Linear and Nonlinear Dimensionality-Reduction-Based Surrogate Models for Real-Time Design Space Exploration of Structural Responses

Gregory David Bird
Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Engineering Commons

BYU ScholarsArchive Citation
Bird, Gregory David, "Linear and Nonlinear Dimensionality-Reduction-Based Surrogate Models for Real-Time Design Space Exploration of Structural Responses" (2020). Theses and Dissertations. 8653. https://scholarsarchive.byu.edu/etd/8653

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.
Linear and Nonlinear Dimensionality-Reduction-Based Surrogate Models for
Real-Time Design Space Exploration of Structural Responses

Gregory David Bird

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

Master of Science

Steven E. Gorrell, Chair
John L. Salmon
S. Andrew Ning

Department of Mechanical Engineering
Brigham Young University

Copyright © 2020 Gregory David Bird
All Rights Reserved
Design space exploration (DSE) is a tool used to evaluate and compare designs as part of the design selection process. While evaluating every possible design in a design space is infeasible, understanding design behavior and response throughout the design space may be accomplished by evaluating a subset of designs and interpolating between them using surrogate models. Surrogate modeling is a technique that uses low-cost calculations to approximate the outcome of more computationally expensive calculations or analyses, such as finite element analysis (FEA). While surrogates make quick predictions, accuracy is not guaranteed and must be considered. This research addressed the need to improve the accuracy of surrogate predictions in order to improve DSE of structural responses. This was accomplished by performing comparative analyses of linear and nonlinear dimensionality-reduction-based radial basis function (RBF) surrogate models for emulating various FEA nodal results. A total of four dimensionality reduction methods were investigated, namely principal component analysis (PCA), kernel principal component analysis (KPCA), isometric feature mapping (ISOMAP), and locally linear embedding (LLE). These methods were used in conjunction with surrogate modeling to predict nodal stresses and coordinates of a compressor blade. The research showed that using an ISOMAP-based dual-RBF surrogate model for predicting nodal stresses decreased the estimated mean error of the surrogate by 35.7% compared to PCA. Using nonlinear dimensionality-reduction-based surrogates did not reduce surrogate error for predicting nodal coordinates. A new metric, the manifold distance ratio (MDR), was introduced to measure the nonlinearity of the data manifolds. When applied to the stress and coordinate data, the stress space was found to be more nonlinear than the coordinate space for this application. The upfront training cost of the nonlinear dimensionality-reduction-based surrogates was larger than that of their linear counterparts but small enough to remain feasible. After training, all the dual-RBF surrogates were capable of making real-time predictions. This same process was repeated for a separate application involving the nodal displacements of mode shapes obtained from a FEA modal analysis. The modal assurance criterion (MAC) calculation was used to compare the predicted mode shapes, as well as their corresponding true mode shapes obtained from FEA, to a set of reference modes. The research showed that two nonlinear techniques, namely LLE and KPCA, resulted in lower surrogate error in the more complex design spaces. Using a RBF kernel, KPCA achieved the largest average reduction in error of 13.57%. The results also showed that surrogate error was greatly affected by mode shape reversal. Four different approaches of identifying reversed mode shapes were explored, all of which resulted in varying amounts of surrogate error. Together, the methods explored in this research were shown to decrease surrogate error when performing DSE of a turbomachine compressor blade. As surrogate accuracy increases, so does the ability to correctly make engineering decisions and judgements throughout the design process. Ultimately, this will help engineers design better turbomachines.
Keywords: design space exploration, surrogate modeling, dimensionality reduction, principal component analysis, kernel principal component analysis, isometric feature mapping, locally linear embedding, finite element analysis, modal analysis, modal assurance criterion, turbomachinery, compressor blades
ACKNOWLEDGMENTS

First, I would like to thank my advisor, Dr. Gorrell, for his support. I would not have been able to complete this research without his direction and expertise. I am grateful to have worked with him on a weekly basis over these last two years. I also thank the rest of my committee, Dr. Salmon and Dr. Ning, for supporting this research. I especially thank Dr. Salmon for serving as a co-advisor and participating in our weekly meetings. As a whole, the time and effort spent on my behalf by my committee has enabled me to succeed in my endeavors.

I would also like to thank all of my colleagues in the research lab. They provided invaluable feedback and suggestions to code, CAD, FEA, and technical writing that guided me in the research. I especially thank Spencer Bunnell for his expertise and friendship throughout this process.

Lastly, I thank my family for their constant and continual support. Their encouragement throughout my life to seek more education ultimately led me to where I am today.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Page</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>Abstract</td>
<td>ii</td>
<td></td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
<td></td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
<td></td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
<td></td>
</tr>
<tr>
<td>Nomenclature</td>
<td>ix</td>
<td></td>
</tr>
<tr>
<td>Chapter 1 Introduction</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1.1 Design Space Exploration</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1.2 Challenges of Design Space Exploration</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1.3 Research Objectives</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Chapter 2 Background</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2.1 Structural Analysis</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2.2 Surrogate Modeling</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>2.2.1 Applications of Surrogate Models of FEA</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>2.3 Dimensionality Reduction</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>2.3.1 Types of Dimensionality Reduction</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>2.3.2 Applications of Dimensionality Reduction</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>2.4 Turbomachinery Blade Model</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Chapter 3 Dimensionality-Reduction-Based Surrogates for Emulating Nodal Stresses and Coordinates</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>3.2 Background</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>3.2.1 Surrogate Modeling</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>3.2.2 Dimensionality Reduction</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>3.3 Methods</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>3.4 Results</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>3.4.1 Measuring Linearity of the Manifold</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>3.4.2 Computational Cost</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>3.5 Conclusion</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>Chapter 4 Dimensionality-Reduction-Based Surrogates for Emulating Nodal Displacements and the Modal Assurance Criterion</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>4.2 Background</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Modal Assurance Criterion</td>
<td>48</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Surrogate Modeling</td>
<td>49</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Dimensionality Reduction</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Methods</td>
<td>56</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Training and Testing Data Generation</td>
<td>58</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Surrogate Model Training and Testing</td>
<td>59</td>
</tr>
<tr>
<td>4.4</td>
<td>Results</td>
<td>62</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Linearity of the Manifold</td>
<td>66</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Mode Shape Reversal</td>
<td>69</td>
</tr>
<tr>
<td>4.5</td>
<td>Conclusion</td>
<td>73</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>Conclusions</td>
<td>75</td>
</tr>
<tr>
<td>5.1</td>
<td>Research Contributions</td>
<td>75</td>
</tr>
<tr>
<td>5.2</td>
<td>Recommended Work</td>
<td>77</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Exploration of More Dimensionality Reduction Methods</td>
<td>77</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Optimization of Hyperparameters</td>
<td>78</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Identification of Mode Shape Reversal</td>
<td>78</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Application Beyond FEA</td>
<td>78</td>
</tr>
<tr>
<td>5.2.5</td>
<td>DSE of Other Parts</td>
<td>79</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>80</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Important Code</td>
<td>84</td>
</tr>
<tr>
<td>A.1</td>
<td>Linear Dimensionality-Reduction-Based Dual-RBF Surrogate Training</td>
<td>84</td>
</tr>
<tr>
<td>A.2</td>
<td>Nonlinear Dimensionality-Reduction-Based Dual-RBF Surrogate Training</td>
<td>89</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

2.1 Mode names and frequencies ......................................................... 10  
2.2 Purdue blade parameters .......................................................... 21  

3.1 Purdue blade parameters for the stress and coordinate study ................. 33  
3.2 Hyperparameters of scikit-learn’s dimensionality reduction classes .......... 35  
3.3 Regression results for predicting the double-NRMSE for stress and coordinates 39  
3.4 Double-NRMSE for PCA and ISOMAP at different levels of the MDR .......... 43  
3.5 Computational cost of the dimensionality reduction algorithms ............... 44  

4.1 Purdue blade parameters for the mode shape study ................................ 57  
4.2 Mode names and frequencies ....................................................... 58  
4.3 Hyperparameters of scikit-learn’s dimensionality reduction classes .......... 61  
4.4 NRMSE in the five parameter design space with 100 training samples .......... 63  
4.5 Mixed model regression results for predicting the NRMSE for MAC values .... 64  
4.6 Predicted NRMSE for each dimensionality reduction method ................. 65  
4.7 MDRs of modes four and five ...................................................... 68  
4.8 Node displacements of various designs .......................................... 70  
4.9 NRMSE for each eigenvector correction approach ................................ 71
# LIST OF FIGURES

1.1 Design space exploration .................................................. 2
1.2 Manifold geometry .......................................................... 4

2.1 Finite element analysis ..................................................... 8
2.2 Structural analysis: static and dynamic ................................. 9
2.3 Mode shapes ................................................................. 10
2.4 Surrogate model example ................................................... 11
2.5 Dimensionality reduction ................................................... 13
2.6 Principal component analysis ............................................. 15
2.7 Kernel principal component analysis ................................... 17
2.8 Isometric feature mapping: geodesic distance ....................... 18
2.9 Locally linear embedding .................................................. 19
2.10 Purdue blade and disk .................................................... 21

3.1 Finite element analysis: mesh and stress contour .................. 25
3.2 Dimensionality reduction ................................................... 26
3.3 Principal component analysis ............................................. 28
3.4 Kernel principal component analysis ................................... 29
3.5 Isometric feature mapping: geodesic distance ....................... 30
3.6 Locally linear embedding .................................................. 31
3.7 Purdue blade and disk .................................................... 32
3.8 Automated workflow for the stress and coordinate study ........ 34
3.9 Box plots of the double-NRMSE of stress and coordinate predictions ........................................ 37
3.10 Manifold distance ratios for the stress and coordinate data manifolds ........................................ 42

4.1 Finite element analysis: mesh and stress contour .................. 50
4.2 Dimensionality reduction ................................................... 51
4.3 Principal component analysis ............................................. 52
4.4 Kernel principal component analysis ................................... 54
4.5 Isometric feature mapping: geodesic distance ....................... 55
4.6 Locally linear embedding .................................................. 56
4.7 Purdue blade and disk .................................................... 57
4.8 Mode shapes ................................................................. 58
4.9 Automated workflow for the mode shape study ..................... 59
4.10 True vs predicted MAC values ......................................... 72
## NOMENCLATURE

### Simulation Modeling

- **CAD**: Computer-Aided Design  
- **CFD**: Computational Fluid Dynamics  
- **D**: Damping matrix  
- **F**: Load matrix  
- **FEA**: Finite Element Analysis  
- **K**: Stiffness matrix  
- **M**: Mass matrix  
- **σ**: Stress  
- **u**: Displacement matrix  
- **\(\dot{u}\)**: First time derivative of **u**  
- **\(\ddot{u}\)**: Second time derivative of **u**

### Surrogate Modeling

- **c**: Design sample matrix  
- **DOE**: Design of experiments  
- **DSE**: Design space exploration  
- **λ**: Kernel evaluation matrix of new designs  
- **n**: Number of design samples  
- **NRMSE**: Normalized Root Mean Square Error  
- **φ**: Kernel function  
- **Φ**: Kernel evaluation matrix of design samples  
- **RBF**: Radial basis function  
- **W**: Kernel function weights  
- **X**: Input matrix  
- **Y**: Output matrix

### Dimensionality Reduction

- **A**: Squared distance matrix  
- **B**: Centering matrix  
- **D**: Distance matrix  
- **ISOMAP**: Isometric Feature Mapping  
- **k**: Number of nearest neighbors  
- **K**: Constructed matrix used to compute eigenvectors and eigenvalues  
- **KPCA**: Kernel Principal Component Analysis  
- **Λ**: Eigenvalue matrix  
- **LLE**: Locally Linear Embedding  
- **MDR**: Manifold distance ratio  
- **MDS**: Multidimensional Scaling  
- **n**: Number of design samples or data points  
- **φ**: Transformation function  
- **PC**: Principal Component  
- **PCA**: Principal Component Analysis
$S$ Matrix from SVD
$SVD$ Singular Value Decomposition
$U$ Matrix from SVD
$V$ Eigenvector matrix
$W$ LLE algorithm weights
$X$ High-dimensional data set
$Y$ Low-dimensional data set
CHAPTER 1. INTRODUCTION

Design space exploration (DSE) is a tool used to evaluate and compare designs as part of the design selection process. When used correctly, DSE helps engineers determine superior designs. Evaluating every possible design in a design space is infeasible. However, understanding design behavior and response throughout the design space may be accomplished by evaluating a subset of designs and interpolating between them. This process is often hindered by computationally expensive analyses, such as finite element analysis (FEA), when time and budget constraints limit the available resources. While computational power continues to increase, so does the need for more complicated models. Evaluating fewer designs will decrease interpolation accuracy between explored designs and in turn decrease the optimality of the design selection. This research addressed the need to improve the accuracy of interpolations within a design space. This was accomplished by comparing dimensionality-reduction-based surrogate models and their ability to accurately predict FEA results of a jet engine compressor blade in real time. To the author’s knowledge, such a comparison has not been published in the literature. Various dimensionality-reduction-based surrogate models were applied to the FEA results of a jet engine compressor blade, as jet engines play an integral part in society.

1.1 Design Space Exploration

Design space exploration (DSE) is a systematic analysis of designs defined by a series of parameters. A design space consists of all possible parameter configurations for a given model. DSE is a tool frequently used by engineers to explore these design possibilities in search of an optimal design. The process of performing DSE typically involves changing parameters within a parameterized model to create new designs that are evaluated and compared to determine their feasibility. This is illustrated in Fig. 1.1. The nominal geometry of a compressor blade has a max von Mises stress of 41.4 kilopound per square inch (ksi) in the lower right corner of the blade.
If a designer seeks to decrease this value to below 35 ksi, one option is to decrease the length of the tip chord. Through DSE, a more robust structural design is found. DSE may be performed behind the scenes as part of an automated optimization routine, or it may be an interactive process manually performed by an engineer who visually interprets and analyzes results. Interactive DSE can provide useful information and insights to a designer that go beyond design selection, such as the relationship between the design variables and the results of interest. Regardless of the method used, DSE helps determine optimal design configurations that satisfy the objective or constraint criteria. DSE is often used in the aerospace industry, such as in the design of turbomachinery, to facilitate better design selection. When performed properly, DSE leads to the design of better performing turbomachines that meet the structural demands, such as withstanding fatigue and centrifugal loading, of increasingly more complex engines.

![Design Space Exploration](image)

Figure 1.1: An example of design space exploration. The nominal geometry (a) has a higher max stress than the shortened chord geometry (b). Each figure has its own scale.

### 1.2 Challenges of Design Space Exploration

DSE is a useful tool that aids engineers in the design selection process. However, DSE faces challenges that make thorough exploration of the design space difficult: an infinite number of possible designs and computationally expensive analyses. The former of these challenges occurs
in a continuous design space (i.e. model parameters are continuous) and can be addressed by using a design of experiments (DOE). A DOE is a clever way of sampling the design space so as to cover it in the best way possible with a limited number of designs. By carefully selecting sections of the design space to explore, a designer can understand the unexplored space by interpolating between designs. These interpolations, whether computed or simply perceived by an engineer, become more accurate as more designs are sampled in the DOE. Many DOE methods exist for determining which points to explore in the design space, such as Latin hypercube sampling and Taguchi arrays [1, 2]. The later challenge mentioned above prevents designers from evaluating designs within the time and computational constraints often imposed during the design process. Individuals and companies have deadlines and budgets that may limit the number of designs that can be evaluated. While the design space can be covered using a DOE, a limited number of sampled designs will lead to larger interpolations and, consequently, larger errors. Ultimately, this may lead to suboptimal design selection. This is especially true with turbomachinery, as models often involve ten or more parameters and require complex analyses, such as high-fidelity finite element analysis (FEA), to obtain useful and accurate results. Increasing interpolation accuracy will help engineers better determine optimal designs.

Surrogate modeling is a commonly used approach to interpolate between explored designs during DSE [3]. Surrogate models rely on a set of training data to create a mathematical relationship between sets of independent and dependent variables. Once trained, surrogate models can predict the outcome of an analysis without having to perform it. The accuracy of these predictions is affected by a number of factors, including the amount of training data, how the training data is selected, and the model complexity. When used with FEA, surrogates can predict FEA nodal properties (e.g. stress, coordinates, displacements) when given a set of geometric inputs (e.g. length, width, height). Surrogate modeling may also be coupled with dimensionality reduction to reduce the number of predictions that must be made by the surrogate [2]. Dimensionality reduction is a technique that seeks to reduce the number of variables, or dimensions, needed to explain the variance in a set of data. When done properly, data in a high-dimensional space may be mapped to a low-dimensional space with little information loss. This mapping may be linear or nonlinear, resulting in linear and nonlinear dimensionality reduction. Dimensionality reduction is most effective when the high-dimensional data lies on some low-dimensional manifold embedded within
the high-dimensional space, as illustrated in Fig. 1.2. When viewed from the correct angle, the three-dimensional data can be seen to lie on a two-dimensional subspace. The two-dimensional subspace could be discovered and unraveled using nonlinear dimensionality reduction, thus reducing the dimensionality of the data. High-dimensional FEA nodal data, where each node contributes to the dimensionality, may be reduced to a lower dimensionality using this technique.

![Three-dimensional view](image1.png) ![Two-dimensional view](image2.png)

Figure 1.2: An example of a low-dimensional manifold embedded in a high-dimensional space. When viewed from the correct angle, the three-dimensional data (a) can be seen to lie on a two-dimensional subspace (b).

Researchers have found that linear dimensionality-reduction-based surrogates can lead to faster predictions of FEA stresses with a negligible effect on accuracy [4]. This increases the number of interpolations that can be evaluated in a given amount of time but does not increase surrogate accuracy. Some data sets, however, contain nonlinear variances that can be discovered and preserved more accurately using nonlinear dimensionality reduction techniques [5–11]. In these cases, nonlinear dimensionality-reduction-based surrogates may lead to improved surrogate accuracy. For example, Franz et al. [12] compared linear and nonlinear dimensionality-reduction-based surrogates for predicting aerodynamic shock locations and intensities and found that the nonlinear dimensionality-reduction-based surrogate models resulted in surface pressure distributions that more accurately represented the shocks. Determining when a data set necessitates nonlinear dimensionality reduction is not straightforward, as neither the geometry of the data nor its
intrinsic dimensionality is typically known. Thus, comparative analyses of linear and nonlinear dimensionality-reduction-based surrogates are necessary to gain understanding of their application for emulating FEA results.

1.3 Research Objectives

The purpose of this research is to compare linear and nonlinear dimensionality-reduction-based surrogate models and their ability to decrease surrogate error when emulating FEA results of turbomachinery. Knowledge gained from this comparison will help build confidence in and improve the accuracy of surrogate models used for DSE. While this research focuses on the structural characteristics of a jet engine compressor blade, the methods used may be applied to other analyses and other components for further comparative studies. To accomplish this, the following objectives were completed:

1. Apply linear and nonlinear dimensionality-reduction-based surrogates to structural DSE of a jet engine compressor blade.

2. Perform a comparative analysis of linear and nonlinear dimensionality-reduction-based surrogates to determine their speed and accuracy in emulating FEA results from a static analysis, including nodal stresses and coordinates.

3. Perform a comparative analysis of linear and nonlinear dimensionality-reduction-based surrogates to determine their speed and accuracy in emulating FEA results from a dynamic analysis, including mode shapes (i.e. relative nodal displacements).

4. Measure the nonlinearity of a data manifold to increase understanding of when nonlinear dimensionality reduction should be applied to a data set.

The thesis proceeds in Chapter 2 with a review of structural analysis, surrogate modeling, and dimensionality reduction techniques. Following this, the research objectives are presented and discussed in Chapter 3 and Chapter 4. Chapter 3 is based on an unpublished manuscript that will be submitted for publication [13] and outlines the application of dimensionality-reduction-based surrogate models for emulating FEA results from a static analysis, including nodal stresses
and coordinates. Four different dimensionality reduction methods, both linear and nonlinear, are compared across 200 design spaces to demonstrate the capability of nonlinear dimensionality reduction to reduce surrogate error. Chapter 4 compares the same dimensionality reduction methods explored in Chapter 3 for a different application. Chapter 4 is based on an unpublished manuscript that will be submitted for publication [14] and demonstrates the use of nonlinear dimensionality-reduction-based surrogate models for emulating FEA results from a dynamic analysis, including mode shapes (i.e. relative nodal displacements) and subsequently the modal assurance criterion, more accurately than their linear counterparts. Finally, a conclusion of this research is presented in Chapter 5.
CHAPTER 2. BACKGROUND

The background section contains an overview of the structural analyses, surrogate modeling, and dimensionality reduction techniques used to perform DSE in this research. First, an overview of static and dynamic structural analysis is presented, detailing how and why these analyses are performed. Next, an introduction to surrogate modeling and its use in DSE, including the emulation of FEA results, is discussed. Then, an overview of linear and nonlinear dimensionality reduction is presented. While many methods exist, this section focuses on those used as part of this research. Finally, the turbomachinery blade geometry used in this research is shown and discussed.

