train and test an algorithm for classification.

4.3.1 Define Training Data

To obtain an initial training set the designer must first run the task mentioned at the beginning of this chapter. For this research we constructed a training set by changing the baseline design of a model by $\pm 10\%$. For example, if a baseline design has a length of 10 inches the training set would include designs with lengths from 9 to 11 inches. After constructing the training data the designer must then label the instances in the ARFF file manually. To correctly label these mode shapes the designer may view each parts corresponding contour plot that results from the modal analysis. The training set can be appended to through consequent runs in order to create a larger training set. A larger training set may be desired if more training examples are needed to more accurately and consistently identify mode shapes. These issues of accuracy will be addressed in subsequent sections.

4.3.2 Selection of Machine Learning Algorithm

As described in Chapter 3, three algorithms to apply to this classification problem were chosen: k-Nearest Neighbors (k-NN), Decision Trees, and Support Vector Machines (SVM). These algorithms are available in Weka's Explorer application. The Explorer is an environment for exploring data with Weka [6]. In Weka the IBk algorithm was chosen which implements the k-NN approach as discussed. To implement a decision tree learner, Weka's J48 and Random Forest were chosen. J48 is a simple decision tree as described in Chapter 3, whereas Random Forests operate by constructing a multitude of decision trees and outputting the class that is the mode of the classes output by individual trees. Weka's SMO algorithm was chosen to implement the SVM approach. More detail about Weka's Explorer application and each algorithm used in this implementation can be viewed in Weka's documentation [6].

4.3.3 Training

The training of each algorithm was also done using Weka's Explorer application with the training set obtained from the previous pre-processing step. Leave-One-Out Cross Validation (LOOCV) was used to evaluate how well each algorithm was able to learn the training data. LOOCV is a special case of the general k-fold cross validation method. LOOCV works by taking a data set with n examples and performs n experiments. LOOCV uses $n-1$ examples for training and the remaining example for testing. An example of how the data is split in LOOCV as described is shown in Figure 4.5. The overall accuracy can be obtained by averaging the accuracies computed on each experiment. LOOCV is used in this research in order to allow for sparse training data so as to train on as many examples as possible. LOOCV is easily implemented in Weka's Explorer environment by simply selecting "Cross-validation" and inputting the appropriate number of folds [6].

Algorithms are then analyzed using Weka's Experimenter which enables users to create, run, modify, and analyze experiments [6]. To ascertain if there were statistical differences between the selected algorithms the Experimenter's paired t-test was used and the results of which can be found in Chapter 5.

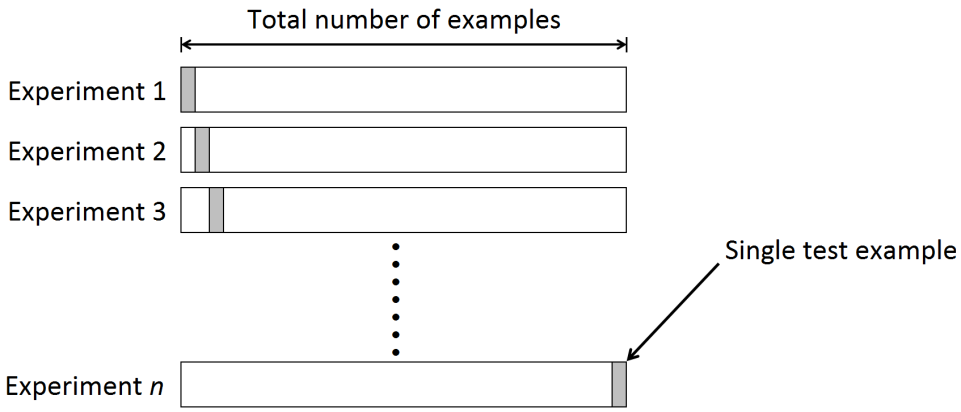

Figure 4.5: Splitting data in leave-one-out cross validation (LOOCV)

4.3.4 Evaluation with Test Set

In the research performed, after an appropriate learning algorithm was chosen it was tested by subjecting the classifier to models that contained changes in their baseline design ranging from $\pm 10\%$ to $\pm 100\%$ in order to ascertain the robustness of the method. In this way we see how far this method can extrapolate outside of the initial training examples given. These tests were evaluated in Weka and the results of the training and testing of these algorithms will be given in Chapter 5.

