Real Time Design Space Exploration of Static and Vibratory Structural Responses in Turbomachinery Through Surrogate Modeling with Principal Components

Spencer Reese Bunnell
Brigham Young University

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

Part of the Engineering Commons

BYU ScholarsArchive Citation
Bunnell, Spencer Reese, "Real Time Design Space Exploration of Static and Vibratory Structural Responses in Turbomachinery Through Surrogate Modeling with Principal Components" (2020). Theses and Dissertations. 8451.
https://scholarsarchive.byu.edu/etd/8451

This Dissertation 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 scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Real-Time Design Space Exploration of Static and Vibratory Structural Responses in Turbomachinery Through Surrogate Modeling with Principal Components

Spencer Reese Bunnell

A dissertation submitted to the faculty of Brigham Young University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Steven E. Gorrell, Chair
Matthew R. Jones
S. Andrew Ning
John L. Salmon
Scott Thomson

Department of Mechanical Engineering
Brigham Young University

Copyright © 2020 Spencer Reese Bunnell
All Rights Reserved
ABSTRACT

Real-Time Design Space Exploration of Static and Vibratory Structural Responses in Turbomachinery Through Surrogate Modeling with Principal Components

Spencer Reese Bunnell
Department of Mechanical Engineering, BYU
Doctor of Philosophy

Design space exploration (DSE) is used to improve and understand engineering designs. Such designs must meet objectives and structural requirements. Design improvement is non-trivial and requires new DSE methods. Turbomachinery manufacturers must continue to improve existing engines to keep up with global demand. Two challenges of turbomachinery DSE are: the time required to evaluate designs, and knowing which designs to evaluate. This research addressed these challenges by developing novel surrogate and principal component analysis (PCA) based DSE methods. Node and PCA-based surrogates were created to allow faster DSE of turbomachinery blades. The surrogates provided static stress estimation within 10% error. Surrogate error was related to the number of sampled finite element (FE) models used to train the surrogate and the variables used to change the designs. Surrogates were able to provide structural evaluations three to five orders of magnitude faster than FEA evaluations. The PCA-based surrogates were then used to create a PCA-based design workflow to help designers know which designs to evaluate. The workflow used either two-point correlation or stress and geometry coupling to relate the design variables to principal component (PC) scores. These scores were projections of the FE models onto the PCs obtained from PCA. Analysis showed that this workflow could be used in DSE to better explore and improve designs. The surrogate methods were then applied to vibratory stress. A computationally simplified analysis workflow was developed to allow for enough fluid and structural analyses to create a surrogate model. The simplified analysis workflow introduced 10% error but decreased the computational cost by 90%. The surrogate methods could not directly be applied to emulation of vibration due to the large spikes which occur near resonance. A novel, indirect emulation method was developed to better estimate vibratory responses. Surrogates were used to estimate the inputs to calculate the vibratory responses. During DSE these estimations were used to calculate the vibratory responses. This method reduced the error between the surrogate and FEA from 85% to 17%. Lastly, a PCA-based multi-fidelity surrogate method was developed. This assumed the PCs of the high and low-fidelities were similar. The high-fidelity FE models had tens of thousands of nodes and the low-fidelity FE models had a few hundred nodes. The computational cost to create the surrogate was decreased by 75% for the same errors. For the same computational cost, the error was reduced by 50%. Together, the methods developed in this research were shown to decrease the cost of evaluating the structural responses of turbomachinery blade designs. They also provided a method to help the designer understand which designs to explore. This research paves the way for better, and more thoroughly understood turbomachinery blade designs.

Keywords: computational fluid dynamics, design space exploration, finite element analysis, multi-fidelity surrogates, principal component analysis, real-time emulation, stress and geometry coupling, surrogates, turbomachinery, two-point correlation, vibratory analysis
ACKNOWLEDGMENTS

First, I thank my graduate advisor, Dr. Gorrell, for his guidance during my education. I am grateful for the opportunity to have studied turbomachinery design and methods under his tutelage. I thank him for the patience shown during the many iterations which were often required to bring out the full potential of the published research.

Next, I am grateful to Brigham Young University for the opportunity to pursue both of my degrees and for my first engineering job. I am grateful for those at Brigham Young University who helped with my research. I thank those on my committee: Dr. Salmon, Dr. Jones, Dr. Thomas, and Dr. Ning. I thank them for their time and effort spent on my behalf. I especially thank Dr. Salmon for being weekly involved with the research. I am grateful to the department of Mechanical Engineering for allowing me to be a teaching and research assistant. I thank my fellow students who spent time reviewing my work. They provided invaluable insight to code, CAD, CFD, turbomachinery design, and technical writing.

I am also grateful to Pratt & Whitney for funding this research and for the opportunity to work as an on-site intern. I express gratitude to Kurt Heinemann, Chris Ruoti, Evan Selin, Clint Collins, Ammon Hepworth, Joseph Calogero, and other engineers at Pratt & Whitney who helped with the research. They provided invaluable guidance into turbomachinery structural design methods. Their contribution and mentorship accelerated my learning and understanding of engine design and design methods.

Finally, I thank my family who encouraged me along my educational journey. I thank my mother for teaching me to study and never give up when things are difficult. I thank my father for engendering a love for engineering and suggesting that I study mechanical engineering. Lastly, I thank my wife, Camille Bunnell, for her unwavering support during my education. I also thank her for bringing joy and encouragement as we navigate the next chapters of life.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE PAGE</td>
<td>i</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>NOMENCLATURE</td>
<td>xi</td>
</tr>
<tr>
<td>Chapter 1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Design Space Exploration</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Turbomachinery Blade Design Process</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Challenges of Design Space Exploration with Turbomachinery</td>
<td>5</td>
</tr>
<tr>
<td>1.4 Proposed Research</td>
<td>6</td>
</tr>
<tr>
<td>Chapter 2 Background</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Compressor Blade Structural Analysis</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Compressor Blade Fluid Analysis</td>
<td>14</td>
</tr>
<tr>
<td>2.3 Surrogate Modeling</td>
<td>15</td>
</tr>
<tr>
<td>2.3.1 Surrogates of FEA</td>
<td>17</td>
</tr>
<tr>
<td>2.4 Principle Component Analysis</td>
<td>18</td>
</tr>
<tr>
<td>2.4.1 PCA for DSE of Result Fields</td>
<td>20</td>
</tr>
<tr>
<td>2.5 Multi-Fidelity Modeling</td>
<td>21</td>
</tr>
<tr>
<td>2.5.1 Types of Fidelities</td>
<td>21</td>
</tr>
<tr>
<td>2.5.2 Types of Multi-Fidelity Surrogate Models</td>
<td>22</td>
</tr>
<tr>
<td>2.6 Turbomachinery Blade Models</td>
<td>23</td>
</tr>
<tr>
<td>Chapter 3 Real-Time Design Space Exploration</td>
<td>26</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Background</td>
<td>28</td>
</tr>
<tr>
<td>3.2.1 Finite Element Analysis</td>
<td>28</td>
</tr>
<tr>
<td>3.2.2 Principal Component Analysis</td>
<td>29</td>
</tr>
<tr>
<td>3.2.3 Design Space Exploration</td>
<td>30</td>
</tr>
<tr>
<td>3.3 Method</td>
<td>31</td>
</tr>
<tr>
<td>3.3.1 Mesh Morphing</td>
<td>32</td>
</tr>
<tr>
<td>3.3.2 Surrogate Model</td>
<td>32</td>
</tr>
<tr>
<td>3.3.3 Training Data Workflow</td>
<td>35</td>
</tr>
<tr>
<td>3.3.4 Surrogate Creation and Visualization</td>
<td>37</td>
</tr>
<tr>
<td>3.4 Results</td>
<td>37</td>
</tr>
<tr>
<td>3.4.1 Mesh Morphing</td>
<td>42</td>
</tr>
<tr>
<td>3.4.2 Surrogate Speed</td>
<td>43</td>
</tr>
</tbody>
</table>
3.4.3 Surrogate Error ................................. 49
3.4.4 PCA-based Error and Speed Trade-off .................. 52
3.5 Discussion ........................................ 53
  3.5.1 Improvement of Speed .............................. 54
  3.5.2 Achieving Accurate Surrogates ....................... 55
3.6 Conclusions ....................................... 56

Chapter 4  Design Space Exploration Method Based on PCA ........ 57
  4.1 Introduction ..................................... 57
  4.2 Background ..................................... 59
    4.2.1 Design Exploration ............................ 59
    4.2.2 Principal Component Analysis ................. 60
  4.3 Method .......................................... 62
    4.3.1 Principal Component Analysis ................... 63
    4.3.2 Two-Point Correlation ......................... 65
  4.4 Results ......................................... 65
    4.4.1 Low Dimensional Relationships ................. 67
    4.4.2 High Dimensional Relationships ................. 69
    4.4.3 Two-Point Correlation .......................... 71
    4.4.4 Stress and Geometry Coupling ................. 72
  4.5 Discussion ...................................... 74
    4.5.1 Process ..................................... 74
    4.5.2 Example ..................................... 75
    4.5.3 Real-Time Design Exploration ................. 78
  4.6 Conclusions ..................................... 79

Chapter 5  Design Space Exploration of Vibratory Results ........ 82
  5.1 Introduction ..................................... 82
  5.2 Background ..................................... 86
    5.2.1 Computational Analysis ......................... 87
    5.2.2 Computational Simplifications ................. 88
  5.3 Method .......................................... 89
    5.3.1 Surrogate Models .............................. 92
  5.4 Results ......................................... 95
    5.4.1 RANS-HMS Methodology ......................... 95
    5.4.2 Surrogate Models .............................. 100
  5.5 Discussion ...................................... 106
    5.5.1 DSE Example ................................ 106
  5.6 Conclusion ...................................... 108

Chapter 6  Multi-Fidelity Surrogates Based on Shared Principal Components 110
  6.1 Introduction ..................................... 110
  6.2 Background ..................................... 113
    6.2.1 Principal Component Analysis .................. 114
  6.3 Methodology .................................... 117
LIST OF TABLES

2.1 Blade vibration analysis workflow ........................................... 10
2.2 Purdue blade mode names and frequencies ................................. 12
2.3 Purdue blade design variables ................................................. 24
2.4 PBS Rotor 4 design variables ................................................... 25
3.1 Surrogate evaluation time ....................................................... 47
4.1 Purdue blade design variables ................................................... 61
5.1 PBS Rotor 4 design variables ................................................... 85
5.2 Error of surrogate models used to calculate vibratory response ....... 102
5.3 Error of the emulated vibratory responses .................................... 104
6.1 Purdue blade design variables ................................................... 112
6.2 Error of current multi-fidelity methods ....................................... 123
6.3 Error of SPC multi-fidelity methods .......................................... 124
6.4 Surrogate error lowered by more than 50% ................................. 126
6.5 Computational cost of sample fidelities ..................................... 127
6.6 Computational cost for fixed error ............................................. 128
6.7 Error for fixed computational cost ............................................. 128
6.8 Error for SPC method with PBS Rotor 4 .................................... 131
LIST OF FIGURES

1.1 DSE Example ................................................. 2
1.2 Turbomachinery blade design process .......................... 5

2.1 Purdue blade coarse mesh .................................. 9
2.2 Purdue blade FEA example .................................. 9
2.3 Purdue blade modes ....................................... 12
2.4 PBS Rotor 4 single blade passage ........................... 14
2.5 RBF surrogate and kernel example ......................... 16
2.6 Surrogates for FEA ........................................ 18
2.7 Cantilever beam finite element model ...................... 18
2.8 PCA example ............................................... 19
2.9 Multi-fidelity samples ...................................... 22
2.10 Purdue blade model ....................................... 23
2.11 PBS Rotor 4 models ...................................... 25

3.1 Purdue blade section splines ................................ 31
3.2 Data matrix ............................................... 34
3.3 PC score matrix .......................................... 35
3.4 Percent of non-surface nodes .............................. 36
3.5 Training data workflow ................................... 36
3.6 Graphical user interface .................................... 38
3.7 Von Mises stress of the Purdue blade suction side ...... 39
3.8 Von Mises stress from sweep of Purdue blade pressure side ................................................. 41
3.9 Mesh congruency during morphing process ............... 42
3.10 Time to solve FEA ......................................... 45
3.11 Time to evaluate surrogates ............................... 46
3.12 Interactive FEA evaluation time ......................... 48
3.13 Interactive surrogate evaluation time ................... 49
3.14 Error testing workflow ................................... 50
3.15 Error of surrogates ....................................... 51
3.16 Error versus samples ..................................... 52
3.17 Error versus number of PCs ............................... 53

4.1 Flow chart for PCA of finite element models ............. 63
4.2 Data matrix ............................................... 64
4.3 PC score matrix .......................................... 65
4.4 PCs for Purdue blade ..................................... 66
4.5 Scree plot ................................................ 67
4.6 PC scores of design space of $C_R$ ......................... 68
4.7 PCs of design space of $C_R$ .............................. 69
4.8 PC scores for design space of $C_T$ and $\alpha$ ............... 70
4.9 Correlation versus PC scores .............................. 71
4.10 Correlation versus design variables ...................... 72
4.11 Geometric coupling of PCs ........................................... 73
4.12 PCA-based DSE workflow ............................................. 75
4.13 PCA-based DSE example ............................................... 76
4.14 PC scores versus $L$ .................................................... 77
4.15 PC scores versus $C_R$ .................................................. 79
4.16 PC scores versus $N$ .................................................... 80

5.1 PBS Rotor 4 geometry ................................................... 85
5.2 PBS Rotor 4 blade row ................................................. 85
5.3 RANS-HMS workflow .................................................. 90
5.4 Surrogate creation and evaluation workflows ......................... 92
5.5 Data matrix .............................................................. 94
5.6 Nominal PBS Rotor 4 resonance diagram .............................. 96
5.7 Performance map of PBS Rotor 4. .................................... 97
5.8 Pressure distribution on PBS Rotor 4 .................................. 98
5.9 Nominal PBS Rotor 4 vibratory response ............................. 99
5.10 Computation and error of various analysis methods ............... 100
5.11 Maximum vibratory stress across $C_M$ and $C_{75}$ ................. 103
5.12 Maximum modified Goodman across $C_M$ and $C_{75}$ .............. 104
5.13 Maximum vibratory stress over $C_M$ ................................ 105
5.14 Modified Goodman values for the design iterations. ............... 107

6.1 Purdue blade disk geometry isometric view ......................... 113
6.2 Purdue blade geometry ................................................. 113
6.3 PCA example ............................................................ 115
6.4 Common grid multi-fidelity workflow ................................ 116
6.5 SPC surrogate creation workflow .................................... 117
6.6 SPC method data matrix ................................................. 118
6.7 SPC method PC score matrix .......................................... 118
6.8 Multi-fidelity PC comparison ......................................... 119
6.9 SPC surrogate evaluation workflow .................................. 120
6.10 Multi-fidelity surrogate error versus number of low-fidelity samples 123
6.11 Multi-fidelity surrogate error versus high-fidelity surrogate error 125
6.12 Surrogate error versus computational cost ........................... 129

B.1 Turbomachinery airfoil cross-sections ............................... 216
B.2 CAD expressions ........................................................ 217
B.3 CAD model with split bodies .......................................... 217
B.4 Purdue coarse mesh .................................................... 218
B.5 CAD model of rotor fluid domain ..................................... 221
B.6 CFD model of rotor fluid domain ...................................... 222
B.7 Convergence of PBS Rotor 4 RANS Simulation ..................... 223
B.8 Residuals for PBS Rotor 4 RANS simulation ......................... 224
B.9 PBS Rotor 4 performance map ....................................... 225
B.10 Static pressure along hub and shroud ............................... 226
B.11 PBS Rotor 4 inlet total pressure and total temperature . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 227
B.12 PBS Rotor 4 outlet total pressure and total temperature . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 228
### Simulation Modeling

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFD</td>
<td>Computational Fluid Dynamics</td>
</tr>
<tr>
<td>D</td>
<td>Damping matrix</td>
</tr>
<tr>
<td>EO</td>
<td>Engine order</td>
</tr>
<tr>
<td>f</td>
<td>Loads vector</td>
</tr>
<tr>
<td>FE</td>
<td>Finite Element</td>
</tr>
<tr>
<td>FEA</td>
<td>Finite Element Analysis</td>
</tr>
<tr>
<td>HMS</td>
<td>Harmonic Mode Superposition</td>
</tr>
<tr>
<td>K</td>
<td>Stiffness matrix</td>
</tr>
<tr>
<td>n</td>
<td>Number of nodes in the FE model</td>
</tr>
<tr>
<td>M</td>
<td>Mass matrix</td>
</tr>
<tr>
<td>RANS</td>
<td>Reynolds Averaged Navier-Stokes</td>
</tr>
<tr>
<td>σ</td>
<td>Stress</td>
</tr>
<tr>
<td>σₘₐₜₜ</td>
<td>Alternating, or vibratory stress</td>
</tr>
<tr>
<td>σₜₚₜₜ</td>
<td>Steady stress</td>
</tr>
<tr>
<td>Sₑ</td>
<td>Endurance strength</td>
</tr>
<tr>
<td>Sₜₜₜₜ</td>
<td>Ultimate strength</td>
</tr>
<tr>
<td>u</td>
<td>Displacements vector</td>
</tr>
<tr>
<td>φᵣ</td>
<td>Mass normalized displacement of mode r</td>
</tr>
<tr>
<td>ωₖ</td>
<td>Forcing load frequency</td>
</tr>
<tr>
<td>ωᵣ</td>
<td>Mode r frequency</td>
</tr>
<tr>
<td>URANS</td>
<td>Unsteady Reynolds Averaged Navier-Stokes</td>
</tr>
</tbody>
</table>

### Surrogate Modeling

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>PC score vector</td>
</tr>
<tr>
<td>A</td>
<td>PC score matrix</td>
</tr>
<tr>
<td>c</td>
<td>Design variable sample matrix</td>
</tr>
<tr>
<td>C</td>
<td>Correlation score matrix</td>
</tr>
<tr>
<td>d</td>
<td>Number of variables in the design space.</td>
</tr>
<tr>
<td>d</td>
<td>Design vector</td>
</tr>
<tr>
<td>DOE</td>
<td>Design of experiments</td>
</tr>
<tr>
<td>DSE</td>
<td>Design space exploration</td>
</tr>
<tr>
<td>ε</td>
<td>Kernel shape parameter</td>
</tr>
<tr>
<td>e</td>
<td>Error</td>
</tr>
<tr>
<td>λ</td>
<td>Vector of kernel evaluations</td>
</tr>
<tr>
<td>n</td>
<td>Number of surface nodes</td>
</tr>
<tr>
<td>ψ</td>
<td>Kernel function</td>
</tr>
<tr>
<td>PC</td>
<td>Principal Component</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>q</td>
<td>Number of retained PCs</td>
</tr>
<tr>
<td>s</td>
<td>Number of samples used to train the surrogates</td>
</tr>
<tr>
<td>t</td>
<td>Time (seconds)</td>
</tr>
<tr>
<td>ˆσ</td>
<td>Estimated stress</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>$S$</td>
<td>Diagonal matrix from SVD used to calculate $A$</td>
</tr>
<tr>
<td>$SVD$</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>$U$</td>
<td>Matrix from SVD used to calculate $A$</td>
</tr>
<tr>
<td>$V$</td>
<td>Matrix of PCs</td>
</tr>
<tr>
<td>$W$</td>
<td>Kernel weights matrix</td>
</tr>
</tbody>
</table>
CHAPTER 1. INTRODUCTION

Design space exploration is a technique used to improve and understand engineering designs. Such designs must meet their design objectives while keeping within structural limitations and requirements. Improving designs and meeting these structural limitations is often non-trivial. The development of new design space exploration methods and tools can open the path for better designs and understanding of those designs. The need for improved design space exploration methods and tools were addressed with this research. This was accomplished by developing and evaluating novel surrogate and principal component analysis (PCA) based methods for design space exploration. The surrogates were used to create computationally cheaper estimates of structural responses. PCA was used to reduce the computational cost of using surrogates based on finite element models. PCA reduced the description of the finite element model from a large set of nodes to a smaller set of principal components. PCA was also used to discover underlying features which were not readily apparent in the finite element analysis samples. These methods were applied to turbomachinery which have become an integral part of day-to-day operations in our society. Turbomachines power aircraft which provide both travel and national security. They also power cities by providing an efficient way to generate electricity. To stay competitive and keep up with global demand, engine manufacturers must continue to innovate and improve existing turbomachinery designs [1]. Applying the methods developed in this research to turbomachinery design will be shown in this dissertation to improve DSE for turbomachinery. This allows for development of better engines to meet global demand.

1.1 Design Space Exploration

Design space exploration (DSE) is the exploration of a model’s response through a space spanned by a set of design variables. Optimization is an automated form of DSE which focuses on the search for the best designs according to some set of constraints and some maximum or min-
imum objective functions. Interactive DSE is a method by which a designer seeks to understand the effect of the design variables on the model’s response by manually and iteratively evaluating designs. This process involves selecting a proposed design, evaluating the response, visually inspecting the results, and then using what was learned from those results to update the proposed design. Interactive DSE can provide powerful insight to the relationship between the design variables and the response, as well as provide understanding concerning the set of possible responses which exist within the design space. An example of interactive DSE is given with Fig. 1.1 which shows compressor blades of different designs with the von Mises stress contoured as color. Figure 1.1(a) shows the nominal design of a compressor blade for which the chord at the root was 1.79 inches (4.55 cm). This design was solved with finite element analysis and found to have a maximum von Mises stress of 40 ksi. High stress is seen as red points at the root leading and trailing edges, or the bottom corners in Fig. 1.1(a). The objective of this example was to lower the maximum stress below 38 ksi. It was suspected, from design experience, that increasing the chord at the root would lower the stress at the root. The design was updated by increasing the chord at the root to 1.9 inches (4.83 cm). This new design was solved with finite element analysis and found to have a maximum von Mises stress of 37 ksi. The lower stress and change in geometry are shown in Fig. 1.1(b) where the stress in the bottom corners is lower and the bottom of the blade is wider.

![Figure 1.1: DSE example of compressor blade. The color represents the von Mises stress. Values above 38 ksi are red. Markers underneath the blades show length of the chord at the root in inches.](image-url)
While this change may seem small, it can mean the difference between an infeasible and feasible design. Further interactive design iterations could be performed to refine the design.

Both optimization-based and interactive-based DSE can help find global and local optimaums, understand how variables influence the result, and explore new design concepts. When done well, DSE can be used to intelligently and quickly understand and traverse the design space to find and understand good designs. DSE can be used anywhere in the design process from concept generation to finalizing the details of a design. Exploring the design space iteratively around some proposed geometry is commonly performed in turbomachinery design to help achieve the increasing demands on engine performance, as done by Benamara et al. [2]. However, many design challenges are presented in these iterations. Such challenges could come from infeasible structures, unmet aerodynamic performances, or changing customer priorities. Research has found that these challenges pose a significant barrier in the design process [3]. These design challenges can be mitigated by understanding how the design variables influence the model. This helps designers come to an acceptable solution in fewer iterations.

Two challenges to DSE are: the time required to evaluate a design, and knowing which designs to evaluate. Solution time has a large effect on user performance in DSE [4–7]. Goodman et al. conducted a study which required users to search a design space composed of several parameters for a design which satisfied given criteria. They found that a delay as low as 1.5 seconds between user input and visual indication of the results led to a 50% increase in the time required to accomplish the task. Others found similar results among different forms of tests and exploration. Each claimed that delays increased both the time to find a satisfactory result and the difference of the result to some known optimum [5, 6]. Decreasing the time needed to obtain these results increases the quality of DSE for compressor blade design. Real-time solutions are defined in this research as speeds faster than a standard monitor’s 60Hz refresh rate or 0.0167 seconds. These speeds, coupled with accurate models, would provide the best possible speeds for interactive DSE. Traditional methods of selecting a design to evaluate come from estimation and experience. This often takes the form of selecting a design, performing an analysis, reviewing the results, and estimating how to change the design to improve the results. Where there is experience with the design space, this process can help to find the designs which should be explored. When experience is lacking, either from novice engineers or when exploring unfamiliar design spaces, finding the de-
signs to explore is more difficult. Improving how a design is selected for exploration would also increase the quality of DSE. Either fewer designs would need to be evaluated to meet the design requirements or more useful designs could be evaluated within a given computational and time budget.