2.1 Structural Analysis

Structural analysis of mechanical parts and systems is typically performed using FEA. When designing a jet engine compressor blade, structural analysis is used to predict how and when a blade will fail. This knowledge helps engineers avoid part failure during operation, thus ensuring the structural integrity of the engine. Structural analysis considers both the static (steady) and dynamic (vibratory) response, where the dynamic response adds a time or frequency component to the analysis. To perform these analyses, a part is discretized into nodes and elements to create a finite element mesh. This is shown in Fig. 2.1(a), where a compressor blade is discretized using a coarse mesh with approximately 1000 nodes for illustration purposes. The low node count makes it easier to see individual nodes and elements but produces less accurate results. High-fidelity FEA meshes may contain tens of thousands to millions of nodes. Boundary conditions may be applied directly to the nodes (typically a subset) to limit their degrees of freedom or possible directions for displacement. For a compressor blade, this might include constraining the nodes at the base of the blade so that they are fixed in place or to prevent them from rotating in a particular direction. Likewise, loading conditions are also applied directly to the nodes, such as pressures on the blade surfaces. Together, the boundary and loading conditions cause displacements in each uncon-
strained degree of freedom at each of the nodes in the finite element mesh. These displacements, $u$, are obtained by solving Eqn. 2.1, where the load matrix, $F$, contains the loading conditions applied in each unconstrained degree of freedom at each node and the stiffness matrix, $K$, contains material and geometric properties [15]. The displacements can be used to calculate other nodal properties, such as static stress, $\sigma$, as shown in Eqn. 2.2. Using this process, the finite element mesh in Fig. 2.1(a) was used to compute the static von Mises stress contour seen in Fig. 2.1(b). The course mesh used in Fig. 2.1(a) allows for better visualization of the individual nodes and elements but often leads to inaccurate stress results, such as large changes in stress from element to element as seen in Fig. 2.1(b). The exact equation used to compute stress is dependent on the stress-strain and strain-displacement relations of the elements used in the model but ultimately is a function of the calculated displacements.

$$Ku = F$$  \hspace{1cm} (2.1)

$$\sigma = f(u)$$  \hspace{1cm} (2.2)

![Finite element mesh and stress contour](image)

(a) Finite element mesh

(b) Finite element stress contour

Figure 2.1: The finite element mesh (a) is used to discretize the compressor blade into smaller nodes and elements. Boundary and loading conditions are applied at these nodes, causing the stress contour depicted in (b).
In addition to static analyses that provide information about the steady state response of a component, dynamic analyses measure the vibratory response. Vibration may occur due to mechanical or aerodynamic excitation and will change the structural response of the component over time. This is illustrated in Fig. 2.2. The static or steady-state response does not change with time. The overall response, however, combines the static and dynamic responses. When added to the static response, the dynamic response increases the maximum stress experienced by the part and adds a cyclical component to the stress level. These additional responses, if not considered, could cause part failure during operation due to yielding (exceeding the elastic limit of a material) or fatigue (weakening of a material due to cyclical loading). Dynamic structural analysis provides critical information that is unobtainable from a static analysis alone.

![Figure 2.2: Static and dynamic stresses combine to form the overall stress experienced by a component.](image)

Modal analysis is a tool within FEA for conducting a dynamic analysis. Modal analysis determines the mode shapes, or relative displacements at each node, and the natural frequencies at which they occur. Exciting these frequencies during operation, whether from a mechanical or aerodynamic phenomenon, could lead to part failure due to the additional vibratory responses. When designing jet engines, engineers must understand how and when these excitations occur so as to design against them. Modal analysis involves solving the equation of motion of a dynamic system
as shown in Eqn. 2.3, where $M$ and $D$ are the mass and damping matrices, respectively, that contain information about the mass and damping characteristics of the part, and $\dot{u}$ and $\ddot{u}$ are the first and second time derivatives of $u$, respectively. This is an eigenvalue problem and involves performing eigenvalue decomposition on the inverse of $M$. The resultant eigenvectors and eigenvalues yield the mode shapes and their frequencies [16]. Examples of these mode shapes for a compressor blade are shown in Fig. 2.3, and their corresponding names and frequencies are shown in Table 2.1.

$$M\ddot{u} + D\dot{u} + Ku = F$$ (2.3)

![Mode shapes of the nominal geometry of the Transonic Purdue Research Compressor.](image)

Figure 2.3: Mode shapes of the nominal geometry of the Transonic Purdue Research Compressor. Displacements have been normalized using min-max normalization.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Name</th>
<th>Frequency (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First bending</td>
<td>1263</td>
</tr>
<tr>
<td>2</td>
<td>First torsion</td>
<td>2850</td>
</tr>
<tr>
<td>3</td>
<td>Second bending</td>
<td>5428</td>
</tr>
<tr>
<td>4</td>
<td>Second torsion</td>
<td>7780</td>
</tr>
<tr>
<td>5</td>
<td>First chordwise bending</td>
<td>8042</td>
</tr>
</tbody>
</table>
2.2 Surrogate Modeling

Surrogate modeling is a technique that uses low-cost calculations to approximate the outcome of more computationally expensive calculations or analyses, such as FEA. “Surrogate” is an all-encompassing term for such approximation techniques, including radial basis functions, artificial neural networks, response surface models, etc [3, 17]. Surrogates are often used with DSE to increase the speed with which designs can be evaluated. A subset of designs from the design space, called a training set, is selected and evaluated using the computationally expensive, higher fidelity analysis. Typically, these training designs are selected using a DOE. The design parameters (inputs) and the analysis results (outputs) are then used to build and train the surrogate. Surrogate models interpolate between explored designs to predict the outcome of unexplored regions of the design space. An example of this is shown in Fig. 2.4, where select points from a polynomial function are selected to train a surrogate that can then interpolate between the training points. If an analysis involving more complex inputs and outputs were used in lieu of the polynomial function, the surrogate could decrease the time required to make new evaluations. While surrogates make quick predictions, accuracy is not guaranteed and must be considered. Surrogate accuracy generally improves with more training data [18].

![Figure 2.4: A surrogate model is created using select points from the data for training.](image-url)
A commonly used surrogate to interpolate between designs is the radial basis function (RBF). A radial function is a function where the value at each point only depends on the Euclidean distance between that point and a fixed point (e.g. the origin). For a set of known independent variables, \( X = \{x_1, x_2, \ldots, x_n\} \), and dependent variables, \( Y = \{y_1, y_2, \ldots, y_n\} \), a radial basis function solves for the weights, \( W \), of the radial kernels, \( \phi() \), centered at each of the \( n \) samples in \( X \), \( c_n \). The radial kernels evaluate the Euclidean distances using some kernel function, such as a Gaussian or multiquadric kernel [17]. The weights are calculated as shown in Eqn. 2.4, where \( \Phi \) contains the evaluations of the radial kernels at the sampled designs. These same weights are then used to approximate out of sample responses, \( y \), as shown in Eqn. 2.5, where \( \lambda \) contains the evaluations of the radial kernels at the new designs, \( x \).

\[
W^T = \Phi^{-1}Y \text{ where } \Phi_{i,j} = \phi(||c_i - c_j||) \text{ for } \forall i \in \{1, 2, \ldots, n\}, \forall j \in \{1, 2, \ldots, n\} \quad (2.4)
\]

\[
y = \lambda W^T \text{ where } \lambda_i = \phi(||x - c_i||) \text{ for } \forall i \in \{1, 2, \ldots, n\} \quad (2.5)
\]

### 2.2.1 Applications of Surrogate Models of FEA

When used with FEA, surrogate models predict the results of the analysis. This could include single values, such as the maximum stress or maximum displacement across the component, or an entire field of nodal results, such as nodal stresses or displacements. When performing the latter, a separate surrogate may be used for each node in order to emulate its result. Kodiyalam et al. [19] used this approach to represent the displacement of every node in a finite element model of a crashworthiness vehicle at varying time steps. The rapid visualization of the results helped designers to more quickly discover a superior design. Heap et al. [20] created surrogate models of FEA results at each node of parametric, two-dimensional meshes for various cantilevered beams under different loading conditions, resulting in real-time design space exploration. Bunnell et al. [18] expanded this to predict the nodal stresses across a jet engine compressor blade. They successfully achieved real-time emulations even with a large number of nodes (25,000 to 500,000) and demonstrated that surrogate accuracy was greatly dependent on the complexity of the model.
involved (e.g. the number of design variables) and the number of training samples used to train the surrogate. They also found that an exponentially decaying relationship existed between error and the number of training samples. As the number of training samples increased, the rate of error reduction decreased, meaning that spending additional time and resources to produce more training samples eventually became unprofitable. Thus, further reducing surrogate error requires a different approach.

2.3 Dimensionality Reduction

Much of the information presented in this section is also repeated in Chapter 3 and Chapter 4 as needed information for journal papers. Dimensionality reduction is a technique that seeks to reduce the number of variables, or dimensions, needed to explain the variance in a set of data. In some cases, the data points in space $\mathbb{R}^p$ lie on some lower dimensional manifold in space $\mathbb{R}^m$, where $m < p$ (Fig. 1.2). If the true dimensionality of the data is known or can be discovered, then the data in the high-dimensional space can be represented in the lower dimensional space with minimal information loss. The mapping from high- to low-dimensional space can be linear or nonlinear, resulting in linear or nonlinear dimensionality reduction. This is illustrated in Fig. 2.5, where two-dimensional data is reduced to one-dimensional data using both linear and nonlinear dimensionality reduction by projecting the data points onto linear and nonlinear variances. The intrinsic geometry of the data (i.e. the spacing between points) is perfectly preserved for the nonlinear case, whereas it is altered for the linear case. For a given set of data, dimensionality

![Figure 2.5: Two-dimensional data is reduced to one-dimensional data using both linear and nonlinear dimensionality reduction.](image)
reduction may produce lower dimensional data that can more easily be predicted by a surrogate, thereby reducing the surrogate’s error.

2.3.1 Types of Dimensionality Reduction

Many linear and nonlinear dimensionality reduction methods exist [21]. For this research, a selection of dimensionality reduction techniques commonly used in the literature were selected, including principal component analysis (PCA), kernel principal component analysis (KPCA), isometric feature mapping (ISOMAP), and locally linear embedding (LLE). No matter the method, the end result of applying dimensionality reduction to a data set is of similar form: a reduced set of dimensions that describe the data.

Principal Component Analysis

Principal component analysis (PCA) was first introduced by Pearson [22] in 1901 and has been used extensively ever since. PCA seeks to discover the principal directions of variance within a data set, as illustrated in Fig. 2.6. Those directions, or principal components (PCs), become the variables of the data when transformed into the lower dimensional space. PCA is a linear dimensionality reduction technique, as it solely performs linear transformations on the data. Jolliffe et al. [23] outline in detail how this is achieved. PCA can be accomplished either by performing singular value decomposition (SVD) on the centered data matrix or by performing eigenvalue decomposition on the covariance matrix. For a \( n \times p \) matrix \( X \), with \( n \) data points of dimension \( p \), the general steps for the SVD approach are as follows: center the data by subtracting the column means \( \bar{X} \), perform singular value decomposition to solve for \( U \), \( S \), and \( V \) (Eqn. 2.6), sort the eigenvectors \( V \) by their corresponding eigenvalues, and select \( m \) eigenvectors to form the lower dimensional space \( \mathbb{R}^m \), where \( m < p \).

\[
X - \bar{X} = USV \quad (2.6)
\]

Each of the \( n \) samples in \( X \) may be represented as a linear combination of the PCs contained in \( V \). The coefficients used in this linear combination are called the PC scores and may be obtained
by multiplying $US$. The data can be transformed to the lower dimensional space $\mathbb{R}^m$ by projecting each of the $n$ data points onto the $m$ PCs through

$$Y_{PCA} = US$$

(2.7)

Given a set of low-dimensional values, $Y_{PCA}$, the data can be transformed back to the high-dimensional space (inverse transform) through

$$X = Y_{PCA}V + \bar{X}$$

(2.8)

Figure 2.6: The principal components discovered using PCA can be used to describe the variance in the data.

Kernel Principal Component Analysis

Kernel principal component analysis (KPCA) is an extension of PCA, first introduced by Schölkopf et al. [24] as a nonlinear form of PCA. KPCA is closely connected to multidimensional scaling (MDS), which constructs a dissimilarity matrix that contains the dissimilarities between all pairwise data points [25]. If the dissimilarities are computed using Euclidean distances, classical MDS aims to reduce the dimensionality of a data set in such a way that the Euclidean distances between points in the low-dimensional space are the same as the Euclidean distances between points in the high-dimensional space. Williams [25] outlines in detail how this is performed.
MDS involves the following steps: construct matrix $A$ of all pairwise dissimilarities (i.e. squared Euclidean distances) and solve for the eigenvalues and eigenvectors of the matrix $K$ defined as

$$K = -\frac{1}{2}BAB = (BX)(BX)^T \tag{2.9}$$

where $B$ is the centering matrix. If the eigenvalue decomposition of $K$ is

$$K = V\Lambda V^{-1} \tag{2.10}$$

where $V$ and $\Lambda$ contain the eigenvectors and eigenvalues of $K$, respectively, the low-dimensional representation of the data can be obtained through

$$Y_{MDS} = \Lambda^{1/2}V \tag{2.11}$$

For a $n \times p$ matrix $X$, with $n$ data points of dimension $p$, MDS performs eigenvalue decomposition on the centered matrix $XX^T$. PCA performs eigenvalue decomposition on the covariance matrix, which is constructed using $X^TX$. Williams [25] shows that the MDS solution, $Y_{MDS}$, is equivalent to the PCA solution, $Y_{PCA}$ (i.e. PC scores), if matrix $A$ is constructed using Euclidean distances.

KPCA utilizes MDS’s approach to compute the PC scores of $\phi(X)$, where $X$ has been transformed to a higher dimensional space, called the feature space, by some transformation $\phi()$. Schölkopf et al. [24] show that $\phi(X)\phi(X)^T$, or equivalently the dot product between all pairwise points in the feature space, can be computed using a kernel. This process is commonly referred to as the “kernel trick”. The key mechanism of the “kernel trick” lies in the fact that kernels, or functions, exist that are mathematically equivalent to performing the dot product of two vectors in some feature space. That is, the kernel can compute the dot products without having to perform the actual transformation. Evaluating the kernel is computationally less expensive than transforming the data to the feature space to compute dot products. Many kernels exist that compute the dot products in different feature spaces. The most common include polynomial, radial basis function, and sigmoid kernels. While PCA is considered a linear dimensionality reduction technique, KPCA may be nonlinear. As shown in Fig. 2.7, the linear variances (i.e. PCs) of the data discovered by PCA in the feature space will not necessarily, and usually are not, linear in the original space due to the transformation $\phi()$. As such, nonlinear variances in the data can be extracted through KPCA.
Figure 2.7: Linear principal components in the high-dimensional feature space (left) will not necessarily be linear in the original space (right) due to the transformation, thus allowing KPCA to discover nonlinear variances in the data.

**Isometric Feature Mapping**

Isometric feature mapping (ISOMAP) was introduced by Tenenbaum et al. [26] as an extension of multidimensional scaling (MDS). When finding an embedding, or low-dimensional representation of the data, classic MDS preserves pairwise distances between points. This process is equivalent to PCA if Euclidean distances are used. ISOMAP differs in that it preserves the pairwise geodesic distances, or the shortest path between two points along a curved surface, in order to preserve the intrinsic geometry of the data. Mathematically, it is not possible to compute a geodesic distance along a surface if the shape of the surface is not known. ISOMAP overcomes this by creating a k-nearest neighbors graph and approximating the geodesic distance between any two points as the shortest path through the k-nearest neighbors graph (i.e. a summation of Euclidean distances) as shown in Fig. 2.8. ISOMAP involves the following steps: construct a k-nearest neighbors graph, compute the geodesic distances between all points to form matrix $D$, and solve for the eigenvalues and eigenvectors of the matrix $K$ defined as

$$K = -\frac{1}{2}BAB$$  \hspace{1cm} (2.12)

where $A$ is the matrix of squared geodesic distances ($A_{ij} = D_{ij}^2$), and $B$ is the centering matrix. The low-dimensional representation of the data, $Y_{ISO}$, can be obtained by using Eqns. 2.10 and 2.11.
Figure 2.8: The Euclidean distance, $AE$, is different from the geodesic distance. The later is computed as the shortest path through the k-nearest neighbor graph and is shown for both $k = 1$ (left) and $k = 2$ (right).

**Locally Linear Embedding**

Locally linear embedding (LLE) models the global structure of the data manifold as a union of locally linear fits. Roweis et al. [27], the first to introduce LLE, outline in detail how this is done. Each data point is reconstructed using a weighted sum of its $k$ nearest neighbors, where $k$ is a parameter of the algorithm. The weights, $W$, are chosen such that the squared distances between the original $n$ data points, $X$, and their reconstructions are minimized according to

$$W = \min \left( \sum_{i=1}^{n} \left| X_i - \sum_{j=1}^{k} W_{ij} X_j \right|^2 \right) \quad (2.13)$$

Once determined, these same weights are used to perform the low-dimensional mapping such that the squared distances between the lower dimensional $n$ data points, $Y$, and their reconstructions are minimized according to

$$Y_{LLE} = \min \left( \sum_{i=1}^{n} \left| Y_i - \sum_{j=1}^{k} W_{ij} Y_j \right|^2 \right) \quad (2.14)$$

This process is illustrated in Fig. 2.9. The local geometry of the k-nearest neighbors is preserved for each point in the data set while still reducing the dimensionality of $X$.

**2.3.2 Applications of Dimensionality Reduction**

Researchers have used dimensionality reduction for various applications. Orsenigo et al. [5] applied seven different dimensionality reduction methods to nine high-dimensional cancer microarray data sets and compared their ability to properly classify the data using its low-dimensional rep-
representation. No single dimensionality reduction method was superior in all cases. Four nonlinear techniques, namely ISOMAP, LLE, Laplacian eigenmaps (LE), and maximum variance unfolding (MVU), each had the highest classification accuracy in at least one of the nine data sets. In a later work, Orsenigo et al. [6] compared a linear method (PCA) to a nonlinear method (ISOMAP) in their ability to improve credit rating predictions of banks. Several classification algorithms were applied to the high-dimensional data, as well as to each of the low-dimensional representations from PCA and ISOMAP. In the majority of cases, ISOMAP’s low-dimensional representation resulted in the highest classification accuracy. Lopez et al. [7] used LLE to enable real-time analysis of heterogeneous microstructures. The low-dimensional embedding produced by LLE led to a more efficient process that required fewer computational resources.

Researchers have also coupled dimensionality reduction with surrogate modeling. Bun nell [4] used PCA’s low-dimensional representation of nodal stresses to train surrogate models, increasing the surrogate prediction speeds without an increase in error. The amount of improvement was dependent on the model complexity, number of training samples, and number of retained PCs, but a model with 100,000 nodes, 100 samples, and 10 retained PCs resulted in a PCA-based surrogate that was 2.05 times faster than a surrogate that directly emulated the nodal stresses. The lin [28] used this same approach to emulate the nodal displacements of mode shapes, which in turn were used to compute the modal assurance criterion (MAC). These surrogates were based on applying PCA to the high-dimensional nodal data (not explicitly stated) and resulted in high root mean square error between 15-25% when comparing several mode shapes. Nonlinear dimensionality reduction, which showed promise in the applications listed above, was not explored by
Bunnell or Thelin. Franz et al. [12] compared PCA- and ISOMAP-based surrogates for predicting aerodynamic shock locations and intensities. The nonlinear dimensionality-reduction-based surrogate models (ISOMAP) resulted in surface pressure distributions that more accurately represented the shocks. Citing this research, Ripepi et al. [29] suggested that similar improvements from nonlinear dimensionality-reduction-based surrogates may be achieved for other applications. In many use cases, nonlinear dimensionality reduction produces low-dimension embeddings that lead to greater efficiency and increased accuracy. No single dimensionality reduction method is superior in all cases. As such, comparing multiple algorithms for each application is necessary in order to achieve the greatest accuracy. In this research, applying and comparing multiple dimensionality reduction methods in conjunction with surrogate models to predict FEA results, including the nodal stresses explored by Bunnell [4] and the nodal displacements investigated by Thelin [28], has led to increased surrogate accuracy as presented and discussed in Chapter 3 and Chapter 4.

2.4 Turbomachinery Blade Model

All of the data presented in this research was produced using the Transonic Purdue Research Compressor, which is a high pressure compressor blade originally developed at Purdue University to perform structural and aerodynamic studies [30]. The blade is defined by a series of cross sections as shown in Fig. 2.10(a). It has a height of about 2.1 inches and a mean chord of about 1.9 inches. A computer-aided design (CAD) drawing of the blade mounted on a disk is shown in Fig. 2.10(b) for reference. The Purdue blade is commonly used in research, including studies involving surrogate models of FEA [18,28,31]. The blade alone model, or the single blade not oriented on a disk, was used for the DSE research presented herein. In addition to the 14 cross sections defining the geometry, the blade was parameterized using common airfoil parameters in order to more easily automate the DSE process. These parameters, including their names, symbols, and ranges, are listed in Table 2.2.