4.4 Identify Mode Shape

After a classifier has been obtained, the task shown at the beginning of this chapter can be run to identify vibrational mode shapes. In the task Weka is called from a batch file that runs a new solution set against the classifier and outputs its predictions. The training dataset location, algorithm, and the location of the dataset to be identified can all be input into the Weka batch file. An example of a command line prompt that is run from this batch file can be seen in the code below.

```
1 java -cp "C:/Program Files/Weka-3-6/weka.jar" weka.classifiers.lazy.IBk -t "C
    :/ModeResearch/attributeFileTrain.arff" -T "C:/ModeResearch/attributeFile.
    arff" > "C:/ModeResearch/myOutputFile1.arff"
```

CHAPTER 5. RESULTS

The goal of this research, as discussed in Chapter 1, is to develop a method to automatically identify mode shapes of a finite element modal analysis that improves on previous automated attempts. The developed method leverages parameterized geometric representations of vibrational mode shapes that allow machine learning to match differing parts of varied geometry or mesh coarseness. This method can be utilized in an iterative design process, such as an optimization or design of experiments as well as in a standalone operation.

While there are a number of modes that may be of interest to a designer, this research only looked at eight modes that result from a modal analysis for identification. These modes can be seen in Figure 5.1. Each of these modes signify their own class by being labeled as “Mode1”, “Mode2”, ... , “Mode8”. Any other mode that results from a modal analysis that cannot be classified as one of these modes is labeled as “Junk”. It is the task of the classifier to distinguish between the different classes given the displacement attributes provided.

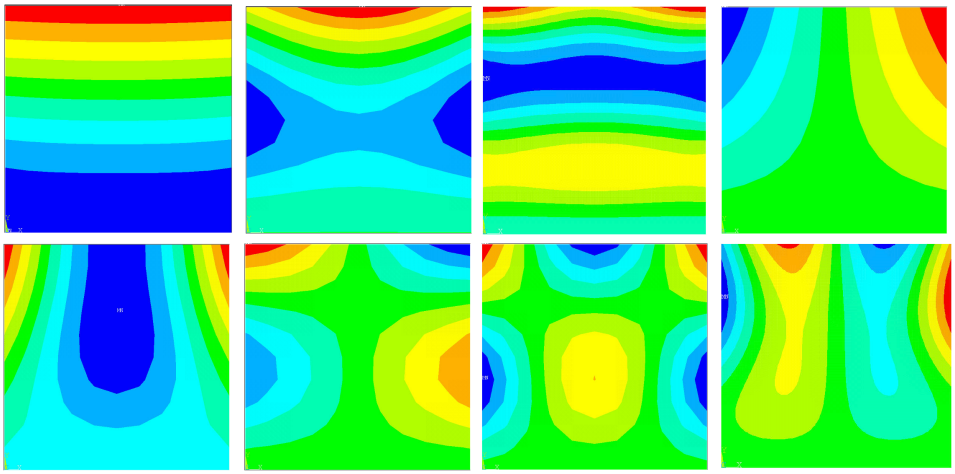

Figure 5.1: Eight trained mode shapes for identification

Section 5.1 of this chapter reports the results of selecting an algorithm based on a supplied training set. Section 5.2 shows the results of using the method to identify models that have been meshed with different levels of detail than of those in the training set, and details the effectiveness of the method in comparing parts with different geometrical definition. Section 5.3 presents the results of utilizing the method in an iterative design process.

5.1 Machine Learning - Algorithm Selection

The method was tested on three different geometries. First a simple rectangular plate defined by four parameters: length, width, thickness, and mesh coarseness. The second was tapered and twisted along with changes in length, top width, bottom width, thickness, twist, and mesh coarseness. Lastly a plate was designed similar to the second plate with the addition of two non-linear edges. A sample of these geometries can be seen in Figure 5.2.