By addressing the solution time challenge and the challenge of selecting which designs should be explored, better designs can be achieved in less time and for less computational cost. Surrogate modeling is a method commonly used to decrease the computational cost of evaluating computationally expensive engineering functions [8]. It forms a cheaper function, trained on sampled evaluations of the expensive function, to estimate the response. An example of this is a surrogate which estimates the maximum stress of a new compressor blade based on finite element analyses of many other sampled compressor blade designs. The surrogate allows for faster and more design evaluations, as will be shown throughout this dissertation. PCA is a method which reduces a set of data which uses a large number of descriptors to a set which uses a smaller set of descriptors. This means the data is less computationally expensive to use. PCA also provides insight into hidden, underlying features of the data set [9]. This is done by providing a reduced set orthogonal bases which describe the principal variations in the data. This research applied surrogate modeling and PCA to develop methods which decrease the time to evaluate a design and provide insight of which designs to evaluate.

1.2 Turbomachinery Blade Design Process

Turbomachinery blade design is a multidisciplinary, iterative process, shown in Fig. 1.2, with many complex design challenges. The figure shows a well established process used in industry and research [10]. The aerodynamic design steps are highlighted as light gray, structural design step is highlighted as dark gray, and design steps which involve both structures and aerodynamics are white. This process begins by determining the requirements of the design with an engine cycle analysis. These requirements can be pressure rise, mass flow, efficiency, weight, fatigue life, structural limitations, or other aerodynamic and structural properties. Next, a mean-line analysis is used to determine a rough estimate of the blade’s cross sections based on the aerodynamic requirements. This method analyzes two-dimensional flow characteristics along the mean-radius flow path through the blade stage [11]. Then, the three-dimensional geometry is defined by stacking
Figure 1.2: Turbomachinery blade design process. The light gray boxes are primarily aerodynamic design processes. The dark gray box is primarily a structural design process. The white boxes involve both aerodynamics and structures.

multiple airfoil sections from hub, the bottom of the blade, to tip, the top of the blade. Once the three-dimensional shape is defined, current practice involves many iterations between aerodynamic and structural groups for which each has competing objectives, computationally expensive analyses, and sub-iterations within the group. During each iteration the disciplines seek to refine the design. This often requires finite element analysis (FEA) and computational fluid dynamics (CFD). This iterative process is computationally expensive, especially as more performance is demanded from turbomachinery blades. This is because increased performance is often obtained through more design iterations and more accurate simulations which use more nodes in the finite element and volume grids. Increased performance is often accompanied by higher pressure loading, the difference in pressure between the blade surfaces, and thinner structures. These lead to thinner structural margins and a greater need to understand the structural static and vibratory responses of the blades [12, 13].

1.3 Challenges of Design Space Exploration with Turbomachinery

Compressor and fan blade design suffers from both DSE challenges described above. The long solution times and high computational cost required for CFD and FEA limit DSE during the blade design process. High-quality FEA meshes may require a few minutes to several hours to analyze a single design. High-quality CFD meshes are even more computationally expensive. This limitation affects the number of designs which may be explored, how well those designs are understood, and thus the quality of the final design. The complexity of the turbomachinery blade
design also inhibits design. The push for more improved blades leads to novel designs where prior engineering experience may be limited. Understanding which designs and regions of the design space should be explored can be a difficult challenge. Additionally, instead of just needing to explore a single value such as maximum stress, an entire spatial field, such as stress on the entire blade surface, must be explored in order to better understand the design. New methods must be developed to address these difficulties of DSE in order to create better engines.

1.4 Proposed Research

The purpose of the research is to develop methods which improve DSE of turbomachinery blade structural responses. These methods include surrogate modeling to improve the speed at which the designs may be explored and PCA to help understand which regions of the design space should be explored. Both surrogates and PCA are used, complimentary to each other, to solve the challenges of performing DSE with turbomachinery. The developed methods are not dependent upon the type of analysis or part. They may be applied to many field variables of interest and parts. However, this research focused on structural responses for compressor and fan blades. Improved DSE of these turbomachinery structural responses is accomplished with the following objectives.

1. Develop surrogates and methods to quickly emulate finite element result fields of turbomachinery blades for DSE.

2. Develop a process to explore the structural design space of turbomachinery blades using PCA.

3. Develop surrogates and methods to quickly emulate vibratory finite element result fields of turbomachinery blades for DSE.

4. Develop multi-fidelity surrogates to decrease the computational cost to train emulators.

The dissertation proceeds in Chapter 2 with a review of current static and vibratory design methods, surrogate modeling, PCA, and multi-fidelity surrogate modeling. Following this review, each of the given research objectives are presented with its own chapter. These chapters, 3 through 6, are based on journal submissions. Each chapter contains the details for the methods used and
the results found. Chapter 3 contains an outline on the application of surrogate models to quickly predict the stress field of turbomachinery blades to improve the solution times. Chapter 4 contains a description of how the information obtained through PCA can be used to improve selection of samples for interactive DSE. Chapter 5 contains a description of the application of surrogate methods to vibratory structural results. Chapter 6 contains details on how multi-fidelity models were used to decrease the number of samples and computational cost needed to create accurate surrogate models. The final chapter is a conclusion of the research with a review of future work.
CHAPTER 2. BACKGROUND

This chapter covers a general overview of the turbomachinery analysis and surrogate methods used for DSE in this research. More detailed and specific discussions of these topics are contained in the following chapters. First, the background presents the state of current structural turbomachinery blade structural analysis. This is focused on the different methods used to analyze the static and dynamic structural responses. Next, the fundamentals of the surrogate modeling and PCA methods used in this research to improve DSE are presented. Then, an overview of multi-fidelity surrogate models is given. Lastly, the test geometries are shown and discussed. The methods developed in this research were applied to DSE of these geometries.

2.1 Compressor Blade Structural Analysis

Compressor blade design must account for both static, also known as steady, and dynamic, otherwise known as vibratory, structural responses. Dynamic structural responses contain time or frequency based results while static structural responses do not. Finite element analysis (FEA) is a method used to analyze a design’s structural response. FEA involves creating a mesh of the part, applying boundary constraints such as a fixed blade root, and applying loads such as rotational and pressure loads. A mesh is a discretization of the geometry into nodes and elements which allow for computation of the structural response at points along the geometry. An example of a coarse mesh for a blade is shown in Fig. 2.1. This mesh had only about 200 nodes so they could be easily seen in the figure. Turbomachinery blade finite element models usually have thousands of nodes. Boundary conditions and loads are applied directly onto these nodes. The loads cause displacements of the model while the boundary conditions constrain those displacements. A fixed boundary condition was applied to the root (bottom of the blade) as shown in Fig. 2.1. A rotational load was also applied to all of the nodes. The rotational load is dependent upon the speed of rotation, the distance of the model from the center of rotation, and material properties of the model. Pressure loads were
also applied on the pressure and suction sides (the two faces) of the blade. Static FEA solves for the steady stresses of the blade. It solves Eqn. 2.1 for the nodal displacements, \( u \), which contains the displacement for each node’s unconstrained degree of freedom [14]. Degrees of freedom are the possible directions of displacement for the nodes. Degrees of freedom are unconstrained when they do not have boundary conditions constraining their displacement. In the equation, \( f \) is the loading conditions placed on each node’s unconstrained degrees of freedom and \( K \) is the stiffness matrix which contains material and geometric properties related to the degrees of freedom for unconstrained nodes. The displacements, \( u \), are then used to obtain the stresses, \( \sigma \), for all of the
nodes, as shown in Eqn. 2.2. The function for stress, $R()$, is dependent upon the types of elements used in the finite element model as well as material properties such as modulus of elasticity.

$$\mathbf{K} \mathbf{u} = f$$  \hspace{1cm} (2.1)

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

While static stresses are relatively simple, vibratory stresses require more advanced analysis. The general steps for solving and understanding turbomachinery vibratory responses are given in Tab. 2.1. These were formulated and presented by Snyder et al. in a compilation of turbomachinery design methods [15]. The sources of excitation can be mechanical or aerodynamic. Mechanical excitation forces can come from manufacturing deviations that cause imbalance or mistuning. Aerodynamic excitation forces can come from flow distortions entering the inlet or caused by neighboring blade rows. The operating speed ranges are the rotational speeds of the engine. Steps three through six are the most computationally expensive, requiring FEA. They are

Table 2.1: The vibratory response analysis process as presented by Snyder et al. [15].

1. Identify possible sources of excitation
2. Determine operating speed ranges
3. Calculate natural frequencies
4. Construct resonance diagram
5. Determine response amplitudes
6. Calculate stress distribution
7. Construct modified Goodman diagram
8. Determine high cycle fatigue (HCF) life (finite or infinite)
9. Redesign if HCF life is not infinite
10. Conduct physical engine testing to verify predicted response amplitudes
necessary to solve for the modified Goodman values. The modified Goodman relies upon static
and vibratory stress as well as the material properties of the blade to determine the high-cycle fa-
tigue (HCF) life [16]. Modified Goodman values less than 100% have infinite HCF life. The most
accurate FEA methods use transient structural simulation, following Eqn. 2.3 which is a more
complete form of the equations of motion than Eqn. 2.1. Unsteady CFD may be used to solve for
the time dependent forces, \( f \), based on aerodynamic sources of excitation. \( M \) is the mass matrix
and \( D \) is the damping matrix. They contain information about the mass and damping distribution
through the part. When this equation is coupled with CFD it is called fluid-structure interaction
(FSI) [17]. Two way coupling involves iterations between the structural solver and the fluid solver
until convergence [18]. Each iteration solves for the pressures on the surface of the blade, these
are used to solve for the structural displacements. FSI converges on the flow and displacements
before taking a time step. Chen et al. used two-way coupling of FSI to understand stresses on
compressor blades under vibratory loads [18]. Their simulations found the amplitudes of vibration
and location on the blade where HCF failure was most likely. Two-way coupled FSI simulations,
however, are computationally very expensive, requiring weeks to run a single simulation on high
performance computers. One way coupling iterates on the fluid solver first. Once fluid conver-
genence is reached, the structural solver computes the displacements and a time step is made. Kou
et al. showed that one way coupling of FSI could accurately model vibratory characteristics in a
study to reverse engineer compressor blades [19, 20]. These simulations gave results within 10%
of experimental data for their application. One-way coupling FSI simulations are computationally
cheaper than two-way coupling but can still require days to run a single simulation on high perf-
formance computers. Simpler methods have been developed which shorten the simulation time and
make analysis more feasible for DSE.

\[
M \ddot{u} + D \dot{u} + Ku = f
\]  

(2.3)

Many of these simpler methods involve using the mode shapes and frequencies of the parts.
This is done by first considering the equation of motion shown in Eqn. 2.3. Modal analysis per-
forms eigenvalue decomposition on the inverse of \( M \). The resulting eigenvalues are the structural
mode frequencies while the eigenvectors are the corresponding mode displacements, or shapes.
Figure 2.3: Purdue blade mode displacements, \( \phi \), coarse mesh.

Table 2.2: Purdue blade mode names and frequencies for a coarse mesh.

<table>
<thead>
<tr>
<th>Mode Number</th>
<th>Mode Name</th>
<th>Mode Frequency, ( \omega ) (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First bending</td>
<td>1462</td>
</tr>
<tr>
<td>2</td>
<td>First torsion</td>
<td>3066</td>
</tr>
<tr>
<td>3</td>
<td>Second bending</td>
<td>6827</td>
</tr>
<tr>
<td>4</td>
<td>Second torsion</td>
<td>9278</td>
</tr>
</tbody>
</table>

Mode shapes are the displacements caused by vibration at the corresponding mode frequencies. The mode shapes for the Purdue blade are shown in Fig. 2.3 with their corresponding frequencies shown in Tab. 2.2. Because modal analysis does not consider loads, and because eigenvalue decomposition provides unscaled vectors, the displacements returned by modal analysis are not true displacement magnitudes for all loading conditions [1]. More analyses must be performed to account for true magnitude. Traditional design methods to account for vibration include some identification of the vibratory sources, the use of Campbell diagrams to check if any mode frequencies are close to the frequencies of the vibratory sources, estimation of the vibratory amplitudes, and assessment of the fatigue life of the part [13]. Common methods estimate vibratory amplitude from engineering experience or experimental testing [12, 21]. Recent advancements in jet engine design methods have sought to better the understanding of the static and dynamic structural results by using probabilistic methods, mixing various numerical methods with experimental measurements, or using advanced reduced-order simulations [22–29]. While most of these methods
found the results to be within 10% of the experimental results, the number of samples required for probabilistic methods is too large, several hundred to a few thousand, and the expense of building physical compressor blades for testing is too great to obtain enough data needed for DSE.

Other structural simulations make use of the mode information and vibratory loads to solve for correctly scaled vibratory stress with reduced computational expense compared to the transient simulation. One such method, harmonic mode superposition (HMS), solves for the excitation amplitude of each mode given specified harmonic loads and frequencies. The vibratory stresses are solved using HMS shown in Eqn. 2.4 through Eqn. 2.7. These equations were derived from the equations of motion for a simple harmonic forcing with no damping [30]. The modal force scalar, $p_r$, is the dot product between the load vector, $f$, and the modal displacement vector, $\phi_r$, for mode $r$. The frequency of mode $r$ is $\omega_r$ while the frequency of the forcing is $\omega_f$. Both $\omega_r$ and $\omega_f$ are used to calculate the frequency ratio, $\Omega_r$. The frequency ratio is then used to calculate the frequency scale factor, $q_r$. When a mode frequency is close to the forcing frequency the frequency ratio is close to one and the frequency scale factor is large. This leads to large spikes in vibratory stress, $\sigma_a$. When the frequencies are the same the response is mathematically infinite in an undamped system. Modern turbomachinery blade rows, which include the disk and blades, are often a single part. While this saves weight, it offers very little structural damping. Small margins of error in modeling the damping may lead to large errors in stress. Thus, to be conservative, this research used undamped HMS. Vishwakarma et al. used HMS, considering vibratory response only at the mode and engine order frequencies [31]. They found that the estimated amplitude of vibration matched within a few percent of the true vibratory amplitude. Danforth used inlet distortion patterns to obtain the forcing function and frequencies for HMS of fan and compressor blades [32]. Inlet distortion is nonuniform flow entering a fan or compressor. Many types of distortion were used to quantify the effect vibratory stress. It was found, with HMS, that aerodynamic inlet distortion had a significant effect upon vibratory stress. Bakhle performed similar simulations to investigate the vibratory stress on a fan blade designed specifically for inlet distortion [33]. They used CFD to solve for steady and unsteady aerodynamic loading. While their research was investigating flutter stability,
their methods found vibratory stress for both clean and distorted inlet flows. While they were able to improve the fan’s efficiency, many of their designs were near the safe structural limits.

\[ p_r = \phi_r \cdot f \]  

\( \Omega_r = \frac{\omega_f}{\omega_r} \)  

\[ q_r = \frac{1/\omega^2}{(1 - \Omega_r^2)} \]  

\[ \sigma_a = \sum_r \sigma_r q_r p_r \]  

2.2 Compressor Blade Fluid Analysis

Various types of CFD simulations may be used to understand the aerodynamic loading on compressor and fan blades. One of the more common solvers is the Reynolds-Averaged Navier-Stokes (RANS). This is a steady-state simulation which often only requires a single blade passage per blade row, like that shown in Fig. 2.4. The boundary conditions, such as pressure at the inlet

Figure 2.4: PBS Rotor 4 single blade passages for rotor and stator rows.
and outlet, define the solution for specific operation conditions. Beginning with an initial velocity field, the solver iterates until convergence [34]. Each iteration solves for the velocity and pressure fields of the fluid regions. Only using a single blade passage per blade row reduces the number of grids needed for solution. RANS, however, does not provide unsteady aerodynamic loading. Unsteady Reynolds-Averaged Navier-Stokes (URANS) provides unsteady loading by iterating to convergence, then taking a step in time. The completed solution includes time dependent pressures on the surfaces of the blade. This allows for unsteady effects from blade row interactions and non-uniform inlet conditions at the cost of increasing the computational cost by more than an order of magnitude. The increased cost comes from solving the simulation at several time steps and the need to model more blade passages for the rotors and stators. The varying levels of fidelity and computational cost with fluid and structural analysis lead to different simulation types being used for different stages in the design process. Steady-state CFD and static FEA are more likely to be used in early stages of DSE. URANS and full transient FEA are used towards the end of the design process to refine the design and provide more confidence prior to physical testing.

2.3 Surrogate Modeling

Surrogate modeling is a method which uses a computationally cheap set of equations to estimate the response of a design where the computationally expensive solution is unknown. According to Forrester et al., surrogates are “educated guesses as to what an engineering function might look like, based on a few points in space where we can afford to measure the function values” [8]. There are many mathematical forms of surrogate models that may be used to explore a design space and they all require the expensive system to be sampled at a selected subset of the design space. These sampled designs are selected with a design of experiment (DOE) which seek to intelligently spread the samples through the design space [8]. The surrogate is created by fitting weights which allow the surrogate to interpolate from the selected expensive samples to obtain results for unknown designs. If the surrogate is well trained, it estimates the result of the computationally expensive system with low error. The error of the surrogate is dependent on how many samples are selected, and where they are selected, from the design space.

One of the more common surrogate models is the radial-basis function (RBF). The RBF training equation is shown in Eqn. 2.8 and the emulation equation is shown in Eqn. 2.9. RBF
is trained by solving for weights of a set of kernel functions, $\psi$, centered at each of the $s$ samples. These kernel functions are evaluated at the euclidean distances between the sampled designs, $c$. The inverse of $\Phi$, a matrix composed of the kernel evaluations, is multiplied by $Y$, a matrix composed of the values from the sampled designs. A new response is estimated, in Eqn. 2.9, by multiplying $\lambda$, the vector created from evaluating each kernel at the new design, and $W$, the matrix of kernel weights obtained through the training equation.

$$W^T = \Psi^{-1}Y$$ where $\Psi_{i,j} = \psi(|c_i - c_j|_2)$ for $i, j = 1, \ldots, s \quad (2.8)$

$$\hat{y}(x) = \lambda W$$ where $\lambda_i = \psi(|x - c_i|_2)$ for $i = 1, \ldots, s \quad (2.9)$

An example of an RBF surrogate model trained on a simple quadratic equation, Eqn. 2.10, is shown in Fig. 2.5. Both images in the figure have the quadratic equation shown with the blue line, the surrogate model shown with the red line, and the samples shown with the red triangles. The samples are evaluations of the quadratic equation at selected values for $x$. The RBF surrogate model is trained by finding the weights ($W$) for kernel functions ($\psi$), shown in green, centered at the samples. This figure uses multiquadric kernels, shown in Eqn. 2.11. The final surrogate evaluation, or red line, is the sum of the weighted kernels, the green lines. The difference between the true function, blue, and the surrogate estimation, red, indicates the error for this example. Adding more, well placed samples, would decrease the error between the surrogate and the equation.

![image](a) Surrogate

![image](b) Kernels

Figure 2.5: Surrogate model of quadratic equation using a radial basis function.
\[ y = -4 \cdot x^2 + 4 \cdot x \]  
\[ \psi(r) = \sqrt{\left(\frac{r}{\varepsilon}\right)^2 + 1} \]

2.3.1 Surrogates of FEA

Because FEA is computationally expensive, surrogate modeling has been applied to allow for better DSE of structural parts. The surrogate allows for DSE which is two or more orders of magnitude faster on parts which require FEA [35, 36]. Most applications have used surrogates to emulate a single value obtained through FEA, such as max stress or displacement. Deng et al. and Houck et al. found these emulators to be accurate enough for useful DSE for aeromechanical responses of fan and turbine blades [35, 36]. The surrogates used by Deng et al. had less than 5% error. They used a single value emulator, based on maximum stress, for aero-mechanical optimization of a fan blade and were able to improve efficiency without exceeding structural limitations [36]. These single value emulator methods were sufficient when DSE was used for optimization, but not for interactive DSE which needed to understand the entire structural response of the part. Heap et al. created unique surrogate models for every node of the finite element model as shown in Fig. 2.6. This allowed for fast visualization of the entire stress field across the design space for thorough interactive DSE [37]. The surrogate inputs were design variables and the surrogate output was stress at a node of a finite element mesh. In order to create a surrogate model for every node, the same relative node placement had to be maintained on all of the finite element models in the design of experiments (DOE). This limited their finite element models to parametric meshes on simple geometries like the cantilever beam model shown in Fig. 2.7. Benzaken et al. and Schulz et al. also used surrogates to emulate the stress field of structural parts [38, 39]. Benzaken et al. used isogeometric methods while Schulz et al. used mesh morphing. This expanded surrogates from simple parametric parts, to more complex geometries and meshes, a step necessary for compressor blade DSE.

While many forms of surrogates are possible, Heap et al. found that real-time DSE could be performed with finite element models by using the radial basis function (RBF) surrogates [37].
This surrogate method and others have been tested for their ability to accurately emulate results across design spaces [40]. The RBF was only slightly surpassed by Kriging for accuracy, within 1% mean squared error, but had the fastest training and emulation time. By using the RBF, the surrogates used by Heap et al. allowed up to 1,400 nodes to be emulated and visualized in less than 0.05 seconds, or “real-time”. These models had a maximum of 15.9% error with respect to the FEA and an average error below 5%. They were able to then explore the structural design space of simple shapes fairly accurately in “real-time”.

2.4 Principle Component Analysis

Principle component analysis, or PCA, is a spectral method which creates a reduced order model from data with large number of dimensions while maintaining a majority of the information [9]. It is also known in other fields as proper orthogonal decomposition (POD) [41–43]. The method finds orthogonal variations in a given data set. This can be seen with Fig. 2.8. The left
image shows some data described by some bases, $Y_1$ and $Y_2$. The image on the right shows a new set of orthogonal bases, $V_1$ and $V_2$, which more efficiently describe the data. It is more efficient because the majority of the variation in the data is described with $V_1$ and because the bases are still orthogonal. Because they are orthogonal, each variation is independent from one another and can assist in better understanding the data. A change along one principal component does not cause a change along another. PCA is performed by the creation of a data matrix, $Y$, in which the rows of the matrix are different observations and the columns of the matrix are different values obtained for all of the observations. Then, singular value decomposition is performed on $Y$ after subtracting the column averages, $\bar{y}$, in order to obtain $U$, $S$, and $V$ as shown in Eqn. 2.12. The matrix $V$ contains the eigenvectors, or principal components (PCs). These PCs are the orthogonal variations of the data present in the columns of $Y$. Every sample of the model, or row of $Y$, may be represented by a linear combination of the PCs. The number of PCs may be truncated to retain only those which explain the majority of the variance in the data. However, the greater the number of PCs, the more the linear combination will represent the true model. The scores, or coefficients, used in the linear combination are the projections of the data, $Y$, onto PCs. The score of each PC in each design is contained in the matrix of PC scores which is obtained by multiplying the matrices $U$ and $S$. For this research this product will be referred to as $A$. This can be seen in Fig. 2.8. The shown data is described by each point’s values along $Y_1$ and $Y_2$ in Fig. 2.8(a). These values make up the data.
matrix, \( \mathbf{Y} \). The new bases in Fig. 2.8(b) are the columns of \( \mathbf{V} \) and projections of the data onto those bases are the matrix \( \mathbf{A} \). In Fig. 2.8(b), the majority of the variation is described with \( V_1 \). Just using this PC reduces the number of dimensions used to describe the data but retains the majority of the variation. The second PC, \( V_2 \), may also be used to explain the rest of the variation but at the cost of returning to a two dimensional space. The PCs are centered in the data due to subtracting \( \bar{y} \) before performing singular value decomposition and helps get better descriptions of the variation. PCA is used extensively in aerospace engineering for many reasons, including; providing real-time analysis, data size reduction, its ability to control the trade-off between error and computation cost, and improved understanding of characteristics in the data [44].

\[
\mathbf{Y} - \bar{y} = \mathbf{USV} = \mathbf{AV}
\]  

(2.12)

### 2.4.1 PCA for DSE of Result Fields

Blanc et al. used PCA to describe the temperature at a large set of nodes on a turbine blade finite element model with a smaller set of PCs [45]. These components represented the orthogonal spatial variations of temperature. A set of samples were created for which each sample had varying temperature and velocity at the inlet boundary. The temperature profile for a new temperature and velocity boundary condition was a linear combination of the PCs with their estimated scores. This method allowed rapid and accurate emulation of the entire temperature field for any design within the trained design space. By doing this, the temperature field of the turbine blade could be measured quickly to within 5% maximum error or within 2% root mean square error for only 64 samples. La et al. used PCA to classify mode shapes of compressor blades across a design space [46, 47]. They found that by using PCA they could classify and order the modes according to their shape.