While not reported as part of this research, a second blade, the Air Force Research Lab (AFRL) Rotor 4, was considered. This blade was originally developed as part of a study involving the aerodynamic properties of fan blades and represents a more modern turbomachinery blade design [32]. The methods presented in this research were applied to select geometries of Rotor 4 and found to produce similar results. However, the surrogate errors, overall, were lower than
Figure 2.10: The Transonic Purdue Research Compressor. The cross sections used to define the blade’s geometry (a) and the blade oriented on a disk (b) are both shown.

Table 2.2: Set of design parameters of the Transonic Purdue Research Compressor used for this research. The ranges for these variables were based on the relationship to a nominal value of the geometry.

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Symbol</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>$H$</td>
<td>± 5% of nominal $H$</td>
</tr>
<tr>
<td>Ave. Radius</td>
<td>$R_o$</td>
<td>± 5% of nominal $R_o$</td>
</tr>
<tr>
<td>Root Chord</td>
<td>$C_R$</td>
<td>± 20% of nominal $C_R$</td>
</tr>
<tr>
<td>Tip Chord</td>
<td>$C_T$</td>
<td>± 20% of nominal $C_T$</td>
</tr>
<tr>
<td>Sweep</td>
<td>$S$</td>
<td>± 15% of nominal $C_T$</td>
</tr>
<tr>
<td>Lean</td>
<td>$L$</td>
<td>± 15% of nominal $C_T$</td>
</tr>
<tr>
<td>Angle</td>
<td>$\alpha$</td>
<td>± 20 deg of nominal $\alpha$</td>
</tr>
<tr>
<td>Rotation Speed</td>
<td>$N$</td>
<td>± 10% of nominal $N$</td>
</tr>
<tr>
<td>Ave. Pressure</td>
<td>$P$</td>
<td>± 10% of nominal $P$</td>
</tr>
</tbody>
</table>

those found for the Purdue blade. This could be due to the more realistic geometry of Rotor 4 being more structurally sound or the design variable ranges being too small for the larger Rotor 4 blade. Lower errors made it more difficult to see improvement with the nonlinear methods explored herein. As such, the Purdue blade was used as the primary candidate for applying and comparing the dimensionality-reduction-based surrogate models to various FEA results.
Surrogate models are most useful when they are capable of making real-time predictions that are accurate. Surrogate accuracy has the potential to be improved through the use of nonlinear dimensionality reduction. The purpose of this chapter is to perform a comparative analysis of linear and nonlinear dimensionality-reduction-based surrogate models for emulating finite element analysis (FEA) nodal results in order to determine which methods are most effective at reducing surrogate error. This chapter is based on an unpublished manuscript that will be submitted for publication [13]. The research presented in this chapter contributes to the field of real-time structural DSE in the following ways:

1. Provide a comparative analysis of linear and nonlinear dimensionality-reduction-based surrogate models in their ability to emulate FEA nodal results from a static analysis, including stresses and coordinates.

2. Provide a comparative analysis of dimensionality-reduction-based surrogate models applied to design spaces of a turbomachinery compressor blade.

3. Introduce a new metric, the manifold distance ratio (MDR), to measure of the nonlinearity of the data manifold in order to increase understanding of when nonlinear dimensionality reduction should be applied to a data set.

4. Identify for this application that the surrogate error and nonlinearity of the nodal stress and coordinate spaces are different.

3.1 Introduction

Design space exploration (DSE) is a systematic analysis of designs based on parameters that are permitted to vary. It is a tool frequently used by engineers for optimal design selection.
DSE, however, often involves computationally expensive analyses, such as finite element analysis (FEA), that can require large amounts of time and computer resources [18,19,33]. This can hinder an engineer’s ability to perform DSE in real time, leading to fewer tested designs. DSE also suffers from the curse of dimensionality [34]. As the number of design variables increases, the number of designs to explore increases exponentially. This, coupled with long analysis times, can make complete exploration of the design space infeasible. This is frequently the case in the aerospace industry as designs often involve ten or more parameters. Incomplete exploration of the design space increases the likelihood of sub-optimal design selection.

To overcome this obstacle, tools and methods have been developed that allow the designer to more quickly explore the design space. One such tool is surrogate modeling [3]. Surrogate models can predict the outcome of an analysis without having to perform it. The accuracy of these predictions generally increases as more data is used to train the surrogate. When used with FEA, surrogate models can create relationships between the design parameters (e.g. length, width, height) and FEA nodal properties (e.g. stresses, coordinates). A designer can input parameters into the surrogate and quickly obtain results. Studies have used surrogate models to predict the nodal responses of finite element models for DSE. These include the emulation of displacement and stress fields over an entire three-dimensional FEA model [18–20, 35]. Surrogate models can also be coupled with dimensionality reduction techniques [36, 37]. This reduces the number of predictions that must be made by the surrogate and in turn may lead to faster, more accurate surrogate models. When applied to a nonlinear data set, nonlinear dimensionality-reduction-based surrogate models may be more accurate than linear dimensionality-reduction-based surrogates. Bunnell et al. [18] did not use dimensionality reduction but demonstrated that, with enough training samples, full-nodal surrogates could have less than 5% error. However, this was only achieved with relatively simple design spaces that had three or fewer parameters. This error increased with more complicated models.

This paper explores surrogate models based on dimensionality reduction to predict the FEA nodal properties of a jet engine compressor blade. Four different dimensionality reduction methods are compared, namely principal component analysis (PCA), kernel principal component analysis (KPCA), isometric feature mapping (ISOMAP), and locally linear embedding (LLE). The Transonic Purdue Research Compressor [30] is used. Each method is applied to the FEA stresses and
coordinates of a training set of designs, the results of which are used as inputs to the surrogate model. Rather than predict the nodal stresses and coordinates directly, the surrogate instead predicts the value of each dimension in the dimensionality reduced space. The results will show that using nonlinear dimensionality reduction techniques (KPCA, ISOMAP, LLE) can lead to reduced error in the surrogate models compared to linear dimensionality reduction (PCA).

This paper continues with a background section on surrogate modeling and the dimensionality reduction techniques used in this research. The process of implementing these techniques and constructing the surrogate models is then outlined in the methods section. To the authors’ knowledge, a direct comparison between linear and nonlinear dimensionality-reduction-based surrogate models and their ability to predict full-nodal stresses and coordinates has not been published in the literature. This comparison is presented and discussed in the results section for the FEA stresses and coordinates of a jet engine compressor blade. Finally, the conclusion summarizes the key findings and discusses the benefits of using nonlinear dimensionality-reduction-based surrogate models for emulating finite element results.

3.2 Background

A background on the surrogate modeling and dimensionality reduction techniques explored in this research is presented herein. The interested reader is referred to the referenced works for further explanations and details.

3.2.1 Surrogate Modeling

Surrogate modeling relates a set of independent variables to a set of dependent variables through the use of a mathematical approximation function. A set of known dependent and independent values, called a training set, is used to create a function that predicts the outputs based on the inputs. “Surrogate” is an all-encompassing term for such approximation techniques, including radial basis functions, artificial neural networks, response surface models, etc [33]. The advantage of surrogate modeling is its ability to approximate the outcome of computationally expensive analyses almost instantaneously and, when trained properly, with minimal error. This makes surrogate models especially useful when performing DSE.
Full-nodal FEA surrogate models emulate results for every node in the finite element mesh, such as those illustrated in Fig. 3.1. A separate surrogate model may be used to emulate the results for each node, such as stress or geometric coordinates, in each degree of freedom. Kodiyalam et al. [19] used this approach to represent the displacement of every node in a finite element model of a crashworthiness vehicle at varying time steps. The rapid visualization of the results helped designers to more quickly discover a superior design. Heap et al. [20] created surrogate models of FEA results at each node of parametric, two-dimensional meshes for various cantilevered beams under different loading conditions, resulting in real-time design space exploration. Bunnell et al. [18] expanded this method to three-dimensional meshes of jet engine compressor blades and demonstrated the capability of full-nodal surrogates to visualize FEA results in real time even with a large number of nodes (25,000 to 500,000).

![Finite element mesh](image1.jpg) ![Stress contour](image2.jpg)

Figure 3.1: An example of the finite element mesh (a) and stress contour (b) on different geometries of a compressor blade.

With an infinite number of possibilities, no designer can explore all combinations of design parameters in a continuous design space. However, in each of these studies, surrogate modeling was used in order to facilitate rapid and more thorough design space exploration of finite element results. Creating a global model that can predict the desired outputs can help a designer discover a more optimal design. While useful, surrogate models must be properly trained in order to minimize
their error. Bunnell et al. [18] found that the error decayed exponentially as the number of training samples increased, meaning that it became increasingly harder to further reduce the error. Spending additional time and computational resources to create more training samples quickly becomes unfruitful. Further reduction in error to produce more accurate surrogate models requires other strategies.

### 3.2.2 Dimensionality Reduction

Dimensionality reduction is a technique that seeks to reduce the number of variables, or dimensions, needed to explain the variance in a set of data. In some cases, the data points in space $\mathbb{R}^p$ lie on some lower dimensional manifold in space $\mathbb{R}^m$, where $m < p$. If the true dimensionality of the data is known or can be discovered, then the data in the high-dimensional space can be represented in the lower dimensional space with minimal information loss. The mapping from high- to low-dimensional space can be linear or nonlinear, resulting in linear or nonlinear dimensionality reduction. This is illustrated in Fig. 3.2, where two-dimensional data is reduced to one-dimensional data using both linear and nonlinear dimensionality reduction by projecting the data points onto linear and nonlinear variances. The intrinsic geometry of the data (i.e. the spacing between points) is perfectly preserved for the nonlinear case, whereas it is altered for the linear case. For a given set of data, dimensionality reduction may produce lower dimensional data that can more easily be predicted by a surrogate, thereby reducing the surrogate’s error. Many linear and nonlinear methods exist, including principal component analysis (PCA), kernel principal component analysis (KPCA), isometric feature mapping (ISOMAP), and locally linear embedding.

![Dimensionality Reduction Diagram](image)

**Figure 3.2:** Dimensionality reduction: two-dimensional data is reduced to one-dimensional data using both linear and nonlinear dimensionality reduction.
Researchers have used these and other dimensionality reduction methods in conjunction with surrogate modeling to predict various structural and aerodynamic results [4, 12, 38, 39]. The remainder of this section is repeated (verbatim) from Sec. 2.3.

**Principal Component Analysis**

Principal component analysis (PCA) was first introduced by Pearson [22] in 1901 and has been used extensively ever since. PCA seeks to discover the principal directions of variance within a data set, as illustrated in Fig. 3.3. Those directions, or principal components (PCs), become the variables of the data when transformed into the lower dimensional space. PCA is a linear dimensionality reduction technique, as it solely performs linear transformations on the data. Jolliffe et al. [23] outline in detail how this is achieved. PCA can be accomplished either by performing singular value decomposition (SVD) on the centered data matrix or by performing eigenvalue decomposition on the covariance matrix. For a \( n \times p \) matrix \( X \), with \( n \) data points of dimension \( p \), the general steps for the SVD approach are as follows: center the data by subtracting the column means \( \bar{X} \), perform singular value decomposition to solve for \( U, S, \) and \( V \) (Eqn. 3.1), sort the eigenvectors \( (V) \) by their corresponding eigenvalues, and select \( m \) eigenvectors to form the lower dimensional space \( \mathbb{R}^m \), where \( m < p \).

\[
X - \bar{X} = USV \tag{3.1}
\]

Each of the \( n \) samples in \( X \) may be represented as a linear combination of the PCs contained in \( V \). The coefficients used in this linear combination are called the PC scores and may be obtained by multiplying \( US \). The data can be transformed to the lower dimensional space \( \mathbb{R}^m \) by projecting each of the \( n \) data points onto the \( m \) PCs through

\[
Y_{PCA} = US \tag{3.2}
\]

Given a set of low-dimensional values, \( Y_{PCA} \), the data can be transformed back to the high-dimensional space (inverse transform) through

\[
X = Y_{PCA}V + \bar{X} \tag{3.3}
\]
Figure 3.3: The principal components discovered using PCA can be used to describe the variance in the data.

**Kernel Principal Component Analysis**

Kernel principal component analysis (KPCA) is an extension of PCA, first introduced by Schölkopf et al. [24] as a nonlinear form of PCA. KPCA is closely connected to multidimensional scaling (MDS), which constructs a dissimilarity matrix that contains the dissimilarities between all pairwise data points [25]. If the dissimilarities are computed using Euclidean distances, classical MDS aims to reduce the dimensionality of a data set in such a way that the Euclidean distances between points in the low-dimensional space are the same as the Euclidean distances between points in the high-dimensional space. Williams [25] outlines in detail how this is performed. MDS involves the following steps: construct matrix $A$ of all pairwise dissimilarities (i.e. squared Euclidean distances) and solve for the eigenvalues and eigenvectors of the matrix $K$ defined as

$$K = -\frac{1}{2}BAB = (BX)(BX)^T \quad (3.4)$$

where $B$ is the centering matrix. If the eigenvalue decomposition of $K$ is

$$K = V\Lambda V^{-1} \quad (3.5)$$

where $V$ and $\Lambda$ contain the eigenvectors and eigenvalues of $K$, respectively, the low-dimensional representation of the data can be obtained through
\[ Y_{MDS} = \Lambda^{1/2} V \]  

(3.6)

For a \( n \times p \) matrix \( X \), with \( n \) data points of dimension \( p \), MDS performs eigenvalue decomposition on the centered matrix \( XX^T \). PCA performs eigenvalue decomposition on the covariance matrix, which is constructed using \( X^T X \). Williams [25] shows that the MDS solution, \( Y_{MDS} \), is equivalent to the PCA solution, \( Y_{PCA} \) (i.e. PC scores), if matrix \( A \) is constructed using Euclidean distances.

KPCA utilizes MDS’s approach to compute the PC scores of \( \phi(X) \), where \( X \) has been transformed to a higher dimensional space, called the feature space, by some transformation \( \phi() \). Schölkopf et al. [24] show that \( \phi(X)\phi(X)^T \), or equivalently the dot product between all pairwise points in the feature space, can be computed using a kernel. This process is commonly referred to as the “kernel trick”. The key mechanism of the “kernel trick” lies in the fact that kernels, or functions, exist that are mathematically equivalent to performing the dot product of two vectors in some feature space. That is, the kernel can compute the dot products without having to perform the actual transformation. Evaluating the kernel is computationally less expensive than transforming the data to the feature space to compute dot products. Many kernels exist that compute the dot products in different feature spaces. The most common include polynomial, radial basis function, and sigmoid kernels. While PCA is considered a linear dimensionality reduction technique, KPCA may be nonlinear. As shown in Fig. 3.4, the linear variances (i.e. PCs) of the data discovered by

\[ \text{Figure 3.4: Linear principal components in the high-dimensional feature space (left) will not necessarily be linear in the original space (right) due to the transformation, thus allowing KPCA to discover nonlinear variances in the data.} \]
PCA in the feature space will not necessarily, and usually are not, linear in the original space due to the transformation $\phi()$. As such, nonlinear variances in the data can be extracted through KPCA.

**Isometric Feature Mapping**

Isometric feature mapping (ISOMAP) was introduced by Tenenbaum et al. [26] as an extension of multidimensional scaling (MDS). When finding an embedding, or low-dimensional representation of the data, classic MDS preserves pairwise distances between points. This process is equivalent to PCA if Euclidean distances are used. ISOMAP differs in that it preserves the pairwise geodesic distances, or the shortest path between two points along a curved surface, in order to preserve the intrinsic geometry of the data. Mathematically, it is not possible to compute a geodesic distance along a surface if the shape of the surface is not known. ISOMAP overcomes this by creating a k-nearest neighbors graph and approximating the geodesic distance between any two points as the shortest path through the k-nearest neighbors graph (i.e. a summation of Euclidean distances) as shown in Fig. 3.5. ISOMAP involves the following steps: construct a k-nearest neighbors graph, compute the geodesic distances between all points to form matrix $D$, and solve for the eigenvalues and eigenvectors of the matrix $K$ defined as

$$K = -\frac{1}{2}BAB$$

where $A$ is the matrix of squared geodesic distances ($A_{ij} = D_{ij}^2$), and $B$ is the centering matrix. The low-dimensional representation of the data, $Y_{ISO}$, can be obtained by using Eqns. 3.5 and 3.6.

![Figure 3.5: The Euclidean distance, AE, is different from the geodesic distance. The later is computed as the shortest path through the k-nearest neighbor graph and is shown for both $k = 1$ (left) and $k = 2$ (right).](image-url)
**Locally Linear Embedding**

Locally linear embedding (LLE) models the global structure of the data manifold as a union of locally linear fits. Roweis et al. [27], the first to introduce LLE, outline in detail how this is done. Each data point is reconstructed using a weighted sum of its $k$ nearest neighbors, where $k$ is a parameter of the algorithm. The weights, $W$, are chosen such that the squared distances between the original $n$ data points, $X$, and their reconstructions are minimized according to

$$W = \min \left( \sum_{i=1}^{n} \left| X_i - \sum_{j=1}^{k} W_{ij}X_j \right|^2 \right) \tag{3.8}$$

Once determined, these same weights are used to perform the low-dimensional mapping such that the squared distances between the lower dimensional $n$ data points, $Y$, and their reconstructions are minimized according to

$$Y_{LLE} = \min \left( \sum_{i=1}^{n} \left| Y_i - \sum_{j=1}^{k} W_{ij}Y_j \right|^2 \right) \tag{3.9}$$

This process is illustrated in Fig. 3.6. The local geometry of the $k$-nearest neighbors is preserved for each point in the data set while still reducing the dimensionality of $X$.

Figure 3.6: Locally linear embedding is performed by taking data in the high-dimensional space(a) and using the $k$ nearest neighbors of each point(b) and their weights to reconstruct the data(c) and find a low-dimensional embedding(d).
3.3 Methods

This paper applies the dimensionality reduction techniques described in Sec. 3.2 to FEA nodal properties of the Transonic Purdue Research Compressor [30] shown in Fig. 3.7. The process utilized an automated workflow that is illustrated in Fig. 3.8 and described here. This includes the creation of the training data (blue boxes) and testing data (orange boxes), as well as the training and testing of the surrogate model (rightmost light gray boxes). The model was parameterized using common jet engine compressor blade parameters, allowing the engineer to easily update the blade’s geometry. The parameters, shown in Table 3.1, were allowed to vary from their nominal values within the specified ranges. To understand how each of the methods performed across the entire blade’s design space, 200 different design spaces were created. A random integer, \( n \), between one and nine was chosen, after which \( n \) design variables from the full set of nine in Table 3.1 were randomly selected. A training design of experiments (DOE) with \( m \) samples was created that varied each of the selected design variables within the ranges specified in Table 3.1. All unselected design variables were held constant at their nominal values. The variable \( m \) was randomly set between 10 and 20 times larger than \( n \). Three different finite element models were created with approximately 10,000, 25,000, and 50,000 nodes. After randomly selecting one of these meshes,

![Figure 3.7: Transonic Purdue Research Compressor blade, both the cross-sections used in the parametric model (a) and the blades oriented on a disk (b).](a) Purdue blade (b) Purdue blade disk)
Table 3.1: Set of design parameters for Transonic Purdue Research Compressor used for this research. The ranges for these variables were based on the relationship to a nominal value of the geometry.

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Symbol</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>(H)</td>
<td>± 5% of nominal (H)</td>
</tr>
<tr>
<td>Ave. Radius</td>
<td>(R_o)</td>
<td>± 5% of nominal (R_o)</td>
</tr>
<tr>
<td>Root Chord</td>
<td>(C_R)</td>
<td>± 20% of nominal (C_R)</td>
</tr>
<tr>
<td>Tip Chord</td>
<td>(C_T)</td>
<td>± 20% of nominal (C_T)</td>
</tr>
<tr>
<td>Sweep</td>
<td>(S)</td>
<td>± 15% of nominal (C_T)</td>
</tr>
<tr>
<td>Lean</td>
<td>(L)</td>
<td>± 15% of nominal (C_T)</td>
</tr>
<tr>
<td>Angle</td>
<td>(\alpha)</td>
<td>± 20 deg of nominal (\alpha)</td>
</tr>
<tr>
<td>Rotation Speed</td>
<td>(N)</td>
<td>± 10% of nominal (N)</td>
</tr>
<tr>
<td>Ave. Pressure</td>
<td>(P)</td>
<td>± 10% of nominal (P)</td>
</tr>
</tbody>
</table>

The parameterized blade was updated for each of the designs in the training DOE (top blue box) and solved in FEA to obtain a training set of high-dimensional static coordinates and von Mises stresses (middle blue box). Using the same blade, a corresponding testing set was created with a separate testing DOE of 50 samples (leftmost orange boxes). This process was repeated 200 times.