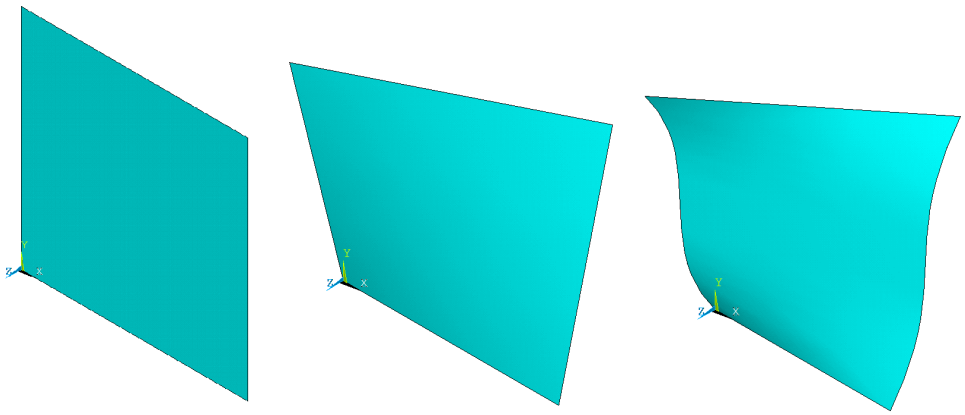

Figure 5.2: Isometric views of each test geometry

A training dataset of 200 instances was compiled containing variations in a model's baseline design of $\pm 10\%$ as described in Section 4.3.1. The simple rectangular plate described above was the only model used to compile this training set. The training set containing 200 instances consists of 20 instances for each class $Mode1$ through $Mode8$ and 40 instances for the *Junk* class. As discussed in Section 4.2.4, 100 features were pulled from each surface to create each instance in this research. This training set was used in the training and evaluation of each algorithm selected. How an algorithm performs can be seen visually in a layout known as a confusion matrix. For

an example, how the k-NN algorithm performs with the training set can be seen in the confusion matrix in Table 5.1 where LOOCV was used.

Table 5.1: Confusion Matrix resulting from LOOCV of training data using k-NN

Actual Class	Predicted Class								
	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode8	Junk
Mode1	20	0	0	0	0	0	0	0	0
Mode2	0	20	0	0	0	0	0	0	0
Mode3	0	0	20	0	0	0	0	0	0
Mode4	0	0	0	20	0	0	0	0	0
Mode5	0	0	0	0	20	0	0	0	0
Mode6	0	0	0	0	0	20	0	0	0
Mode7	0	0	0	0	0	0	20	0	0
Mode8	0	0	0	0	0	0	0	20	0
Junk	0	0	0	0	1	0	0	0	39

As can be seen by the values in the diagonal elements of the Table, the k-NN algorithm was able to learn the training data with 99.5% accuracy. Similar assessments were made with the other algorithms chosen. The algorithms were subjected to a paired t-test at a 5% significance level in Weka's Experimenter to assess their differences, and the results of which can be seen in Table 5.2. It is shown that there is no significant difference at the 5% significance level between the IBk algorithm and the RandomForest algorithm, whereas there is significant degradation in the J48 and SMO algorithms. Since it was found that there is not a significant difference between the IBk and Random Forest algorithms, IBk was used due to its simplicity, speed, and ability to learn the training set.

The machine learning algorithms used in this research were able to induce very accurate models to distinguish between classes based on displacement values. Once the displacement data that resulted from the modal analysis was processed into feature vectors, machine learning was able to learn the mode shapes with low error. Several tests were performed to show that these results hold over a range of models and parameters. These tests are presented in the next section. The success seen here using machine learning is consistent with others who have also utilized machine learning to realize great benefits [11, 26]

Table 5.2: Results of a Paired T-test

Dataset	(1)	(2)	(3)	(4)	
Mode-Shape	99.50	95.50 •	98.85	81.00	•

○, • statistically significant improvement or degradation

(1) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(2) trees.J48 '-C 0.25 -M 2' -217733168393644444
(3) trees.RandomForest '-I 10 -K 0 -S 1' -2260823972777004705
(4) functions.SMO '-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyKernel -C 250007 -E 1.0\"' -6585883636378691736

5.2 Mode Identification

By utilizing parametric geometries in this research, vibrational mode shapes are easily identified without regards to mesh coarseness. Figure 5.3 shows three geometrically identical models that have differing mesh densities and therefore a differing number of nodal displacements in their modal shape vectors. As expected, because the models in Figure 5.3 are geometrically identical the mode shapes determined from a modal analysis are also identical. The results of testing this method include such variations of mesh densities. Mesh density is dictated by the number of nodes along the width and length of the model and are defined in the baseline designs reported in Table 5.3. Table 5.3 contains the parameters that define each model used in this research.

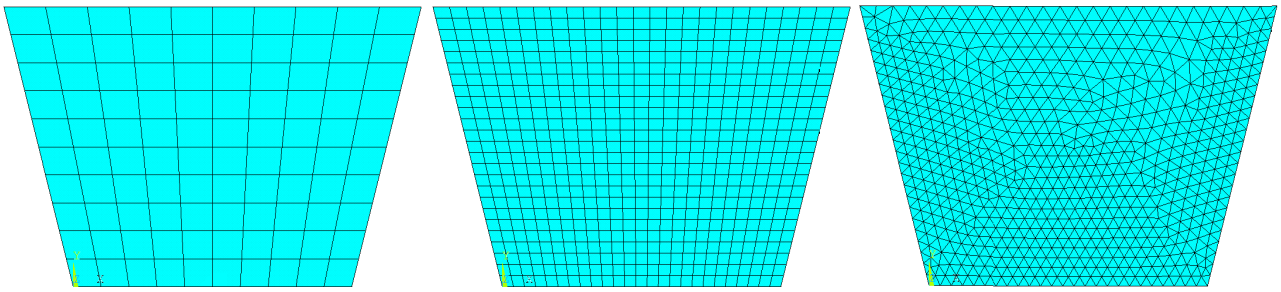

Figure 5.3: Three geometrically identical models meshed differently

In order for this method to be used in an iterative design process it must be able to identify modes that result from models that have varied some parameter value(s) of its baseline design. These changes in parameters may be small or large. Such change is illustrated in Figure 5.4 where the length of the model on the right has increased from its baseline design on the left.