Similar research also applied this method to FEA and CFD [25, 48–50]. Ni et al. emulated stress contours for specific vibratory mode shapes on the NASA ROTOR 67 compressor blade [25]. Each node was represented by a column in the data matrix while each row represented a variation of the blade geometry. Singular value decomposition led to a set of orthogonal stress distribution bases. They were able to use this method to estimate and explore peak blade stress from a limited set of experimental tests. Their method achieved errors below 5%. Ross et al. used the
displacement of the nodes to view the PCs of strain under various loading conditions [48]. Yeh et al. was able to perform DSE using a PCA-based surrogate method of flow in combustors [49]. They were able to emulate the flow accurately in a few minutes. While this was a great improvement over traditional CFD, there was still much room for increased speed to allow for DSE improvement.

2.5 Multi-Fidelity Modeling

Multi-fidelity surrogate modeling is a method which adds computationally cheaper, albeit less accurate, samples to improve a surrogate model created from more accurate, yet more computationally expensive samples. An example multi-fidelity sample set is shown in Fig. 2.9. The low-fidelity models can provide greater coverage of the design space due to their lower computational cost while the high-fidelity models provide a better representation of the system to be emulated. Multi-fidelity surrogates utilize both of these fidelities to increase accuracy and decrease computational expense. In a study on the optimization of compressor blades, Benamara et al. found that using multi-fidelity surrogates, instead of surrogates based only on high-fidelity models, decreased emulation error on a compressor blade for the same computational costs [43]. The multi-fidelity surrogate increased the accuracy of isentropic efficiency from 74% to 87% and the total-to-total pressure ratio from 79% to 95% as calculated by the Pearson correlation coefficient [51]. This coefficient correlates the emulated designs to the true CFD values with a match of 100% indicating all nodes were the same between the two solutions across all tested designs.

2.5.1 Types of Fidelities

The first step in forming multi-fidelity models is the selection of fidelities. Simulations of lower fidelity can take different forms. Iterative simulations, such as CFD or transient FEA, may have relaxed convergence tolerances. Both CFD and FEA may use meshes with fewer grids as a lower fidelity model as suggested by Yondo et al. and Clark et al. [44, 52]. Benamara et al. used a fine mesh for high-fidelity modeling of compressor blades and a coarse mesh for lower fidelity modeling [43]. They found that, while the CFD could obtain results for the two fidelities, in order to combine the results into a multi-fidelity model they needed to map both fidelities to a common grid template. The use of low-fidelity models increased the accuracy of the surrogates for the
same computation cost. Yondo et al. further suggested that instead of just two levels of fidelity, a more continuous, variable-fidelity surrogate could be used [44]. In the review of multi-fidelity modeling, it has been found that, whatever forms of fidelity are chosen, the low-fidelity model should be orders of magnitude cheaper to compute in order to reduce error without an increase in computational cost [43, 52, 53].

2.5.2 Types of Multi-Fidelity Surrogate Models

The next step in multi-fidelity modeling is the selection of the surrogate method which emulates the high-fidelity response from the multi-fidelity samples. Many surrogates have been developed to achieve this purpose. Most notable among these is Co-Kriging which is well researched and robust to various simulations [44]. Co-Kriging treats the training data as if they were a stochastic process where the cheap samples have error and the expensive samples do not [8]. A low-fidelity stochastic-based estimation is made from the low-fidelity samples. Then multiplicative and additive transformations are made to estimate the high-fidelity response. Other forms of multi-fidelity surrogate models include correcting the low-fidelity model to match the high-fidelity model by creating a correction map across the design space based on multiplicative transformations, additive transformations, or a combination of both. Many authors have studied the nuances of these methods, each finding various strengths and weakness [44, 52, 54]. Their research suggests

Figure 2.9: Possible design space sampled for multi-fidelity modeling.
that, for most cases, a method which is a hybrid of the additive and multiplicative transformation performs the best. Fernandez et al. also covered these methods but introduced scaling based on uncertainty rather than deterministic methods [55]. Multi-fidelity surrogates have also been applied through PCA by centering the analysis closer to the high-fidelity samples. Benamara et al. used PCA in their multi-fidelity modeling by using this centering technique [43]. The results for the high-fidelity CFD were combined with the results for the low-fidelity CFD. Their method improved the accuracy of fluid density contours around the compressor blade by 17%. This dissertation used PCA as the basis for the development of novel multi-fidelity methods. The research tested this novel method and compared it to other multi-fidelity. Each method was evaluated for its ability to improve the accuracy of PCA-based surrogates of compressor blades for a set amount of computational cost. These methods were tested and evaluated by measuring the error between the surrogate model to high-fidelity samples.

2.6 Turbomachinery Blade Models

Two geometries were used to create the surrogate model training data in this research. First, the Purdue blade, is a high pressure compressor blade which was developed to study transonic aerodynamic effects. The blade with disk are shown in Fig. 2.10(a). It has been the subject of many academic studies [56]. Some include structural PCA analysis [57]. This research used a

![Figure 2.10: Purdue blade model.](image)
Table 2.3: Set of possible design variables for Purdue blade used for this research.

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Symbol</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>$H$</td>
<td>± 10% of nominal $H$</td>
</tr>
<tr>
<td>Ave. Radius</td>
<td>$R_o$</td>
<td>± 10% of nominal $R_o$</td>
</tr>
<tr>
<td>Root Chord</td>
<td>$C_R$</td>
<td>± 10% of nominal $C_R$</td>
</tr>
<tr>
<td>Tip Chord</td>
<td>$C_T$</td>
<td>± 10% of nominal $C_T$</td>
</tr>
<tr>
<td>Sweep</td>
<td>$S$</td>
<td>± 10% of nominal $S$</td>
</tr>
<tr>
<td>Lean</td>
<td>$L$</td>
<td>± 10% of nominal $L$</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>

blade-alone model of the Purdue blade, shown in Fig. 2.10(b). Blade-alone indicates that only the blade, with no disk, was used. These types of models are often used in industry early in the design process to reduce the computational cost by modeling less geometry while still understanding the blade response. The leading edge of the blade is on the right when viewing the blade from the suction side, as in Fig. 2.10(b). The nominal design has a height of 2.0 in (5.08 cm), a mean chord of 1.9 in (4.83 cm), and a mean thickness to chord ratio of 8%. The design variables which are explored in this research are found in Tab. 2.3. In this table, the first column is the name of the design variable, the second column is the symbol which will be used in this dissertation to represent the design variable, and the third column is the range the variable is permitted to change. Other geometric dimensions for the nominal blade, as well as the test stand configuration used in the experimental analysis, can be found in given references [56, 57]. Several finite element models were created from the Purdue blade, each with a different number of nodes. The FE models used SOLID158(8) elements of 25K, 50K, 100K, 250K, and 500K nodes [58] with a linear static solver.

The second geometry is the Air Force Research Laboratory (AFRL) Parametric Blade Study (PBS) Rotor 4. This blade, shown in Fig. 2.11, is a transonic fan blade which was developed as a part of a study to understand the effect of various design variables on the aerodynamic properties of the fan [59]. Soderquist and List found that URANS could produce results close to the experimental findings from Rotor 4 [60, 61]. The design variables used for PBS Rotor 4 are found in Tab. 2.4. While the Purdue blade is commonly used for academic research of design and analysis methods, PBS Rotor 4 represents a more modern turbomachinery blade. The geometry
(a) PBS Rotor 4 disk  

(b) PBS Rotor 4

Figure 2.11: PBS Rotor 4 model.

Table 2.4: Set of possible design variables for the PBS Rotor 4 fan blade used for this research.

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Symbol</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mid-Span Chord</td>
<td>$C_M$</td>
<td>± 10%</td>
</tr>
<tr>
<td>75%-Span Chord</td>
<td>$C_{75}$</td>
<td>± 10%</td>
</tr>
<tr>
<td>Tip Chord</td>
<td>$C_T$</td>
<td>± 10%</td>
</tr>
<tr>
<td>Mid-Span Thickness-to-Chord Ratio</td>
<td>$TC_M$</td>
<td>± 10%</td>
</tr>
<tr>
<td>75%-Span Thickness-to-Chord Ratio</td>
<td>$TC_{75}$</td>
<td>± 10%</td>
</tr>
<tr>
<td>Tip Thickness-to-Chord Ratio</td>
<td>$TC_T$</td>
<td>± 10%</td>
</tr>
</tbody>
</table>

used in this research includes the fillet to reduce artificial stress concentrators from FEA and provide results which more accurately model real world blades. The FE models used SOLID158(8) and SOLID187 with 220K nodes [58]. This model used a non-linear solver which is more accurate with larger deflections. Both blades were made from and modeled with Ti-6Al-4V, a material commonly used in turbomachinery blades [33]. The finite element models were not grid refined to a converged solution as the scope of the research was to provide DSE methods for models which are used by industry. The models described are representative of those which are used for DSE at various stages of the design process in industry.
CHAPTER 3. REAL-TIME DESIGN SPACE EXPLORATION

The length of time necessary to perform structural analysis of turbomachinery blades limits the number of designs which may be explored. Surrogate modeling of finite element analysis was developed to reduce the time required to perform a large number of structural evaluations so better designs may be found more quickly. The two surrogate methods developed and used in this research were the node-based surrogates and surrogates based on principal component analysis (PCA). The emulation speed and error were measured to evaluate the ability of the surrogate methods to provide useful DSE of turbomachinery blades. The node-based results were presented at the 2018 Turbomachinery Technical Conference & Exposition [62]. This chapter is based on this publication but also includes the results for the PCA-based surrogates which were developed after the conference. The chapter advances the body of research by testing the ability of surrogates to provide DSE for turbomachinery blades in the following manner.

- Apply mesh morphing to allow for more complex geometry than those used by Heap et al. [37].
- Develop a single surrogate model which can emulate the stresses at all points at a faster speed.
- Develop a PCA-based surrogate for further speed improvement.
- Quantify and compare the speed increase of DSE for both surrogate methods.
- Quantify and compare the error associated with creating a surrogate to emulate turbomachinery blade static stress for both surrogate methods.
3.1 Introduction

Structural analysis of turbomachinery compressor blades requires computationally expensive finite element analysis (FEA). To accurately model the structural responses, the blades require a very large number of nodes, between 1K and 1M [19,63]. In general, the larger the node count the greater the computational expense. Exploring how the structural response behaves across different parametric combinations compounds this expense because many analyses are required. This exploration, called design space exploration (DSE), allows engineers to better understand and optimize a design. Traditionally, DSE of structural responses requires FEA of several different parameter combinations. More analyses lead to a better exploration while fewer lead to a lower possibility of finding an optimum final design. Often, relatively few analyses are solved due to the time and computation costs that are required for solving a large number of analyses.

One method used to decrease the expense of analysis is surrogate modeling [8, 64–67]. This method predicts the result of a new design by using training data from previously analyzed designs. The expense lies in creating training data of specific designs but provides the ability to predict the results of any number of designs with negligible additional expense. The application of these “surrogates” for turbomachinery simulations have assisted engineers in exploring the design space of various engine properties [68,69]. Vasilopoulos et al. used surrogates to predict values for aerodynamic objective functions based on compressor blade geometry. Leylek et al. and Wagner et al. both applied the same methods to predict aerodynamic properties of turbine blades [70–72].

Recent work has applied surrogate models to predict structural responses from FEA of turbomachinery. Zhang et al., for instance, used surrogates to quickly predict the maximum stresses of a turbine disk [64]. Geller et al. used surrogates to predict the maximum stress on compressor impellers [73]. Both studies found that the estimation for maximum stress of turbomachinery parts was performed accurately at negligible computation cost compared to FEA and allowed for the design to be thoroughly explored in the optimization process. These surrogate methods, however, only provided a prediction for a single structural value instead of predicting the full stress field. This limits DSE when engineers seek to understand the full stress response.

This research studies surrogate modeling of entire stress fields as a feasible method to perform DSE of turbomachinery compressor blades. This is a novel turbomachinery design method which has not been published in open literature. This method may help engineers improve com-
pressor blade designs within allotted time and resource constraints. The speed and accuracy of the surrogates are measured in order to evaluate how and where they may be applied to provide the most benefit to DSE of compressor blades. As a part of the creation of training samples for the surrogates, mesh morphing is studied as a means to maintain the same nodal structure across the entire design space. This research shows that mesh morphing a compressor blade finite element (FE) model across the entire design space is sufficient to provide accurate surrogate predictions.

This research will focus on the application and evaluation of surrogate models to perform DSE of stress fields for the Purdue blade, a compressor blade used for research [56]. This will include prediction and visualization of the surrogates built from FE models. The FE models are composed of 25K to 500K nodes, and one to ten design variables.

The paper will proceed as follows. First, the paper will discuss recent advances and applications in the areas of surrogate modeling for: turbomachinery, FEA, and DSE. Next, methods for applying surrogate modeling to visualize FEA of compressor blades for rapid design space exploration will be outlined. Finally, the results of this method and a discussion of their impact will then be presented.

3.2 Background

Surrogate models have been applied to make fast structural predictions for DSE of turbomachinery parts. In 2013, surrogate models were used to quickly find the cause of an engine blade failure [35]. Because the surrogate model allowed for a fast and thorough exploration of the design space, the time between blade failure and installation of a new part was decreased by 50%.

3.2.1 Finite Element Analysis

Surrogate modeling may be used to quickly emulate results of FE models. This has been used to predict stresses in parts that would be very expensive to compute in iterative processes. The structural results in Zhang et al.’s study were required to ensure the design did not exceed specified constraints [64]. Using FEA in the optimization loop would have been too computationally expensive. The surrogates, predicting the results of FEA, allowed more evaluations of the structural
results which assisted the optimization process in exploring the design space. This led to a 7% decrease in the maximum von Mises stress of the part.

Until recently, surrogate models of FEA were used to represent only the maximum or average stress of parts. This method is sufficient where only a single result is needed. A better understanding, however, of the structural response and the design space is obtained when the stress contours of the entire part are visualized. Heap et al. found that the entire stress field could be predicted by representing the stress of every node with a unique surrogate [37]. To visualize the results, each node’s surrogate was evaluated and the results were presented together in a reconstructed FE model. This method required that each simulation, used to train the surrogates, have the same mesh structure. If surrogates are to represent specific nodes, those nodes must be the same relative position across each of the FE models used in training. Heap et al. used a parametric mesh to ensure the nodes remained in the same relative location on the part. When a new design was sampled for training, the nodal locations were adjusted parametrically to match the new geometry. For visualization of the surrogates in OpenGL, the nodal locations were calculated parametrically and the stresses mapped as colors between them. This meshing method limited the study to simple parts that could perform well with parametric meshes. Walther et al., in a study of fast mesh regeneration, found that nodal locations, instead of stresses, of very complex geometry could be accurately predicted by surrogates [74].

### 3.2.2 Principal Component Analysis

Instead of representing each node with a unique surrogate, many have used principal component analysis (PCA) to reduce the number of needed surrogates. PCA is a spectral method which creates a reduced order model from data with a large number of dimensions while retaining as much information as possible [9, 75]. It finds the orthogonal variations, or principal components (PC), which explain the most variation in the data. Because they are orthogonal, each PC is independent from one another. PCA also provides the scores, or projections of the samples onto the PCs. Surrogate models may then be created to emulate the PC scores. By training surrogates on only the scores of a few PCs, which still explain the most variation, the number of values the surrogates needed to produce for an estimation of the result field was reduced. The surrogate predicted
scores can be transformed back to values for the nodes. This allows the data to be more quickly evaluated. This research followed the basic method presented by Jolliffe et al. [9].

PCA-based surrogates have been applied to represent flow fields, heat transfer, and geometric tolerances of compressor blades, turbine blades, and other parts. [41, 45, 76]. Blanc et al. used PCA to reduce the set of nodal temperatures on a turbine blade finite element model to a small set of PCs [45]. These represented the orthogonal variations of the temperature field on the blade. A surrogate was trained on the PC scores to create a PCA-based surrogate. This surrogate emulated the scores for each PC and then transformed those scores into an estimation of the temperature field. The PCA-based surrogate predicted a full temperature field as linear combination of the PCs with their predicted scores. Brown et al. used PCA to get geometric variations in as-manufactured compressor blades and used the scores to create surrogates of blade mode shapes and frequencies [77]. Using the PC scores as inputs to the model allowed them to predict how changes in geometric variation affected the frequencies and modes.

3.2.3 Design Space Exploration

The speed at which the surrogates in Heap et al.’s research were solved allowed for quick exploration of the stresses across the design space of the parts under investigation. The surrogates decreased the time to visualize new results by more than 99% from the FEA. Models with up to 1,392 nodes were solved and visualized in less than 0.05 seconds [37]. The time between user entry of the design and the display of the results was so small that it was not perceivable to the user. The time between design entry and visualization of the results will be termed interactive speed. This metric will be used to assess the benefits of using surrogates for prediction of compressor blade stresses. The interactive speeds found by Heap et al.’s study would allow engineers to more thoroughly explore design spaces.

The parametric mesh used by Heap et al., however, limited DSE to parts that could be represented with simple, parametric meshes. Only flat beams, with relatively low node counts, were explored in their research. In order to allow exploration of more complex parts, studies have applied different meshing methods [38]. Schulz et al. used a discretized design space where each discrete region had a unique mesh and set of surrogate models [39]. Mesh morphing was used within each discrete region to adjust the mesh across small geometric changes. As DSE crossed the
boundaries of the discrete regions the method changed which surrogates were used in prediction. The method allowed the same kind of interactive DSE as provided by Heap et al., but required that multiple samples be taken of the same design in order to provide a smooth transition between regions of the design space.

3.3 Method

This research developed a method which is a combination of the methods presented by Heap et al. and Walther et al. to predict both the stresses and the geometry of a turbomachinery compressor blade [37, 74]. Instead of using a parametric mesh, as done by Heap et al., this method implemented mesh morphing, as done by Schulz et al., to allow prediction of more complex geometry [39]. Their method used mesh morphing across small parts of the design space. The method developed for this present research mesh morphed the compressor blade across the entire design space.

This method was applied to analysis of the Purdue blade, a turbomachinery compressor blade developed at Purdue for verification of methods of compressor blade analysis [56, 78–80]. The geometry of this blade can be seen in Fig. 3.1. The leading edge of the blade is on the right when viewing the blade from the suction side, as in this figure. The geometric dimensions for the nominal blade, as well as the test stand configuration used in the experimental analysis, can be found in given references. While this study uses the Purdue blade, the method is general enough

![Purdue blade section splines](image)
for any DSE of any FE model. For the present research, the blade was parameterized with sweep, lean, chord, angle, inner and outer radius, as well as rotation speed and pressure loading. Each of the geometric parameters adjusted either the tip or root of the blade while each section was adjusted by the square of its percentage along the span. To test this method over a range of mesh densities, several finite element models were created from the Purdue blade, each with a different number of nodes. The FE models used SOLID158(8) elements of 25K, 50K, 100K, 250K, and 500K nodes [58].

3.3.1 Mesh Morphing

The prediction of a stress field required unique predictions of the stress values at multiple locations in the field. The predicted stresses for each location were later graphically interpolated to visualize the full field. For surrogate modeling, this meant that the same locations must be selected in each sample used to train the surrogate. While there were various ways this could be accomplished, the research studied using the location of the nodes of the FE model. Mesh morphing was used to ensure the nodes stayed in the same relative location on the compressor blade.

The method presented here used a nominal mesh that morphed to changed geometry. A parametric NX part was used to provide a changeable compressor geometry. Then, the nominal mesh was morphed to the new geometry using NX Advanced Simulation’s automatic mesh morphing tool. This tool performs well when the geometric changes are relatively simple, such as seen with the compressor blades used for this research. The simulation was then solved using the newly updated mesh. On the next iteration, the mesh was morphed from the nominal mesh again and not from the previously used mesh. This ensured that the quality of the mesh did not degrade through the creation of the training samples.

3.3.2 Surrogate Model

An important step of this method was the selection of a suitable surrogate model. There are many models that have been studied and found to perform well in certain circumstances. Two, however, are consistently better with sparse training data in design spaces with many parameters:
the Kriging model and the radial basis function (RBF) [37, 40, 81, 82]. Implementations for both models were tested in this research for their speed to predict a new response. Because both models were shown to have high accuracy in sparse data, the speed of prediction was considered the most important model characteristic for this research. Faster solutions meant better DSE. Therefore, either a larger model could be explored or faster interactive speed could be attained. The speeds for surrogate prediction of compressor blades are later presented and discussed.

It was found that, for this application, the specific implementation of the RBF method tested outperformed the specific implementation of the Kriging model tested. In seeking to compare the time of prediction between the two methods, it was discovered that the time to train an RBF on 84 samples with six input variables was less than 0.005 seconds. The Kriging model took 133 seconds to perform the same task. That means training 4,000 Kriging models in order to predict 4,000 nodal stresses would take close to a week. Both methods were tested in the same hardware and software environment. From this result, it was concluded that RBF should be used. Eqn. 3.1 shows the training equation for RBF. This solves for a matrix of weights, \( W \), by evaluating a kernel function, \( \psi \), at the euclidean distances between the samples, \( c \). The kernel function chosen for this research was the multiquadric equation, shown in Eqn. 3.2. In this equation, \( \varepsilon \) is a constant which is equal to the average distance among the designs. These kernel evaluations were compiled into a matrix, \( \Psi \). \( W \) was equal to the inverse of this matrix multiplied by a matrix \( Y \). For the node-based surrogates, \( Y \) was the data matrix, shown in Fig. 3.2, which contained as rows the samples for each FEA evaluation and as columns the stress and x, y, and z coordinates of each node. Eqn. 3.3 was used to predict a new response after the surrogate was trained. The kernel functions were evaluated at the euclidean distances between the new design, \( x \), and the samples, \( c \). The vector containing these evaluations, \( \lambda \), was multiplied by \( W \) to obtain the predicted response, \( \hat{y}(x) \). For the node-based surrogates, \( \hat{y}(x) \), contains the von Mises stresses and x, y, and z coordinate location of each node. This node-based method provided a speed improvement from the method used by Heap et al. because only one surrogate was used to evaluate all of the nodes.

\[
W^T = \Psi^{-1}Y \quad \text{where} \quad \Psi_{i,j} = \psi(|e^{(i)} - e^{(j)}|_2) \quad \text{for} \quad i, j = 1, \ldots, s
\] (3.1)
Figure 3.2: The \((n, p)\) data matrix, \(Y\). Each row was a unique finite element model sample, of which there were \(n\). The columns, of which there were \(p\), represented the \(m\) nodes on the finite element model.

\[
\psi(r) = \sqrt{\left(\frac{r}{\varepsilon}\right)^2 + 1}
\]  

\[
\hat{y}(x) = \lambda \mathbf{W} \text{ where } \lambda_i = \psi(|x - c^{(i)}|_2) \text{ for } i = 1, \ldots, s
\]

The PCA-based surrogates used the same RBF equation as the node-based method. However, PCA was performed on the data matrix, as shown in Eqn. 3.4. First, the column means, \(\bar{y}\), was subtracted from \(Y\). Then, singular value decomposition was performed to solve for \(U, S, V\). The PCs, or the variations of the data through the design space, are represented with \(V\). Multiplying \(U\) and \(S\) returns the PC scores, \(A\), which the PC scores each sample as shown in Fig. 3.3. The RBF surrogate model was created to predict the PC scores, \(\hat{a}(x)\), at a new design, \(x\). Eqn. 3.5 was used to transform the predicted PC scores into a stress field and coordinate values.

\[
Y - \bar{y} = USV = AV
\]
Figure 3.3: The \((n, q)\) PC score matrix, \(\mathbf{A}\). Each row was one of \(n\) samples. Each column was a PC, of which there were \(q\).

\[
\hat{\mathbf{Y}} = \hat{\mathbf{a}}(\mathbf{x})\mathbf{V} + \bar{\mathbf{y}}
\]

(3.5)

In order to further decrease the time required to predict the entire solution field, surrogates were only created on nodes which were located on the surface of the FE model. This was a reasonable adjustment because FEA software used in industry generally only displays the results of the surface nodes. The percentage of nodes that did not lie on the surface varied based on the geometry and the mesh density. Fig. 3.4 shows how the percentage of non-surface nodes, which could be ignored, increased as the number of nodes in the mesh of the Purdue blade increased. The figure’s horizontal axis uses a log scale, showing that the percentage of ignored nodes increased towards some upper limit as the order of magnitude of nodes increased.