The training data was then used to create a surrogate for predicting properties of every node in the finite element mesh, as illustrated in Figure 3.8 with the rightmost light gray boxes. This was done once using the stress data and once using the coordinate data. Stresses and coordinates were treated separately, as they exhibit different nodal properties. While many surrogate models exist [33], Jin et al. [40] showed that radial basis functions (RBFs) performed well across a variety of design spaces. Bunnell et al. [18] found that RBFs made fast predictions with low error when used to emulate FEA results. An implementation of RBF, SciPy’s radial basis function, was used for this application [41]. Rather than use the high-dimensional coordinates and stresses directly to train the surrogate, the various dimensionality reduction methods were performed on the high-dimensional nodal data to obtain its low-dimensional representation (bottom blue box). This significantly reduced the dimensionality from 10,000-50,000 dimensions for stress (stress values at each of the 10,000-50,000 nodes) and 30,000-150,000 dimensions for coordinates (x-, y-, z-coordinate values at each of the 10,000-50,000 nodes) to a dimensionality set on the order of ten. PCA, ISOMAP, LLE, and KPCA each employ different techniques to perform the dimensionality reduction, but the
results are of similar form: a reduced set of dimensions and their corresponding values (scores) that describe the variance in the original high-dimensional data with minimal information loss. If the variances in the data follow a nonlinear profile, the nonlinear dimensionality reduction techniques may more accurately preserve the intrinsic geometry of the data in the simpler, low-dimensional space. Such a preservation may produce a more accurate surrogate, as design-to-design variations remain similar in both the high- and low-dimensional spaces. The training DOE design parameter values and the corresponding low-dimensional training data were used to train a RBF surrogate (Surrogate 1). Once trained, Surrogate 1 predicted the low-dimensional scores when given the geometric parameter values as an input. To return to the high-dimensional space, a secondary surrogate (Surrogate 2) was created using the low- and high-dimensional training data. When given low-dimensional scores, Surrogate 2 predicted the high-dimensional nodal coordinates or stresses (rightmost orange box). This dual surrogate approach inherently introduced more sources of error than a single surrogate alone; however, dimensionality reduction may lead to more accurate

Figure 3.8: Automated workflow used to create the training data, testing data, and surrogate models.
surrogates that have a lower combined error. This process was repeated for PCA using both the
dual-surrogate as well as replacing Surrogate 2 with PCA’s inverse transform (Eqn. 3.3).

A separate dual-surrogate was created using the low-dimensional representation of the
nodal data (bottom blue box) from each of the dimensionality reduction methods. To perform the
dimensionality reduction, scikit-learn’s PCA, Isomap, LocallyLinearEmbedding, and KernelPCA
classes were used [42]. Each class has specific hyperparameters that, when varied, change the per-
formance of the algorithm. Table 3.2 lists the hyperparameters that were changed from their default
values for each class and the ranges within which they were changed. These hyperparameters are
described in detail in scikit-learn’s documentation of these algorithms [42]. As the true dimen-
sionality and shape of the nodal data was unknown, many different combinations of hyperparameters
were used. A DOE of up to 1000 different hyperparameter configurations was tested for each of
the dimensionality reduction methods, once on the stresses and once on the coordinates. Because
the PCA and ISOMAP algorithms had fewer hyperparameters, it was not always possible to have

<table>
<thead>
<tr>
<th>Class</th>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>n_components</td>
<td>[1, num_samples]</td>
</tr>
<tr>
<td>ISOMAP</td>
<td>n_neighbors</td>
<td>[1, num_samples-1]</td>
</tr>
<tr>
<td></td>
<td>n_components</td>
<td>[1, num_samples]</td>
</tr>
<tr>
<td></td>
<td>eigen_solver</td>
<td>dense</td>
</tr>
<tr>
<td>LLE</td>
<td>n_neighbors</td>
<td>[1, num_samples-1]</td>
</tr>
<tr>
<td></td>
<td>n_components</td>
<td>[1, num_samples]</td>
</tr>
<tr>
<td></td>
<td>reg</td>
<td>[0.0005, 0.25]</td>
</tr>
<tr>
<td></td>
<td>eigen_solver</td>
<td>dense</td>
</tr>
<tr>
<td>KPCA-RBF</td>
<td>kernel</td>
<td>rbf</td>
</tr>
<tr>
<td></td>
<td>n_components</td>
<td>[1, num_samples]</td>
</tr>
<tr>
<td></td>
<td>gamma</td>
<td>[0, 0.001]</td>
</tr>
<tr>
<td></td>
<td>eigen_solver</td>
<td>dense</td>
</tr>
<tr>
<td>KPCA-Poly</td>
<td>kernel</td>
<td>poly</td>
</tr>
<tr>
<td></td>
<td>n_components</td>
<td>[1, num_samples]</td>
</tr>
<tr>
<td></td>
<td>gamma</td>
<td>[0, 0.001]</td>
</tr>
<tr>
<td></td>
<td>degree</td>
<td>[1, 5]</td>
</tr>
<tr>
<td></td>
<td>coef0</td>
<td>[-1000, 1000]</td>
</tr>
<tr>
<td></td>
<td>eigen_solver</td>
<td>dense</td>
</tr>
</tbody>
</table>
1000 different configurations. In such cases all possible configurations were tested. This amount was equal to the number of samples for PCA and the number of samples multiplied by the the number of samples minus one for ISOMAP (Table 3.2). For each configuration, the high-dimensional nodal data (stresses or coordinates) from the training set were embedded and used to create the dual-surrogate. The surrogate was then tested using the true and predicted high-dimensional nodal data to determine its error. The hyperparameter configuration that resulted in the lowest surrogate error was chosen.

3.4 Results

Once created, the dual-surrogate was tested for accuracy using double normalized root mean square error (double-NRMSE) to obtain a single value for the error across each node of every design in the testing set. This metric was evaluated separately for the stress and coordinates at each node. Eqn. 3.10 shows how the double-NRMSE is computed for stress, where \( \sigma_{i,j} \) is the stress at node \( i \) of test sample \( j \), \( \hat{\sigma}_{i,j} \) is the surrogate’s prediction for that same stress, and \( \sigma_{\text{max}} \) is the maximum stress across all nodes, \( n \), and across all test samples, \( t \). A similar equation was used to compute the double-NRMSE for coordinates, where stress terms were replaced with coordinate terms.

\[
\text{double-NRMSE} = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{t} (\hat{\sigma}_{i,j} - \sigma_{i,j})^2}{nt\sigma_{\text{max}}^2}} \quad (3.10)
\]

The lowest double-NRMSE across all tested hyperparameter configurations for each method was saved for each of the 200 tested design spaces. Box plots of these double-NRMSEs, both for predicting stress and x-, y-, z-coordinates, are shown in Fig. 3.9, where double-NRMSE is reported as a percentage. For PCA, both the inverse transform (PCA-IT) and dual-RBF (PCA-RBF) surrogates are shown. For KPCA, both the polynomial kernel (KPCA-Poly) and the RBF kernel (KPCA-RBF) surrogates are shown. The mean double-NRMSE for each method is also displayed with the “\( \times \)” symbol. Stresses and coordinates are reported separately because dimensionality reduction was performed separately on both. ISOMAP had the lowest double-NRMSE in predicting the nodal stresses on the blades, as it had a lower mean and median and a more narrow distribution than all of the other tested algorithms. Apart from ISOMAP, all of the other methods had similar
shaped distributions with roughly the same variance and range. Using the inverse-transform-based PCA as a baseline, ISOMAP reduced both mean and median error for predicting stress by 35.7% and 34.4%, respectively. This baseline was chosen due to PCA’s unique properties of being linear and having the capability to directly transform back to the high-dimensional space. The LLE and polynomial kernel KPCA methods also had less mean and median error than the baseline PCA but showed less improvement than ISOMAP, while the RBF kernel KPCA had more mean and median error than the baseline PCA. For PCA, it should be noted that a reduction in error was achieved by using a secondary surrogate rather than the inverse transform (baseline) to return to the high-dimensional space. The dual-RBF surrogate reduced both mean and median error for predicting stress by 8.6% and 10.7%, respectively.

Figure 3.9: Box plots of the double-NRMSE for stress (a) and coordinates (b). Errors (NRMSE) are reported as a percentage. Mean errors are indicated with the “×” symbol.

Several factors may play a role in the performance differences among the nonlinear methods. The LLE algorithm is sensitive to noise when determining the weights used in the embedding process [43]. It is also sensitive to the values of the neighbors and regularization parameters. Due to these and other shortcomings, LLE is known to be unstable and may produce distorted embeddings if the manifold has more than one dimension [44]. With only 1000 configurations
in the parameter tuning DOE, appropriate values for the number of neighbors, manifold dimensionality, and regularization parameters may not have been found in order to produce a stable and accurate embedding. Modifications to the LLE algorithm exist to help address these issues, including modified locally linear embedding (MLLE) and Hessian eigenmapping or Hessian-based LLE (HLLE) [42]; however, these methods put further constraints on the algorithm and will not guarantee improved performance for this application. The results of KPCA, on the other hand, are highly dependent on the kernel function used to compute the inner product in the higher dimensional feature space. Ham et al. [45] show that ISOMAP and LLE are actually implementations of KPCA that use a specially constructed Gram matrix. From this perspective, four different implementations of KPCA were performed using the ISOMAP, LLE, RBF, and polynomial kernels, with the best results coming from ISOMAP. Kernel selection is not a straightforward process. Schölkopf et al. [24] introduce KPCA acknowledging that the question of how to choose the best kernel remains unanswered. While researchers have proposed methods that assist with kernel selection [46], determining the best kernel remains, to some degree, a trial and error process. For this specific application, ISOMAP’s specially constructed Gram matrix proved to be the most effective. Thus, the ISOMAP algorithm resulted in the lowest error when predicting the nodal stresses. The method with the lowest error in predicting the nodal coordinates was the inverse-transform-based PCA, as it had a lower mean and median and a more narrow distribution than all of the other tested algorithms. Apart from PCA-IT, all of the other methods had similar shaped distributions with roughly the same variance and range. Compared to PCA-IT, the other algorithms had between 2.4 and 3.2 times as much mean and median error. While all of these errors were small (less than 0.23%), the PCA-IT method outperformed the others.

Bunnell et al. [18] found that error was highly dependent on the number of training samples and the number of geometric parameters used in the design space. It was also believed that the number of nodes in the finite element mesh had an effect. As such, multiple regression was performed to create models that predicted the mean double-NRMSE error for stress and coordinates after accounting for the number of training samples, the number of geometric parameters, and the number of nodes. Multiple regression was used to compare the different methods to see if there were statistically significant differences among them. Tables 3.3(a) and 3.3(b) show the results of the multiple regressions, one for stress and one for coordinates. For both regressions, the two
PCA methods were combined into one by taking the lowest double-NRMSE of the two. As the number of nodes and the method variables were categorical, a reference level for each was used as a comparison for all other levels. In these tables, the “Intercept[50000, PCA]” term denotes the predicted double-NRMSE for a finite element mesh with 50,000 nodes, zero parameters, zero training samples, and a PCA-based surrogate. All other intercepts listed are additional errors to this reference error for other numbers of nodes (10,000, 25,000) and other methods (ISOMAP, LLE, KPCA). The total predicted double-NRMSE is found by combining the appropriate intercepts with the added error from the slope terms based on the number of parameters and number of training samples present in the design space of interest. Tables 3.3(a) and 3.3(b) each list the regression’s

Table 3.3: Coefficient estimates, p-values, and confidence intervals for each term in the multiple regression analysis that predicts the double-NRMSE for stress and coordinates.

(a) Stress

<table>
<thead>
<tr>
<th>Term</th>
<th>Estimate</th>
<th>p-value</th>
<th>95% CI Lower</th>
<th>95% CI Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression Intercept</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intercept[50000, PCA]</td>
<td>-0.6251</td>
<td>&lt;0.0001</td>
<td>-0.8445</td>
<td>-0.4058</td>
</tr>
<tr>
<td>Num Nodes[10000]</td>
<td>0.1312</td>
<td>0.1205</td>
<td>-0.0345</td>
<td>0.2969</td>
</tr>
<tr>
<td>Num Nodes[25000]</td>
<td>-0.0092</td>
<td>0.9126</td>
<td>-0.1734</td>
<td>0.1550</td>
</tr>
<tr>
<td>Method[ISOMAP]</td>
<td>-0.6457</td>
<td>&lt;0.0001</td>
<td>-0.8588</td>
<td>-0.4326</td>
</tr>
<tr>
<td>Method[LLE]</td>
<td>0.2016</td>
<td>0.0637</td>
<td>-0.0115</td>
<td>0.4147</td>
</tr>
<tr>
<td>Method[KPCA-Poly]</td>
<td>0.0121</td>
<td>0.9112</td>
<td>-0.2010</td>
<td>0.2252</td>
</tr>
<tr>
<td>Method[KPCA-RBF]</td>
<td>0.9712</td>
<td>&lt;0.0001</td>
<td>0.7581</td>
<td>1.1843</td>
</tr>
<tr>
<td>Regression Slopes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Num Parameters</td>
<td>0.8400</td>
<td>&lt;0.0001</td>
<td>0.7828</td>
<td>0.8972</td>
</tr>
<tr>
<td>Num Training</td>
<td>-0.0186</td>
<td>&lt;0.0001</td>
<td>-0.0226</td>
<td>-0.0146</td>
</tr>
</tbody>
</table>

(b) Coordinates

<table>
<thead>
<tr>
<th>Term</th>
<th>Estimate</th>
<th>p-value</th>
<th>95% CI Lower</th>
<th>95% CI Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression Intercept</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intercept[50000, PCA]</td>
<td>-0.0006</td>
<td>0.9334</td>
<td>-0.0136</td>
<td>0.0125</td>
</tr>
<tr>
<td>Num Nodes[10000]</td>
<td>0.0004</td>
<td>0.9406</td>
<td>-0.0093</td>
<td>0.0100</td>
</tr>
<tr>
<td>Num Nodes[25000]</td>
<td>-0.0062</td>
<td>0.2032</td>
<td>-0.0158</td>
<td>0.0034</td>
</tr>
<tr>
<td>Method[ISOMAP]</td>
<td>0.1374</td>
<td>&lt;0.0001</td>
<td>0.1250</td>
<td>0.1498</td>
</tr>
<tr>
<td>Method[LLE]</td>
<td>0.1015</td>
<td>&lt;0.0001</td>
<td>0.0891</td>
<td>0.1139</td>
</tr>
<tr>
<td>Method[KPCA-Poly]</td>
<td>0.1321</td>
<td>&lt;0.0001</td>
<td>0.1197</td>
<td>0.1444</td>
</tr>
<tr>
<td>Method[KPCA-RBF]</td>
<td>0.1482</td>
<td>&lt;0.0001</td>
<td>0.1358</td>
<td>0.1606</td>
</tr>
<tr>
<td>Regression Slopes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Num Parameters</td>
<td>0.0359</td>
<td>&lt;0.0001</td>
<td>0.0326</td>
<td>0.0392</td>
</tr>
<tr>
<td>Num Training</td>
<td>-0.0016</td>
<td>&lt;0.0001</td>
<td>-0.0019</td>
<td>-0.0014</td>
</tr>
</tbody>
</table>
intercept and slope values along with their 95% confidence intervals. Each intercept’s and slope’s p-value is also included to indicate whether or not it is different from zero, with p-values less than 0.1 in bold.

For the stress results, a negative estimate with a p-value of less than 0.0001 is convincing evidence that ISOMAP had less error than PCA. ISOMAP is estimated to have 0.65% less double-NRMSE (mean) than PCA, with a 95% confidence interval of 0.43% to 0.86%. As this is a prediction on the mean double-NRMSE, some design spaces will experience greater improvement using ISOMAP-based surrogate models. Of the 200 tested design spaces, the largest reduction in double-NRMSE occurred in a seven parameter design space, where the double-NRMSE was reduced by a magnitude of 3.77% from 6.24% (PCA) to 2.47% (ISOMAP). LLE is estimated to have 0.20% more double-NRMSE than PCA, with a 95% confidence interval of -0.01% to 0.41% and a p-value of 0.06. KPCA with a RBF kernel is estimated to have 0.97% more double-NRMSE than PCA, with a 95% confidence interval of 0.76% to 1.18% and a p-value of less than 0.0001. A p-value of 0.91 is not suggestive that the error of KPCA with a polynomial kernel is statistically different from the error of PCA. For the coordinate results, positive estimates with p-values of less than 0.0001 are convincing evidence that each of the nonlinear methods had more error than PCA. For both stress and coordinates, there were not statistically significant differences among finite element meshes with 10,000, 25,000, and 50,000 nodes. As expected in both regressions, positive slopes for the “Num Parameters” variable suggest that double-NRMSE increased as more parameters were changed in the design space and negative slopes for the “Num Training” variable suggest that double-NRMSE decreased as more training samples were used to train the surrogates. After accounting for the number of training samples (“Num Training”), the number of geometric parameters (“Num Parameters”), and the number of nodes (“Num Nodes”), multiple regression suggests that ISOMAP performs best at predicting nodal stresses and PCA performs best at predicting nodal coordinates.

### 3.4.1 Measuring Linearity of the Manifold

A surrogate model’s predictions are based off the relationship between the inputs and the outputs. If this relationship is nonlinear, a nonlinear dimensionality reduction technique may be able to unfold the data and create a linear relationship by discovering the data’s true dimensionality.
The resultant surrogate could be more accurate due to the simplified relationship between the inputs and outputs. Thus, it is believed that the nonlinear dimensionality-reduction-based surrogates can outperform linear dimensionality-reduction-based surrogates if the manifold geometry on which the dimensionality reduction is performed is nonlinear. If this is true, ISOMAP’s improvement over PCA should increase as the nonlinearity of the manifold increases. The results presented above therefore imply that the stress manifold should be more nonlinear than the coordinates manifold. To measure the linearity of the manifold, a new term, called the manifold distance ratio (MDR), is introduced. The MDR is computed as shown in Eqn. 3.11, where $x_{geoij}$ is the geodesic distance between points $i$ and $j$, $x_{eucij}$ is the Euclidean distance between points $i$ and $j$, and $n$ is the number of points on the manifold.

$$
MDR = \max \left( \frac{x_{geoij}}{x_{eucij}} \right), \quad \forall i \in \{1, 2, \ldots, n\}, \quad \forall j \in \{1, 2, \ldots, n\}, \quad i \neq j \quad (3.11)
$$

The geodesic distance is the shortest path along the curvature of the manifold surface (Fig. 3.5). Without an equation defining the surface geometry the exact geodesic distance cannot be computed, but it can be approximated using the same technique used in the ISOMAP algorithm. That is, calculating the geodesic distance as the shortest path through the k-nearest neighbor graph. By definition, the geodesic distance will equal the Euclidean distance in a perfectly linear space. As the curvature of the manifold increases, the geodesic distance will also increase. The ratio of the geodesic distance to the equivalent Euclidean distance provides insight into how much curvature exists on the manifold.

Fig. 3.10 shows the double-NRMSE for stress and coordinates as a function of the MDR for the 200 tested design spaces. Whether for PCA or ISOMAP, the double-NRMSE for stress did not exceed 0.8% for MDRs below 1.7. Error quickly increased with larger MDRs, with ISOMAP generally having less error than PCA. In design spaces with MDRs larger than 1.7, the mean double-NRMSE using PCA- and ISOMAP-based surrogates was 3.31% and 2.33%, respectively. The spread in the double-NRMSE above MDRs of 1.7 could be due to several factors. A nonlinear manifold could take on any number of shapes (e.g. twist, concavity, sinuosity), and variations in the shape itself could affect how well ISOMAP and PCA embed the data. Also, the geodesic distance used in computing the manifold distance ratio is an approximation and is greatly influ-
Figure 3.10: Manifold distance ratios versus double-NRMSE for predicting stress (a) and coordinates (b).

enced by the density and allocation of points on the manifold and the number of neighbors used in constructing the k-nearest neighbor graph. Varying these hyperparameters changed the value of the MDR for each design space to some degree; however, the overall trends remained the same. The reported MDRs all used the ISOMAP hyperparameter configuration that resulted in the lowest double-NRMSE from the 1000 tested configurations with one additional constraint: the number of neighbors was between 20% and 40% of the number of samples. This constraint helped ensure a more accurate measure of the geodesic distance in sparse design spaces while still reducing the risk of short-circuiting, an ISOMAP shortcoming that occurs when the distance between neighbors exceeds the distance between folds on the manifold [26, 47]. Design spaces with a more linear manifold (e.g. one design parameter), and thus lower MDRs, also had lower double-NRMSEs. This same trend was seen with the coordinates where 98% of the design spaces had a MDR lower than 1.7 and a double-NRMSE below 0.5%. Only four design spaces had MDRs above 1.7, with the highest being 1.9. Table 3.4 summarizes these findings. Nonlinear dimensionality-reduction-based surrogate models can have less error than PCA-based surrogate models when the manifold is sufficiently nonlinear (MDR > 1.7), such as in the stress space where the mean double-NRMSE was nearly 1% lower for ISOMAP than for PCA. The lower MDRs in the coordinate space suggest
that it is more linear than the stress space, thus a linear dimensionality reduction technique such as PCA is sufficient.

Table 3.4: Double-NRMSE for PCA and ISOMAP at different levels of the manifold distance ratio (MDR).