Table 5.3: Parameters of baseline designs

Parameters of Baseline Designs (*inches*)			
	Rectangular Plate	Tapered Twisted Linear Plate	Tapered Twisted Non-linear Plate
Length	20	20	20
Width	20	N/A	N/A
Top Width	N/A	20	20
Bottom Width	N/A	20	20
Thickness	0.5	0.5	0.5
Nodes along Width	25	25	25
Nodes along Length	25	25	25
Twist	N/A	1	1

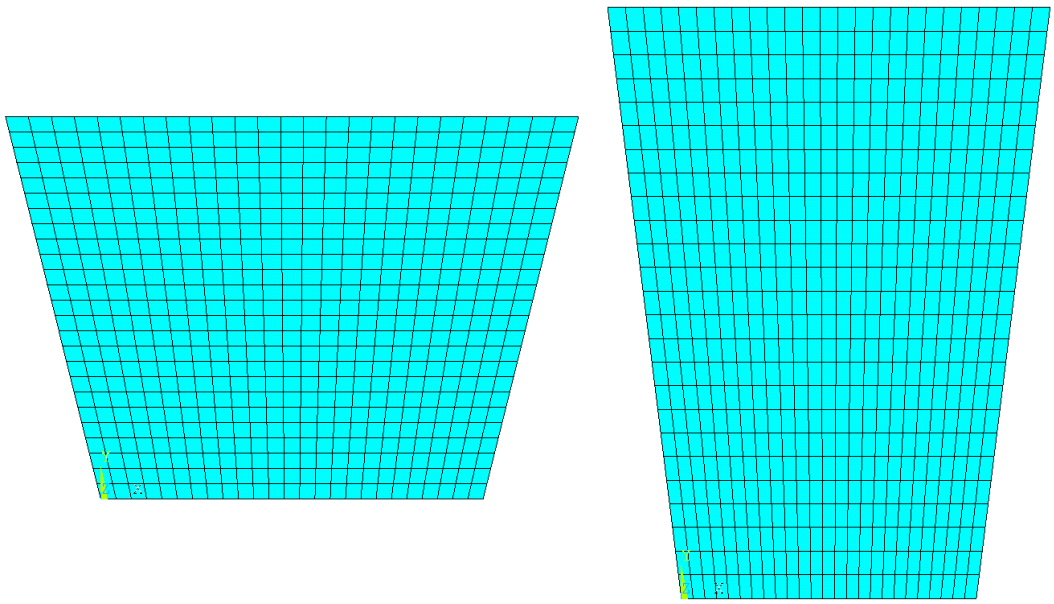

Figure 5.4: A baseline model and modified design

This method was subjected to three types of geometry as described earlier where each model's parameters were subjected to changes in their baseline design ranging from $\pm 10\%$ to $\pm 100\%$. The DOE in this method was used to produce the varied geometry according to a defined parameter percent variation. The tests were performed by compiling test sets of various sizes ranging from 20 to 50 instances per test set. Upon compiling a test set it was manually labeled to ensure that each instance that represented a mode shape was correctly labeled via visual inspection.

By labeling the instances in the test set Weka could then report how accurate it was in its prediction by comparison to the correctly labeled test set.

Tests were performed for each geometry type three times to produce an average accuracy. The average accuracies for the three generated tests are reported in Table 5.4. These results can be compared to the results from the research performed by Selin et al. shown in Table 5.5.

Table 5.4: Accuracy of the method averaged over three tests

Mode Identification Method Accuracy over 3 Tests										
	DOE Parameter Variation									
	±10%	±20%	±30%	±40%	±50%	±60%	±70%	±80%	±90%	±100%
Rectangular Plate	100%	100%	100%	100%	100%	99%	97%	97%	90%	98%
Tapered Twisted Linear Plate	100%	100%	100%	99%	96%	94%	92%	81%	76%	93%
Tapared Twisted Non-linear Plate	99%	100%	98%	97%	96%	97%	97%	89%	86%	84%

Table 5.5: Results from Selin et al. 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%
Tapared Twisted Non-linear Plate	91%	88%	87%	82%

The results of testing the rectangular plate show significant improvement over previous research. It can be seen from Table 5.4 that the accuracy of the mode identification method developed in this research is related to the amount of variation in model parameters. However, such degradation of accuracy does not come into effect until the variation reaches ±60%. Throughout the full range of parameter variation the results show that the average accuracy never falls below 90% for the rectangular plate. Through ±50% variation of the rectangular plate the accuracy was found to be 100%.