### 3.3.3 Training Data Workflow

A design of experiments (DOE) was used to create a set of designs, or parameter combinations, to govern the geometry of the compressor blade. The models had up to ten design parameters that controlled the geometry. The DOE was created in Python, which generated a Latin-Hypercube that maximized the distance between samples in the design space. A large number of these DOE were generated and the one with the best maximized spacing was used.
The percent of interior nodes is shown in Figure 3.4. For each design, the parameters were passed to a Java file used to control NXOpen API. Through the NX10 API, used in batch mode, the compressor blade .prt file, .sim file, and .fem file were each opened. The part file contained a list of expressions that parametrically governed the geometry of the blade. The Java file iterated over each parameter in the design. First, in the .prt file, the expression for that parameter was changed, updating the geometry of the blade. Next, in the .fem file, the blade was mesh morphed using NX Advanced Simulation’s Automated Morph tool to match the new geometry. The parameters were morphed one at a time in order to help guide the morphing through the complete geometry change. This resulted in fewer failed elements which broadened the design space over which the mesh could be morphed. After the blade was morphed across each geometric parameter the .sim file was opened. This file contained expressions that governed loading conditions. Once these expressions...
were changed the FEA file was written. This process was repeated for every design produced by the DOE.

After all the FEA files were written, Python was used to launch the FEA in batch mode. This process read the FEA files, solved the FEA, and wrote the stresses and locations for the surface nodes to text files. When this process finished, Python was used to write the DOE to a text file.

3.3.4 Surrogate Creation and Visualization

After the training data was created, the surrogates were trained. Heap et al. used surrogates to predict nodal stresses while Walther et al. used surrogates to predict nodal locations [37,74]. This research combined these methods to create a surrogates to predict both stresses and coordinates of every node. Python was used to create a surrogate model to represent every node’s stress and three-dimensional coordinate location. The surrogate model, therefore, predicted four values per surface node of the finite element model. To train the surrogate, the DOE and the values of that node for each design in the DOE were passed to the RBF. The surrogates create a relationship between the designs in the DOE and the value at the nodes. Once the surrogate is trained, Python passes a set of parameters, representing a single design, to the created surrogate. These returned the new nodal stress and coordinate values which were passed to the visualizer.

The visualizer used PyOpenGL to draw the results. PyOpenGL draws square elements to the screen when passed an x, y, and z coordinate along with a color for four different points. The structural simulation was run in batch mode to produce a list of the element faces that lie on the surface of the FE model and the nodes that correspond to those faces. In Python, the list of predicted stresses was converted to a list of colors. These colors, with the predicted coordinates and the face data, were used to draw the elements of the FE model. This reconstructed FE model showed the predicted stress field and geometric shape of the compressor blade for the given design.

3.4 Results

The method described allowed DSE of the Purdue blade at interactive speeds while accurately representing the full response of the part. To assess the success and value of this method for DSE of compressor blades, the surrogate models are compared to the FEA. The metrics of predic-
tion speed and prediction accuracy are discussed. The model accuracy will provide understanding as to how well the surrogates represent the structural model. Prediction speed will show how fast the structural results may be obtained and demonstrate how useful this method will be for DSE.

The GUI, showing the predicted results of the Purdue blade, along with the parameter controls, is shown in Fig. 3.6. The blade in part (b) of the figure was a visualization of a reconstructed FE model with each node’s von Mises stress and coordinate location predicted according to the design shown in part (a) of the figure. The design variables accessible to the user are seen in part (a). For the test shown in Fig. 3.6, parameter values of outer blade radius, inner blade radius, chord at the root, chord at the tip, sweep, lean, and angle were used. The GUI shows the outer radius parameter was set to 10% above nominal and the chord at the root was set to 10% below nominal. All other parameters were set to nominal.

Fig. 3.7 shows a side by side comparison of the FE model as predicted with the surrogate models and the FE model as solved by FEA. The color scale signifies, as standard, that red is high stress and blue is low stress. Both the FEA and the surrogate models show the full von Mises stress field of the Purdue blade with the same parameter values. Both models were set to the color scale produced by the FEA simulations with the maximum stress shown as red and the minimum stress as dark blue. The color scale is discretized such that stresses within a certain range are colored the same. Thus the stresses in the darkest blue region on the FEA simulation are the same stresses in the darkest blue region of the surrogate. Boundaries between the color bands may be viewed as
Figure 3.7: Purdue blade suction side von Mises stress.
lines of constant stress. The maximum and minimum stress locations are indicated by \( X \) and \( O \) respectively. The leading and trailing edges are also indicated by \( LE \) and \( TE \) respectively.

The figure shows that the maximum stress on both models is at the root of the blade centered along the chord. The blade was swept such that the tip was displaced aft a distance of 10% of the chord at the tip. Both the FEA results and the surrogate results showed comparable geometric shape, stress contour shape, and stress magnitudes. The predicted results correctly showed a region of high stress located at the root of the blade at the middle of the chord. This stress decreased towards the tip of the blade and toward the edges. The surrogates, however, show error where they predicted that the stress shape does not decrease as quickly towards the tip as does the FEA. The yellow region, for example, extended from the root to 7.5% of the blade span on the FEA model. On the predicted model, the same region extended to about 15% of the blade span. Towards the tip and edges of the blade, however, the stress shapes matched more closely the results of the FEA. A region of slightly higher stress of bright blue was correctly predicted on the trailing edge that goes from the root to about 15% of the blade span. The surrogates also predicted the small region of stress concentrated at the leading edge of the root. For the example shown in Fig. 3.7 the surrogates predicted a maximum stress 0.26% less than the FEA. Yet, because both models were set to use the color bar produced by the FEA simulation with 12 discretized color bands, the maximum stress of the predicted results still falls in the red color band. The location of the predicted maximum stress is about the same as the predicted location of maximum stress on the FEA result.

As the parameters in the GUI shown by Fig. 3.6 were changed, the surrogate models predicted new values and the new response was then visualized. Designers could use this to explore how their choice of parameter values affected the part’s results around any given design. Fig. 3.8 shows how the stress contours and the geometry were affected by adding sweep to the blade. Fig. 3.8, from part (b) to (c), shows designs with negative sweep exploration while from part (b) to (a) shows the response as positive sweep was explored. The designs were explored between no sweep to a maximum sweep causing displacement at the tip of 10% of the chord. These changes showed that as the section was swept forward, away from the nominal design, the region of higher stress at the root increased. When the design had maximum backward sweep the stress at the root increased by about 15% compared to the nominal design as shown by the yellow region in Fig. 3.8 part (c). In this design, the maximum stress decreased 0.53%. As the tip was swept forward to
maximum sweep, part (a), the same region also increased stress by about 15% while the maximum stress increased by 1%. This region was smaller when the blade was swept forward versus swept back, covering 28% of the root chord instead of the 40% seen by backward sweep. This shows that the backward sweep design had slightly more evenly distributed stress than the forward swept design. It can also be seen from the figures that forward sweep caused a region of low stress to cover up to 25% of the root chord at the trailing edge. This low stress region was not present when the blade was swept backward. It is also shown that the trailing edge stresses decreased with positive sweep and increased with negative sweep while the leading edge stresses were much less

Figure 3.8: Von Mises stress from sweep of Purdue blade pressure side.
affected. When compared to the FEA results, the surrogates at the backward sweep design under predicted a maximum stress by 0.26%, the zero sweep design was over predicted by 0.01%, and the forward swept design maximum stress did not change. The surrogates predicted that the maximum stress moved 1.86% of the chord further from the leading edge with positive sweep while moving 3.64% of the chord further from the leading edge with negative sweep. The error of the predicted maximum stress location in the negative sweep design was 0.26% while the error from the positive sweep design was 1%. Because the error is less than the motion of the maximum stress, the surrogates help visualize trends which allow designers to understand how the design parameters affect the structural response of the compressor blade.

3.4.1 Mesh Morphing

Mesh morphing the Purdue blade provided the meshes necessary for surrogate modeling within a reasonable design space of compressor blades. Fig. 3.9 shows a mesh of the Purdue blade with about 400 nodes. The low node count allows the nodes to be easily seen in the figure. The mesh was morphed from the nominal mesh, part (a), to new designs. Part (b) shows a morph from the nominal case to a blade with forward sweep which caused displacement of 5% of the chord at the tip. The mesh was then morphed in part (c) to a blade with sweep which caused 10% displacement. As shown by the figure, the arrangement of elements in the mesh stayed constant. Each node remained in the same location relative to the geometry of the blade while no nodes were either created or destroyed. This was necessary so that each node would be present, at the
same relative location, through all of the training data, in order to create their respective surrogate models.

If a blade was morphed too far then the resulting mesh could contain failed elements. Using the method of morphing one parameter at a time, none of the morphs contained failed elements within the tested design space. The bounds of this space were: +/-10% of both inner and outer blade radius, a change of +/-20 degrees in blade angle, and +/-10% of the chord to adjust the chord length, the sweep, and the lean of the blade. To further test the limits of the design space, the sweep described in Fig. 3.9 was tested on the 25K node mesh and the 500K mesh. They were able to morph to sweeps that caused displacements up to 200% of the tip chord without failed elements. For the sweep, this represented a reasonably large space over which the design could be thoroughly explored. This means that the mesh morphing was able to provide results within that design space for that parameter.

### 3.4.2 Surrogate Speed

The first step to testing the surrogates ability to provide useful DSE was to evaluate the speed at which the results could be emulated. This was done by performing an analytical evaluation of emulation computational cost, measuring the emulation time, and measuring the time for interactive design evaluation. The analytical evaluation used Big-O evaluation to understand how the time to emulate would scale with certain aspects of the surrogates, such as number of nodes or number of design variables. The emulation time was measured as the wall-clock time needed for the surrogate to emulate a new design. For FEA, this time was the measured as the time needed to solve the analysis. This speed is important to understand the surrogate method’s benefits to non-interactive DSE, such as optimization. The time for interactive design evaluation was the wall-clock time between the user querying a new design and when the result was drawn to the screen. For FEA, this time was measured as the time needed to update the CAD model to the correct variables, morph the mesh, and solve the FEA. This speed is important to understand the surrogate method’s benefits to interactive DSE. The experimental measurements were performed on over 50 randomly selected test cases that compared various numbers of nodes, number of training samples, and parameter combinations. These tests were done for both the node-based
surrogates and PCA-based surrogates and compared to the speeds which are achieved by DSE with FEA. They were performed on a computer with 16 GB of RAM and a CPU speed of 3.7 GHz.

**Big-O Analysis**

The Big-O analysis is a measurement which helps understand how the computational complexity of an algorithm scales with certain inputs to that algorithm. This computational complexity scales the same as the time required for the algorithm to compute. The analysis is performed by counting the number of floating point operations (FLOPS) needed for the calculations. Traditional linear, static FEA, for example, has the complexity shown in Eqn. 3.6. In this equation, $n$ represents the number of nodes in the finite element model. Eqn. 3.6 states that FEA scales between the square and the cube of the $n$. The most basic of FEA algorithms scale by the cube, but the use of more advanced matrix algorithms for FEA inversions has brought this closer to the square. The node-based surrogates have Big-O complexity shown in Eqn. 3.7 in which $s$ represents the number of samples and $d$ represents the number of design variables. Because $s$ and $d$ are usually one or more orders of magnitude lower than $n$, and because this equation shows that the computational complexity scales linearly with $n$, emulation time will likely be much faster than FEA. The improvement in time from between FEA to node and PCA-based surrogates increases as the number of nodes in the model increases. The PCA-based surrogates have Big-O complexity shown in Eqn. 3.8 in which $q$ represents the number of retained principal components. This method also scales better than FEA with respect to the number of nodes, but it is not always better than the node-based surrogate. Because the PCs can not be greater than the number of samples, the complexity of the PC-based surrogate without a truncated set of PCs is close to the complexity of the node-based surrogate. Truncating even just a few PCs will make the scaling of the PC-based surrogate better than the node-based surrogate.

$$O(n^2) \text{ to } O(n^3)$$ \hspace{1cm} (3.6)

$$O(sd + sn)$$ \hspace{1cm} (3.7)
Compute Time

The wall-clock times for computation of FEA, the node-based surrogates, and PCA-based surrogates are shown in Fig. 3.10, Fig. 3.11. The FEA was tested with models from 25,000 to 1,000,000 nodes. The node-based surrogate was tested on design spaces with the same finite element models as the FEA tests, one to nine design variables, and 10 to 100 samples. The PCA-based surrogate was tested over the same ranges as the node-based surrogate with the addition of PCs tested between 10 and 100. The maximum number of PCs is limited to number of samples in the design space. It should be noted, that though the same finite element models were used, design exploration only requires emulation of the surface nodes. About 10% of nodes were on the surface of compressor and fan finite element models. For this reason, the presented figures and equations shows 2,500 to 100,000 nodes for the surrogates. These values, however, reflect the same range of models as those tested with FEA. The FEA simulations in Fig. 3.10 shows a trend line, dotted,

\[ O(sd + sq + nq) \]

(3.8)

Figure 3.10: Wall-clock time required to solve FEA compared to the number of nodes, \( n \).
which confirms that the time required to solve an FEA simulation scales with the square of the number of nodes. The data was fit to a polynomial trend line shown in Eqn. 3.9 which has an $R^2$ value of 0.997.

$$
t = (0.1) \left( \frac{n}{10,000} \right)^2 + (2.0) \frac{n}{10,000} \tag{3.9}
$$

Fig. 3.11 shows that the compute time of the node-based surrogate is mostly related to the number of nodes, $n$, but is also influenced by the number of samples, $s$. The data is fit to a polynomial trend surface shown in Eqn. 3.10 which has an $R^2$ value of 0.98. Each of the estimated coefficients has a $p$-value of less than 0.01. The equation and figures both show that the compute speed of the node-based method can compute the results in real-time for the tested training sets.

Fig. 3.11 also shows how the wall-clock time required for computation of the PCA-based surrogate is mostly related to $n$, but also changed with the number of principal components, $q$. The data is fit to a polynomial trend surface shown in Eqn. 3.11 which has an $R^2$ value of 0.98. Similar trends to those found with node-based surrogates are found with the PCA-based surrogates. The figures and equations also show that the compute time for the PCA-based surrogates is slightly higher than the node-based surrogates when all possible PCs are included, but faster than the node-based surrogate when the number of PCs is lowered. Tab. 3.1 gives compute times for an example model based on the fits in Eqn. 3.9, Eqn. 3.10 and Eqn. 3.11. As shown in the table, both surrogates show vast

![Figure 3.11: Wall-clock time required to evaluate surrogates compared to the number of nodes, $n$.](image-url)
Table 3.1: Computation speed example for model with 100,000 nodes, or 10,000 surface nodes, 100 samples, and 10 retained PCs

<table>
<thead>
<tr>
<th>Method</th>
<th>Computation Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEA</td>
<td>30.0</td>
</tr>
<tr>
<td>Node-Based Surrogate</td>
<td>0.00229</td>
</tr>
<tr>
<td>PCA-Based Surrogate</td>
<td>0.00112</td>
</tr>
</tbody>
</table>

improvement in solution time over performing FEA. The PCA-based surrogate, when retaining only 10 of the 100 PCs is twice as fast as the node-based surrogate.

\[
t = (1.0 \times 10^{-5}) \frac{n}{10,000} s + (6.9 \times 10^{-4}) \frac{n}{10,000} + (6.0 \times 10^{-6}) s \tag{3.10}
\]

\[
t = (1.2 \times 10^{-5}) \frac{n}{10,000} q + (9.0 \times 10^{-4}) \frac{n}{10,000} + (1.0 \times 10^{-5}) q \tag{3.11}
\]

Interactive Design Evaluation Time

Interactive DSE performance profiling revealed that between 90 and 95% of the interactive design evaluation time was spent passing the stress colors and node coordinates to the graphics card. Thus, because the surrogate methods developed in this have such small compute times, further algorithm improvements have little effect on interactive design evaluation speed. Significant performance improvement may be achieved only by obtaining better hardware which is not the purpose of this research. The only factor which effects the time required to pass the results to the graphics card is \(n\). Fig. 3.12 and Fig. 3.13 are used to better understand how \(n\) affects this time for both FEA and the surrogate models. Fig. 3.12 shows how the FEA computation times, the time required to explore a design space by using FEA, is quadratic. The times are almost all above about a minute, with the higher fidelity models showing times of up to thirty minutes. The fit quadratic equation is shown in Eqn. 3.12 and has parameter estimates with p-values of less than 0.0001. While these times may not seem very high, delays in obtaining results negatively impact how many design may be checked and the quality of the final design which is found.
Figure 3.12: Wall-clock time required for interactive design evaluation of FEA compared to $n$.

$$
t = (0.1) \left( \frac{n}{10,000} \right)^2 + (4) \frac{n}{10,000}
$$

(3.12)

Fig. 3.13 shows the how the times for surrogate emulation change with $n$. The node-based surrogates and PCA-based surrogates are grouped together in this figure due to the relatively small difference between the methods when considering interactive design evaluation time. While the values of $n$ here are about an order of magnitude less than those shown in Fig. 3.12, they are directly comparable. The finite element models emulated in Fig. 3.13 were the same solved with FEA in Fig. 3.12. Only the nodes which are on the surface of the blades are useful for DSE, therefore about 90% are neglected. These interactive design evaluation times in Fig. 3.13 show that near real-time DSE may be performed by using surrogates of the finite element models. The average interactive time for the model with 9,000 surface nodes, or 25,000 total nodes, is between 0.02 and 0.03 seconds. This means that, for this model, the time needed to emulate and visualize is close to real time. As the number of nodes increased the time needed also increased, but still stays low. Emulating a model with 12,000 surface nodes, or a 1,000,000 total nodes, only takes about 0.2 seconds. This is a large improvement over the 30 minutes needed for exploring with...
Fig. 3.13 also shows that the time did not increase quadratically as did the interactive FEA speeds. This means that the higher fidelities had better interactive time improvement by changing to surrogate-based design space exploration.

\[ t = (0.0159) \frac{n}{10,000} \]  

(3.13)

3.4.3 Surrogate Error

The second step to testing the surrogates ability to provide useful DSE was to evaluate the error of the surrogates. The true value was taken to be the FEA model which the surrogate was seeking to emulate. Lower error equates to surrogates which better approximate the finite element solution. In turn, this provides confidence to the designer that the solutions being explored accurately represent the models they use to analyze the designs. The tests for this chapter were carried out on the Purdue blade finite element model with 25,000 nodes. This model was used due to no significant change in error being detected for models with more nodes. The error tests sought to understand the magnitude of error and how error changed with varying \( s \) and \( d \). Because emulation was found to be near real-time speeds and interactive speeds were found to mostly be
Randomly select design variables

Use DOE to select training sets by varying $s$ and $d$

Create a training set

Create a separate testing set

Get error between surrogate stress ($\hat{\sigma}$) and test stress ($\sigma$)

Use surrogate to emulate stress at every test samples

Create surrogate from training set

Figure 3.14: Error testing procedure for the nodal surrogate models.

influenced by the number of nodes, no truncation of PCs was used for this error testing. This truncation was seen as unnecessary for interactive DSE. However, the effect of truncating $q$ will be shown later.

The error testing procedure for a single training set is described in Fig. 3.14. As shown in the first row, a DOE was used to randomly select training sets with varying number $s$, and $d$. Using a DOE ensured that the selected training sets covered the set of reasonable training sets performed by industry. The second and third rows in the figure were performed for each of the selected training sets. The specific variables for each training set were randomly selected. Then, the surrogate model training set was created by the same method described in Fig. 3.5. A testing set was created independent of, but with the same design space as the corresponding training set. The test set used 50 samples which were not used to train the surrogate. A surrogate created from the training set then was used to emulate the stress at the designs in the test test. The error of stress between the test samples and the emulated results was measured by using Eqn. 3.14.

Double normalized root-mean square error (double-NRMSE) was developed and used to measure the error of the surrogate model. It is labeled as double because it must evaluate NRMSE across all nodes of each sample to obtain the error for that sample and then evaluate the error over all of the samples in the design space. The equation for double NRMSE is given in Eqn. 3.14. In this equation $d$ represents the total number of design variables, $n$ represents the total number of nodes, $\hat{\sigma}_{i,j}$ represents the estimated stress for node $i$ at design $j$, $\sigma_{i,j}$ represents the true stress for node $i$ at design $j$, and $\sigma_{max}^2$ represents the maximum stress across all nodes and samples.
NRMSE was chosen as the base for this equation due to the academic and industry acceptance of the equation, as well as its ability to provide a single value to represent error.

\[ e = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{d} (\hat{\sigma}_{i,j} - \sigma_{i,j})^2}{nd\sigma_{\text{max}}^2}} \] 

(3.14)

The errors for the node-based surrogates and PCA-based surrogates are shown in Fig. 3.15a and Fig. 3.15b respectively. These figures show how the error of the surrogates changes with respect to \( s \) and \( d \). These figures show that there is noise in the emulation error across both \( s \) and \( d \). This noise comes from the differing effects each design variable, and their varying interactions, has on error. To provide evidence of this theory, double-NRMSE for several selected design spaces are plotted in Fig. 3.16. The error, shown on a logarithmic scale, shows that each design space has a general decrease in error as \( s \) increases. This decrease is logarithmic. The next item to be learned from Fig. 3.15 is that the node-based surrogates and PCA-based surrogates have very similar errors. Even considering the noise due to varying design variable selection, the error for any given training set was similar between the two methods. The errors, in general, decrease with \( s \) and increase with \( d \). These trends may also be seen in Fig. 3.16. Both the node-based and PCA-based show similar
errors. One noticeable difference is that the design space of $H$ is better with PCA-based surrogates and the design space of $L$ is better with node-based surrogates. Generally, however, both methods had similar error. A least squares regression was performed on the log of both errors. A paired $t$-test was performed between the errors obtained from the two methods. This test showed that there was no statistically significant difference between the errors from the node-based surrogates and the PCA-based surrogates. This conclusion was based a $p$-value of 0.29 when testing for a difference in the mean error between the two methods. It can also be seen that, generally, design spaces with more variables have higher error. Lastly, the discussed figures show that the tested surrogates provide errors of 10% or less over the tested design spaces. Most of the tests, those with a more sufficient number of samples, showed error of about 1%.

### 3.4.4 PCA-based Error and Speed Trade-off

The number of principal components retained, $q$, has an effect both on computation speed and error. While this effect was not tested in the previous section due to the small effect excluding PCs has on interactive DSE speed, these trends are still worth noting. Several design spaces were tested measuring the compute time and double-NRMSE with varying $q$. One of these is shown in Fig. 3.17. This design space had 76 samples and four design variables. The figures show a nearly linear relationship between the computation time and $q$. The double-NRMSE, however, decreases for the first few PCs but then levels off after just the first few PCs. This means that the computation time can be reduced by near an order of magnitude, for some design spaces, without a significant
Figure 3.17: Double-NRMSE and time to compute values for a design space with 76 samples and four design variables with varying $q$. The variables are $C_R$, $C_T$, $P$, and $H$.

increase in the emulation error. Other design spaces with varying numbers of design variables and samples had similar trends. The example in Tab. 3.1 would achieve a PCA-based surrogate which was twice as fast, with no significant increase in error. The number of PCs needed to be retained for this savings will vary per design space but is related to the explained variance. For the shown examples, if 99.99% of the explained variance is retained then the lowest possible error is achieved with just a few PCs. Using 99.99% explained variance for the design space in Fig. 3.17 would result in using just the first twelve PCs. This would result in an emulator which computes in about 0.0003 seconds and has an error of about 0.15% double-NRMSE. This ability to trade off between speed and error is helpful when large numbers of non-interactive emulations are needed, such as optimization processes.

3.5 Discussion

From the results shown by the speed testing, interactive DSE could be performed with near real-time visualized solutions. The 25K node count used in the research is representative of some compressor blade meshes used in industry. For an engineer, this means that exploring how the
interaction between sweep and blade angle affects the stresses at the root of a blade can take just a few seconds as they check various designs. DSE of meshes with larger node counts also benefit greatly from this method. With a reduction in solution time by three to five orders of magnitude, structural iteration time, though not sub-second, is significantly shortened. As the time to predict and visualize designs decreases, the number of designs that may be explored increases. This leads to a better understood and optimized design.