<table>
<thead>
<tr>
<th>Method</th>
<th>Stress MDRs ≤ 1.7</th>
<th>Stress MDRs &gt; 1.7</th>
<th>Coordinates All MDRs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Max</td>
<td>Mean</td>
</tr>
<tr>
<td>PCA</td>
<td>0.17</td>
<td>0.67</td>
<td>3.31</td>
</tr>
<tr>
<td>ISOMAP</td>
<td>0.24</td>
<td>0.77</td>
<td>2.33</td>
</tr>
</tbody>
</table>

### 3.4.2 Computational Cost

Understanding the computational cost of each dimensionality reduction technique is critical in order to determine its practicality. While ISOMAP-based surrogate models may have a lower double-NRMSE when predicting stress than PCA-based surrogate models, they may be unusable if the computational cost is too extreme. Fortunately, this was not the case. Table 3.5 lists the total time required for each method to tune its parameters, embed the data, and create the surrogates. The reported values are the maximum times across all 200 tested design spaces. Times were measured on a computer with 32 GB of RAM running at 3.40 GHz. PCA was the fastest algorithm, requiring only three minutes to complete. It was closely followed by the RBF and polynomial kernel KPCA and ISOMAP algorithms at four, nine, and 14 minutes, respectively. LLE took more than 10 times longer than the next closest algorithm at 144 minutes. While ISOMAP did require more time to embed the data than PCA, the time required to do so was sufficiently low for the method to be practical. No matter the dimensionality reduction technique used, the end result was a dimensionality-reduction-based surrogate. Once the data was embedded and the surrogates were created, all of the methods performed similarly at making real-time predictions. An ISOMAP-based dual-RBF surrogate model decreases surrogate error without becoming computationally infeasible.
Table 3.5: Maximum total time (minutes) required for each method to tune hyperparameters, embed data, and create surrogates.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>3</td>
</tr>
<tr>
<td>ISOMAP</td>
<td>14</td>
</tr>
<tr>
<td>LLE</td>
<td>144</td>
</tr>
<tr>
<td>KPCA-Poly</td>
<td>4</td>
</tr>
<tr>
<td>KPCA-RBF</td>
<td>9</td>
</tr>
</tbody>
</table>

3.5 Conclusion

When applied to a nonlinear data set, nonlinear dimensionality-reduction-based surrogate models may be more accurate than linear dimensionality-reduction-based surrogates. As such, four different dimensionality reduction techniques, one linear and three nonlinear, were implemented in conjunction with surrogate modeling to predict the FEA results of a jet engine compressor blade. The results show that using an ISOMAP-based dual-RBF surrogate model for predicting nodal stress decreased the error of the surrogate. While LLE- and KPCA-based surrogate models also resulted in a lower mean error, ISOMAP-based surrogates had the largest reduction in error. The mean error using ISOMAP was estimated to be 35.7% lower than the mean error using the baseline PCA. Using nonlinear dimensionality reduction in lieu of PCA did not reduce error for predicting nodal coordinates. Multiple regression was also used to predict the double-NRMSE of each method after accounting for the number of training samples used to train the surrogates, the number of geometric parameters changed in the model, and the number of nodes in the finite element mesh. From these models, there was statistically significant evidence to suggest that ISOMAP-based surrogate models had the largest reduction in error over PCA-based surrogates for predicting nodal stress. Statistically significant evidence also suggested that PCA-based surrogates had the least error for predicting nodal coordinates.

The computational costs of the different methods were also evaluated. PCA was the fastest algorithm tested, closely followed by KPCA and ISOMAP. LLE required significantly longer (10x) to tune its parameters, embed the data, and create the surrogates. For all 200 tested design spaces, the ISOMAP algorithm completed in under 15 minutes. This allowed the ISOMAP-based surro-
gates to remain computationally feasible for practical use. Once the surrogates were created, all of the methods performed similarly at making real-time predictions.

The results also show that the double-NRMSE was higher for design spaces with more nonlinear manifolds. A new metric, called the manifold distance ratio (MDR), was introduced that measured the linearity of a manifold. The MDRs suggested that the stress space was more nonlinear than the coordinates space, resulting in higher double-NRMSEs for predicting nodal stress than for predicting nodal coordinates. When predicting nodal stress, the reduction in error achieved by using nonlinear dimensionality reduction (ISOMAP) was attributed to the nonlinear geometry of the manifold. Nonlinear dimensionality-reduction-based surrogates use a simplified relationship between the inputs and outputs that can lead to more accurate predictions when the manifold geometry is sufficiently nonlinear. This leads to improved accuracy when performing DSE, allowing engineers to explore design spaces with greater confidence in the predicted results.
CHAPTER 4. DIMENSIONALITY-REDUCTION-BASED SURROGATES FOR EMULATING NODAL DISPLACEMENTS AND THE MODAL ASSURANCE CRITERION

As demonstrated in Chapter 3, nonlinear dimensionality-reduction-based surrogate models can increase surrogate accuracy compared to their linear counterparts in sufficiently nonlinear design spaces. The purpose of this chapter is to apply the same methods presented in Chapter 3 to a new application in order to further explore their ability to create more accurate surrogate models for emulating FEA nodal properties, including mode shapes and the modal assurance criterion. This chapter is based on an unpublished manuscript that will be submitted for publication [14]. The research presented in this chapter contributes to the field of real-time structural DSE in the following ways:

1. Provide a comparative analysis of linear and nonlinear dimensionality-reduction-based surrogate models in their ability to emulate FEA nodal results from a dynamic analysis, including mode shapes and the modal assurance criterion.

2. Provide a comparative analysis of dimensionality-reduction-based surrogate models applied to design spaces of a turbomachinery compressor blade.

3. Identify that mode shape reversal is a real phenomenon that has a great effect on surrogate accuracy.

4.1 Introduction

Modal analysis is a tool used to understand the dynamic or vibratory characteristics of designs, including turbomachines. Modal analysis is a finite element analysis (FEA) process that determines the mode shapes, or relative displacements at each node, and the natural frequencies at which they occur. This analysis helps engineers understand and estimate how and when a part will vibrate during operation. If these design characteristics are not considered, mode excitation may
occur and lead to part failure. It is common practice to use the modal assurance criterion (MAC) to measure the similarity between two mode shapes in order to classify modes [28, 48, 49].

Modal analysis may also be coupled with design space exploration (DSE), a systematic analysis of designs based on a parameterized model. DSE allows engineers to explore a suite of potential designs by changing the parameters within the model, a process that helps determine a more optimal design. Modal analysis, however, can be a computationally expensive process. As model complexity increases, so does the required solution time. This may hinder an engineer’s ability to perform DSE due to time and computational constraints. To help address this issue, tools and methods have been developed that decrease the time required to evaluate the FEA results of a design. One such tool commonly used to emulate FEA results as part of DSE is surrogate modeling [3, 4, 13, 28]. Surrogate models rely on a set of training data to create a mathematical relationship between sets of independent and dependent variables. Once trained, the surrogate can quickly predict a set of outputs for a given set of inputs. The accuracy of these predictions is closely related to the amount of training data used to define this relationship [18]. In general, accuracy improves with more training data. When used with FEA, surrogates can predict FEA nodal properties, such as stress, coordinates, or displacements, when given a set of geometric inputs, such as length, width, and height. Surrogate modeling may also be coupled with dimensionality reduction. Researchers have found that this can lead to faster and more accurate predictions [4, 13]. Thelin [28] used surrogates to predict MAC values but did not explore the use of dimensionality-reduction-based surrogate models. The high errors they found for various mode shape comparisons could potentially be reduced by using nonlinear dimensionality reduction methods.

This paper explores surrogate models based on dimensionality reduction to predict the MAC values obtained by comparing the FEA mode shapes of a jet engine compressor blade, as nonlinear dimensionality reduction may improve surrogate accuracy. To the authors’ knowledge, no such study has been published in the literature. Four different dimensionality reduction methods are compared, namely principal component analysis (PCA), kernel principal component analysis (KPCA), isometric feature mapping (ISOMAP), and locally linear embedding (LLE). These methods are tested on the Transonic Purdue Research Compressor [30]. Each method is applied to the FEA nodal displacements of a training set of designs, the results of which are used as inputs to the surrogate model. Rather than predict the nodal displacements directly, an intermediate surrogate is
used that predicts the value of each dimension in the dimensionality reduced space. A secondary surrogate is then used that predicts the high-dimensional values, after which MAC values are calculated from the predicted mode shapes. The results will show that using nonlinear dimensionality reduction techniques (KPCA, ISOMAP, LLE) can lead to reduced error in the surrogate models compared to linear dimensionality reduction (PCA).

This paper proceeds with a background section on the modal assurance criterion, surrogate modeling, and the dimensionality reduction techniques used in this research. The process of implementing these techniques and constructing the surrogate models is then outlined in the methods section. In the results section, a direct comparison is made between linear and nonlinear dimensionality-reduction-based surrogate models and their ability to predict the MAC values obtained by comparing the FEA mode shapes of a jet engine compressor blade. Finally, the conclusion summarizes the key findings and discusses the benefits of using nonlinear dimensionality-reduction-based surrogate models for performing DSE of modal analysis results.

4.2 Background

A background on the modal assurance criterion, surrogate modeling, and dimensionality reduction is presented herein. The interested reader is referred to the referenced works for further explanations and details.

4.2.1 Modal Assurance Criterion

The Modal Assurance Criterion (MAC) computes the similarity between two mode shape vectors consisting of the $X$, $Y$, and $Z$ displacements ($\delta_x, \delta_y, \delta_z$) at every node in the finite element mesh. Each vector ($\phi$) is structured as shown in Eqn. 4.1, where $n$ represents the total number of nodes.

$$\phi = \{\delta_{\text{x},1}, \ldots, \delta_{\text{x},n}, \delta_{\text{y},1}, \ldots, \delta_{\text{y},n}, \delta_{\text{z},1}, \ldots, \delta_{\text{z},n}\}$$ (4.1)

Given mode shape $A$ and mode shape $B$, the MAC value is obtained from Eqn. 4.2, where $\{\phi_A\}_i$ represents the displacement of the $i^{th}$ entry in mode shape vector $A$, $\{\phi_B\}_i$ represents the displacement of the $i^{th}$ entry in mode shape vector $B$, and $N$ represents the total number of entries
in the vectors. The MAC value is bounded between 0 and 1, with 0 indicating no consistency between mode shapes and 1 indicating perfect consistency between mode shapes (i.e. the mode shapes are the same) [48].

\[
MAC_{A:B} = \frac{\left| \sum_{i=1}^{N} \{\phi_A\}_i \{\phi_B\}_i \right|^2}{\left( \sum_{i=1}^{N} \{\phi_A\}_i \{\phi_A\}_i \right) \left( \sum_{i=1}^{N} \{\phi_B\}_i \{\phi_B\}_i \right)} \tag{4.2}
\]

The MAC value is often used to identify mode types, such as first bending mode or second torsional mode, by comparing mode displacement vectors of unknown type to a reference mode displacement vector whose type is known. Design to design variations often change mode shapes. In general, MAC values above 0.9 indicate matching mode types [48].

4.2.2 Surrogate Modeling

Surrogate modeling is an all-encompassing term for approximation techniques that create a mathematical relationship between sets of independent and dependent variables. A set of known dependent and independent values, called a training set, is used to define this relationship and create a function that predicts a set of outputs when given a set of inputs. Many types of surrogate models exist, including radial basis functions, artificial neural networks, and response surface models [33]. In general, the mathematical functions used by surrogates make predictions much faster than the original process used to obtain results, such as FEA. In some case, these approximations can be evaluated almost instantaneously. This means that the results of computationally expensive analyses can be predicted by a surrogate in real time and, when trained properly, with minimal error. Thus, surrogate models enable DSE in ways that would otherwise be impractical.

Full-nodal FEA surrogate models emulate results for every node in the finite element mesh, such as those illustrated in Fig. 4.1. A separate surrogate model may be used to emulate the results for each node (nodal surrogate), such as geometric displacements, in each degree of freedom. Heap et al. [20] used this approach to emulate the FEA results at each node of parametric, two-dimensional meshes for various cantilevered beams under different loading conditions, resulting in real-time design space exploration. Bunnell et al. [18] expanded this method to three-dimensional meshes of jet engine compressor blades and demonstrated the capability of full-nodal surrogates to visualize FEA stresses in real time even with a large number of nodes (25,000 to 500,000).
In a later work, Bunnell [4] made a comparison between nodal surrogates and dimensionality-reduction-based surrogates, demonstrating that the dimensionality-reduction-based surrogates had increased speed without losing accuracy. Bird et al. [13] further explored the use of dimensionality-reduction-based surrogate models for real-time design space exploration of FEA results, showing that nonlinear methods can decrease surrogate error in nonlinear design spaces by as much as 36%.

Each of these studies used surrogate modeling to help designers more thoroughly explore design spaces. Minimizing surrogate error leads to greater confidence in surrogate predictions and in turn safer, more reliable designs. Thelin [28] explored the use of full-nodal surrogates to predict the MAC values for various mode shapes of a jet engine compressor blade. These surrogates predicted nodal displacements (mode shapes) as an intermediate step to calculate MAC values. Even with an increased number of training samples, the surrogate error was high enough to establish inconfidence in the surrogate predictions. Nonlinear dimensionality reduction techniques were not applied to the nodal displacements, yet these methods have the potential to reduce the surrogate error of the full-nodal FEA surrogates if the design spaces are sufficiently nonlinear [13].
Dimensionality reduction is a technique that seeks to reduce the number of variables, or dimensions, needed to explain the variance in a set of data. In some cases, the data points in space $\mathbb{R}^p$ lie on some lower dimensional manifold in space $\mathbb{R}^m$, where $m < p$. If the true dimensionality of the data is known or can be discovered, then the data in the high-dimensional space can be represented in the lower dimensional space with minimal information loss. The mapping from high- to low-dimensional space can be linear or nonlinear, resulting in linear or nonlinear dimensionality reduction. This is illustrated in Fig. 4.2, where two-dimensional data is reduced to one-dimensional data using both linear and nonlinear dimensionality reduction by projecting the data points onto linear and nonlinear variances. The intrinsic geometry of the data (i.e. the spacing between points) is perfectly preserved for the nonlinear case, whereas it is altered for the linear case. For a given set of data, dimensionality reduction may produce lower dimensional data that can more easily be predicted by a surrogate, thereby reducing the surrogate’s error. Many linear and nonlinear methods exist, including principal component analysis (PCA), kernel principal component analysis (KPCA), isometric feature mapping (ISOMAP), and locally linear embedding (LLE). The remainder of this section is repeated (verbatim) from Sec. 2.3.

Figure 4.2: Dimensionality reduction: two-dimensional data is reduced to one-dimensional data using both linear and nonlinear dimensionality reduction.

**Principal Component Analysis**

Principal component analysis (PCA) was first introduced by Pearson [22] in 1901 and has been used extensively ever since. PCA seeks to discover the principal directions of variance within
a data set, as illustrated in Fig. 4.3. Those directions, or principal components (PCs), become the variables of the data when transformed into the lower dimensional space. PCA is a linear dimensionality reduction technique, as it solely performs linear transformations on the data. Jolliffe et al. [23] outline in detail how this is achieved. PCA can be accomplished either by performing singular value decomposition (SVD) on the centered data matrix or by performing eigenvalue decomposition on the covariance matrix. For a $n \times p$ matrix $X$, with $n$ data points of dimension $p$, the general steps for the SVD approach are as follows: center the data by subtracting the column means $\bar{X}$, perform singular value decomposition to solve for $U$, $S$, and $V$ (Eqn. 4.3), sort the eigenvectors ($V$) by their corresponding eigenvalues, and select $m$ eigenvectors to form the lower dimensional space $\mathbb{R}^m$, where $m < p$.

$$X - \bar{X} = USV$$

(4.3)

![Figure 4.3: The principal components discovered using PCA can be used to describe the variance in the data.](image)

Each of the $n$ samples in $X$ may be represented as a linear combination of the PCs contained in $V$. The coefficients used in this linear combination are called the PC scores and may be obtained by multiplying $US$. The data can be transformed to the lower dimensional space $\mathbb{R}^m$ by projecting each of the $n$ data points onto the $m$ PCs through

$$Y_{PCA} = US$$

(4.4)
Given a set of low-dimensional values, \( Y_{\text{PCA}} \), the data can be transformed back to the high-dimensional space (inverse transform) through

\[
X = Y_{\text{PCA}}V + \bar{X}
\]  

(4.5)

**Kernel Principal Component Analysis**

Kernel principal component analysis (KPCA) is an extension of PCA, first introduced by Schölkopf et al. [24] as a nonlinear form of PCA. KPCA is closely connected to multidimensional scaling (MDS), which constructs a dissimilarity matrix that contains the dissimilarities between all pairwise data points [25]. If the dissimilarities are computed using Euclidean distances, classical MDS aims to reduce the dimensionality of a data set in such a way that the Euclidean distances between points in the low-dimensional space are the same as the Euclidean distances between points in the high-dimensional space. Williams [25] outlines in detail how this is performed.

MDS involves the following steps: construct matrix \( A \) of all pairwise dissimilarities (i.e. squared Euclidean distances) and solve for the eigenvalues and eigenvectors of the matrix \( K \) defined as

\[
K = -\frac{1}{2}BAB = (BX)(BX)^T
\]  

(4.6)

where \( B \) is the centering matrix. If the eigenvalue decomposition of \( K \) is

\[
K = V \Lambda V^{-1}
\]  

(4.7)

where \( V \) and \( \Lambda \) contain the eigenvectors and eigenvalues of \( K \), respectively, the low-dimensional representation of the data can be obtained through

\[
Y_{\text{MDS}} = \Lambda^{1/2}V
\]  

(4.8)

For a \( n \times p \) matrix \( X \), with \( n \) data points of dimension \( p \), MDS performs eigenvalue decomposition on the centered matrix \( XX^T \). PCA performs eigenvalue decomposition on the covariance matrix, which is constructed using \( X^TX \). Williams [25] shows that the MDS solution, \( Y_{\text{MDS}} \), is equivalent to the PCA solution, \( Y_{\text{PCA}} \) (i.e. PC scores), if matrix \( A \) is constructed using Euclidean distances.
KPCA utilizes MDS’s approach to compute the PC scores of $\phi(X)$, where $X$ has been transformed to a higher dimensional space, called the feature space, by some transformation $\phi()$. Schölkopf et al. [24] show that $\phi(X)\phi(X)^T$, or equivalently the dot product between all pairwise points in the feature space, can be computed using a kernel. This process is commonly referred to as the “kernel trick”. The key mechanism of the “kernel trick” lies in the fact that kernels, or functions, exist that are mathematically equivalent to performing the dot product of two vectors in some feature space. That is, the kernel can compute the dot products without having to perform the actual transformation. Evaluating the kernel is computationally less expensive than transforming the data to the feature space to compute dot products. Many kernels exist that compute the dot products in different feature spaces. The most common include polynomial, radial basis function, and sigmoid kernels. While PCA is considered a linear dimensionality reduction technique, KPCA may be nonlinear. As shown in Fig. 4.4, the linear variances (i.e. PCs) of the data discovered by PCA in the feature space will not necessarily, and usually are not, linear in the original space due to the transformation $\phi()$. As such, nonlinear variances in the data can be extracted through KPCA.

![Figure 4.4: Linear principal components in the high-dimensional feature space (left) will not necessarily be linear in the original space (right) due to the transformation, thus allowing KPCA to discover nonlinear variances in the data.](image)

**Isometric Feature Mapping**

Isometric feature mapping (ISOMAP) was introduced by Tenenbaum et al. [26] as an extension of multidimensional scaling (MDS). When finding an embedding, or low-dimensional representation of the data, classic MDS preserves pairwise distances between points. This process is
equivalent to PCA if Euclidean distances are used. ISOMAP differs in that it preserves the pairwise geodesic distances, or the shortest path between two points along a curved surface, in order to preserve the intrinsic geometry of the data. Mathematically, it is not possible to compute a geodesic distance along a surface if the shape of the surface is not known. ISOMAP overcomes this by creating a k-nearest neighbors graph and approximating the geodesic distance between any two points as the shortest path through the k-nearest neighbors graph (i.e. a summation of Euclidean distances) as shown in Fig. 4.5. ISOMAP involves the following steps: construct a k-nearest neighbors graph, compute the geodesic distances between all points to form matrix $D$, and solve for the eigenvalues and eigenvectors of the matrix $K$ defined as

$$K = -\frac{1}{2}BAB \quad (4.9)$$

where $A$ is the matrix of squared geodesic distances ($A_{ij} = D_{ij}^2$), and $B$ is the centering matrix. The low-dimensional representation of the data, $Y_{ISO}$, can be obtained by using Eqns. 4.7 and 4.8.

**Locally Linear Embedding**

Locally linear embedding (LLE) models the global structure of the data manifold as a union of locally linear fits. Roweis et al. [27], the first to introduce LLE, outline in detail how this is done. Each data point is reconstructed using a weighted sum of its $k$ nearest neighbors, where $k$ is a parameter of the algorithm. The weights, $W$, are chosen such that the squared distances between the original $n$ data points, $X$, and their reconstructions are minimized according to
\[ W = \min \left( \sum_{i=1}^{n} \left| X_i - \sum_{j=1}^{k} W_{ij} X_j \right|^2 \right) \] (4.10)

Once determined, these same weights are used to perform the low-dimensional mapping such that the squared distances between the lower dimensional \( n \) data points, \( Y \), and their reconstructions are minimized according to

\[ Y_{LLE} = \min \left( \sum_{i=1}^{n} \left| Y_i - \sum_{j=1}^{k} W_{ij} Y_j \right|^2 \right) \] (4.11)

This process is illustrated in Fig. 4.6. The local geometry of the \( k \)-nearest neighbors is preserved for each point in the data set while still reducing the dimensionality of \( X \).