The tapered twisted linear plate showed a similar relationship between the amount of variation and the method accuracy with amplified effects. This loss of accuracy is largely due to the fact that the baseline design of the tapered twisted linear plate has different parameters than those

for the rectangular plate used in compiling the training data. By observing Table 5.4 it can be seen that this method was able to identify mode shapes with 100% accuracy through ±30% variation. At ±90% variation is found the lowest accuracy in the method of 76%.

Next, the tapered twisted non-linear plate model was used for testing. This model, having the most differences from the model used in the training set, showed results consistent with the relationship between variation and the method accuracy. While staying above 95% accurate through ±70% parameter variation, the average of the three tests show that for a tapered twisted non-linear plate this method is never 100% accurate. While the results show that at ±20% variation the method has 100% accuracy, because it is only 99% accurate at the ±10% variation level, we cannot say that this method is ever 100% accurate for the tapered twisted non-linear plate model.

By investigating which mode shapes were incorrectly matched, it was determined that the mismatches were mainly caused by two problems. The first problem was when one mode closely resembled another and the distinctions between the two were difficult to identify even by visual inspection. This problem may be amplified when looking at higher order modes when distinguishing between mode shapes is subtle. The function used to create a NURBS surface from nodal displacements determines an approximate surface from a cloud of points. As it is an approximation and does not pass through each point, the surface may not represent the mode shape as accurately as possible. This approximation could add to the problem of misidentifying similar mode shapes. An example of how mode shapes can blend into one another can be seen in Figure 5.5 where a distinct mode is represented at the ends of the Figure and are conflated with each other towards the middle.

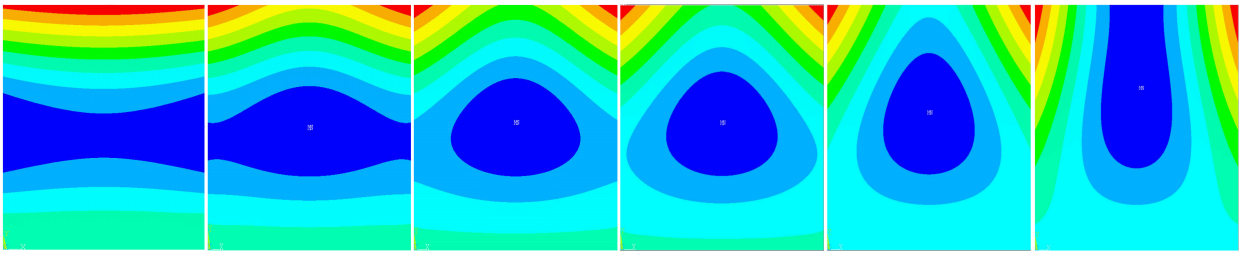

Figure 5.5: Sample of modes blending into another

The second problem that was observed was when a mode shape had not been properly trained. For example in Figure 5.6 both contour plots represent the same mode in bending, but

while examples of the mode on the left was included in the training set the mode on the right was not thus causing some error when the untrained mode was encountered. Both of these problems could be relieved by a larger training set that includes more examples of distinct mode shapes. The research performed has verified that by expanding the training set to include more examples these problems could be overcome. However, through continual use of this method new mode shapes would arise that exhibit the problems just mentioned and required further adjustment to the training set.

A final caveat that should be clear is that in Section 4.2.4 it was discussed that the displacements collected as attributes were in the z direction. This method looks at displacements perpendicular to a working plane. In this research the working plane was the xy plane and as such z displacements were used as feature attributes. If the designed model is significantly displaced outside of the working plane information about the model could be lost. If in varying parameters of a design a model should be displaced in a direction that is not primarily perpendicular to the working plane, then the proposed methods effectiveness is not guaranteed and new attributes that can describe the model should be explored. For this reason a designed model should stay relatively in a working plane for best results. For established design processes this should not pose a problem as a working plane can be readily identified.