While the study was performed specifically with the Purdue compressor blade, similar results may be expected on other parts as well. Different parts will vary in the percentage of nodes that are located on the surface of the FE model. This will change the number of surrogate models and in turn the time it takes to solve all of them. The number of training samples that are required for accuracy will also change based on how the different parameters will affect the response being predicted. Parts similar to the Purdue blade, such as fan blades or turbine blades, should see speed and accuracy results similar to those presented on the Purdue compressor blade.

3.5.1 Improvement of Speed

Further speed improvement may be gained by only emulating the nodal stresses and not the nodal locations. The coordinates for the baseline design may be used for visualization of the FE model. This would remove Walther et al.’s contribution to the method. However, because prediction of the nodal locations requires three surrogates per node, this can cut down the interactive time for the surrogates to predict by 75%. Such a method should be used with caution because it may limit the user’s understanding of the design space by not showing geometric changes. It is a viable option, however, when the benefit of faster DSE outweighs the benefit to seeing the geometric changes. This may be the case with compressor blades, depending on which parameters are being explored. If the maximum thickness along the chord, for example, is increased then both the stress shapes and magnitudes will vary while the geometric changes will be very small. Thus, by neglecting the geometric changes, the interactive speed may be real-time, or near real-time, for models with more nodes.
3.5.2 Achieving Accurate Surrogates

The results showed that each parameter, and combination of parameters, has a different effect upon the metamodel efficiency. A general trend seen is that as the number of dimensions increases so does the number of required samples. This increase is not a linear correlation but a power correlation. This phenomenon is known as the curse of dimensionality [83]. However, this does not scale perfectly. Because each parameter has a different effect on metamodel efficiency, predicting the number of samples needed for a given accuracy based solely on the given parameters is difficult. The interactions between the parameters also impede understanding of how many samples are needed even if the number is known for each parameter individually. The interactions between any given set of parameters may cause more or fewer training samples to be needed. These complex interactions may also be experienced with mesh morphing. Morphs across different combinations of parameters could cause element failure at different parameter values. This could change the extents of the design space. Because of these complex interactions, care should be taken to track which designs cause failed elements during the morphing process.

One possible explanation for the variation of accuracy among parameters could be how each parameter affects the overall geometry of the blade. Because each of the blade sections were adjusted by their percent span along the blade, each parameter changed every element in the FE model to some degree. For instance, if the chord at just the blade root was included in the design space then the chord at all cross-sections were adjusted some percentage of the total change in the chord at the root even though chord at the root was the only design parameter.

In order to help achieve a certain accuracy with such complex interactions, adaptive sampling may be used to provide the least amount of samples necessary. For many cases, the FEA will be solved using supercomputers. A DOE with fewer samples than thought necessary may be created and their FEA run on the supercomputers. When the supercomputers return results, the error of the model may be evaluated. If the model does not have the required accuracy, more points may be sampled where the model is least accurate [81].
3.6 Conclusions

The results of this research show that interactive DSE may be performed on turbomachinery compressor blades. By using either node or PCA-based surrogates the full structural response and geometry of the blade may be predicted and visualized very quickly. The node-based surrogates were trained on a set of stresses and coordinate positions at all of the nodes. The PCA-based surrogates were trained on a set of PC scores which were obtained by performing PCA on the stresses and coordinate positions of the nodes. Training data for the surrogates relied on similar meshes across all FE models. This was accomplished with mesh morphing. The novelty of the method came by morphing across the entire design space to allow surrogate modeling of both stress contours and geometry. This allowed for accurate surrogates with fewer training samples than previous methods which partitioned the design space into morphing regions [39]. Partitioning with morphing regions allows more drastic geometric changes but required duplicate samples at the boundaries between the regions. By being able to morph across the entire design space the number of needed samples was lowered. It was found that the solution and visualization time of this method depends first on how many nodes are being predicted and second on the number of parameters in the design space. The time for prediction and visualization may be given by Eqn. 3.13. This equation is accurate for one to ten parameters and 25K to 1M nodes.

It was found that a design space based on a model with 25K nodes may be predicted in less than 0.5 seconds. This allows the designer to see and visually gain understanding of how those two parameters affect the full system response of the compressor blade. Designers may interact with the surrogates to visualize the structural trends in their designs. If more parameters are desired, then the prediction will take longer. Regardless of the number of nodes, however, it was found that, on average, the prediction of the solution required three to five orders of magnitude less time than a full simulation. This means that, even if the solution is not solving real-time, the exploration of the design space is more feasible.

Visual exploration of the design space is crucial to compressor blade design. This research provided methods for visual design space exploration in order to provide a design engineer with increased understanding of how their design decisions affect the full solution response of a compressor blade. Such a complete understanding allows for faster structural iterations and better performance of the final design.
CHAPTER 4. DESIGN SPACE EXPLORATION METHOD BASED ON PCA

The objective of this chapter was to develop processes to explore the structural design space of turbomachinery blades using PCA. The previous chapter studied improving the DSE speed for turbomachinery blades by using surrogates to emulate the blade stress [62]. This chapter focuses on methods to understand the relationship between design variables and structural results. This addresses the challenge of helping designers understand which design variables and regions of the design space should be explored. The content of this chapter was submitted as a paper to the Journal of Computing and Information Science in Engineering with the title, “Structural Design Space Exploration Using Principal Component Analysis”. This chapter will contribute to the body of research as follows.

• Develop methods to relate design variables to PC scores.

• Develop a workflow to use PCA for DSE.

• Demonstrate that the developed methods can be used to achieve a desired structural response.

4.1 Introduction

Design space exploration (DSE) is the study or analysis of designs defined by a set of variables which are permitted to vary. When done well, it can provide powerful insight into the influence of design variables on the local and global results. Optimization is a form of DSE which seeks to find a maximum or minimum objective in a design space subject to constraints. DSE is a broader concept which seeks to understand and explore the design space. It helps find optimums, understand sensitivities, and explore new design concepts. This form of exploration is useful for many types of design problems. One application is the multidisciplinary design of turbomachinery compressor blades. A geometry optimized aerodynamically may not be structurally feasible.
This requires structural DSE around a proposed aerodynamic design to find a structurally feasible design.

A challenge of exploring the structural design space of compressor blades is knowing which design variables to vary, and by how much, to satisfy structural criteria. Compressor blade structural design often involves many iterations of trying different design variations, based on a proposed aerodynamic geometry, to find one which is structurally feasible. The structural and aerodynamic groups iterate together with each group also performing sub-iterations. This design method has a twofold problem which limits the ability to find quality, feasible designs. First, the long solution times required for high fidelity simulations limit the number of designs which can be explored. Second, a lack of understanding about how design variables influence compressor blade structural results obscures a designer’s knowledge of which design variables to vary and which regions of the design space to explore. This may lead to a larger percentage of simulations being wasted on designs which do not improve the compressor blade design objectives. Previous research studied improving the DSE speed for compressor blades by using surrogates to emulate the blade stress [37, 62]. The present research focuses on methods to understand the relationship between design variables and structural results to improve DSE. This will help designers understand which design variables and regions of the design space should be explored.

This research uses principal component analysis (PCA) to develop methods which improve understanding of compressor blade structural design spaces. PCA is also known in the field of turbomachinery and CFD as proper orthogonal decomposition (POD) [41–43]. It provides the orthogonal variations in the structural data and scores which represent the magnitude of those variations for each sample. The orthogonality of the PCs means that the variations are the most efficient set possible from the given data. While this paper uses PCA, other latent or spectral methods may also be used if they provide a set of orthogonal variations and the magnitude of each variation for each sample. Other such acceptable methods include exploratory factor analysis or least squares regression [84, 85]. Latent methods which would not work include hidden Markov models, Isomap, or other non-linear techniques because they do not provide orthogonal vectors which represent the variations [86]. PCA was selected due to its simplicity and given that the transformation preserves as much of the variability as possible within the first few PCs [9]. For DSE of compressor blades, the samples are design variations from a baseline compressor blade
and the data is structural results from finite element analysis (FEA). Two methods are used to understand the relationship between design variables and PCs for high dimensional spaces. The first method applies two-point correlation between design variables and PC scores. The second method was developed in this research from coupling the node locations and node stresses in a single PCA. These methods are applied to DSE of the von Mises stress on blade-alone finite element models of the Transonic Purdue compressor [56]. This compressor blade was developed to study transonic aerodynamic effects, but has been the subject of many academic studies including structural PCA analysis [57]. The nominal design has a height of about 2.0 in, a mean chord of about 1.9 in, and a mean thickness-to-chord ratio is about 8%.

The paper proceeds with a background on DSE and PCA of result fields. Then the methods for PCA and the two-point correlation are described. Following this, the results of the PCA, two-point correlations, and stress and geometry coupling are shown. The results show PCs of the compressor blade across various design spaces and how the stated methods relate the variables to the PCs. The discussion then provides an example to demonstrate how to use these methods to improve compressor blade DSE.

4.2 Background

The background covers deficiencies to common practices of compressor blade DSE, the basics of PCA, and methods which have been used to better understand turbomachinery with PCA.

4.2.1 Design Exploration

DSE of compressor blades is a complicated and computationally expensive process. It is common to start from a baseline design and rely on experience to adjust design variables until a desired result is obtained. For structural design of compressor blades, each iteration requires FEA with possibly millions of nodes [19]. The long simulation times and computational expense which accompany these high fidelity models limit the number of designs which may be tested. Research has shown that, during design exploration, longer times to reach an acceptable design lead to worse final designs [4–6]. This means that the simulation times should be as low as possible and that the designs which are tested should be chosen well. Knowing which designs to test can be challenging
without an understanding of how each design variable influences the structural results. While engineering experience can help with this, that understanding is limited in unfamiliar or complex design spaces.

More advanced design exploration methods use a design of experiments (DOE) to choose which designs to analyze. A DOE, also known as a sampling plan, seeks to select a set of designs which are, in some sense, evenly distributed through the design space [8]. Analysis from the selected samples can help understand global trends across the design space. Borer et al. and Huang et al. [87, 88] performed DSE by using a Latin hypercube DOE to understand how performance was related to specific propeller and aerospace engine design characteristics. After selecting and analyzing the samples, they plotted their performance parameters with their design parameters to understand the global trends and relationships in propeller and engine design variables. They were only exploring results based on single values. Many have also used DOEs and fit surrogate models to emulate the response fields of structural parts [37–39, 62]. Their methods allowed for real-time design queries, but lacked the ability to relate the design variables to the global response due to the response being represented by fields instead of single values.

4.2.2 Principal Component Analysis

PCA is a spectral method which creates a reduced order model from data with a large number of dimensions while retaining as much information as possible [9, 75]. It finds the orthogonal variations, or principal components (PC), which explain the most variation in the data. Because they are orthogonal, each PC is independent from one another. By using only a few PCs which explain the most variation, the data can be more quickly analyzed and understood. Originally developed for statistics, it is used in other fields, often by different names and with slight variations. Despite these differences, the governing principles are the same. This research followed the basic method presented by Jolliffe et al. [9]. PCA is used in aerospace engineering for many reasons, including; real-time analysis, data size reduction, control of the trade-off between error and computation cost, and improved understanding of characteristics in the data [44]. Hajikolaei et al., used PCA to reduce high dimensional design spaces to design spaces of lower dimensions [89]. They found that by using just a few PCs they could reconstruct the results from the original high dimen-
Table 4.1: Set of possible design variables for Transonic Purdue blade used for this research.

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Symbol</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>H</td>
<td>± 10% of nominal H</td>
</tr>
<tr>
<td>Ave. Radius</td>
<td>( R_o )</td>
<td>± 10% of nominal ( R_o )</td>
</tr>
<tr>
<td>Root Chord</td>
<td>( C_R )</td>
<td>± 10% of nominal ( C_R )</td>
</tr>
<tr>
<td>Tip Chord</td>
<td>( C_T )</td>
<td>± 10% of nominal ( C_T )</td>
</tr>
<tr>
<td>Sweep</td>
<td>( S )</td>
<td>± 10% of nominal ( C_T )</td>
</tr>
<tr>
<td>Lean</td>
<td>( L )</td>
<td>± 10% 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>

...sional space with less than 5% relative average absolute error. This low error provides confidence that just the first few PCs indicate meaningful variations through the design space.

PCA has been applied to better understand and explore design spaces based on flow fields, heat transfer, and geometric tolerances of compressor blades, turbine blades, and other parts. [41, 45, 76]. Blanc et al. used PCA to reduce the set of nodal temperatures on a turbine blade finite element model to a small set of PCs [45]. These represented the orthogonal variations of the temperature field on the blade. The full temperature field was a linear combination of the PCs with their respective scores. The PCs provided better understanding of the characteristics of this temperature field. Spencer et al. used PCs to understand unique characteristics about flow fields [76]. They were able to better understand specific flow characteristics and relate them to the simulation’s inlet conditions. Brown et al. used PCA to get geometric variations in as-manufactured compressor blades and used the scores to emulate blade mode shapes and frequencies [77]. Using the PC scores as inputs to the model allowed them to predict how changes in geometric variation affected the frequencies and modes. Yan et al. compared methods to understand spatial variation of parts [75]. They found that, among these methods, PCA performed the best at displaying and understanding the spatial variations of manufacturing defects. These studies used PCA as an analysis tool to better understand flow and temperature characteristics. The present research shows how similar PCA methods may be used as a DSE tool when coupled with methods which relate the design variables to PCs.
4.3 Method

PCA was used to develop methods to understand variations in the von Mises stress of compressor blade design spaces. This allowed global trends to be discovered when exploring result fields. The tested design spaces were made up of various combinations of the design variables shown in Tab. 4.1 for the Transonic Purdue compressor blade. The table is made up of nine design variables in which the first and second columns have the names and symbols, respectively, of the design variables. The third column gives the range that each variable may deviate from its nominal value. Methods similar to Blanc et al., as described in the background, were applied to reduce the nodal results for sampled designs into the PCs [45]. These nodal results, which together represented the spatial distribution of von Mises stress in the samples, were reduced into PCs which described variations of spatial stress distribution among those samples. The spatial distributions describe the how the stress varies over the surface of the blade. The PCs describe how that spatial distribution varies across the design space. This research, however, coupled nodal von Mises stress and nodal coordinates together in PCA. These PCs represented the orthogonal variations which coupled stress and geometry variations present among the samples. The orthogonality of the PCs means that variations are independent of each other such that a change in the score of one PC does not necessitate the change in score of another PC. This property is helpful when exploring the design space and isolating the effect of design variables upon the stress of the blade. The full spatial stress field and geometry for a sample in the design space could be reconstructed by linear combination of the PCs.

The general workflow to create the data sets and model is shown in Fig. 4.1 where rectangles represent numerical processes and cylinders represent data. To train the model, data sets for the stresses and coordinates of each node on the finite element model were obtained for a number of design variations from the following steps. The first step was to choose which design space to explore and then select samples from that design space for simulation. This required selecting the design variables and creating a design of experiments specifying the designs to sample. Latin Hyper-sampling with a maximin criterion was used to select samples spread across the design space. Then, a finite element model was mesh morphed to match the designs. This mesh morph was performed to keep the nodes in the same location, relative to the geometry, among all of the samples, which was necessary so that the stress at each point on the part could be recorded for each
Figure 4.1: Flow chart for PCA of finite element models. Rectangular boxes are processes while cylinders represent data.

Sample Data
Select Design Variables to Explore
Select Samples in Design Space
Create / Solve Finite Element Models
Stress, X, Y, Z Location for All Nodes

Create Model
Assemble Data into Matrix X
Perform PCA
Principal Components
Principal Component Scores

Sample. The finite element model of the Transonic Purdue compressor blade used in this study had 25,000 nodes with a fixed constraint at the blade root, a uniform pressure load, \( P \), applied to the pressure surface, and a rotational load, \( N \). Next, FEA was used to solve for the stresses of the finite element model. Upon finishing FEA, the stress and location of each node was saved into a data set. The data sets used in this research contained stresses and coordinates for 25,000 nodes and up to 100 samples. The stresses had values up to 72 ksi, with an average near 15 ksi. The coordinates had values between 0 to 6 inches.

4.3.1 Principal Component Analysis

The stress and coordinate data at the nodes was reduced through PCA. While the PCs of the stress show variation of the stress through the design space, the PCs of the coordinates were used to relate stress variations to geometric variations. The reduction was performed on the data matrix, \( X \), shown in Eqn. 4.1 and assembled as shown in Fig. 4.2. Each of \( n \) rows in the matrix represented a finite element model from one sample. The columns, of which there were \( p \), represented the nodes of the finite element model. The first quarter of the columns were stresses at the nodes and the remaining columns were coordinates at the nodes. This meant that \( p \) was equal to four times the number of nodes, \( m \). For these tests \( m \) was 25,000 and \( n \) was between 50 and 100. Only nodes
Figure 4.2: The \((n, p)\) data matrix, \(X\). Each row was a unique finite element model sample, of which there were \(n\). The columns, of which there were \(p\), represented the \(m\) nodes on the finite element model.

on the surface of the finite element model were used to decrease the computation while retaining all visible node information useful in design space exploration. Common practice for structural design of compressor blades only considers stress on the surface.

\[
X - \bar{x} = USV = AV
\]  \hspace{1cm} (4.1)

The PCA method used to decompose the data matrix is thoroughly explained by Jolliffe et al. [9]. This research used a data matrix with the data centered by subtracting the column means, \(\bar{x}\). The result of PCA on the data matrix was twofold. First, a matrix, \(V\), containing the PCs as rows. Next, a matrix, \(U\), and a diagonal matrix, \(S\). When these two latter matrices were multiplied they return a matrix, \(A\), representing the scores of each PC for each sample as shown in Fig. 4.3. PCA finds the matrices \(A\) and \(V\) that satisfy Eqn. 4.1. A reverse transform may be used to obtain the nodal stress and coordinate data, \(x\), for a design. This is shown in Eqn. 4.2 where \(a\) is a vector containing a single score for each PC.

\[
x = aV + \bar{x}
\]  \hspace{1cm} (4.2)
The two-point correlation was performed between the design variables and PC scores. The correlation for every combination of PC and design variable was obtained by Eqn. 4.3. In this equation, $d_i$ is a vector of design variable values for the $i^{th}$ design variable and $pc_j$ is a vector of PC scores for the $j^{th}$ PC. The elements in the vectors are from the samples used in the PCA. The value of the correlation between the $i^{th}$ design variable and the $j^{th}$ PC is $c_{i,j}$ and is used to understand the relationship between that design variable and PC. Higher correlation values indicate a better relationship between the variable and PC.

$$c_{i,j} = d_i \cdot pc_j \quad (4.3)$$

4.4 Results

Several design spaces of the Transonic Purdue compressor blade were tested to show how two-point correlation and stress and geometry coupling improve DSE. The first few PCs and mean stress from a design space with all nine design variables are shown in Fig. 4.4. The number of samples, $n$, was 100 in this example. The upper color scale in the figure is for the stress in Fig. 4.4(b) and uses the standard color palette in understanding the stress distribution for FEA post processing. Fig. 4.4(c)-4.4(e) use the lower color scale in the figure to show the stress variation of
Figure 4.4: Mean stress and three PCs for design space of nine variables in Tab. 4.1. Image b) is scaled typical to FEA, using the upper color scale with units of psi. Images c) - e) are PCs which represent normalized stress variation and use the lower color scale. The variations are normalized by dividing each node’s value by the maximum absolute stress variation.

The PCs. Red shows increase of stress and blue shows decrease while white shows no variation. Each PC in the figure shows normalized stress variation obtained by dividing the variation by the maximum absolute stress variation of that PC. Each PC is normalized to its own variation so that each PC may be clearly seen and understood. All other PC figures will use this same scale. The stress contours for any design within the nine dimensional design space may be reconstructed by a linear combination of these PCs and the mean stress. The scores associated with each design indicate the coefficient for the linear combination. The figure shows the pressure surface of the blade with the leading edge of the blade on the right. The mean stress profile among the samples, shown in Fig. 4.4(b), shows the highest stress of about 34,500 psi at the bottom corners of the blade.
Figure 4.5: Scree plot of PCs explaining 99.9% of variation in design space of nine variables in Tab. 4.1. Bars are each PCs explained variance and dashed line is cumulative.

and high stress in the center of the blade reaching up to about 25,000 psi. PC-1 in Fig. 4.4(c) shows a stress increase at the trailing and leading edge, especially at the bottom stress concentrators. It also has decreased stress in the upper right and lower left regions of the blade. PC-2 in Fig. 4.4(d) displays an area of large stress increase mid-way between the leading and trailing edges from the bottom to about 60% up the blade with gradual reduced variation away from this region. It also shows decrease in stress along the leading and trailing edges. PC-3 in Fig. 4.4(e) shows alternating increase and decrease along the chord of the blade.

The explained variance, or percent of total variation in the data each PC describes, decreases with each consecutive PC. This is shown with a scree plot in Fig. 4.5 where bars are the explained variance of each PC and the dashed line is the cumulative explained variation. The figure shows PC-1 explains close to 70% of the variation in the stress contours across the given design space. This is followed by 20% with PC-2 while the others are less than 10%. The first three PCs explain about 95% of the variation among the samples.

4.4.1 Low Dimensional Relationships

To use the PCs as a DSE tool, a designer must understand the relationship between the PC scores and the design variables. For design spaces with one variable, every sample’s PC scores can be plotted against the variable as shown in Fig. 4.6. This shows the PC scores for the Purdue blade with a design space which only varies $C_R$ and where the number of samples, $n$ was 50. As this
variable increases the score of PC-1 increases significantly. This means that the variation of stress shown in Fig. 4.7(a) is increased in the full stress field of the design as $C_R$ increases. While the score of PC-2 increases in either direction as $C_R$ moves away from a value of one, this change is much smaller than that of PC-1. The amount each PC score may vary in the design space is related to the explained variance of the PC. Over this design space, PC-1 explains most of the variation causing its score to change more than the other PCs. According to Fig. 4.6, as $C_R$ moves away from a value of 1.0 the full stress profile of the blade changes in small amounts according to the variation shown with PC-2 in Fig. 4.7(b). Using this figure and Fig. 4.6, a designer can know how changing $C_R$ increases or decreases the stress at the regions described by the PCs.

Design spaces with two variables can use three dimensional plots to show the scores across the design space. Fig. 4.8 is similar to Fig. 4.6 but extended to three dimensions to allow two design variables. Because of interaction between design variables, the scores do not line up as nicely with respect to any one design variable as in Fig. 4.6. Plotting the scores against only one dimension shows more noise and obscures the effect of other variable on the score.

Fig. 4.8(a) shows a three dimensional plot from the $\alpha$ axis. The number of samples, $n$, for this example was 100. This perspective shows that PC-1 decreases with increasing $\alpha$ and PC-2 increases as $\alpha$ is moved away from zero. As $\alpha$ increases from -20 to about -5 the contribution

Figure 4.6: Scores for all PCs explaining 99.9% of the variation across a design space of $C_R$. 

![Graph showing scores for all PCs explaining 99.9% of the variation across a design space of $C_R$.]
of PC-1 and PC-2 to the stress distribution decreases. As $\alpha$ increases beyond -5, PC-1 continues to decrease while PC-2 increases. Because PC-3 has no noticeable trend across the range of $\alpha$, this design variable can likely be changed without influencing that PC. Fig. 4.8(b) shows the plot from the view of the $C_T$ axis. Due to the larger scatter of data in PC-1 and PC-2, the image shows that they do not correlate as much with $C_T$ as they do with $\alpha$. The scores for PC-3 exhibit a tighter grouping with respect to this variable than with $\alpha$. This shows that this variable correlates more with $C_T$ than with $\alpha$. As such plots are used interactively, designers can view the plot from different perspectives to understand how PC scores change across the design space of two design variables. This can help them know which design variables to change and which regions of the design space to explore to reduce the stress at areas of the blade indicated by the PCs.