Figure 4.6: Locally linear embedding is performed by taking data in the high-dimensional space\(^{(a)}\) and using the \( k \) nearest neighbors of each point\(^{(b)}\) and their weights to reconstruct the data\(^{(c)}\) and find a low-dimensional embedding\(^{(d)}\).

4.3 Methods

If the design spaces are sufficiently nonlinear, the high surrogate errors observed by The- \( \text{lin} \) [28] may be reduced using nonlinear dimensionality reduction. This paper applies the dimensionality reduction techniques described in Section 4.2 to mode shapes obtained from FEA of the Transonic Purdue Research Compressor [30] shown in Figure 4.7. A parameterized model of the blade was created, a practice commonly used by engineers when performing DSE. The parameters, shown in Table 4.1, were allowed to vary from their nominal values within the specified ranges. The first five modes of the nominal geometry are shown in Figure 4.8, and their corresponding names and frequencies are listed in Table 4.2. The process utilized an automated workflow that is
illustrated in Figure 4.9 and described here. This includes the training data (blue boxes) and testing data (orange boxes) generation, as well as the training and testing of the surrogate model (right-most light gray boxes). A total of six different design spaces were created from the parametric model: three that varied angle, root chord, and lean, and three that varied all five parameters listed in Table 4.1. The three design spaces for each parameter set contained 100, 250, and 350 training designs, respectively.

![Purdue blade](image1)

![Purdue blade disk](image2)

Figure 4.7: Transonic Purdue Research Compressor blade, both the cross-sections used in the parametric model (a) and the blades oriented on a disk (b).

Table 4.1: Set of design parameters for the Transonic Purdue Research Compressor used for this research. The ranges for these variables were based on the relationship to a nominal value of the geometry.

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Symbol</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Chord</td>
<td>$C_R$</td>
<td>$\pm 20%$ of nominal $C_R$</td>
</tr>
<tr>
<td>Tip Chord</td>
<td>$C_T$</td>
<td>$\pm 20%$ of nominal $C_T$</td>
</tr>
<tr>
<td>Sweep</td>
<td>$S$</td>
<td>$\pm 15%$ of nominal $C_T$</td>
</tr>
<tr>
<td>Lean</td>
<td>$L$</td>
<td>$\pm 15%$ of nominal $C_T$</td>
</tr>
<tr>
<td>Angle</td>
<td>$\alpha$</td>
<td>$\pm 20 \deg$ of nominal $\alpha$</td>
</tr>
</tbody>
</table>
The first five modes are shown. Displacements have been normalized using min-max normalization.

Table 4.2: Mode names and frequencies of the nominal geometry of the Transonic Purdue Research Compressor.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Name</th>
<th>Frequency (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First bending</td>
<td>1263</td>
</tr>
<tr>
<td>2</td>
<td>First torsion</td>
<td>2850</td>
</tr>
<tr>
<td>3</td>
<td>Second bending</td>
<td>5428</td>
</tr>
<tr>
<td>4</td>
<td>Second torsion</td>
<td>7780</td>
</tr>
<tr>
<td>5</td>
<td>First chordwise bending</td>
<td>8042</td>
</tr>
</tbody>
</table>

4.3.1 Training and Testing Data Generation

The first step was to create training data from a design of experiments (DOE) with $m$ samples using Latin Hyper-sampling that varied each of the selected design variables within the ranges specified in Table 4.1, where $m$ was equal to 100, 250, or 350. All unselected design variables, such as tip chord and sweep for the three parameter design spaces, were held constant at their nominal values. As shown with the blue boxes in Figure 4.9, the parameterized blade geometry was updated using computer-aided design (NX) for each of the designs in the training DOE and solved in FEA (ANSYS) using a finite element mesh with approximately 25,000 nodes, as previous research successfully used this node count on this blade to perform structural DSE studies [13,28,31]. The modal analysis returned the mode shapes and the frequencies at which they occurred. The first five modes, or those with the lowest frequencies, were saved and used as the high-dimensional training data (middle blue box). Displacements were also normalized using min-
max normalization. Using the same parametric model and the same design space characteristics, including the parameters and number of samples, the corresponding high-dimensional testing data was created from a separate testing DOE (leftmost orange boxes). This process was performed separately for each mode shape and was repeated for each of the six design spaces. The nominal blade geometry was also solved in FEA to produce five reference mode shapes used in the MAC calculation (Equation 4.2).

![Diagram of workflow](image)

Figure 4.9: Automated workflow used to create the training data, testing data, and surrogate models.

### 4.3.2 Surrogate Model Training and Testing

The next step was to train and test the surrogate model, as illustrated in Figure 4.9 with the rightmost light gray boxes. The training DOE and data (blue boxes) was used to create a surrogate for predicting the x, y, and z displacements of every node in the finite element mesh. Radial basis functions (RBFs) have been shown to emulate FEA results in real time with minimal error [4, 13, 18]. An implementation of RBF, SciPy’s radial basis function, was used for this
Bunnell [4] found that linear dimensionality-reduction-based surrogate models (PCA) resulted in faster and in some cases more accurate predictions than nodal surrogates. Bird et al. [13] discovered that nonlinear methods could further reduce surrogate error in sufficiently nonlinear design spaces. Rather than use the nodal displacements directly to train the RBF surrogate, the low-dimensional representation of the data (bottom blue box) was used as an intermediate predictive step. The geometric design parameter values from the training DOE and the corresponding low-dimensional training data scores were used to train a RBF surrogate (Surrogate 1) that, given geometric parameters as an input, predicted the low-dimensional scores. To return to the high-dimensional space, a secondary RBF surrogate (Surrogate 2) was trained using the low- and high-dimensional training data to predict the high-dimensional nodal displacements when given low-dimensional scores. When combined, Surrogate 1 and Surrogate 2 formed a dual-surrogate that could be used to predict nodal displacements (rightmost orange box) of blade geometries throughout the design space. The predicted high-dimensional mode shapes from the testing data, as well as their true high-dimensional mode shapes obtained using FEA, were compared to the reference mode shapes of the nominal geometry using the MAC calculation (Equation 4.2). This process was repeated for PCA using both the dual-surrogate as well as replacing Surrogate 2 with PCA’s inverse transform (Equation 4.5).

To obtain the low-dimensional representation of the data (bottom blue box) used to train the dual-surrogate, dimensionality reduction was performed on the high-dimensional training data. This significantly reduced the dimensionality from 75,000 dimensions (x-, y-, z-displacement values at each of the 25,000 nodes) to a dimensionality set on the order of ten. PCA, ISOMAP, LLE, and KPCA each utilize different techniques when performing dimensionality reduction, yet they all share the same strategy: reduce the dimensionality of the data without losing information. Nonlinear dimensionality reduction may be able to better preserve the intrinsic geometry of the data in the low-dimensional space if nonlinear variances are present and can accurately be discovered (Figure 4.2). This can lead to more accurate surrogate models, as demonstrated by Bird et al. [13] for predicting FEA stresses. This process was repeated for each of the dimensionality reduction methods using scikit-learn’s PCA, Isomap, LocallyLinearEmbedding, and KernelPCA classes [42]. It is well known that the performance of these algorithms is greatly dependent on the values of their respective hyperparameters [50]. In general, these values are chosen based on the
true dimensionality and shape of the data. As these characteristics of the node displacement data were unknown, many hyperparameter configurations were tested. Table 4.3 lists the hyperparameters of each algorithm and the ranges within which they were varied. These hyperparameters are described in detail in scikit-learn’s documentation of these algorithms [42]. For each dimensionality reduction method, a DOE of up to 1000 different hyperparameter configurations was generated for tuning. If 1000 unique configurations using the available hyperparameter values were not possible, as was the case for the PCA algorithm, all possible configurations were tested. This amount was equal to the number of samples for PCA (Table 4.3). For each hyperparameter configuration, a dimensionality-reduction-based dual-surrogate was created as described above and tested to determine its error in predicting MAC values. The hyperparameter configuration that resulted in the lowest surrogate error was chosen.

<table>
<thead>
<tr>
<th>Class</th>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>n_components</td>
<td>[1, num_samples]</td>
</tr>
<tr>
<td></td>
<td>n_neighbors</td>
<td>[1, num_samples-1]</td>
</tr>
<tr>
<td>ISOMAP</td>
<td>n_components</td>
<td>[1, num_samples]</td>
</tr>
<tr>
<td></td>
<td>eigen_solver</td>
<td>dense</td>
</tr>
<tr>
<td>LLE</td>
<td>n_neighbors</td>
<td>[1, num_samples-1]</td>
</tr>
<tr>
<td></td>
<td>n_components</td>
<td>[1, num_samples]</td>
</tr>
<tr>
<td></td>
<td>reg</td>
<td>[0.0005, 0.25]</td>
</tr>
<tr>
<td></td>
<td>eigen_solver</td>
<td>dense</td>
</tr>
<tr>
<td>KPCA-RBF</td>
<td>kernel</td>
<td>rbf</td>
</tr>
<tr>
<td></td>
<td>n_components</td>
<td>[1, num_samples]</td>
</tr>
<tr>
<td></td>
<td>gamma</td>
<td>[0, 0.001]</td>
</tr>
<tr>
<td></td>
<td>eigen_solver</td>
<td>dense</td>
</tr>
<tr>
<td>KPCA-Poly</td>
<td>kernel</td>
<td>poly</td>
</tr>
<tr>
<td></td>
<td>n_components</td>
<td>[1, num_samples]</td>
</tr>
<tr>
<td></td>
<td>gamma</td>
<td>[0, 0.001]</td>
</tr>
<tr>
<td></td>
<td>degree</td>
<td>[1, 5]</td>
</tr>
<tr>
<td></td>
<td>coef0</td>
<td>[-1000, 1000]</td>
</tr>
<tr>
<td></td>
<td>eigen_solver</td>
<td>dense</td>
</tr>
</tbody>
</table>
4.4 Results

Once created, the dual-surrogate was tested for accuracy using normalized root mean square error (NRMSE) to obtain a single value for the error of predicting MAC values across every design in the testing set. This metric was evaluated separately for each design mode \((d)\) to reference mode \((r)\) comparison \((d:r)\). Each of the five modes from each design (design modes) were compared to each of the five modes from the reference, or nominal geometry, design (reference modes) that are shown in Fig. 4.8. For each blade geometry, the five mode shapes used for these comparisons were obtained from the modal analysis and consisted of the first five occurring modes when sorted by frequency. Eqn. 4.12 shows how the NRMSE is computed, where \(MAC_{d:r,i}\) is the MAC value of mode comparison \(d:r\) of test sample \(i\), \(\hat{MAC}_{d:r,i}\) is the surrogate’s prediction for that same MAC value, and \(MAC_{d:r,max}\) is the maximum MAC value across all test samples, \(n\), for mode comparison \(d:r\).

\[
NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{MAC}_{d:r,i} - MAC_{d:r,i})^2}{nMAC_{d:r,max}^2}} \tag{4.12}
\]

Using this metric, the various dimensionality-reduction-based surrogates were compared by examining the raw errors themselves as well as performing a statistical analysis to determine any statistically significant differences between them. The lowest NRMSE across all tested hyperparameter configurations for each method was saved for each of the six tested design spaces. This was performed for each of the 25 \(d:r\) mode comparisons (five design modes, five reference modes). Table 4.4 lists the NRMSEs of the five parameter design space with 100 training samples for each of the dimensionality reduction techniques. For PCA, both the inverse transform (PCA-IT) and dual-RBF (PCA-RBF) surrogates are shown. For KPCA, both the polynomial kernel (KPCA-Poly) and the RBF kernel (KPCA-RBF) surrogates are shown. NRMSEs are reported as a percentage. Mode comparisons 4:4, 4:5, 5:4, and 5:5 all had NRMSEs at least one order of magnitude higher than all other mode comparisons, with NRMSEs ranging from 11-19%. Using similar data sets, Thelin [28] found that this occurred in part due to mode switching, where modes four and five would switch places in their order of occurrence. The other 21 mode comparisons, regardless of the dimensionality reduction method, each had NRMSEs below 1.5%. This error is
Table 4.4: NRMSE of the predicted MAC values for each dimensionality reduction method in the five parameter design space with 100 training samples for the design mode (d) to reference mode (r) comparisons. Reported values are a percentage.

|      | PCA-IT | | | | | PCA-RBF | | | | | | ISOMAP | | | | | | LLE | | | | | | KPCA-Poly | | | | | | KPCA-RBF | | | | | |
|------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
|      | r = 1  | r = 2  | r = 3  | r = 4  | r = 5  | r = 1  | r = 2  | r = 3  | r = 4  | r = 5  | r = 1  | r = 2  | r = 3  | r = 4  | r = 5  | r = 1  | r = 2  | r = 3  | r = 4  | r = 5  | r = 1  | r = 2  | r = 3  | r = 4  | r = 5  |
| d = 1 | 0.22   | 0.12   | 0.02   | 0.01   | 0.03   | 0.22   | 0.13   | 0.03   | 0.01   | 0.06   | 0.23   | 0.12   | 0.03   | 0.01   | 0.04   | 0.06   |
| d = 2 | 0.07   | 0.90   | 0.74   | 0.04   | 0.07   | 0.07   | 0.94   | 0.77   | 0.05   | 0.07   | 0.10   | 0.66   | 1.15   | 0.25   | 0.48   | 0.25   |
| d = 3 | 0.11   | 1.02   | 1.32   | 0.25   | 0.69   | 0.11   | 0.98   | 1.46   | 0.26   | 0.64   | 0.10   | 0.66   | 1.15   | 0.25   | 0.48   | 0.25   |
| d = 4 | 1.11   | 0.57   | 0.71   | 15.03  | 15.33  | 1.38   | 0.66   | 0.62   | 14.39  | 14.84  | 1.44   | 0.42   | 1.20   | 14.48  | 18.50  |
| d = 5 | 1.23   | 0.39   | 1.49   | 14.62  | 18.85  | 1.44   | 0.42   | 1.20   | 14.48  | 18.50  | 1.44   | 0.42   | 1.20   | 14.48  | 18.50  |

These same trends were found in the other tested design spaces. In order to statistically compare the mode comparisons with high errors, a mixed model multiple regression analysis was performed to create a model that predicted the mean NRMSE for emulating MAC values after accounting for the number of training samples used to train the surrogate. The three and five parameter design spaces were treated separately, as error is dependent on the design variables used in the design space [13]. The dimensionality reduction method was used as a fixed effect in the regression in order to see any differences between the methods. The mode comparison was also included as a random effect to isolate the effects of the dimensionality reduction methods across the analyzed mode comparisons. Only the 4:4, 4:5, 5:4, and 5:5 mode...
comparisons were considered, as they exhibited the highest error. Tables 4.5(a) and 4.5(b) show the results of the mixed model multiple regressions for the three and five parameter design spaces, respectively. As the dimensionality reduction method variable was categorical, a reference level (PCA-IT) was used as a comparison for all other levels to determine if there were statistically significant differences between them. In these tables, the “Intercept[PCA-IT]” term denotes the predicted NRMSE of a PCA-based surrogate that uses the inverse transform. This estimate must be adjusted based on the number of training samples in the design space. All other intercepts listed are additional errors to this reference error for other methods (PCA-RBF, ISOMAP, LLE, KPCA-Poly, KPCA-RBF). The total predicted NRMSE is found by combining the appropriate intercepts with the added error from the slope term. For example, the estimated NRMSE using the

<table>
<thead>
<tr>
<th>Term</th>
<th>Estimate</th>
<th>p-value</th>
<th>95% CI Lower</th>
<th>95% CI Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intercepts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intercept[PCA-IT]</td>
<td>13.7448</td>
<td>&lt;0.0001</td>
<td>12.2159</td>
<td>15.2736</td>
</tr>
<tr>
<td>Method[PCA-RBF]</td>
<td>-0.1051</td>
<td>0.8152</td>
<td>-0.9998</td>
<td>0.7897</td>
</tr>
<tr>
<td>Method[ISOMAP]</td>
<td>-0.5503</td>
<td>0.2235</td>
<td>-1.4450</td>
<td>0.3444</td>
</tr>
<tr>
<td>Method[LLE]</td>
<td>-0.3949</td>
<td>0.3810</td>
<td>-1.2896</td>
<td>0.4998</td>
</tr>
<tr>
<td>Method[KPCA-Poly]</td>
<td>-0.2313</td>
<td>0.6072</td>
<td>-1.1260</td>
<td>0.6635</td>
</tr>
<tr>
<td>Method[KPCA-RBF]</td>
<td>-0.3093</td>
<td>0.4921</td>
<td>-1.2041</td>
<td>0.5854</td>
</tr>
<tr>
<td>Regression Slope</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Num Training</td>
<td>-0.0197</td>
<td>&lt;0.0001</td>
<td>-0.0222</td>
<td>-0.0172</td>
</tr>
</tbody>
</table>

Table 4.5: Coefficient estimates, p-values, and confidence intervals for each term in the mixed model multiple regression analysis that predicts the NRMSE for the three and five parameter design spaces. Reported estimates are a percentage.

<table>
<thead>
<tr>
<th>Term</th>
<th>Estimate</th>
<th>p-value</th>
<th>95% CI Lower</th>
<th>95% CI Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intercepts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intercept[PCA-IT]</td>
<td>15.6556</td>
<td>&lt;0.0001</td>
<td>13.6312</td>
<td>17.6800</td>
</tr>
<tr>
<td>Method[PCA-RBF]</td>
<td>-0.5849</td>
<td>0.2245</td>
<td>-1.5379</td>
<td>0.3680</td>
</tr>
<tr>
<td>Method[ISOMAP]</td>
<td>-0.5170</td>
<td>0.2824</td>
<td>-1.4700</td>
<td>0.4360</td>
</tr>
<tr>
<td>Method[LLE]</td>
<td>-1.6987</td>
<td>0.0007</td>
<td>-2.6516</td>
<td>-0.7457</td>
</tr>
<tr>
<td>Method[KPCA-Poly]</td>
<td>-0.7137</td>
<td>0.1395</td>
<td>-1.6666</td>
<td>0.2393</td>
</tr>
<tr>
<td>Method[KPCA-RBF]</td>
<td>-2.1249</td>
<td>&lt;0.0001</td>
<td>-3.0779</td>
<td>-1.1719</td>
</tr>
<tr>
<td>Regression Slope</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Num Training</td>
<td>-0.0028</td>
<td>0.0426</td>
<td>-0.0055</td>
<td>-0.0001</td>
</tr>
</tbody>
</table>
ISOMAP-based surrogate in a three parameter design space with 100 training samples is obtained by combining the reference intercept (13.75) with the additional intercept for ISOMAP (-0.55) and the additional slope with 100 training samples \((-0.02 \times 100)\) to arrive at 11.22% NRMSE. Table 4.6 lists the predicted NRMSEs for each method after making these necessary adjustments. Tables 4.5(a) and 4.5(b) list each regression’s intercept and slope values along with their 95% confidence intervals. Each intercept’s and slope’s p-value is also included to indicate whether or not it is different from zero, with p-values less than 0.1 in bold.

Table 4.6: Predicted NRMSE of each method in each of the tested design spaces using the results of the mixed model multiple regression analysis. Reported values are a percentage.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Samples (100)</th>
<th>Training Samples (250)</th>
<th>Training Samples (350)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA-IT</td>
<td>11.77</td>
<td>8.82</td>
<td>6.85</td>
</tr>
<tr>
<td>PCA-RBF</td>
<td>11.67</td>
<td>8.71</td>
<td>6.74</td>
</tr>
<tr>
<td>ISOMAP</td>
<td>11.22</td>
<td>8.27</td>
<td>6.30</td>
</tr>
<tr>
<td>LLE</td>
<td>11.38</td>
<td>8.42</td>
<td>6.45</td>
</tr>
<tr>
<td>KPCA-Poly</td>
<td>11.54</td>
<td>8.59</td>
<td>6.62</td>
</tr>
<tr>
<td>KPCA-RBF</td>
<td>11.47</td>
<td>8.51</td>
<td>6.54</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Samples (100)</th>
<th>Training Samples (250)</th>
<th>Training Samples (350)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA-IT</td>
<td>15.38</td>
<td>14.96</td>
<td>14.68</td>
</tr>
<tr>
<td>PCA-RBF</td>
<td>14.79</td>
<td>14.37</td>
<td>14.09</td>
</tr>
<tr>
<td>ISOMAP</td>
<td>14.86</td>
<td>14.44</td>
<td>14.16</td>
</tr>
<tr>
<td>LLE</td>
<td>13.68</td>
<td>13.26</td>
<td>12.98</td>
</tr>
<tr>
<td>KPCA-Poly</td>
<td>14.66</td>
<td>14.24</td>
<td>13.96</td>
</tr>
<tr>
<td>KPCA-RBF</td>
<td>13.25</td>
<td>12.83</td>
<td>12.55</td>
</tr>
</tbody>
</table>

In the three parameter design space, none of the p-values for the dimensionality reduction methods are statistically significant. All p-values are greater than 0.1, which fails to even suggest that any of the methods are statistically different from the inverse-transform-based PCA. In the five parameter design space, a negative estimate with a p-value of 0.0007 is convincing evidence that LLE had less error than the inverse-transform-based PCA. LLE is estimated to have 1.70% less NRMSE (mean), with a 95% confidence interval of 0.75% to 2.65%. Similarly, a negative estimate with a p-value of less than 0.0001 is convincing evidence that KPCA with a RBF kernel (KPCA-RBF) had less error than the inverse-transform-based PCA. KPCA-RBF is estimated to have 2.12% less NRMSE (mean), with a 95% confidence interval of 1.17% to 3.08%. This estimate equates to a 13.57% reduction in error compared to the baseline PCA. Negative slopes for the “Num Training” variable in both regressions suggest that NRMSE decreased as more training samples
were used to train the surrogates for the tested design spaces. After accounting for the number of training samples, a mixed model multiple regression analysis suggests that the high errors for mode comparisons 4:4, 4:5, 5:4, and 5:5 obtained by Thelin [28] can be reduced using nonlinear dimensionality reduction.