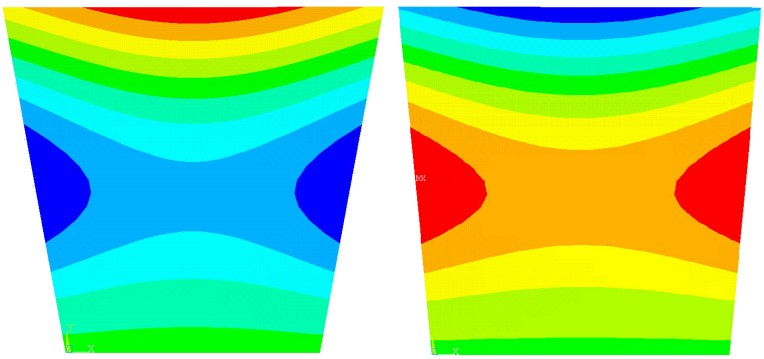

Figure 5.6: Sample of trained and untrained mode shape of same mode

Despite the inefficiencies found in this method, the results show that the identification of mode shapes using machine learning and parametric surfaces is feasible. The results reported here demonstrate improvement in overall accuracy over a larger design space using machine learning

over previous attempts using a MAC calculation. The advantage of using machine learning in this method is its ability to be modified to more accurately identify mode shapes. By modifying or adjusting algorithms or training data, a classifier can become more accurate in its predictions. Machine learning also allows for additional descriptive attributes to be used in conjunction with such attributes of displacements used in this research thus allowing for the identification of more complex parts. For example, by using the number of peaks and valleys present in a mode shape may help identify higher order modes such as the mode shown in Figure 5.7.

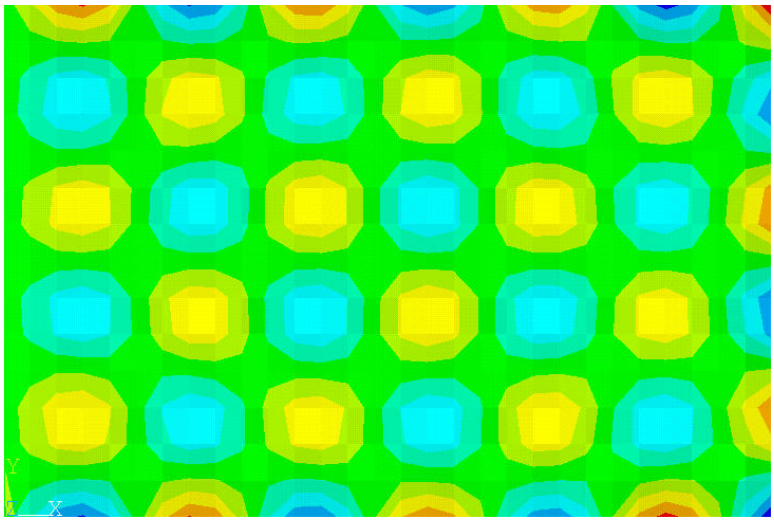

Figure 5.7: Example of a high order mode shape

5.3 Mode Identification in Iterative Design

As discussed in previous chapters, this method may be implemented as an iterative design process. Figure 5.8 shows an example of this method set up to run through the task shown. The task starts with a DOE that initializes parameter values and passes them into the subflow which creates a new modal analysis script, and solves the modal analysis in ANSYS. The task then performs the pre-processing step as described in Section 4.2 to prepare the modal analysis results for identification. The last component in the task calls Weka for the identification of mode shapes resulting from the modal analysis results.

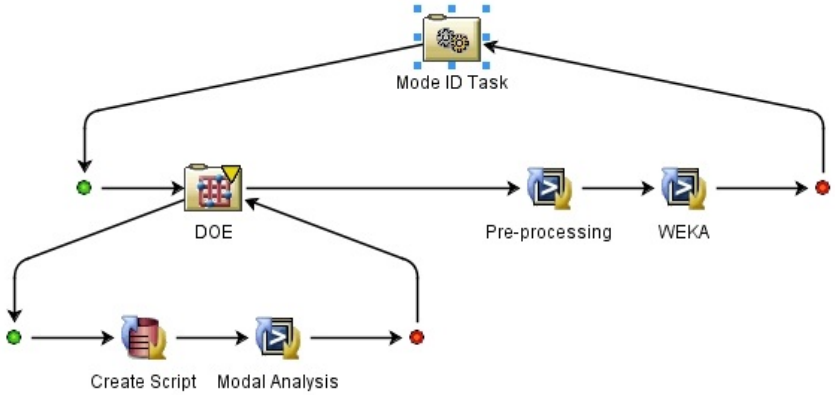

Figure 5.8: Isight mode identification task

By automatically identifying mode shapes significant time savings have been realized over the current method of visual inspection. To test the time required to identify vibrational mode shapes in this method 40 runs were completed, each run producing 5 mode shapes to be identified. These results are benchmarked against the current method of visual inspection and to previous work performed by Selin et al. in Table 5.6. Due to the similarities in approach, the results for time required via visual inspection of mode shapes will be those reported by Selin et al. [2]. Selin's research used both parametric curves as well as surfaces. Since the proposed research uses parametric surfaces, the time saving reported herein will be comparing time to identify mode shapes using parametric surfaces only.