**4.4.2 High Dimensional Relationships**

Understanding these relationships when a design space has more than two variables is more difficult. Plots similar to Fig. 4.6 and Fig. 4.8 can be created if the designer can find one or two design variables which have the greatest influence on the PC scores. Discovering these variables is difficult and there may not be only two which clearly relate to the PC scores. This section discusses two novel methods to find the variables which most relate to the PC scores.
Figure 4.8: Scores for the first three PCs of the variation across a design space of $C_T$ and $\alpha$. 
4.4.3 Two-Point Correlation

The first method applies two-point correlation to relate the design variables to the PCs for DSE of compressor blades. Fig. 4.9 shows correlation scores with PCs along the x-axis and Fig. 4.10 shows scores with design variables along the x-axis. These figures use the same design space and training samples discussed with Fig. 4.4. Both figures show correlation between the PCs and the design variables but with different groupings. It is more intuitive to find which variables relate to each PC in Fig. 4.9 and more intuitive to find which PCs relate to design variable in Fig. 4.10. \( L \) is shown to have the largest effect on PC-1 in Fig. 4.9. \( C_R \) has the next largest effect upon PC-1 but its influence upon the component is much less than \( L \). While PC-2 does have some correlation with \( L \), it doesn’t have a single design variable with which it correlates significantly more than the others. Fig. 4.9 may show the correlation with \( L \) and multiple PCs, but reading and comparing the correlations of multiple PCs with \( L \) in the figure can be difficult. Fig. 4.10 helps visualize the correlation better from this view point. This shows that as \( L \) is changed there is a significant change in PC-2, but not as much as PC-1.
Figure 4.10: Correlation scores for the first four PCs of the nine dimensional design space. Design variables are along the x-axis while the PCs are colored bars.

4.4.4 Stress and Geometry Coupling

The second method to understand the relationship between the PCs of the stress and the design variables is to consider the geometric portion of the PCs in a stress and geometry coupled PCA. Because the node locations were coupled with the node stress for PCA, the PCs contain variations not only of stress, but also of geometry. However, because the magnitude of the values which represent the stress are several orders of magnitude greater than the values which represent the geometry, the PCA is weighted towards the stress. The stress portions of the PCs in the coupled analysis are nearly identical to the stress components when stress and location are segregated. The geometric portions of the coupled PCs are only the geometric variations which align with the stress variations. Fig. 4.4 shows only the stress portion of the PCs contoured onto a blade of nominal geometry. The geometric portion of the component, however, may also be graphically displayed along with the stress contour. It is easier to understand how the design variables relate to the geometric variations because the variables directly control the geometry. These geometric portions of the PC can be used to know which design variables corresponded to the PCs.

Fig. 4.11 displays the first four PCs of the Purdue blade with the nine dimensional design space. The mesh is the nominal blade geometry while the solid, contoured geometry shows the
Figure 4.11: First four PCs with geometric variation over design space of all nine design variables shown in Tab. 4.1. Mesh shows nominal geometry. The displacements of the geometry are scaled in each image such that the shape of the variation is visible.
geometric variation. The colors still represent the normalized stress portion of the PCs. Fig. 4.11(a) shows PC-1 has a geometric variation which includes mostly $L$. This is determined by visual inspection. The solid geometry leans back from the mesh perpendicular to the blade faces. This is how $L$ changes the geometry. While there is geometric variation shown for PC-2 in Fig. 4.11(b), no single design variable from Tab. 4.1 describes the majority of this variation. PC-3 and PC-4 show twist of the solid geometry around its vertical axis. This twist is caused by a change in $\alpha$. These all agree with the two-point correlations shown in Fig. 4.9. This method, however, also provides an indication to directionality that the two-point correlation does not. Fig. 4.11 show that the score of PC-1 increases with negative $L$. This means that as $L$ is changed such that the blade leans back away from nominal the stress will change by an increase of the variation described by PC-1. Fig. 4.9 only shows that $L$ is related to PC-1, but doesn’t show the direction the relationship. The two-point correlation method, however, also works for correlations to the design variables which do not change geometry such as $N$ and $P$. The stress and geometry coupling method would not capture these correlations.

4.5 Discussion

This discussion shows how to use the PC-based methods to choose which design variables and regions of the design space should be explored to change the blade stress distribution. The process for using these methods is outlined as shown in Fig. 4.12, then an example is used to show each step of the process in detail. This example uses nine design variables and 100 samples. This example will show that these methods help improve the understanding of the structural design space by showing the effect of specific variables on the stress distribution of the blades.

4.5.1 Process

The process in Fig. 4.12 is based on understanding the relationship between the design variables and the PCs. First, after reviewing the stress distribution for a specific design, regions of the blade are selected based on where the designer wishes to explore how to change the stress and the resulting effects. Second, the PCs which influence stress at the desired regions are selected. Because the PCs show stress contour changes through the design space, changing the score of a
Figure 4.12: Process for using relationship between PCs and design variables to change stress.

given PC will change the stress as described by the PC. The explained variance, like that shown in Fig. 4.5 may also be used in this step. Using PCs with higher explained variance will have the greatest influence on the stress. Third, the design variables which relate to the selected PCs are chosen. This is done by using the correlations or stress and geometry coupling methods described in the methods and results section. Fourth, the relationship between the selected design variables and PCs is discovered. The correlations and coupling, from the third step, indicate a relationship between design variables and changes of stress described by the PC. Plots, such as those described in Fig. 4.6 and Fig. 4.8, may be used to understand how they are related. Lastly, once an understanding of how the design variable changes the stress is determined, the design change is made to change the stress as desired in step one. The effects of this change are observed to improve understanding of how the design variables change the stress and how stress changes at specific regions affects stress at other regions.

4.5.2 Example

An example of the process is given using Fig. 4.13. The von Mises stress for a nominal design is shown in Fig. 4.13(a). Most of the stress at this design is below 10,000 psi, however, the stress at the bottom corners is greater than 30,000 psi. For step one in Fig. 4.12, it is desired to explore how to reduce the stress at the bottom corners to below 20,000 psi and the resulting effects of those changes. Second, the PCs from the design space in Fig. 4.4 are used. Fig. 4.4(c)
shows an increase in the score of PC-1 increases the stress at the bottom corners. Fig. 4.4(d) shows an increase in the score of PC-2 decreases the stress at the bottom corners. Fig. 4.4(e) shows an increase in the score of PC-3 increases the stress at the bottom corners, but this PC also shows a decrease in stress very near the bottom left corner for the same increase of score. From these PCs it is determined that the score of PC-1 should decrease and the score of PC-2 should increase. Fig. 4.5 shows PC-1 and PC-2 explain about 70% and 20%, respectively, of the stress variation through the design space. This means that using variables which relate to the scores of PC-1 is likely to achieve greater stress change than those variables which relate to the scores of PC-2.

Step three requires finding design parameters which relate to PC-1 and PC-2. Fig. 4.9 shows PC-1 is mostly influenced by $L$. Therefore, $L$ should be used to change the stress. Fig. 4.9 also shows there is no design variable which correlates with PC-2 significantly more than other variables. Therefore, no additional variables will be used for this iteration of the process. Similar conclusions may also have been obtained by using the stress and geometry coupling in Fig. 4.11 instead of the correlations in Fig. 4.9. Step four finds the relationship between $L$ and all PCs which relate to $L$. Fig. 4.10 shows that the only significant correlations with $L$ are PC-1 and PC-2. Fig. 4.14 shows the scores of these PCs across $L$. A value of 0.15 is chosen for $L$ because the score of PC-1 is at a minimum and the score of PC-2 is close to a maximum. The fifth step is to change
Figure 4.14: The scores for the first two PCs with respect to $L$ in the nine dimensional design space.

The design and check the solution. Fig. 4.13(b) shows that the stress at the bottom corners was decreased from greater than 30,000 psi to less than 10,000 psi. Through this process the designer learned that $L$ may be used to change the stress at the bottom corners of the blade. The effect of changing $L$ from the nominal design also increased the stress to greater than 30,000 psi at the center of the blade.

To gain more precise influence and understanding of the stress, more iterations of this process may be performed which explore using design variables not selected in previous iterations. Exploring $L$ successfully decreased the stress at the bottom corners. However, because an increase in the score of PC-2 also increased the stress at the center of the blade from less than 10,000 psi to greater than 30,000 psi, it may be desired to further explore the design space. The process is begun again by choosing to explore how to reduce stress at the center of the blade and the resulting effects. PC-2, from Fig. 4.4(d) is chosen for step two because it describes a change in stress at the center of the blade. It is desired to decrease the score of PC-2 without a significant increase in the score of PC-1 in Fig. 4.4(c). Steps three and four are more difficult with subsequent iterations. Changing the stress is more challenging when the variables which correlate to a PC have close
to the same or greater correlation with other PCs as well. For step three, Fig. 4.9 shows that any variable that has significant correlation with PC-2 also has significant correlation with PC-1. This suggests that multiple variables need to be investigated.

The next variable for this example will be $C_R$. Fig. 4.10 shows that only PC-1 and PC-2 have significant correlations with $C_R$. For step four, Fig. 4.15 is used to show how the PC scores vary across $C_R$ and $L$. This plot shows a surface which is fitted to the samples across these variables with a black line indicating where $L$ equals 0.15 on the surface. Because $L$ was set to 0.15, any change in $C_R$ should result in the scores following these lines. Fig. 4.15 shows that changing $C_R$ at this value of $L$ will not result in significant change of score for PC-1 or PC-2. Using $C_R$ will not likely help achieve a reduction in stress at the middle of the blade. Next the process returns to step three and $N$ is chosen as the design variable. In step four, Fig. 4.16 shows a decrease of $N$, where $L$ equals 0.15, will decrease the score of both PC-1 and PC-2. This will decrease the stress center of the blade due to PC-2. Decreasing the score of PC-2 would also increase stress at the bottom edges, but because the score of PC-1 also decreases, this effect should be canceled out. For the fifth step in this iteration of the design process, $N$ is changed to a value of 1.1e6. This value is chosen because Fig. 4.16 indicates that the scores PC-2 and PC-1 are close to their minimums for $L$ equals 0.15. Fig. 4.13(c) shows that, with a new design of increased $L$ and decreased $N$, stress was reduced to about 20,000 in the center of the blade without also increasing the stress above 10,000 at the bottom corners. This example has shown how using the methods which discover the relationship between the PCs and the design variables can assist in exploring the design space, can increase understanding of the design space, and can be used to achieve design objectives.

### 4.5.3 Real-Time Design Exploration

The methods applied and developed in this research help designers know which variables and regions of the design space should be explored for desired structural results. As shown through the example, the process may take a few iterations to find a satisfactory design. These methods could be applied with real-time DSE as done by Bunnell et al. [62] to further improve DSE. The exercise shown in Fig. 4.13 provides guidance to indicate which variables to explore in large design spaces. Fine tuning a design around a given region of the design space may require many iterations.
A desired result can be quickly achieved by using real-time DSE to quickly search regions of the design space indicated by the methods described in this paper.

4.6 Conclusions

This research used PCA to reduce structural results from FEA into PCs describing structural variation of compressor blades across design spaces. Two methods were applied which related the design variables to PCs. These methods allowed the PCs to be used as a DSE tool by indicating which variables and regions of the design space should be explored to changes at specific regions of the blade and the effect of those changes. The two-point correlation, an existing method, was applied to the design variables and the PC scores. This correlation indicated the design variables which had the greatest influence upon each PC. It was shown how these correlations can guide DSE and find the design variables which need to be changed to best influence the structural results. The geometric coupling method was developed and applied to relate variations in stress

Figure 4.15: The scores for the first two PCs with respect to \( C_R \) in the nine dimensional design space.
Figure 4.16: The scores for the first two PCs with respect to $N$ in the nine dimensional design space.

directly to variations in the geometry of the compressor. Like the correlation method, this method indicated which design variables should be changed to achieve desire structural results. The geometric coupling method also provided an indication to the direction the design variables should be changed. These methods performed well in high dimensional design spaces that are common in turbomachinery compressor blade design.

The methods presented in this paper should be used when performing structural design exploration based on finite element analysis. A DOE is used to select a set of samples distributed globally through the design space. This research found useful exploration with relatively few samples, i.e. 100 when using nine design variables. FEA is then performed for the samples to obtain the structural response and coordinates which will be explored. The results from FEA need to be represented on a grid which is common among all of the samples in order to keep the nodes in the same location relative to the geometry. This research accomplished common grid representation with mesh morphing [62]. PCA should then be performed as described in the
methods section. The PCs describe the variation of the structural response contours throughout the design spaces. These are used to understand how the structural response may be changed with the used design variables. PCs which describe more variance allow more change in the structural results. This research found the first two or three PCs explained more than 95% of the variation with tests that included nine design variables. If a desired change is not obtainable, other variables should be used. This will change the PCs and the amount of variation each one describes. The scores of the PCs can then be related to the design variables to understand the effect of the variables on the structural response. Either two-point correlation or stress and geometry coupling may be used to relate the designs. Once the understanding between variables and structural response is known, a new design may be wisely selected and analyzed.

As design spaces for compressor blades become more complex, better methods are needed to more fully understand them. This research developed and applied methods that use PCA of the design space to show variations which are not easily understood otherwise. These methods provide guides to change the design variables by which the stress is better understood and controlled. The improved understanding provided by these methods can help meet designs objectives and decrease the computation cost needed to achieve them.
CHAPTER 5. DESIGN SPACE EXPLORATION OF VIBRATORY RESULTS

Turbomachinery blade design must account for vibratory responses of the blade. The objective of this chapter is to extend the DSE methods developed in the previous chapter to account for structural vibratory responses. The surrogate evaluation and interactive design space exploration speeds are dependent upon the same factors described in the previous chapter. Due to this, speed was not tested and measured with vibratory responses. The error of the vibratory emulations, on the other hand, does not have the same assurance of being similar to the static emulations. The contribution of the research is provided in the following manner.

- Develop an analysis workflow to solve vibratory responses with low enough computational cost to allow for enough samples to be obtained for DSE in a reasonable time frame.

- Develop a surrogate-based emulation method which can handle challenges unique to vibratory responses.

- Quantify improvement of methods for DSE.

This research did not develop novel FEA or CFD analyses, but selected analysis methods which are common to compressor blade design. These selected methods were used to create novel vibratory analysis and surrogate workflows which are useful for DSE. The methods needed to be computationally cheap enough to provide enough samples to create accurate surrogate models. The contents of this chapter are from a paper submitted for publication in the Journal of Computers and Structures.

5.1 Introduction

Gas turbine engines are integral to society with global trends demanding more efficient designs. Fan blades must be designed to account for both steady and vibratory structural responses if
they are to meet structural requirements. Vibratory responses may be mechanical or aerodynamic, such as flow distortions like those studied by Soderquist et al. [90]. Recent fan blades have even been designed to take advantage of boundary layer ingestion, a form of aerodynamic distortion, at the engine inlet to improve efficiency [33, 91–93]. Such designs found efficiency improvements up to 10% but also found that the structural vibratory response caused by the aerodynamically distorted inlet often pushed the structural limits.

Design space exploration (DSE) is the study of the relationship between the design variables and the responses of interest [94]. DSE of turbomachinery blades provide insight to the relationship between the structural responses and the blade design variables [62, 69, 95, 96]. DSE can help find fan blade designs which are more efficient and meet acceptable vibratory stress limits [2, 57, 97]. Other researchers performed DSE to discover the relationship between the inlet Mach number and incidence angle on the amplitude of the vibratory responses [98]. While useful, DSE requires analyses to be performed on a set of designs. Accurately accounting for the vibratory response of fan blades involve computationally expensive analyses which limit the number of designs which may be analyzed in an acceptable time frame. This is especially true with analysis which requires both computational fluid dynamics (CFD) and finite element analysis (FEA), as done by Bakhle et al. [33]. High computational expense limits the effectiveness of DSE and the ability to find designs that meet requirements and constraints. Simplified analyses are often used to mitigate high computational cost for DSE. These can sacrifice a small amount of accuracy for a large reduction in computational cost. Laxalde et al. and others have used simplified analysis methods to perform vibratory analysis on turbomachinery blades [99,100]. Their simplified models had errors of less than 2% but they were able to perform their analyses with low enough computational cost to allow many samples to be analyzed and perform DSE to make important findings on the design of these blades.

Surrogate models are also used to mitigate the high computational cost of DSE. These models create mathematical estimates of the responses based on a few samples of analyzed designs and allow almost instantaneous emulation of the response [8]. This allows for more evaluations, albeit estimations, of the response. Bunnell et al. found that surrogates could accurately be used to represent steady stress of turbomachinery blades for DSE [62]. They found that, given enough samples, their surrogates could achieve error of less than 10% and could be emulated in less than 1
second. Research has shown that, during DSE, longer times to reach an acceptable design lead to worse final designs [4–6]. Whether for optimization-based DSE or interactive, visualization-based DSE, surrogates that lower the time to evaluate a design means more designs can be evaluated. The findings from Bunnell et al. show that using surrogates to emulate the structural response allows for better engine blade designs to be found. For instance, Huang et al., used surrogates based on FEA of turbine disks to reduce the design weight by 18.7% while staying within structural limits [101]. Such findings were performed on steady stress which has a relatively smooth response over the design space. A unique challenge of emulating vibratory stress lies in the large spikes in vibratory response found at resonance, where the frequency of vibratory load matches the frequency of the structural modes [30]. Surrogates which directly emulate the vibratory stress are less able to capture these spikes with sparse sample sets. The decreased ability to capture the spikes limits the ability of DSE to estimate the vibratory response and indicate if the design is within structural limits.

This paper presents a method to improve DSE of structural vibratory response on turbomachinery blades. The method includes a simplified vibratory response amplitude analysis which allows for a larger number of samples to be used in DSE. The analysis, called RANS-HMS, relies on steady-state Reynolds-Averaged Navier Stokes (RANS) CFD and Harmonic Mode Superposition (HMS) FEA simulations. The method also includes a surrogate model which can accurately emulate vibratory responses which include large spikes near resonance. These methods are applicable to interactive DSE and optimization of structural vibratory responses of turbomachinery blades. The fan blade model used in this research is the Parametric Blade Study (PBS) Rotor 4. This geometry, shown in Fig. 5.1 and Fig. 5.2, is a transonic fan which was developed as a part of a study to understand the effect of various design variables on the aerodynamic properties of the fan [59,102]. Fig. 5.1 shows the cross sections of a single blade while Fig. 5.2 shows the full blade row with 20 blades. This researched used design variables, shown in Tab. 5.1, which are common to structural DSE of gas turbine blades.

The paper proceeds with a background section which covers current blade design process and analysis methods. The methods section details the RANS-HMS analysis and surrogate methods developed for DSE of vibratory response. The results and discussion sections use PBS Rotor
Table 5.1: Set of possible design variables for the PBS Rotor 4 fan blade used for this research. The ranges for these variables were from minus 20% to plus 20% of the nominal value.

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mid-Span Chord</td>
<td>$C_M$</td>
</tr>
<tr>
<td>75%-Span Chord</td>
<td>$C_{75}$</td>
</tr>
<tr>
<td>Tip Chord</td>
<td>$C_T$</td>
</tr>
<tr>
<td>Mid-Span Thickness-to-Chord Ratio</td>
<td>$T_{CM}$</td>
</tr>
<tr>
<td>75%-Span Thickness-to-Chord Ratio</td>
<td>$T_{C75}$</td>
</tr>
<tr>
<td>Tip Thickness-to-Chord Ratio</td>
<td>$T_{CT}$</td>
</tr>
</tbody>
</table>
4 to validate the performance of the developed methods and show how their use leads to better turbomachinery blade designs.

5.2 Background

A common vibration analysis workflow, used to account for vibration in turbomachinery blades, is given in the following list presented by Snyder et al. [15]. Following this workflow ensures the blade is within structural limits. Performing DSE while accounting for vibration should include these steps.

1. Identify possible sources of excitation
2. Determine operating speed ranges
3. Calculate natural frequencies
4. Construct resonance diagram
5. Determine response amplitudes
6. Calculate stress distribution
7. Calculate modified Goodman
8. Determine high cycle fatigue (HCF)
9. Redesign if HCF life is not infinite
10. Conduct physical engine testing

Sources of excitation may be mechanical, such as structural mistuning studied by Philippe et al., or aerodynamic, such as inlet distortion as studied by Bakhle et al. [33, 100]. The forcing frequency of the excitation source is then determined from the operating speed range and the nature of the excitation source. The natural frequencies, or modal frequencies, are calculated and used in the resonance diagram to determine which modes may be excited by the vibratory sources [13]. Calculation of the response amplitude finds the magnitude of displacement during vibration. This step requires the most computationally expensive analysis. The amplitude and the mode shapes are
combined to calculate the vibratory stress distribution. The modified Goodman diagram is used to inform the designer on high-cycle fatigue (HCF), the structural response of the blade over repeated loading [16, 103]. High-cycle fatigue is the largest cause of component failures in engines, and is therefore a very important factor in design [104]. Blades with modified Goodman values below 100% have infinite HCF life. Once the expected life is determined, redesign is considered. If the vibratory stress is not within limits or the expected life is not infinite then the blade is redesigned and analyzed again. If the blade is satisfactory then redesign may still be attempted to allow for improvements in other design factors, such as weight or aerodynamic efficiency, while keeping the blade within structural limitations. This redesign step, however, may be difficult when blade designs are more complicated. An understanding of the effect each design variable has upon the vibratory stress and modified Goodman values may be difficult to obtain. However, this step is important because the blade must be satisfactory before physical testing is attempted.

5.2.1 Computational Analysis

Modern analyses use scientific computing in the common vibration analysis workflow. Calculation of mode frequencies and shapes are performed with FEA modal analysis [14]. This analysis is common to turbomachinery blade design [12, 21, 105]. Calculation of response amplitudes involves solving for the aerodynamic excitation forces on the blade. This is often performed with a combination unsteady CFD to solve for the forcing on the blade and FEA to solve for the structural response to that forcing. Unsteady Reynolds Averaged Navier Stokes (URANS) CFD simulations have been performed on PBS Rotor 4 with inlet distortion by List et al. and Soderquist et al. to understand the effect of the distortion had on the aerodynamic qualities of the rotor blade [60, 61, 106]. Bakhle et al. used URANS simulations to solve for the time-dependent pressure variation on a different blade which was specifically designed for boundary layer ingestion [33]. The pressure variations cause structural vibrations. Steady stress is obtained by FEA static simulations and vibratory stress may be obtained by transient simulations. Steady FEA was used by Bunnell et al. to explore the structural design space of compressor blades [62]. Advancements in computing have allowed coupling of unsteady CFD and transient FEA into a single simulation called fluid structure interaction (FSI). Kou et al. showed that FSI could model vibratory responses of compressor blades within 10% of the experimental data [19, 20].
5.2.2 Computational Simplifications

While the discussed computational vibration analysis methods are accurate, they are also too computationally expensive to collect a sufficient number of samples within an acceptable time frame for DSE. Even with high-performance computing machines, coupled FSI simulations can take weeks to months to fully solve. URANS simulations of PBS Rotor 4 require a few days of wall time when running on 1000 processors to solve for a single design and operating condition. This high computational cost especially inhibits DSE or the redesign phase when they require several iterations. Whether for the initial design exploration or for a subsequent redesign iteration, simplified computational vibration analysis methods improve DSE by allowing more designs to be analyzed. Fluid analysis simplifications are often made to decrease this computational cost when design exploration must be performed. Reynolds-Averaged Navier-Stokes (RANS) is a steady-state simplification of URANS that does not include time marching. Montomoli et al. studied and compared RANS to URANS on a four stage axial research compressor. Their RANS simulations were a fourth of the computational cost and within 10% error to the URANS simulation, except near stall where higher error was found [107]. RANS has been used in industry and academia to study the aerodynamic response of turbomachinery blades. Knapke et al. used RANS to understand the response from geometric variations caused by repair on the PBS Rotor 4 [97]. The reduction in computational cost given by RANS, although it comes with an increase in model error, is beneficial for exploring the design space. However, the error and cost reduction is dependent upon the application and modeling conditions as stated by Mondal et al. [108].

Vibratory structural analysis simplifications are also made to decrease the computational cost. Thelin et al. assumed that the vibratory stress was equivalent to the stress induced by a single mode and scaled to a fixed nominal value obtained through empirical experience [105]. This assumption was made due to the lower computational cost of obtaining the vibratory stress from modal analysis instead of CFD. The scaled-mode method was overly conservative throughout most of the design space, but when the mode frequencies were close to a vibratory forcing frequency the scale factor was too low. Harmonic mode superposition (HMS) may be used, instead of full transient analysis or scaled-modes. HMS assumes the structural vibration is a superposition of the mode shapes scaled to calculated amplitudes. However, it still requires a vibratory force to determine the amplitude of excitation. Danforth et al. used inlet distortion patterns to obtain the
vibratory force and frequencies for HMS of fan and compressor blades [32]. Many types of distortion were used to quantify the effect of vibratory stress. It was found, with HMS, that the various types of distortion had a significant effect upon vibratory stress. Bakhle et al. performed similar simulations to investigate the vibratory stress on a fan blade designed specifically for inlet distortion [33]. They used URANS simulations to solve for the aerodynamic vibratory forces on the blade and HMS to solve for the vibratory stresses. While HMS decreased the computational cost of the structural simulation, the URANS fluid simulations still caused the computational cost to be high. The present research developed a simplified computational vibration analysis workflow, with accompanying surrogates, which relied on RANS and HMS to provide enough samples to create accurate emulation for DSE.