4.4.1 Linearity of the Manifold

The predictions made by a surrogate model are based on the relationship between the inputs and outputs. The secondary surrogates used in this research to return to the high-dimensional space thus depend on the relationship between the low- and high-dimensional values. If this relationship can be simplified to have less variability from design to design, the surrogate may be able to make more accurate predictions. The intrinsic geometry of a nonlinear data manifold can be discovered and preserved more accurately using nonlinear dimensionality reduction rather than linear dimensionality reduction. If this leads to a more predictable relationship between low- and high-dimensional values, nonlinear dimensionality-reduction-based surrogate models could outperform linear dimensionality-reduction-based surrogate models. As the nonlinearity of the manifold increases, so does the potential for improved accuracy using nonlinear methods. Bird et al. [13] found that more nonlinear data manifolds often resulted in improved accuracy using nonlinear dimensionality-reduction-based surrogate models. As shown above, statistically significant improvements using nonlinear dimensionality reduction were only seen in the five parameter design spaces. This suggests that these design spaces were more nonlinear than the three parameter design spaces. Bird et al. [13] introduced a new metric, the manifold distance ratio (MDR), to measure the linearity of the design space data manifolds. The same process described by Bird et al. [13] was used here to calculate the MDR for each of the six tested design spaces. The MDR is computed as shown in Eqn. 4.13, where $x_{geo_{ij}}$ is the geodesic distance between points $i$ and $j$, $x_{euc_{ij}}$ is the Euclidean distance between points $i$ and $j$, and n is the number of points on the manifold.

$$MDR = \max \left( \frac{x_{geo_{ij}}}{x_{euc_{ij}}} \right), \quad \forall i \in \{1, 2, ..., n\}, \quad \forall j \in \{1, 2, ..., n\}, \quad i \neq j \quad (4.13)$$

The geodesic distance is the shortest path along the manifold surface (Fig. 2.8). As curvature is introduced to this surface, the geodesic distance becomes longer than the Euclidean distance.

66
The ratio of the geodesic distance to the equivalent Euclidean distance provides a measure of the curvature, or nonlinearity, of the manifold. The exact geodesic distance between two points on the manifold cannot be computed without an equation defining the surface geometry. It can, however, be approximated as in the ISOMAP algorithm, where the geodesic distance is computed as the shortest path through the k-nearest neighbor graph. While not exact, this method closely approximates the true geodesic distance as long as k is appropriately chosen. If k is too large, the calculated geodesic distance may be too small due to short-circuiting, an ISOMAP shortcoming that occurs when distances between neighbors is large enough to bypass the manifold folds [26, 47]. If k is too small, the calculated geodesic distance may be too large due to excessive traversal of the manifold. This is especially true in sparse design spaces where points lie far apart on the manifold. Appropriate values of k are therefore dependent on the manifold geometry and density. Bird et al. [13] addressed this issue by forcing k, or the number of nearest neighbors, to be between 20% and 40% of the number of samples on the manifold. This constraint helped ensure a more accurate measure of the geodesic distance in sparse design spaces while still reducing the risk of short-circuiting. For each tested design space in this study, the geodesic distance was determined using the ISOMAP hyperparameter configuration that resulted in the lowest NRMSE while subject to the identified constraint.

The mixed model multiple regression of mode comparisons between modes four and five, as these mode comparisons exhibited exceptionally high errors, suggested that improved accuracy using nonlinear dimensionality-reduction-based surrogate models could be achieved in the five parameter design spaces but not in the three parameter design spaces. As such, it was expected that the MDR for modes four and five would be higher (more nonlinear) in the five parameter design spaces than in the three parameter design spaces. Tables 4.7(a) and 4.7(b) list the MDRs for these design spaces. Three levels of training samples were used for both parameter configurations, providing three measure of the MDR for each set of design spaces. For mode four, the average MDR of the five parameter design spaces was 0.27 higher than the three parameter design spaces. This 20.27% increase clearly shows that mode four was more nonlinear in the five parameter set. For mode five, this was not the case. The average MDR of the five parameter design spaces was only 0.06 higher than the three parameter design spaces, a 4.03% increase. As described above, this calculation utilized the ISOMAP algorithm with the hyperparameter configuration resulting in
Table 4.7: Manifold distance ratios (MDRs) of modes four and five for each of the tested design spaces.

(a) Mode 4

<table>
<thead>
<tr>
<th>Design Space</th>
<th>Number of Training Samples</th>
<th>Average MDR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>250</td>
</tr>
<tr>
<td>3 Parameters</td>
<td>1.40</td>
<td>1.31</td>
</tr>
<tr>
<td>5 Parameters</td>
<td>1.59</td>
<td>1.60</td>
</tr>
</tbody>
</table>

(b) Mode 5

<table>
<thead>
<tr>
<th>Design Space</th>
<th>Number of Training Samples</th>
<th>Average MDR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>250</td>
</tr>
<tr>
<td>3 Parameters</td>
<td>1.52</td>
<td>1.33</td>
</tr>
<tr>
<td>5 Parameters</td>
<td>1.48</td>
<td>1.51</td>
</tr>
</tbody>
</table>

the lowest error and constrained such that the number of nearest neighbors was between 20%-40% of the number of samples on the manifold. Bird et al. [13] used this constraint for manifolds with as few as ten data points, where the sparsity of points necessitated a larger number of neighbors to obtain a better measure of the true geodesic distance. The data manifolds used in this research were more dense, with 100, 250, or 350 data points, and thus should require fewer neighbors to accurately measure the geodesic distance. Changing the neighbors constraint from 20%-40% to 5%-15% in order to account for denser data manifolds resulted in the same trends as discussed. While nonlinear methods showed improved accuracy over linear methods, the MDRs calculated for mode five do not show this.

One possible explanation for the MDRs in Table 4.7(b) is that the ISOMAP algorithm was used to calculate the geodesic distance used in the MDR calculation. ISOMAP-based surrogates were statistically shown to have no improvement over PCA-based surrogates (Table 4.5), meaning that no ISOMAP hyperparameter configuration was able to discover a meaningful dimensionality that PCA could not. While the true shape of the mode five manifold is unknown, its geometry may be too complex for ISOMAP to accurately discover and preserve. In such a case, the geodesic distances calculated for mode five using the ISOMAP algorithm would not be accurate and could explain the discrepancy with the mode five MDR values. Conversely, if the ISOMAP algorithm can accurately estimate the geodesic distances for mode five, the mode five MDRs may lie close to the MDR threshold as demonstrated in Fig. 3.10 in Chapter 3. Using this same method of calculating the MDR, Bird et al. [13] found that ratios above 1.7 generally resulted in improved accuracy from nonlinear dimensionality-reduction-based surrogate models. For both modes four and five, the MDRs do not reach this threshold even though improvement was seen in the five
parameter design spaces. It is possible that higher NRMSEs, such as those present in this data set (10%-20%), lead to improvement with lower MDRs. The results of Table 4.7(a) suggest that the MDR threshold for the mode shape surrogates, or the point at which improvement is achieved from nonlinear methods, is somewhere between 1.33 and 1.60. The MDRs for mode five in Table 4.7(b) fall in the middle of this range and likely lie close to this threshold. Exploring more design spaces would help determine a more exact location of the MDR threshold.

4.4.2 Mode Shape Reversal

The mode shapes obtained from a modal analysis are eigenvectors [16]. For a given eigenvector, $\mathbf{v}$, of a matrix, $\mathbf{A}$, the eigenvector may be reversed as long as the sign of the corresponding eigenvalue, $\lambda$, is also reversed. This is shown in Eqn. 4.14, where both the positive and negative eigenvectors are acceptable solutions. Thus, the direction of an eigenvector is arbitrary.

$$A\mathbf{v} = \lambda \mathbf{v} = -\lambda (-\mathbf{v}) \quad (4.14)$$

When solving a set of designs in FEA, there is no guarantee that the mode shapes, or eigenvectors, from design to design will all be oriented in the same direction. While not tested as part of this research, this should occur independent of the mesh characteristics and the software used to perform the modal analysis. This phenomenon of eigenvector, or mode shape, reversal could affect surrogate accuracy, as the surrogate models were trained using these eigenvectors. If the affected designs can be identified, then they can be corrected by changing the sign of the displacements at every node. Identifying the affected designs, however, is not always a straightforward process. This is illustrated in Tables 4.8(a) and 4.8(b), where the displacements of modes two and four are shown for several designs in the five parameter design space with 100 training samples. Only the x-displacements for the first five nodes are shown, but the same patterns are seen for y- and z-displacements at other nodes. For mode two, the displacements of each node (columns) are of the same magnitude from design to design (rows). Each design also follows the same sign pattern with no sign change from node to node. Using this logic and assuming that design 1 is oriented correctly, the reversed designs (2, 3, 4, and 5) can easily be identified and corrected by changing the sign of each displacement. For mode four, this is not the case. Design 2 should be reversed to
Table 4.8: Node displacements ($\delta$) for the same five selected designs from the five parameter design space with 100 training samples.

(a) Mode 2

<table>
<thead>
<tr>
<th>Design</th>
<th>$\delta_{x,1}$</th>
<th>$\delta_{x,2}$</th>
<th>$\delta_{x,3}$</th>
<th>$\delta_{x,4}$</th>
<th>$\delta_{x,5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.951</td>
<td>0.921</td>
<td>0.889</td>
<td>0.856</td>
<td>0.823</td>
</tr>
<tr>
<td>2</td>
<td>-0.636</td>
<td>-0.621</td>
<td>-0.604</td>
<td>-0.587</td>
<td>-0.569</td>
</tr>
<tr>
<td>3</td>
<td>-0.792</td>
<td>-0.772</td>
<td>-0.750</td>
<td>-0.728</td>
<td>-0.705</td>
</tr>
<tr>
<td>4</td>
<td>-0.758</td>
<td>-0.738</td>
<td>-0.716</td>
<td>-0.694</td>
<td>-0.671</td>
</tr>
<tr>
<td>5</td>
<td>-0.809</td>
<td>-0.786</td>
<td>-0.762</td>
<td>-0.737</td>
<td>-0.711</td>
</tr>
</tbody>
</table>

(b) Mode 4

<table>
<thead>
<tr>
<th>Design</th>
<th>$\delta_{x,1}$</th>
<th>$\delta_{x,2}$</th>
<th>$\delta_{x,3}$</th>
<th>$\delta_{x,4}$</th>
<th>$\delta_{x,5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.894</td>
<td>0.846</td>
<td>0.797</td>
<td>0.747</td>
<td>0.698</td>
</tr>
<tr>
<td>2</td>
<td>-0.589</td>
<td>-0.574</td>
<td>-0.559</td>
<td>-0.542</td>
<td>-0.525</td>
</tr>
<tr>
<td>3</td>
<td>0.799</td>
<td>0.775</td>
<td>0.748</td>
<td>0.721</td>
<td>0.694</td>
</tr>
<tr>
<td>4</td>
<td>-0.034</td>
<td>-0.042</td>
<td>-0.050</td>
<td>-0.058</td>
<td>-0.067</td>
</tr>
<tr>
<td>5</td>
<td>0.008</td>
<td>-0.005</td>
<td>-0.019</td>
<td>-0.033</td>
<td>-0.048</td>
</tr>
</tbody>
</table>

match designs 1 and 3, but the correct orientation for designs 4 and 5 is ambiguous. The displacements for designs 4 and 5 are of a different order of magnitude and approach zero, and design 5 has a different sign pattern with sign reversal between nodes 1 and 2. Reversal is also inconsistent from mode to mode, as design 3 is only reversed for mode two. Thus, reversing designs 4 and 5 for mode two does not imply reversal for mode four. Identifying the correct orientation for designs such as these requires more complicated logic.

Problematic designs, such as designs 4 and 5 in Table 4.8(b), were seen most frequently in modes four and five. This could be due to mode switching, as identified by Thelin [28]. Mode shape reversal may lead to inaccurate surrogates, as a mixture of positive and negative values at each displacement creates more variability in the training data. This could contribute to the high errors seen in mode comparisons 4:4, 4:5, 5:4, and 5:5 (Table 4.4). To quantify the effect of mode shape reversal on surrogate error, four different approaches to identify reversed mode shapes were implemented and compared. The first approach was to simply do nothing. The mode shapes obtained from FEA were not corrected before training or testing the surrogate. The second approach involved randomly reversing 20% of the designs in the training and testing sets using a randomization function in Python. It was believed that 20% represented a sufficiently large proportion of the population to see the effect of random reversal. For the third approach, the node with the largest average (absolute value) x-, y-, or z-displacement was determined for both the training and testing sets. A design was reversed if the displacement at this node was negative. The fourth approach involved computing the dot product between each training or testing design and the reference design. If the dot product was negative, then the training or testing design was
reversed. None of these correction approaches changed the true MAC values, or the MAC values calculated from the FEA displacements. The MAC calculation (Eqn. 4.2) squares all terms, so a reversed mode shape will return the same MAC value. They did, however, affect the predicted MAC values, as the surrogates’ predicted mode shapes changed. To this point, all results shown have used approach three to identify and correct the reversed mode shapes. The results will show that this approach, overall, produced the lowest surrogate error. Table 4.9 lists the NRMSE of the d:r comparisons for the RBF kernel KPCA in the five parameter design space with 100 training samples, as this dimensionality reduction method had the largest reduction in error in this design space (Table 4.6(b)). NRMSEs above 5% are highlighted. The same patterns shown here were present in the other tested design spaces and for the other dimensionality reduction methods.

Table 4.9: NRMSE of the predicted MAC values using KPCA-RBF in the five parameter design space with 100 training samples. Each mode shape correction method is shown for every design mode (d) to reference mode (r) comparisons. Reported values are a percentage.

<table>
<thead>
<tr>
<th></th>
<th>r = 1</th>
<th>r = 2</th>
<th>r = 3</th>
<th>r = 4</th>
<th>r = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>d = 1</td>
<td>0.21</td>
<td>0.13</td>
<td>0.03</td>
<td>0.01</td>
<td>0.06</td>
</tr>
<tr>
<td>d = 2</td>
<td>0.46</td>
<td>17.15</td>
<td>1.09</td>
<td>0.28</td>
<td>0.31</td>
</tr>
<tr>
<td>d = 3</td>
<td>0.82</td>
<td>1.42</td>
<td>17.07</td>
<td>0.95</td>
<td>2.61</td>
</tr>
<tr>
<td>d = 4</td>
<td>1.34</td>
<td>1.03</td>
<td>0.78</td>
<td>18.41</td>
<td>14.24</td>
</tr>
<tr>
<td>d = 5</td>
<td>1.45</td>
<td>0.47</td>
<td>2.24</td>
<td>23.03</td>
<td>23.92</td>
</tr>
</tbody>
</table>

(a) Approach 1

<table>
<thead>
<tr>
<th></th>
<th>r = 1</th>
<th>r = 2</th>
<th>r = 3</th>
<th>r = 4</th>
<th>r = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>d = 1</td>
<td>0.22</td>
<td>0.13</td>
<td>0.03</td>
<td>0.01</td>
<td>0.06</td>
</tr>
<tr>
<td>d = 2</td>
<td>0.07</td>
<td>0.90</td>
<td>0.77</td>
<td>0.05</td>
<td>0.07</td>
</tr>
<tr>
<td>d = 3</td>
<td>0.11</td>
<td>0.91</td>
<td>1.33</td>
<td>0.26</td>
<td>0.64</td>
</tr>
<tr>
<td>d = 4</td>
<td>1.11</td>
<td>0.61</td>
<td>0.61</td>
<td>11.78</td>
<td>13.08</td>
</tr>
<tr>
<td>d = 5</td>
<td>1.19</td>
<td>0.34</td>
<td>1.10</td>
<td>14.00</td>
<td>14.54</td>
</tr>
</tbody>
</table>

(c) Approach 3

<table>
<thead>
<tr>
<th></th>
<th>r = 1</th>
<th>r = 2</th>
<th>r = 3</th>
<th>r = 4</th>
<th>r = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>d = 1</td>
<td>0.22</td>
<td>0.13</td>
<td>0.03</td>
<td>0.01</td>
<td>0.06</td>
</tr>
<tr>
<td>d = 2</td>
<td>0.07</td>
<td>0.90</td>
<td>0.77</td>
<td>0.05</td>
<td>0.07</td>
</tr>
<tr>
<td>d = 3</td>
<td>0.11</td>
<td>0.91</td>
<td>1.33</td>
<td>0.26</td>
<td>0.64</td>
</tr>
<tr>
<td>d = 4</td>
<td>1.11</td>
<td>0.61</td>
<td>0.61</td>
<td>11.78</td>
<td>13.08</td>
</tr>
<tr>
<td>d = 5</td>
<td>1.19</td>
<td>0.34</td>
<td>1.10</td>
<td>14.00</td>
<td>14.54</td>
</tr>
</tbody>
</table>

(d) Approach 4

<table>
<thead>
<tr>
<th></th>
<th>r = 1</th>
<th>r = 2</th>
<th>r = 3</th>
<th>r = 4</th>
<th>r = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>d = 1</td>
<td>0.22</td>
<td>0.13</td>
<td>0.03</td>
<td>0.01</td>
<td>0.06</td>
</tr>
<tr>
<td>d = 2</td>
<td>0.07</td>
<td>0.90</td>
<td>0.77</td>
<td>0.05</td>
<td>0.07</td>
</tr>
<tr>
<td>d = 3</td>
<td>0.11</td>
<td>0.91</td>
<td>1.32</td>
<td>0.26</td>
<td>0.64</td>
</tr>
<tr>
<td>d = 4</td>
<td>1.31</td>
<td>0.44</td>
<td>0.58</td>
<td>9.81</td>
<td>18.35</td>
</tr>
<tr>
<td>d = 5</td>
<td>1.06</td>
<td>0.38</td>
<td>1.24</td>
<td>19.73</td>
<td>15.66</td>
</tr>
</tbody>
</table>

Each of the mode shape correction approaches performed differently. As expected, randomly switching 20% of the designs (approach two) tended to increase the error when compared to doing nothing (approach one). Forcing the node with the largest average displacement to be positive (approach three) and forcing the dot product to be positive (approach four) performed similarly with high error only occurring in mode comparisons 4:4, 4:5, 5:4, and 5:5, with the for-
mer in general resulting in lower error. While approach three had the overall best performance, it still did not reduce the error low enough to be on par with the single-value surrogate tested by Thelin [28]. The single-value surrogate, or a surrogate that directly predicted the MAC value rather than calculate it from a predicted mode shape, was unaffected by the mode shape reversal identified here. The mode shapes were bypassed and never used in the creation of the surrogate, leading to more accurate predictions (Fig. 6 in Thelin [28]). Fig. 4.10 shows that the problems identified with mode shape surrogates still existed even when using approach three, the highest performing mode shape correction approach. A total of 2500 MAC values are plotted, with 25 mode comparisons for each of the 100 tested designs. If the surrogate were 100% accurate, then the true and predicted MAC values would follow a straight line diagonally through the plot from the bottom left to the top right. This, however, is not the case, as MAC predictions vary from the true values. For

![Figure 4.10: True (FEA) vs predicted (surrogate) MAC values using the RBF kernel KPCA in the five parameter design space with 100 training samples.](image-url)
example, a predicted MAC value of 0.13 has a true MAC value of 0.89. This point is found in the bottom right of Fig. 4.10 and is represented with the ▲ symbol. Deviations this large from the true MAC value were not present in the single-value surrogate. While using nonlinear dimensionality reduction reduced the error of the mode shape surrogate, the single-value surrogate still resulted in less variance from the true MAC values. Inconsistent mode shapes from design to design, as illustrated in Table 4.8, complicated the identification of reversed mode shapes, and it is believed that each of the tested approaches failed to correctly identify 100% of the affected designs. Thus, further improvement to the mode shape surrogate can be achieved by implementing an approach that fully corrects the reversed mode shapes.