Table 5.6: Benchmarking time required of mode shape identification methods

Identification Method	Total Time (sec)	# of Matches	Time per Match (sec)
Proposed Method	159.86	200	0.7993
Selin et al.	311.07	200	1.55535
Visual	2433.28	200	12.1664

When this method was used as illustrated in Figure 5.8 it took an average of 0.7993 seconds per match. Benchmarking the proposed method delivers time savings twice as fast as the method using the MAC and parametric surfaces proposed by Selin et al. and 15 times faster than visual inspection. While it is possible that using this method may result in an incorrect identification

of mode shapes as discussed in Section 5.2, the significant time savings that are achieved present a compelling argument for the use of this method even if the identification is not always 100% accurate.

CHAPTER 6. CONCLUSIONS

This research shows that machine learning, which has been used for a wide array of applications to solve classification and regression problems, can be effectively used to identify the mode shapes of dissimilar finite element models. This is possible through the representation of the model's modal analysis results as parametric surfaces which allows parts with different geometric definitions and mesh coarseness to be matched to trained mode shapes. By using parametric surfaces in this research, the behavior of a mode shape can be well defined over the entire part given that descriptive attributes that use information over the whole model are utilized.

The surface approximation function in the NX API make it possible to use the nodal position and displacement data to create parameterized surfaces that represent specific mode shapes to be used in this method. Once the results from a finite element model are transferred into a parameterized geometric form, attributes are then easily collected to be used for classification using machine learning. Machine learning is then able to identify mode shapes between models of different mesh density and geometric definition given a trained classifier. This method has shown to have high accuracy and over a large design space when applied to four-edge surfaces. The high accuracy achieved by this method suggests that the chosen attributes of displacements was sufficient in describing the mode shape surfaces used. The results of this research also suggests that machine learning could be a valuable tool in identifying vibrational mode shapes of geometries other than four-edge surfaces provided proper feature attributes can be obtained.

By automating the mode identification process, more detail about a parts dynamic properties can be obtained without having to visually inspect each modal solution. The greatest benefit of this research can be realized when implementing an iterative design process, such as optimization or design of experiment. In iterative design processes the geometric parameters of a model are modified with each iteration. By using parametric surfaces to represent mode shapes, models of differing geometric or mesh properties to be successfully identified. While this method is not

100% accurate in the identification of the modes contained in the analysis results over all possible variations, its high accuracy can provide a designer with an understanding of the part's properties in the design process.

Comparing the accuracies of this research that uses machine learning with prior work by Selin et al. that uses a MAC calculation to identify mode shapes indicate that machine learning provides more accurate results over a broader design space. The time required to execute the proposed method has twice the time saving realized by Selin's work using parametric surfaces and 15 times faster than visual inspection of results. These time savings can become even more significant when a large number of modes are identified within an iterative process.

6.1 Recommendations

A limitation of the developed mode identification method, as mentioned previously, is that only the nodal displacements in the z direction are being taken into account to define the attributes that represent mode shapes. An improvement would be realized if features of the model could be identified that described the model without respect to its initial positioning in real (xyz) space. This would retain the level of detail captured in the NURBS surface representations of mode shapes, and allow for an even broader design space to be used. This is due to the fact that information about the model could be retained regardless of a relation to a workplane.

The surface fitting algorithm that is called using the NX Open C API is another limitation in the developed method. The method is not reliable in fitting a surface through a set of points that accurately describes the finite element model, especially as higher order mode shapes are encountered. To improve the surface fitting required by this research another method that is better capable of fitting surfaces through each point in a point cloud as opposed to using a least squares approximation would be have to be realized. In so doing, NURBS surface representations of mode shapes would become more accurate and could help alleviate the problem of misidentification.

If another method were to be realized that does not require a CAD environment for the fitting of surfaces, then more benefits in the form of time savings could be realized. Currently, a portion of the execution time within the process is starting a CAD session when the method is run. This is done in order to call the function from the API which constitutes a significant amount

of time. If a surface fitting function that is independent from CAD could be used then more time savings would be achievable.

Identification of mode shapes using machine learning comes with some uncertainty. Quantified uncertainty in the mode shape identification process is an important issue that can be leveraged to aid a designer in understanding the results of the proposed method. What kind of uncertainty or confidence level that is available is algorithm dependent and may not always be easily available. Developing a means to leverage this uncertainty would improve this method by allowing the identification of mode shapes that fall below a specified uncertainty level to be handled by the designer.