5.3 Method

A simplified vibration analysis workflow that lowers the computational cost to evaluate a design is presented. This was necessary to create enough samples to explore the design space of a fan blade. Like Bakhle et al., this research used the effect of unsteady forces on the surface of the blade caused by aerodynamic inlet distortion as the excitation [33]. However, the simplified analysis methods are consistent for other aerodynamic vibratory sources such as flow distortions, which cause cyclic forcing, from neighboring blade rows. This simplified computational vibration analysis workflow, shown in Fig. 5.3, is referred to as RANS-HMS in this work. First, a computer-aided design (CAD) model was used to parametrically change the geometry to a new design. The updated geometry, labeled in the figure as A, was passed to a CFD model so the fluid mesh could be updated to match the new design. The CFD model was created in Star-CCM+. RANS was solved to obtain the steady-state aerodynamic pressures on the surface of the blade. The steady-state pressures were used to make estimations of the unsteady pressures on the blade. The updated geometry was also passed into the FEA model where the structural mesh was updated with mesh morphing. Mesh morphing, as used by Bunnell et al., was used to keep the FEA mesh congruent among all of the samples [62]. This means the nodes of the mesh remain at the same blade location for each sampled design. A node on the leading edge tip stayed at the leading edge tip. This was required to later create the surrogates. The steady-state pressures from the RANS simulation, labeled in Fig. 5.3 as B, were applied to the surface of the structural mesh. A rotational load was
also applied along with a fixed constraint at the blade root. Static FEA, performed with ANSYS, solved for the steady stress. A pre-stressed modal analysis was then performed. This means modal analysis was performed on the mesh after it was displaced under the loading conditions in static FEA, shown as C in Fig. 5.3. The modal analysis returned the mode displacements and frequencies. The mode displacements were mass normalized, a later requirement for HMS. The mode stresses were calculated from the mode displacements.

The calculation of the vibratory stress with HMS required the mode displacements, mode stresses, mode frequencies, the vibratory force distribution, and forcing frequency. The first three came from the modal analysis. The vibratory forces were calculated from the pressures on the surface of the blade obtained from CFD. Bakhle et al. used the unsteady portion of the pressure...
distribution from URANS as the vibratory force [33]. Because RANS is used in the presented RANS-HMS method, assumptions must be made about the unsteady portion of the pressure distribution. For inlet distortion, the unsteady loads on the blade and the magnitude of vibration may be estimated from the magnitude of the inlet distortion [32, 109]. The unsteady pressure distribution was assumed to be the steady-state pressure distribution scaled by the magnitude of the inlet distortion. This assumption was sufficient for vibration near the first few modes, but may not be sufficient for higher, more complex modes. Because inlet distortion excites the first few modes more than the higher modes, this assumption was deemed sufficient. The vibratory stresses are then solved using HMS shown in Eqn. 5.1-Eqn. 5.4. These equations were derived from the equations of motion for a simple harmonic forcing with no damping [30]. The modal force vector, $p_r$, is the dot product between the modal displacement matrix, $\phi_r$, and the load vector, $f$, for mode $r$. The load vector in the RANS-HMS method is the unsteady pressure distribution. The frequency of mode $r$ is $\omega_r$ while the frequency of the vibratory load is $\omega_f$. When a mode frequency is close to the forcing frequency the vibratory response spikes to large values. When the frequencies are the same the response is mathematically infinite in an undamped system. Modern turbomachinery blade rows, which include the disk and all of its blades, are often manufactured as a single part. This saves weight but offers very little structural damping. The lower damping leads to higher vibratory stresses, especially near resonance. Small margins of error in damping could lead to large errors in vibratory responses near resonance. Undamped HMS was used to ensure conservative estimates.

$$p_r = \phi_r \cdot f \tag{5.1}$$

$$\Omega_r = \frac{\omega_f}{\omega_r} \tag{5.2}$$

$$q_r = \frac{1/\omega^2}{(1 - \Omega^2_r)} \tag{5.3}$$

$$\sigma_a = \sum_r \sigma_r q_r p_r \tag{5.4}$$
The modified Goodman, $G$ in Eqn. 5.5 given by Thelin et al., utilizes the steady stress, $\sigma_s$, and the vibratory stress, $\sigma_a$. In Eqn. 5.5, $S_e$ is the endurance and $S_{ut}$ is the ultimate strength. They are material and geometric properties of the blade. It is used in DSE to understand the expected life-time when accounting for high-cycle fatigue [105]. Because it is dependent upon the vibratory stress it also experiences large spikes near resonance. At and around those spikes, the modified Goodman values can become greater than 100% which means the blade does not have infinite HCF life.

$$
G = \frac{\sigma_a}{S_e - \frac{S_e}{S_{ut}} \sigma_s}
$$

(5.5)

### 5.3.1 Surrogate Models

Surrogate models were created to allow for instantaneous evaluation of the vibratory results for any design within the trained design space. The values for all nodes were used in DSE rather than only the maximum stress or modified Goodman values. This improved the ability to understand the blade’s response during DSE by showing the location on the blade where failure would occur. Emulating all of the nodes has also been shown to reduce error of estimating the maximum stress and Goodman values by 50% and 30% respectively [105]. The surrogate model was based on the following steps:

- **Select Design Samples**
- **Design Variable Values, $C$**
- **Simulation Analyses**
- **Simulation Responses, $Y$**
- **Data Creation**
- **Surrogate Weights, $W$**
  - **Train RBF Scores, $A$**
  - **Perform PCA Principal Components, $V$**
  - **Surrogate Training**
- **Design Variable Values for New Design, $x$**
  - **Evaluate RBF Estimated Scores, $\bar{A}$**
  - **Inverse Transform Estimated Response, $\bar{Y}$**
- **Surrogate Evaluation**

Figure 5.4: Surrogate creation and evaluation work flows. The boxes with squared corners are processes and those with rounded corners are data. Capitalized variables are matrices and lower-case variables are vectors.
on a spectral representation of the structural responses as formulated by Bunnell et al. [95]. The process to create and evaluate the surrogate model is shown in Fig. ???. The data for the surrogate was created by first selecting which design samples would be evaluated with RANS-HMS. A latin-hypercube design of experiments (DOE) was used to select the design samples. The DOE produced a matrix, \( C \), which contained the values for each design variable for all chosen samples. The simulation, RANS-HMS, used the design variable values to update the geometry and solve for the response. A data matrix, \( Y \) shown in Fig. 5.5, was assembled from the simulation responses of each sampled design. The columns of \( Y \) represent the nodes and the rows represent the design samples.

Principal component analysis (PCA) was performed on \( Y \) to solve for the matrices \( V \) and \( A \) which represented the principal components (PCs) and PC scores respectively. PCA was performed by subtracting the column averages, \( \bar{y} \). Then, singular value decomposition was performed to solve for \( U, S, V \) as shown in Eqn. 5.6. \( V \) represents PCs, or the variations of the data through the design space. Multiplying \( U \) and \( S \) returned the PC scores, \( A \), which represented the projection of each sample onto the PCs.

\[
Y - \bar{y} = USV = AV
\] (5.6)

Next the surrogate model used the matrix of PC scores, \( A \), and the design variable values, \( C \), to train a radial basis function (RBF), shown in Eqn. 5.7. The RBF was based on a multiquadric kernel, shown in Eqn. 5.8, the same kernel utilized by Bunnell et al. [62]. Training the RBF produced a matrix of weights, \( W \) which was necessary for surrogate evaluation.

\[
W^T = \Psi^{-1} A \text{ where } \Psi_{i,j} = \psi(||C_i - C_j||) \text{ for } i, j = 1, ..., s
\] (5.7)

\[
\psi(r) = \sqrt{\left(\frac{r}{\varepsilon}\right)^2 + 1}
\] (5.8)

The process to evaluate the surrogate is shown with Fig. ?? in the box on the right. First, the RBF was evaluated at a new design as shown in Eqn. 5.9. The design vector, \( x \), contains the design variable values for the new design. This, with the surrogate weights, \( W \), was used to estimate the PC scores, \( \hat{a} \), as shown in Eqn. 5.9. The estimated PC scores were then used with the PCs, \( V \), to
Figure 5.5: The \((s, n)\) data matrix, \(Y\). Each row was a unique FEA sample, of which there were \(s\). The columns, of which there were \(n\), represented the stresses for \(n\) nodes on the finite element model.

Invert the PCA transformation, Eqn. 5.10. This produces a vector, \(\hat{y}\), which is an estimation of the response at the nodes. This surrogate model emulation process was evaluated with “real-time” speeds which allowed for quick and efficient design exploration [62].

\[
\hat{a}(x) = \lambda W \text{ where } \lambda_i = \psi(||x - C_i||) \text{ for } i = 1, \ldots, s 
\]  

\[
\hat{y} = \hat{a}(x)V + \bar{y} 
\]  

The standard method of DSE suggests training the surrogate directly on the vibratory stress and modified Goodman values, the dark gray boxes in Fig. 5.3. However, the spikes in vibratory stress found through the design space when using HMS make direct emulation of vibratory responses inaccurate at the designs which are structurally unacceptable. When a design space is relatively sparsely sampled, which is often necessary when using any form of CFD in the analysis, the samples may completely miss all spikes in stress. Samples which are close to stress spikes will cause the surrogate to be more accurate at the sampled design, but they will lessen the surrogate accuracy around the spike by erroneously emulating the local feature as a global trend. Instead, this research proposes emulating the vibratory responses indirectly by creating surrogates from the inputs to the HMS calculation, represented by the light gray values in Fig. 5.3. The static stress, mode displacements, mode stresses, and mode frequencies do not have the sharp spikes that the vibratory responses have through the design space. Bunnell et al. found less than 10% error with
static stress surrogates used for design space exploration [62]. Brown et al. created surrogates of mode displacement and frequency from a set of blades with manufacturing deviations [57]. Their surrogate error was also low. While these two studies were performed on different analyses and different models, they show that surrogates may be used effectively for static stress and modal data. The pressures do not have spikes in the feasible space because such samples, like those near blade stall, fail in CFD. Surrogates created on these more smooth spaces are more accurate. DSE therefore uses the set of surrogates and calculates the HMS vibratory response to indirectly emulate the response and obtain more accurate results.

5.4 Results

This section shows the results from the RANS-HMS analysis. It quantifies the slightly increased error, but decrease in computational cost achieved with the RANS-HMS method. It also quantifies the error of the indirect emulation method and shows the improvement from direct emulation. These quantifications show the capability of the presented methods to allow for DSE of vibratory responses in gas turbine blades. The research used PBS Rotor 4 to demonstrate DSE while accounting for aerodynamic induced vibration. The steps in the common vibration analysis workflow, discussed in the background section, were followed. First, the source of excitation was inlet distortion which decreased the total pressure by 15% across 90° sectors of the inlet [90]. Thus, the source of excitation comes from an unsteady pressure distribution which is estimated to be the same as the steady-state pressure distribution but scaled to 15%. Second, the design operating speed of the rotor was 20200 rpm. Only this speed was considered, though the DSE methods presented in this research could also be used to explore varying the rotor operating speed. RANS-HMS was used to perform the rest of the common vibration analysis workflow. This was performed on a set of samples selected through Latin hypercube sampling to make a surrogate.

5.4.1 RANS-HMS Methodology

The RANS-HMS method, as described in the methods section, determined mode frequencies, mode displacements, response amplitudes, stress distributions, and modified Goodman distributions as required in the third through seventh steps of the general vibratory analysis workflow.
Figure 5.6: Resonance diagram for Rotor 4 nominal design. The engine orders (EOs) frequencies are the horizontal lines. The mode frequencies are the vertical lines. The diagonal line indicates where forcing frequency equals the mode frequency.

The mode frequencies were obtained through a modal analysis method. The frequencies of the nominal geometry were plotted on the resonance diagram in Fig. 5.6 which shows the mode frequencies and engine orders (EOs), which are possible forcing frequencies in turbomachinery. The closer a mode frequency is to crossing an EO along the diagonal line, the more strongly it is excited when forced at the EO frequency. Fig. 5.6 shows that the frequency of the first mode was close to crossing EO-2 where it meets the diagonal lines (this crossing is indicated by the circle). This indicated that the first mode may be highly excited by a EO-2 forcing. The figure also shows EO-5 and EO-8 are near the second and fourth mode frequencies respectively. Inlet distortion is more likely to excite the first mode because that mode has the frequency closest to that of inlet distortion. As such, this section will focus on EO-2, however, similar design exploration methods could be applied to EO-5 and EO-8.

The RANS CFD model was used to calculate the vibratory forces on the blade. This RANS model was developed from URANS simulations used and validated by previous studies [60, 61, 106, 110, 111]. Peterson et al. describes the creation of the CFD model in detail. Their model includes an inlet, the rotor row with 20 fan blades, a stator row with 31 blades, and an outlet. Studies have
Figure 5.7: Performance map of PBS Rotor 4 with clean inlet. The plotted performance values were measured between the rotor inlet and the stator outlet. The experimental data is from Law et al. law1988parametric. URANS is by List et al. list2014numerical. The RANS were simulations performed in the present research.

validated the URANS model with no inlet distortion by comparing corrected mass flow, pressure ratio, and adiabatic efficiency of the simulation to experimental data [61]. This comparison, given in Fig. 5.7, shows how the pressure ratio and adiabatic efficiency change with corrected mass flow. List’s URANS simulation found peak adiabatic efficiency of 87.0% at a corrected mass flow of 28.1 (kg/s) and pressure ratio of 2.01 while the experimental data showed 89.6% at a corrected mass flow of 27.6 (kg/s) and pressure ratio of 1.99. The RANS simulation used in this research is also shown in Fig. 5.7. The RANS simulation obtained peak adiabatic efficiency within 1% of
the URANS simulation. The corrected mass flow and pressure ratio were within 5% of the values found with URANS. These comparisons show the error of the RANS model to the URANS model and experimental data.

The pressure distributions from the RANS simulation were used in FEA to solve for the steady stresses and modes. The pressures for the nominal design are shown in Fig. 5.8. These in turn were used to calculate the vibratory amplitudes and then vibratory stress distributions with HMS. This research used 25 modes for calculation of the stress distribution. While including more modes decreases error, it was found that including more than 25 changed the vibratory stress distribution less than 1%. The steady and vibratory stresses were used to calculate the modified Goodman values as required in the seventh step of the common vibration analysis workflow. Fig. 5.9 shows the steady stress, vibratory stress, and modified Goodman contours for the nominal PBS Rotor 4 design under forcing at a frequency equal to EO-2. The maximum steady stress was 84.0 ksi, the maximum vibratory stress was 18.4 ksi, and the maximum modified Goodman was 45%. The ultimate strength was 170 ksi and endurance strength was 46 ksi. Despite having a maximum steady stress which was 50% of yield and a relatively high vibratory stress, the maximum modified Goodman was relatively low due to the differing locations of maximum steady and vibratory stress.
RANS-HMS was compared to the URANS-HMS method used by Bakhle et al. [33]. Normalized-root mean square error (NRMSE) of the vibratory stress was used to evaluate the error of the RANS-HMS to the URANS-HMS. The computational cost of the methods were measured in the average wall-clock time required to perform the simulations using high performance computing. These simulations were performed on about 1000 processors. Fig. 5.10 shows the NRMSE values for the vibratory stress with the computational cost required to obtain the stress. The RANS-HMS method introduced about 10% error compared to the URANS-HMS method for about a 90% reduction in computational cost. This means 10 times the number of samples may be used for DSE when RANS-HMS is employed compared to URANS-HMS, making surrogates more accurate and DSE more effective. Though it had 10% error, the RANS-HMS method was conservative which provided more confidence that the method could be used to produce designs which were within structural limits. Fig. 5.10 also compares the error of the scaled-mode method used by Thelin et al. to the URANS-HMS method [105]. The scaled-mode method assumes a fixed magnitude of vibratory response usually obtained through empirical data. It is computationally cheap because it only requires modal analysis, but tends to be overly conservative for most designs and underes-
estimate the response near resonance. In this case, the first mode was scaled to a maximum stress of 12 ksi, as performed by Thelin et al. This led to an error of about 40%, but estimated lower vibratory stresses than the higher-fidelity methods. The lower estimation can falsely lead to concluding that the design is within structural limits. While the computational cost is low compared to the RANS-HMS method, the error will fluctuate over the design space, producing errors close to 100% where the mode frequencies match the forcing frequency. When the scaled-mode method is used it is important to ensure that the designs are far from resonance. The RANS-HMS method balanced computational cost savings with analysis accuracy, especially near resonance.

5.4.2 Surrogate Models

Surrogates were created to allow for DSE of the vibratory response of the fan blade. A training set was created to test the analysis and surrogate methods developed in this research for DSE. This training set had all of the design variables given in Tab. 5.1 with 83 samples selected through Latin hypercube sampling. These samples were run through the RANS-HMS workflow. Not all samples were feasible due to the large range of the design variables. The samples which were structurally infeasible in FEA still produced stress data, but the samples which were aerodynamically infeasible did not produce pressure data. These aerodynamically infeasible designs experienced blade stall which caused numerical instabilities and failure of the CFD analysis. The
number of samples which failed due to blade stall was 36, leaving a remaining 47 samples for DSE. The failed samples had low values for Chord, between -20% and -10%, on at least two cross sections. The loss in samples, however, does not diminish the quality of DSE because those samples indicate a region of the design space which results in infeasible designs. This region would be unproductive to explore. The 47 samples which successfully completed had steady stress up to 122 ksi.

The indirect emulation method used a set of surrogates to emulate the inputs in to HMS. Double-NRMSE was used to measure the global error of those surrogates which were based on result fields, e.g. unsteady pressure, steady stress, mode stress, and mode displacement. Double-NRMSE evaluates the square-root of the mean of the squared difference between the estimated response and the true response. The resulting value is normalized by the maximum true value for the high node and sample, estimated from a separate testing set. This error, shown in Eqn. 5.11, is the equivalent of NRMSE over all of the samples, of which there were $s$, and all of the nodes of the finite element model, of which there were $n$.

$$
e = \sqrt{\frac{\sum_{i=1}^{s} \sum_{j=1}^{n} (\hat{y}_{i,j} - y_{i,j})^2}{ns}} \cdot \frac{1}{y_{max}} \cdot 100\% \quad (5.11)$$

The error, $e$, is calculated from the difference between the estimated value, $\hat{y}$, and the true value, $y$, at node $j$ and sample $i$. The error of the pressure distribution surrogate is performed with leave-one-out cross-validation due to the high computational cost of creating a separate set of CFD testing data. The error of the steady stress, mode displacement, mode stresses, and mode frequency surrogates are evaluated by comparing the surrogate to a separate set of testing data with about 350 samples. Such a larger number was needed to measure the global accuracy of the responses in a design space with large spikes in the response. These test samples were selected through Latin hypercube sampling which was independent of the samples selected for training the surrogate. However, an abbreviated analysis method was used to create the testing set due to the high computational cost of the CFD simulations needed to evaluate the pressure data. This abbreviated analysis used the pressure surrogate created from the training samples to emulate the pressure at each of the test samples. The emulated pressure was used in FEA to solve for the rest of the data. The error of the mode frequency surrogate uses standard NRMSE because each
Table 5.2: Error of the surrogate models used to calculate vibratory response.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure</td>
<td>1.5%</td>
</tr>
<tr>
<td>Steady Stress</td>
<td>1.2%</td>
</tr>
<tr>
<td>Mode Displacements</td>
<td>1.9%</td>
</tr>
<tr>
<td>Mode Stresses</td>
<td>1.7%</td>
</tr>
<tr>
<td>Mode Frequency</td>
<td>2.1%</td>
</tr>
</tbody>
</table>

sample is represented by a single value. These errors, shown in Tab. 5.2, are all below 3%. This level of error is within the 10% error produced by those by Bunnell et al. and found to provide for acceptable DSE with the indicated responses [62]. These low errors provide confidence that the indirect emulators will also have lower error.

The vibratory stresses and modified Goodman values were calculated with on the fly estimations provided by the surrogates. The vibratory stresses and modified Goodman values had large spikes near resonance. These spikes, however, were not conical spikes distributed through the design space. Instead, they were boundaries of n-dimensions, where n is the number of design variables, which divide the design space into feasible regions. This is seen in Fig. 5.11 which shows the maximum vibratory stress on the blade across the design space of $C_M$ and $C_{75}$. The values in this plot were produced by the indirect emulation method. Regions colored yellow had stresses which were 100 ksi or greater. This is seen as a narrow band of high vibratory stress that creates a boundary between the upper right and lower left regions of this design space. The figure shows a sharp change from high vibratory stress, shown with the yellow band, to low vibratory stress as the design is moved away from resonance. The plot shows the ability of the indirect method to capture both the global trends of vibratory stress and important, prominent local features. The spikes in modified Goodman occur in the same location, near resonance, as the spikes in vibratory stress, but they taper away from the spike differently due to the contribution of steady stress. This is shown in Fig. 5.12 which displays a wider boundary and shallower taper than that of vibratory stress. Designs with a modified Goodman above 100%, shown with the yellow band, are not structurally feasible. These figures show that DSE which includes increasing $C_M$ and $C_{75}$ may include crossing this boundary of infeasible designs. DSE should be able to accurately find
Figure 5.11: Maximum vibratory stress from indirect emulation of the tested design space. Varying $C_M$ and $C_{75}$ is shown while the other design variables are held constant.

and account for these spikes in vibratory response in the design space in order to provide designs which meet structural limitations. The indirect emulation method has been shown in Fig. 5.11 and Fig. 5.12 to have the capability to find these vibratory spikes.

NRMSE was used to measure the global error across the design space by comparing the response to the separate set of testing data with 350 samples. Double-NRMSE, as performed with the surrogates in Tab. 5.2, was not used to evaluate the global error because the true maximum vibratory response through the design space, which would be used to normalize the error, is near infinity. Instead, NRMSE was calculated for each test sample to measure the error of the direct and indirect vibratory response methods. The error for each sample was normalized according to its own maximum true value on the blade as solved through RANS-HMS. The mean and median NRMSE of both found among all of the test samples is given in Tab. 5.3. This table shows both direct emulation, which creates a surrogate directly on the vibratory responses, and the indirect method. The table shows that using a direct surrogate to emulate vibratory responses produced high error, above 85% on average with a median of about 50%. The indirect method produces error more reasonable for DSE with a mean error of about 17% and a median of about 7%. The
Figure 5.12: Maximum modified Goodman from indirect emulation of the tested design space. Varying $C_M$ and $C_{75}$ is shown while the other design variables are held constant.

Table 5.3: Error of the emulated vibratory responses.

<table>
<thead>
<tr>
<th>Method</th>
<th>Response</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>Vibratory stress</td>
<td>85.1%</td>
<td>51.5%</td>
</tr>
<tr>
<td>Direct</td>
<td>modified Goodman</td>
<td>82.7%</td>
<td>47.1%</td>
</tr>
<tr>
<td>Indirect</td>
<td>Vibratory stress</td>
<td>17.1%</td>
<td>7.35%</td>
</tr>
<tr>
<td>Indirect</td>
<td>modified Goodman</td>
<td>17.0%</td>
<td>7.28%</td>
</tr>
</tbody>
</table>

medians in these results give a sense of the global error of the surrogates. The mean errors are higher because of higher normalized error found at the spikes in stress where the results approach infinity.

The indirect emulation method was more accurate than direct emulation because sparse sample sets are better at capturing global trends than local features. The vibratory response spikes were prominent local features found in the tested design spaces. The data, Tab. 5.2, used to calculate the vibratory response did not have the same prominent local features but had more dominant global trends. This is demonstrated in Fig. 5.13 which shows how the indirect emulation found
Figure 5.13: Maximum stress from vibratory emulation of the tested design space. Varying $C_M$ is shown while the other design variables were held constant.