4.5 Conclusion

Compared to linear dimensionality reduction, nonlinear dimensionality reduction has the potential to more accurately discover and preserve the intrinsic geometry of a nonlinear data set. As such, linear and nonlinear dimensionality reduction were explored in conjunction with surrogate modeling to predict the MAC values calculated from the modal analysis results of a jet engine compressor blade. Four different dimensionality reduction methods, including PCA, KPCA, ISOMAP, and LLE, were applied to the node displacement data sets of the first five modes. The results show that two nonlinear techniques, namely LLE and KPCA, resulted in lower surrogate error in the more complex five parameter design spaces. High NRMSE was only found in mode comparisons 4:4, 4:5, 5:4, and 5:5. Compared to the baseline PCA, statistical evidence suggested that LLE and KPCA using a radial basis function kernel reduced the mean NRMSE in predicting MAC values for these mode comparisons by a magnitude of 1.70% and 2.12%, respectively. No statistical differences existed among the dimensionality reduction methods in the three parameter design spaces.

Using the manifold distance ratio (MDR), a metric that measures the linearity of a manifold, mode four was shown to be more nonlinear in the five parameter design spaces than in the three parameter design spaces. This led to the aforementioned improved accuracy from nonlinear dimensionality reduction in the five parameter set. Mode five was not found to be more nonlinear in the five parameter design spaces, yet improvement using nonlinear dimensionality-reduction-based surrogate models was still achieved with mode five comparisons. The exact cause of this is
unknown but may be a result of the ISOMAP algorithm’s inability to accurately discover and pre-
serve the intrinsic geometry of mode five in a low-dimensional space, as evidenced by no statistical
difference between the NRMSE of ISOMAP and the baseline PCA.

The results also show that surrogate error was greatly affected by mode shape reversal. As
the mode shapes obtained from a modal analysis are eigenvectors, there is no guarantee that the
mode shapes from design to design will all be oriented in the same direction when solving a set of
designs in FEA. Four different approaches were explored to detect mode shape reversal, with each
approach resulting in different NRMSEs for each of the mode comparisons. The approach resulting
in the lowest overall NRMSE, which forced the node with the largest average displacement to be
positive for all designs, still experienced high error in predicting MAC values for various designs
throughout the design space. Additional research is needed in order to discover a more correct
approach for mode shape reversal detection. If all reversed mode shapes can be identified and
corrected, the high errors in mode comparisons 4:4, 4:5, 5:4, and 5:5 may be further reduced.
Nonlinear dimensionality-reduction-based surrogates can also be used in nonlinear design spaces
to further reduce the error, allowing engineers to have improved confidence in the results obtained
from design space exploration of modal data.
CHAPTER 5. CONCLUSIONS

While DSE is a useful tool for exploring engineering design possibilities, it suffers from challenges that, if not properly addressed, decrease its effectiveness. Many researchers have used surrogate modeling to speed up the design evaluation process and enable the use of DSE for real-time predictions of structural analysis results obtained from FEA \[18, 20, 28, 31\]. Both the computational cost of creating the surrogate as well as its predictive speed must be considered. Real-time predictions are only useful, however, when they can be trusted. Thus, speed and accuracy both play integral roles in surrogate usefulness. Combining surrogate modeling with dimensionality reduction has led to quicker and, in some cases, more accurate surrogate models \[4, 12, 29\]. This thesis addressed surrogate accuracy when conducting DSE by performing comparative analyses of dimensionality-reduction-based surrogate models for emulating various FEA nodal results in order to find the methods with the greatest reduction in surrogate error.

5.1 Research Contributions

Chapter 3 addressed the challenge of reducing surrogate error when performing DSE by conducting a comparative analysis of linear and nonlinear dimensionality-reduction-based surrogate models. A total of four dimensionality reduction methods were investigated, namely PCA, KPCA, ISOMAP, and LLE. Each of these methods were applied to the high-dimensional nodal stresses and coordinates of a compressor blade, the Transonic Purdue Research Compressor \[30\], to create low-dimensional representations of the nodal data. Dual-RBF surrogate models were created, consisting of a primary and secondary surrogate, to predict the full-nodal stresses and coordinates of unexplored designs within the design space. This was done in each of 200 different design spaces. The research showed that using an ISOMAP-based dual-RBF surrogate model for predicting nodal stresses decreased the error of the surrogate. The mean error using ISOMAP was estimated to be 35.7% lower than the mean error using PCA. Using nonlinear dimensionality-
reduction-based surrogates did not reduce surrogate error for predicting nodal coordinates. The stress space was found to be more nonlinear than the coordinate space for this application. Thus, reducing surrogate error using nonlinear dimensionality reduction was achieved in the stress space. The upfront training cost of the nonlinear dimensionality-reduction-based surrogates was larger than that of their linear counterparts but small enough (on the order of 10 minutes) to remain feasible. After training, all the dual-RBF surrogates were capable of making real-time predictions.

Chapter 3 contributed to the improvement of DSE of turbomachinery blades by demonstrating the ability of nonlinear dimensionality-reduction-based surrogate models to reduce surrogate error in emulating FEA nodal properties from a static analysis of a turbomachine compressor blade. Surrogate models are most useful when they can make quick and accurate predictions. More accurate surrogate models increase the confidence that engineers can have in the results obtained from DSE. Furthermore, Chapter 3 introduced a new metric, the manifold distance ratio (MDR), to measure the nonlinearity of the data manifolds. High-dimensional data can be difficult to visualize, making it difficult to determine when a data set is sufficiently nonlinear to require nonlinear dimensionality reduction. The newly introduced MDR provides an easily calculable metric to make this evaluation. Combined, these contributions enhance an engineer’s ability to perform DSE.

Chapter 4 further addressed the challenge of reducing surrogate error when performing DSE by conducting another comparative analysis of linear and nonlinear dimensionality-reduction-based surrogate models for a separate application. The same dimensionality reduction methods, namely PCA, KPCA, ISOMAP, and LLE, were applied to the high-dimensional nodal displacements of mode shapes obtained from a FEA modal analysis of the Transonic Purdue Research Compressor [30]. The dual-RBF surrogates created using the low-dimensional representation of the data were used to predict the mode shapes of unexplored designs within the design space. The MAC calculation was then used to compare the predicted mode shapes, as well as their corresponding true mode shapes obtained from FEA, to a set of reference modes. This was done for a set of three and five parameter design spaces. The research showed that high surrogate error was only found in mode comparisons 4:4, 4:5, 5:4, and 5:5 and that two nonlinear techniques, namely LLE and KPCA, resulted in lower surrogate error in the more complex five parameter design spaces. Using a RBF kernel, KPCA achieved the largest average reduction in error of 13.57%. The results also showed that surrogate error was greatly affected by mode shape reversal. As the mode shapes
obtained from a modal analysis are eigenvectors, there is no guarantee that the mode shapes from
every design in a DOE will all be oriented in the same direction when solved in FEA. Four differ-
ent approaches of identifying reversed mode shapes were explored, all of which resulted in varying
amounts of surrogate error.

Chapter 4 contributed to the improvement of DSE of turbomachinery blades by demonstrat-
ing the ability of nonlinear dimensionality-reduction-based surrogate models to reduce surrogate
error in emulating FEA nodal properties from a dynamic analysis of a turbomachine compressor
blade. These more accurate surrogate models increase an engineer’s confidence in the results ob-
tained during DSE. Also, Chapter 4 discovered that, when emulating mode shapes, mode shape
reversal was a real phenomenon that had a great effect on surrogate accuracy. While nonlinear di-
mensionality reduction increased surrogate accuracy, errors may be further reduced by addressing
the reversed mode shape issue discovered and identified herein.

Together, the methods explored in this research were shown to decrease surrogate error
when performing DSE of a turbomachine compressor blade. This was done for the results of both
static and dynamic FEA analyses, demonstrating that dimensionality-reduction-based surrogate
models can be applied to a variety of engineering problems and domains. The new discoveries
in this research enhance an engineer’s ability to perform DSE. As surrogate accuracy increases,
so does the ability to correctly make engineering decisions and judgements throughout the design
process. Ultimately, this will help engineers design better turbomachines.

5.2 Recommended Work

This section outlines recommended work to be done in future studies in order to further
enhance the understanding of dimensionality-reduction-based surrogate models.

5.2.1 Exploration of More Dimensionality Reduction Methods

The four dimensionality reduction methods used in this research were selected based on
their frequent utilization in the literature. While only four methods were investigated, more tech-
niques exist. These include other linear methods (multidimensional scaling, linear discriminant
analysis, canonical correlations analysis, etc.) [51, 52] and other nonlinear methods (Laplacian
eigenmaps, local tangent space alignment, maximum variance unfolding, etc.) [52, 53]. Expanding the comparative analyses to include more dimensionality reduction techniques could further increase the surrogate error reduction achieved in this research.

5.2.2 Optimization of Hyperparameters

It is well understood that the dimensionality reduction algorithms are sensitive to the values of their respective hyperparameters [50]. In this research, a DOE was used to explore a range of possible hyperparameter configurations in order to find the configuration that resulted in the lowest predictive error. This was an effective method but could be improved by using an optimization routine rather than a DOE. Many optimization algorithms exist and would need to be explored to find those that are most effective at determining optimal hyperparameter configurations. This could improve the performance of the dimensionality reduction methods and reduce their required training time.

5.2.3 Identification of Mode Shape Reversal

This research discovered that mode shape reversal greatly affected surrogate accuracy when predicting mode shapes. Correctly identifying reversed mode shapes, however, was proven to be difficult, particularly in the presence of mode switching. The identification method resulting in the lowest overall error, which forced the node with the largest average displacement to be positive for all designs, still experienced high error in predicting MAC values for various designs throughout the design space. It is believed that this method, along with the others that were tested, failed to correctly identify and correct all of the reversed designs. Further research is needed in order to discover a more correct method for mode shape reversal detection. If all reversed mode shapes can be identified and corrected, the high errors in mode comparisons 4:4, 4:5, 5:4, and 5:5 may be reduced.

5.2.4 Application Beyond FEA

Further comparative analyses of dimensionality-reduction-based surrogate models involving other sets and types of data are recommended. There is no guarantee that a nonlinear
A dimensionality-reduction-based surrogate model will reduce surrogate error for every application. Ultimately, nonlinear dimensionality-reduction-based surrogate models have the potential to increase surrogate accuracy when applied to sufficiently nonlinear data sets. While the results herein pertain to FEA nodal stresses, coordinates, and displacements, the methods used can be applied to any data set. This could include additional FEA results, such as nodal strains, or even results from other applications, including computational fluid dynamics (CFD) or heat transfer. The dual-RBF surrogate models are agnostic to the type or source of data. It is expected, however, for results to vary when these methods are applied to other applications. The amount of error reduction and the best performing dimensionality reduction methods will likely change from application to application.

5.2.5 DSE of Other Parts

Likewise, further comparative analyses of dimensionality-reduction-based surrogate models involving other geometries are recommended. The methods explored herein can be applied to the FEA results of any geometry, so long as the geometry can be successfully solved in FEA. Successfully applying these methods to DSE of other geometries would validate the results obtained in this research.
REFERENCES


APPENDIX A. IMPORTANT CODE

This appendix presents the Python code necessary to train the dual-RBF surrogates for each of the dimensionality reduction techniques discussed in this research. All packages used, with the exception of “rbf”, are standard Python packages. The code for the “rbf” package was originally created by Bunnell [4].

A.1 Linear Dimensionality-Reduction-Based Dual-RBF Surrogate Training

```python
import numpy as np
import warnings
import random

import realtime_emulation.emulation.rbf as rbf
from sklearn.metrics import mean_squared_error
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split as tts

warnings.filterwarnings('ignore')

class LinearReductionEmulator(object):
    
    def __init__(self, x, y, **kwargs):
        
        LinearReductionEmulator
```
Creating an instance trains the model.

Parameters
-----------

\(x\) : np.ndarray

Labels, or design variable values, which correspond to \(y\).
Shape = [number of samples, number of variables]

\(y\) : np.ndarray

Samples of the features to emulate. Shape = [number of samples, number of features].

**kwargs:

\(\epsilon\) : str, float

The epsilon or sigma which controls the width of the kernel function. If a string then it should be the name of the function used to calculate the value. This function should be in the kernel module. If the value is a float then that value is used.

\(\text{kernel}\) : str

The name of the kernel function, in the kernel module, to use with the radial basis function.

\(\text{norm}\) : str

The name of the norm function, in the norm module, to use with the radial basis function.

\(\text{smoothing}\) : float, np.ndarray

The smoothing factor for the radial basis function. If the value is a float then the smoothing is the same for all samples. If the value is an np.ndarray then it must be of shape [Samples, ]. Each item in the array is the smoothing value for the corresponding sample in \(x\) and \(y\).

\(\text{method}\) : str
The type of linear reduction technique which should be used.

```python
self._x = x

self.method = kwargs.pop('method', 'pca')
self._score_surrogate = None
self._feature_surrogate = None
self.emulated_pc_scores = None
self._num_comp = None

self.reduction_methods = {'pca': {'num_hyper': 1,
                                 'model': PCA}}

self.train(y, **kwargs)
```

def __call__(self, x):
    
    Emulate or predict the model

    Parameters
    ----------

    x : np.ndarray
        Labels, or design variable values, at which to emulate or predict the trained features.
        Shape = [number of evaluations, number of variables] if multiple evaluations are desired.
        Shape = [number of variables, ] if only one evaluation is desired.
Return
--------

np.ndarray: emulated or predicted features.

Shape = [number of evaluations, number of features] if multiple evaluations are supplied.
Shape = [number of features, ] if only one evaluation is supplied.

```
self.emulated_pc_scores = self._score_surrogate(x)
return self._feature_surrogate(self.emulated_pc_scores)
```

@property
def pc_scores(self):
    return self.emulated_pc_scores

@property
def num_comp(self):
    return self._num_comp

def train(self, y, **kwargs):
    ""

    ""

    num_hyperparameters = self.reduction_methods[self.method]['num_hyper']
    train_x, test_x, train_y, test_y = tts(self._x, y, test_size=0.25)

    best_hyperparameter = []
    best_error = 100

87
hyper_doe = self.hyperparameter_doe()

for hyper_sample in hyper_doe:

    try:
        reduction_model = self.reduction_methods[self.method]['model'](*hyper_sample)
        scores = reduction_model.fit_transform(train_y)
        self._score_surrogate = rbf.Rbf(train_x, scores, **kwargs)
        self._feature_surrogate = rbf.Rbf(scores, train_y, **kwargs)
        estimate_y = self(test_x)
        error = np.sqrt(np.mean((estimate_y - test_y)**2)) / np.max(test_y)

        if error < best_error:
            best_error = error
            best_hyperparameter = hyper_sample

    except ValueError:
        pass
    except np.linalg.LinAlgError:
        pass
    except ZeroDivisionError:
        pass
reduction_model = self.reduction_methods[self.method]['model']
    (*best_hyperparameter)

scores = reduction_model.fit_transform(y)
self._score_surrogate = rbf.Rbf(self._x, scores, **kwargs)
self._feature_surrogate = rbf.Rbf(scores, y, **kwargs)
self._num_comp = scores.shape[1]

def hyperparameter_doe(self, num_designs=5000):
    # The code shown here does not use a DOE generator

    samples = self._x.shape[0]
    combos = []

    if self.method == 'pca':
        n_comps_list = range(1, samples+1)

        for n_comp in n_comps_list:
            combos.append([n_comp])

        if len(combos) > num_designs:
            random.shuffle(combos)
            combos = combos[:num_designs]

    return combos

A.2 Nonlinear Dimensionality-Reduction-Based Dual-RBF Surrogate Training

import numpy as np
import warnings
import random

import realtime_emulation.emulation.rbf as rbf
from sklearn.metrics import mean_squared_error
from sklearn.decomposition import KernelPCA as kPCA
from sklearn.manifold import LocallyLinearEmbedding as LLE
from sklearn.manifold import Isomap
from sklearn.model_selection import train_test_split as tts

warnings.filterwarnings('ignore')

class NonlinearReductionEmulator(object):
    def __init__(self, x, y, **kwargs):
        
        """
        NonlinearReductionEmulator

        Creating an instance trains the model.

        Parameters
        ----------
        x : np.ndarray
            Labels, or design variable values, which correspond to y.
            Shape = [number of samples, number of variables]
        y : np.ndarray
            Samples of the features to emulate. Shape = [number of
            samples, number of features].
        **kwargs:
        epsilon : str, float
The epsilon or sigma which controls the width of the kernel function. If a string then it should be the name of the function used to calculate the value. This function should be in the kernel module. If the value is a float then that value is used.

`kernel : str`

The name of the kernel function, in the kernel module, to use with the radial basis function.

`kernel_kpca : str`

The name of the kernel function to be used by kernel principal component analysis. Default is None.

`norm : str`

The name of the norm function, in the norm module, to use with the radial basis function.

`smeoothing : float, np.ndarry`

The smoothing factor for the radial basis function. If the value is a float then the smoothing is the same for all samples. If the value is an np.ndarray then it must be of shape [Samples, ]. Each item in the array is the smoothing value for the corresponding sample in x and y.

`method : str`

The type of nonlinear reduction technique which should be used.

```
self._x = x

self.method = kwargs.pop('method', 'kpca')
self._score_surrogate = None
self._feature_surrogate = None
self.emulated_pc_scores = None
```
self._num_comp = None

self.reduction_methods = {'isomap': {'num_hyper': 3,
    'model': Isomap},
    'lle': {'num_hyper': 4,
    'model': LLE},
    'kpca': {'num_hyper': 6,
    'model': kPCA}}

self.train(y, **kwargs)

def __call__(self, x):
    
    """
    Emulate or predict the model
    
    Parameters
    ----------
    x : np.ndarray
        Labels, or design variable values, at which to emulate or predict the trained features.
        Shape = [number of evaluations, number of variables] if multiple evaluations are desired.
        Shape = [number of variables, ] if only one evaluation is desired.
    
    Return
    -------
    np.ndarray : emulated or predicted features.
        Shape = [number of evaluations, number of features] if multiple evaluations are supplied.
    """

92
Shape = [number of features, ] if only one evaluation is supplied.

```python
self.emulated_pc_scores = self._score_surrogate(x)
return self._feature_surrogate(self.emulated_pc_scores)
```

@property
def pc_scores(self):
    return self.emulated_pc_scores

@property
def num_comp(self):
    return self._num_comp

def train(self, y, **kwargs):
    ```
    num_hyperparameters = self.reduction_methods[self.method]['num_hyper']
    train_x, test_x, train_y, test_y = tts(self._x, y, test_size=0.25)
    best_hyperparameter = []
    best_error = 100

    hyper_doe = self.hyperparameter_doe(kernel=kernel_kpca)

    for hyper_sample in hyper_doe:
```

93
try:
    reduction_model = self.reduction_methods[self.method]['model'](*hyper_sample,
        eigen_solver='dense')
    scores = reduction_model.fit_transform(train_y)
    self._score_surrogate = rbf.Rbf(train_x, scores,
        **kwargs)
    self._feature_surrogate = rbf.Rbf(scores, train_y,
        **kwargs)

    estimate_y = self(test_x)
    error = np.sqrt(np.mean((estimate_y - test_y)**2)) / np.max(test_y)

    if error < best_error:
        best_error = error
        best_hyperparameter = hyper_sample

except ValueError:
    pass
except np.linalg.LinAlgError:
    pass
except ZeroDivisionError:
    pass

reduction_model = self.reduction_methods[self.method]['model'](*best_hyperparameter, eigen_solver='dense')
scores = reduction_model.fit_transform(y)
self._score_surrogate = rbf.Rbf(self._x, scores, **kwargs)
self._feature_surrogate = rbf.Rbf(scores, y, **kwargs)
self._num_comp = scores.shape[1]

def hyperparameter_doe(self, kernel='poly', num_designs=5000):
    # The code shown here does not use a DOE generator

    samples = self._x.shape[0]
    combos = []

    if self.method == 'isomap':
        neighbors_list = range(1, samples)
        n_comps_list = range(1, samples+1)

        for neighbor in neighbors_list:
            for n_comp in n_comps_list:
                combos.append([neighbor, n_comp])

        if len(combos) > num_designs:
            random.shuffle(combos)
            combos = combos[:num_designs]

    elif self.method == 'lle':
        neighbors_list = range(1, samples-1)
        n_comps_list = range(1, samples+1)
        reg_list = np.linspace(5e-4, .25, 100)

        for neighbor in neighbors_list:
            for n_comp in n_comps_list:
                for reg in reg_list:
                    combos.append([neighbor, n_comp, reg])
for reg in reg_list:
    combos.append([neighbor, n_comp, reg])

if len(combos) > num_designs:
    random.shuffle(combos)
    combos = combos[:num_designs]

elif self.method == 'kpca':

    n_comps_list = range(1, samples+1)
    gamma_list = np.linspace(0, 0.001, 100)

    if kernel == 'rbf':
        for n_comp in n_comps_list:
            for gamma in gamma_list:
                combos.append([n_comp, 'rbf', gamma, 1, 0])

    if len(combos) > num_designs:
        random.shuffle(combos)
        combos = combos[:num_designs]

elif kernel == 'poly':
    degree_list = range(1, 5)
    coefficient_list = range(1, 1000)

    for n_comp in n_comps_list:
        for gamma in gamma_list:
            for degree in degree_list:
                for coeff in coefficient_list:
                    combos.append([n_comp, 'poly', gamma,
if len(combos) > num_designs:
    random.shuffle(combos)
    combos = combos[:num_designs]

return combos