While this research only applies to two dimensional finite element models, the ability of machine learning to choose attributes that can describe a model's features can also be used to identify more complex models. Further research in obtaining feature attributes that can describe three dimensional finite element results would prove useful in more complex analyses.

REFERENCES

[1] Gade, S., Herlufsen, H., and Konstantin-hansen, H., 2002. "How to determine the modal parameters of simple structures." *Sound & Vib.,* **36**(1), pp. 72–73. 1, 7

[2] Selin, E., 2012. "Application of parametric nurbs geometry to mode shape identification and the modal assurance criterion." Master's thesis, Brigham Young University, April. 1, 7, 46

[3] Mitchell, T. M., 1997. *Machine Learning*. McGraw-Hill. 1, 8, 9, 21, 23, 24

[4] Alpaydin, E., 2010. *Introduction to Machine Learning*., 2 ed. MIT, Cambridge, Massachusetts. 1, 8, 9, 21, 22, 23, 24

[5] Kotsiantis, S. B., 2007. "Supervised machine learning: A review of classification techniques." *Informatica,* **31**, pp. 249–268. 3, 23, 24

[6] Mark Hall, Eibe Frank, G. H. B. P. P. R. I. H. W., 2009. "The weka data mining software: An update." *SIGKDD Explorations,* **11**(1). 4, 23, 35

[7] Burns, L., 2004. "MAC Evaluations Utilized in FEA Analysis for Mode Identification." *Conference & Exposition on Structural Dynamics,* **4**(1), pp. 2008–2017. 7

[8] Ewins, D. J., 2009. *Modal Testing: Theory, Practice and Application*., 2nd ed. Wiley. 7

[9] Hearn, G., and Testa, R., 1991. "Modal analysis for damage detection in structures." *Journal of Structural Engineering,* **170**(10), pp. 3042–3063. 7

[10] Harrington, P., 2012. *Machine Learning in Action*. Manning. 7

[11] Satish B. Satpal, Yogesh Khandare, A. G., and Banerjee, S., 2013. "Structural health monitering of a cantilever beam using support vector machine." *Journal of the Acoustical Society of America,* **5**(1), pp. 1–7. 8, 39

[12] Liu, C., and Fujisawa, H., 2007. Classification and learning for character recognition: Comparison of methods and remaining problems National Laboratory of Pattern Recognition (NLPR),Beijing, China. 8

[13] Hung, S., and Jan, J., 1997. "Machine Learning in Engineering Design - An Unsupervised Fuzzy Neural Network Case-Based Learning Model." *Intelligent Information Systems*, pp. 156–160. 9

[14] Bishop, C. M., 2006. *Pattern Recognition and Machine Learning*. Springer. 9

[15] Witten, I. H., and Frank, E., 2005. *Data Mining: Practical Machine Learning Tools and Techniques*., 2nd ed. Elsevier. 9

[16] Piegl, L., and Tiller, W., 1997. *The NURBS Book*. Springer-Verlag Berlin Heidelberg, New York, New York. 9, 11

[17] Zeid, I., 2005. *Mastering CAD/CAM*. McGraw-Hill, Boston. 10, 11

[18] Sederberg, T. W., 2012. *Computer Aided Geometric Design - Course Notes*. 11

[19] Wolpert, D. H., and Macready, W. G., 1995. No free lunch for search The Sante Fe Institute, February. 21

[20] Zanifa Omary, F. M., 2010. "Machine learning approach to identifying the dataset threshold for the performance estimators in supervised learning." *International Journal for Infonomics,* **3**(3), pp. 314–325. 21

[21] Alpha, W., 2012. Paired t-test http://mathworld.wolfram.com/Pairedt-Test.html. 21

[22] David W. Aha, Dennis Kibler, M. K. A., 1991. *Instance-based learning algorithms*., Vol. 6 Springer. 22

[23] Georgios Paliouras, Vangelis Karkaletsis, C. D. S., 2001. *Machine Learning and Its Applications: Advanced Lectures*. Springer. 22

[24] Ins M. Galvn, Jo M. Valls, M. G. P. I., 2011. "A lazy learning approach for building classification models." *International Journal of Intelligent Systems,* **26**(8), pp. 773–786. 22

[25] Rong Xiao, Jicheng Wang, F. Z., 2000. "An approach to incremental svm learning algorithm." *Tools with Artificial Intelligence,* **12**, pp. 268–273. 24

[26] Skowronski, M., and Harris, J., 2006. "Acoustic detection and classification of microchiroptera using machine learning: Lessons learned from automatic speech recognition." *International Journal of Advanced Structural Engineering,* **119**(3), pp. 1817–1833. 39