The vibratory stress spike when $C_M$ is increased by about 10%. The true samples, created independently of the training data, also show the spike in stress near the indirect emulation of spike in stress. The figure also shows close approximation to the true samples globally, away from the spike. The direct emulation of vibratory stress did not find the spike in vibratory stress, which indicates poor local accuracy around most of the spikes in stress. Fig. 5.13 also shows a large difference between the true response and the direct emulation at other regions of the design space. At -15% mid-span chord, the true value was close to 10 ksi while the direct emulation method estimated about 75 ksi. In the training data used for this research, a single sample was placed close to a spike in vibratory stress found at a region of the design space not shown in this figure. This increased accuracy at the selected design, but decreased global accuracy. The data shown in Fig. 5.13 and Tab. 5.3 show that the indirect surrogate method was more accurate than the direct surrogate method and could provide DSE which captured spikes in vibratory response.
5.5 Discussion

These results have shown that the presented RANS-HMS analysis method was about ten times faster than the URANS-HMS method. This allows for ten times as many samples to be created for DSE in exchange for a relatively small increase in error. This increase in number of samples allowed a surrogate to be created for DSE. The results also showed how indirect emulation of vibratory response based on the RANS-HMS sample set was able to more accurately emulate vibratory response in the presence of resonance caused by mode frequencies aligning with forcing frequencies. When performing DSE with one or two design variables, plots like that of Fig. 5.12 can be made to understand how the modified Goodman values change over the design space. When more than two variables are explored, DSE becomes more difficult. The PCA-based DSE methods presented by Bunnell et al. will not work in this application because the vibratory responses are not directly emulated with principal components [95]. Instead, DSE should be performed by iteratively checking designs across the set of design variables. The surrogates are used in the process to quickly evaluate the modified Goodman values for each design. The use of surrogates allows for real-time exploration of the modified Goodman which improves DSE and the redesign process. This section continues with an example to show how manual, interactive DSE can be performed to find an improved design while accounting for modified Goodman. This will show the usefulness of the developed surrogate in DSE of fan blade vibratory responses. The example begins by discussing the nominal design. Then, a few iterations of DSE are presented. Each new design was manually selected through trial and error. Each iteration relies on the samples and surrogate methods discussed in the result section to quickly find an improved design.

5.5.1 DSE Example

Analysis of the nominal PBS Rotor 4 design was presented in the results section. The modified Goodman for this design was acceptable, less than 100% as shown in Fig. 5.9(c). Therefore, a proposed increase in the chord design variables and decrease thickness-to-chord ratio design variables is used to guide this DSE example. Such proposed changes often originate from aero-dynamic groups seeking to increase the fan blade performance. The structural groups will then explore the structural responses around the proposed designs. The DSE iterations below will show
evaluation of the vibratory response at several designs in the proposed region of the design space. This is shown by iteratively changing the values of the design variables and checking the vibratory response.

**Iteration 1**

We begin by decreasing $TC_M$ by 15%. The resulting modified Goodman values are shown in Fig. 5.14(a). The maximum modified Goodman value was 36%, indicating this design would also have infinite HCF life.

**Iteration 2**

The next modification started with the design from iteration 1 and explored increasing $C_M$ by 15%. The modified Goodman values are shown in Fig. 5.14(b). The figure shows that this design would not have infinite HCF life. This is shown by the regions of the blade which indicate, in red, a modified Goodman value of greater than 100%. The maximum modified Goodman on the blade was 393%. This design is close to a spike modified Goodman found near resonance. DSE then continued to find an acceptable structural design.
Iteration 3

The next design started with the iteration 2 design and explored a decrease of $C_M$ to only 8% above nominal. The modified Goodman values are shown in Fig. 5.14(c). This design would have infinite HCF life because the maximum modified Goodman on the blade was 47%.

Subsequent Iterations

With the use of the surrogate, less than a minute was required to evaluate all three of these design iterations. The design example showed that the indirect emulation method could be used to quickly explore the feasible boundaries of the design space during DSE. The structural groups can quickly evaluate the proposed designs and also report on the boundaries of feasible designs. This means further, and better, design iterations could be performed between the structural and aerodynamic groups. This was possible because the reduced computational cost of the RAN-HMS method allowed enough samples to be evaluated for surrogate creation. The designs which were infeasible due to large spikes in vibratory response near resonance were captured with the indirect surrogate method. The developed methods improve the ability of designers to find quality designs within tight time constraints.

5.6 Conclusion

This paper presented new and novel analysis and surrogate modeling methods which allowed for more accurate DSE of turbomachinery blades under vibration. The novel analysis method, RANS-HMS, used steady-state CFD to evaluate the pressures on the surface of the blade. The unsteady pressures were then estimated from these steady state pressures. This research assumed the unsteady pressures were the steady-state pressures scaled to the magnitude of an inlet distortion. The unsteady pressures were used in HMS to solve for the vibratory stress and modified Goodman. The novel surrogate method developed indirectly solved for the vibratory responses by emulating the unsteady pressures, mode displacements, mode stresses, and mode frequencies. These were then used in DSE to calculate the vibratory responses. This indirect emulation method allowed for capture of the vibratory response spikes found through the turbomachinery design spaces.
It was found that the RANS-HMS analysis method provided an estimation of vibratory response within 10% error of the higher fidelity URANS-HMS method used by Bakhle et al. for a 90% reduction in computational cost [33]. A direct emulation of vibratory response was unable to emulate the vibratory spikes found when the mode frequencies were close to the forcing frequencies. The indirect emulation of vibratory response, however, was able to capture the spikes found near resonance. It also allowed for better global accuracy of vibratory response with an error of about 17% on average compared to the direct emulation error of about 85% on average for vibratory stress. These errors were achieved with only 47 samples in a six dimensional design space. This shows that DSE accounting for vibratory responses may be performed within limited time and computational resources.

An example was given which followed an iterative design process. It was shown that the design space could be quickly explored to find an improved design. This could happen while identifying designs which are structurally unsatisfactory. This developed methodology offers engineers and engine manufacturers the ability to find, test, and offer better designs with less chance of needing an expensive redesign. The final design was found from exploration of a proposed design change while keeping the fan blade within structural limits. Such improvements can have long-term benefits for engine operators as they seek to increase fan aerodynamic performance while keeping the design within structural limits.
CHAPTER 6. MULTI-FIDELITY SURROGATES BASED ON SHARED PRINCIPAL COMPONENTS

This objective of this research was to develop multi-fidelity (MF) methods which decrease the computational cost required to create emulators with improved accuracy. This was achieved by using MF surrogates based on PCA. The method relies on the assumption that the PCs are common among the fidelities. The contents of this chapter are from a paper submitted for publication in the journal of Structural and Multidisciplinary Optimization. While this chapter only presents examples of static stress, the method is also applicable to vibratory stress. The work in this chapter also allows for the PCA-based DSE methods for static and vibratory stress developed in the previous chapters. The contribution of the research is provided in the following manner.

- Develop a multi-fidelity surrogate for DSE of finite element models without the need for a common grid.
- Quantify the reduction in error.
- Quantify the reduction in computational cost.

6.1 Introduction

Despite increasing computational power available to engineers, computational cost of high-fidelity (HF) simulations limit the number of evaluations which may be performed in an acceptable time frame during design space exploration (DSE). DSE is the study or analysis of designs within a range of specific variables. Optimization is a form of DSE which seeks to find maximum or minimum objectives in a design space subject to constraints. Interactive DSE is another form of DSE in which the designer iteratively explores different design options to understand how the parameters influence the response. Limiting the number of evaluations performed during DSE, whether for optimization or interactive DSE, reduces its effectiveness. Fewer iterations means optimization is
less likely to approach the true optimum and interactive DSE is less able to thoroughly explore the design space. In turn, this reduces the quality of engineering designs.

Many have turned to surrogate modeling to reduce the computational cost needed to perform a large number of evaluations [64–67, 112]. Surrogate models use a limited number of samples to construct a mathematical estimate of the model’s response [8]. Once the samples are collected, estimation of a new response with the surrogate may be obtained in fractions of a second [62]. Chen et al. discussed the benefits of surrogate assisted design exploration, concluding that benefits vary depending upon the type of surrogate model and its application [113]. Application of well selected surrogate modeling to optimization and design exploration have resulted in better optima and better understood design spaces [43, 64, 65].

A challenge of using surrogates in DSE is that the error to the HF response is related to the number of samples used to train them [62]. For some engineering applications, the computational cost of evaluating the HF response is too high to create enough samples for acceptable surrogates [19]. To address this challenge, MF methods have been developed to lower computational cost and error to create and use surrogates [8, 114]. These methods involve creating many lower-fidelity samples to supplement the few HF samples. The low-fidelity (LF) response has some error to the HF response, but the samples are obtained at much less computational cost. The MF surrogates take advantage of the better coverage of the design space with the LF samples, despite their inherent error [2, 43]. Such methods have been found to decrease the computational cost, reduce surrogate error, or both [2, 115–117].

Design exploration and optimization of compressor blades is an application in which the computational cost of HF finite element analysis (FEA) limits the number of samples which may be used in creating surrogate models. Bunnell et al. evaluated single-fidelity (SF) surrogate models of the Transonic Purdue Research Compressor (TPRC) as a means to decrease the computational cost of evaluating the structural results [62]. They evaluated the computational cost of creating FEA samples for the surrogates, the computational cost of evaluating the surrogates, and the error of the surrogates in various design spaces. They produced a sufficient amount of samples to create surrogates with low enough error to warrant their use in DSE. However, some of their design spaces still had between 10-15% error with 80 samples, the maximum number they tested. Because error decayed exponentially with the number of samples, exponentially more samples
Table 6.1: Set of possible design variables for Transonic Purdue Research Compressor used for this research.

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Symbol</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>$H$</td>
<td>± 10% of nominal $H$</td>
</tr>
<tr>
<td>Ave. Radius</td>
<td>$R_o$</td>
<td>± 10% of nominal $R_o$</td>
</tr>
<tr>
<td>Root Chord</td>
<td>$C_R$</td>
<td>± 10% of nominal $C_R$</td>
</tr>
<tr>
<td>Tip Chord</td>
<td>$C_T$</td>
<td>± 10% of nominal $C_T$</td>
</tr>
<tr>
<td>Sweep</td>
<td>$S$</td>
<td>± 10% of nominal $C_T$</td>
</tr>
<tr>
<td>Lean</td>
<td>$L$</td>
<td>± 10% 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>Pressure Load</td>
<td>$P$</td>
<td>± 10% of nominal $P$</td>
</tr>
</tbody>
</table>

would be required to significantly decrease error. Whether for interactive DSE or optimization, using surrogate models to emulate structural results allows more designs to be evaluated in less time. Further decreasing error improves effectiveness in optimization and design exploration. This would lead better structural designs that are more thoroughly understood.

This paper presents a novel MF surrogate method to significantly decrease the error of surrogates based on finite element analysis for a relatively small increase in computational cost. The method is based on the HF and LF samples sharing the same set of principal components and is referred to as the shared-principal component (SPC) MF method. It is tested with the von Mises stress of finite element models based on the same TPRC model and design spaces used by Bun nell et al. [62]. The blade row of the compressor is shown in Fig. 6.1, however, only a blade-alone model, shown in Fig. 6.2, is used. The list of design variables used in these tests is given in Tab. 6.1. Each variable either changes the geometry or loads, structural and pressure, on the nominal geometry shown in Fig. 6.2. The paper continues with a background section which describes common and more advanced MF methods currently used in research. Then, construction of the SPC MF surrogate is presented in the methods section. Its error and computational cost are given in the results section. The error and computational cost of the SPC method are compared to those of SF surrogates and other MF surrogates to validate that the new approach is an improvement. Finally, the conclusion reviews the results and discusses the benefit of using SPC MF surrogates for design exploration and optimization.
6.2 Background

Several MF surrogate methods have been developed to improve surrogate model accuracy for design exploration and optimization. Co-Kriging, as described by Forrester et al., is one of the most commonly referenced MF methods [8]. This method treats each sample as the result of a Gaussian stochastic process to train the surrogate. It emulates the LF response at the new design, multiplies it by a uniform scale factor, then emulates an additive correction at the new design. Forrester et al. show how this method is more accurate to the HF solution than using only the HF samples with a Kriging surrogate.
Song et al. compared MF surrogates based on radial basis functions (MF-RBF) to other forms of MF models [117]. The formulation by Song et al. followed the same emulation equation as described with Co-Kriging, but the weights were solved deterministically instead of as the result of a stochastic process. Song et al. compared this model to others including Co-Kriging, linear regression based MF models, and Co-radial basis functions (Co-RBF). They found that, for most cases, the MF-RBF version of the comprehensive method was the most accurate and least sensitive to the correlation between HF and LF. Kou et al. used the same surrogate method as Song et al. to create accurate emulators of unsteady flow around airfoils [118]. They found that the system was modeled with a reduction in error by using MF samples rather than with HF samples only. The addition of lower-fidelity samples led to better coverage of the design space which reduced the error. Their models reduced 35% to 98% of the error with an average of 75% error reduction.

Cai et al. discussed more simple MF surrogates, such as additive correction, multiplicative correction, and Space-Mapping [115]. While the LF and correction surrogates can take any form, this research refers to them as based on RBF. Each one relies upon an LF surrogate which is fit to the LF samples and separate correction surrogate is fit to correct the LF to the HF samples. The Space-Mapping method uses the LF result as the input correction surrogate. Cai et al. also discussed a more advanced model in which each fidelity has its own set of radial basis function weights. The final HF result is a summation of each fidelity’s RBF result. They found that this more advanced method reduced error by 75% in simple test cases and 32% in a more practical engineering application. They used it to optimize a marine propeller for fatigue life. The improvement was obtained because the addition of LF samples allowed the design space to be more thoroughly sampled. A possible challenge, however, with MF modeling is the increase in error when the samples are not well placed in the design space. This was found in the research presented by Guo et al. and Wang et al. [53, 119]. Cai et al. also found that the placement of the samples had a large impact on surrogate error [115]. They found a further decrease in error by using adaptive sampling methods which selected new samples at regions of the design space with higher error.

### 6.2.1 Principal Component Analysis

All of the methods discussed above emulate a single value. However, they may be extended to emulate every node on the finite element model as done with the SF surrogate used by Bunnell
et al. [62]. This allows for the entire result field to be emulated for improved optimization and design exploration [105]. Principal component analysis (PCA) is a more efficient method to emulate a field of results. PCA is a spectral method which creates a reduced order model from data with a large number of dimensions while maintaining a majority of the information [9]. PCA is also known in other fields as proper orthogonal decomposition (POD) [41–43]. The method finds vectors which represent orthogonal variations in a given data set. Because they are orthogonal, each variation is independent from one another and can assist in understanding the data. A change along one principal component (PC) does not cause a change along another. An example is shown with Fig. 6.3(a) which shows some data, as points, described by some original bases, $Y_1$ and $Y_2$. PCA was performed to find new bases, or PCs, shown as the vectors in Fig. 6.3(b) as $V_1$ and $V_2$. The projections of the data on the PCs are called the PC scores. For applications to FEA, the method transforms the data containing the stress at each node for each sample into PCs and scores for those PCs. These PCs are a set of orthogonal stress profile variations from the mean stress profile. The PC scores describe how much of each stress variation is present at each sample. Instead of creating a surrogate model which emulates each node, a PCA-based surrogate emulates the much smaller set of PC scores and uses the PCs to transform the scores back into the nodal result field. The use of PCA in surrogate models can reduce the computational cost and the error of the surrogate evaluation, furthering the effectiveness of optimization and design exploration.
Fig. 6.4: Workflow required for LF samples when surrogate requires common grid.

Benamara et al. and Mifsud et al. both introduced different PCA-based MF method which emulate a result field [43, 116]. Both methods combine fidelities into a single principal component analysis and use a surrogate to emulate the scores for the new design. Benamara et al. applied their method to optimize a compressor blade for total pressure ratio and isentropic efficiency [43]. Their method was able to find a better optimum than surrogates built on only the HF samples with 40% fewer iterations. The improvement was obtained because the use of LF samples allowed more samples to be collected which improved the accuracy of the surrogate. Mifsud et al. applied their method for DSE of three-dimensional flow fields [116]. The error of their model was acceptable, generally within 5%, but depended on high correlation between the HF and LF samples.

When emulating a result field, all of these discussed methods require that the LF samples and HF samples share a common grid. Using mesh density as the definition of fidelity, as done with MF FEA, requires additional analysis for the LF samples which increases computational cost. This is shown in Fig. 6.4. The LF mesh is first adjusted to the new design and used in analysis. Then, the HF mesh must also be adjusted to the new design. The results from the LF simulation are then interpolated onto the HF mesh. Because the HF mesh is not used in the simulation at the LF samples, the computational cost for the LF samples is lower than the HF samples. However, the computational cost of adjusting the HF mesh and interpolating the LF results onto that mesh is not negligible. A further significant decrease in computational cost could be gained if the LF and HF samples did not need to share a common grid.
Figure 6.5: Training of shared principal component surrogate model. The colors signify fidelity level associated with the data, process, or model. White is HF, dark-grey is LF, and light-grey is MF.

6.3 Methodology

This paper presents the novel SPC MF surrogate model for use in DSE of FEA models. This method is PCA-based and, unlike other methods, does not require a common grid between the fidelities. This decreases the computational cost to create samples. It will be shown that this method also significantly decreased surrogate error. The workflow for training this surrogate model is given in Fig. 6.5. First, PCA is performed on the HF and LF samples separately. To perform PCA, stresses for each node and sample are arranged into the data matrix, $\mathbf{Y}$, as shown in Fig. 6.6. This paper uses emulation of von Mises stress from FEA. The figure shows stress, $\sigma$, for the $n$ nodes on the finite element model with $s$ samples. The matrix $\mathbf{Y}$ has $s$ rows and $n$ columns. To perform PCA, the column means, $\bar{\mathbf{y}}$, are subtracted and singular value decomposition is performed to solve for $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$ in Eqn. 6.1. Multiplying $\mathbf{U}$ and $\mathbf{S}$ results in the matrix of PC scores, $\mathbf{A}$, shown in Fig. 6.7 and $\mathbf{V}$ contains the PCs. The matrix $\mathbf{A}$ has $s$ columns and $q$ rows, where $q$ is equal to the number of PCs. The matrix $\mathbf{V}$ has $q$ rows and $n$ columns. Because PCA is performed separately, each of the variables in Eqn. 6.1 has an LF and HF counterpart.

$$\mathbf{Y} - \bar{\mathbf{y}} = \mathbf{USV} = \mathbf{AV} \quad (6.1)$$

The HF and LF PCs, $\mathbf{V}_H$ and $\mathbf{V}_L$, are numerically different, but the method assumes the physical phenomena they describe are nearly the same. Thus, the PCs are shared between the fidelities. This means that the particular variation described by the first PC in the HF PC will
Finite Element Nodes

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Node 2</th>
<th>...</th>
<th>Node n</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_{1,1} )</td>
<td>( \sigma_{1,2} )</td>
<td>...</td>
<td>( \sigma_{1,n} )</td>
</tr>
<tr>
<td>( \sigma_{2,1} )</td>
<td>( \sigma_{2,2} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>( \sigma_{s,1} )</td>
<td></td>
<td></td>
<td>( \sigma_{s,n} )</td>
</tr>
</tbody>
</table>

Figure 6.6: The \((s, n)\) data matrix, \( Y \). Each row was a unique FEA sample, of which there were \( s \). The columns, of which there were \( n \), represented the stresses for \( n \) nodes on the finite element model.

Principle Components

<table>
<thead>
<tr>
<th>PC 1</th>
<th>PC 2</th>
<th>...</th>
<th>PC q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score 1, 1</td>
<td>Score 1, 2</td>
<td>...</td>
<td>Score 1, q</td>
</tr>
<tr>
<td>Score 2, 1</td>
<td>Score 2, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Score n, 1</td>
<td></td>
<td></td>
<td>Score n, q</td>
</tr>
</tbody>
</table>

Figure 6.7: The \((s, q)\) PC score matrix, \( A \). Each row was one of \( s \) samples. Each column contained the score for a particular PC, of which there were \( q \).

also be found in the first PC of the LF PC. This assumption allows PCA to be performed on each fidelity separately which, in turn, allows each fidelity to have different meshes. If the assumption is true, the PC scores between the HF and LF would share the same trends across the design space, making MF modeling more accurate [116]. This assumption is investigated with Fig. 6.8 which shows the first four PCs in a design space which includes all of the variables in Tab. 6.1. A direct
Figure 6.8: The first four PCs of the Purdue blade in a design space with the nine design variables given in Tab. 6.1. These four PCs describe 99% of the stress variation in the design space. The top row are the PCs of the HF model (25,000 nodes, 35 samples) and the bottom are the PCs of the LF model (500 nodes, 1000 samples). The color scale shows stress variation with stress increase as red and stress decrease as blue.

Numerical, quantitative comparison between the PCs would require interpolation to a common grid, therefore a graphical, qualitative comparison will serve to validate the assumption. The HF model in the figure used 25,000 nodes and the LF model used 500 nodes. Each column is a different PC ordered, from left to right, by decreasing amount of variation each PC explains. Each PC describes a particular variation from mean stress across the design space. Red areas indicate increased stress and blue is decreased stress. The figure shows the PCs are similar between HF and LF. The LF PCs describe, in general, similar stress variation patterns as the HF PCs, but with a coarse grid. In some instances, the PC of the LF may represent the inverse of the HF, as with the third PC. Because these PCs are vectors which describe variation from the mean, the LF PC and its scores may be multiplied by -1. Doing so is not necessary but was found to have a slight decrease in surrogate error. Validation of the assumption that the fidelities share PCs means the LF PCs are not needed, but the scores of the LF PCs are kept. The HF PCs are a higher-fidelity representation of the same phenomena.

Training the SPC MF surrogate method continues after the shared-PC assumption has been validated. Next, the HF and LF PC scores, $A_H$ and $A_L$, are used to create a secondary MF surrogate, $y_S(x)$, which uses both fidelities to emulate HF scores. The secondary surrogate can take the form
of any non-PCA-based MF method. This research tests the non-PCA-based methods discussed in section 2. They were based on RBF with a multiquadric kernel. This secondary surrogate adjusts emulation of the LF scores, $a_L(x)$, to emulation of HF scores, $a_H(x)$.

Fig. 6.9 shows the process to use the SPC MF model to emulate the HF structural response at a new design, $x$. First, the secondary surrogate is used to emulate the HF PC scores at the new design as shown in Eqn. 6.2.

$$a_H(x) = y_s(x) \quad (6.2)$$

The emulated HF scores are used with the HF PCs, $V_H$, obtained from the training process in Fig. 6.5, to solve for the HF result field through an inverse transform. This is done by multiplying the emulated HF PC score vector, $a_H(x)$, with the HF PCs, $V_H$, and adding the HF column mean, $\bar{y}_H$ as shown in Eqn. 6.3.

$$y(x) = a_H(x)V_H + \bar{y}_H \quad (6.3)$$

### 6.4 Results

The SPC MF method was tested on 180 design spaces of the TPRC. This compressor blade was developed to study transonic aerodynamic effects and has been the subject of many academic studies including structural PCA analysis [56, 57]. The nominal design has a height of 2.0 in (5.1 cm), a mean chord of 1.9 in (4.8 cm), and a mean thickness-to-chord ratio of 8%. The HF finite
element model had about 25,000 nodes. The tested design spaces had 10 to 100 samples and 1 to 9 design variables. The selected samples for each design space were chosen with Latin hypercube sampling which seeks to evenly spread the samples through the design space [8]. The variables for each design space were randomly selected from those shown in Tab. 6.1. The design spaces and the HF finite element model were similar to those used by Bunnell et al. [62]. In addition, a matching set of LF samples were created for each HF design space. The number of LF samples was between 100 and 1,000, a range which was an order of magnitude greater than the HF samples. The LF finite element model had about 500 nodes. Testing was performed on the SPC method and compared against the MF methods described in section 2. Each of the non-PCA-based MF surrogates discussed in section 2 were also tested as secondary surrogates in the SPC method.

Training a surrogate resulted in one of three outcomes: numerical instability which failed to train, surrogates which trained but produced unreasonably high error, or surrogates trained and produced reasonable results. Most methods produced reasonable results with more than 95% of the tests. However, the SPC method with a Co-Kriging secondary surrogate failed to train about 15% of the tests. The method by Cai et al. failed to train with about 12% of the design spaces. These failures were due to the formation of ill-conditioned matrices in training. The multiplicative and comprehensive methods, as the main surrogate or the secondary surrogate produced unreasonably high errors with about 20% and 5% of the design spaces. The method by Mifsud et al. also produced unreasonably high errors with about 5% of the design spaces. These methods use some form of scaling in their training. When the emulated values were very small at a few samples, the scale factor became very large, leading to inaccurate surrogates. Surrogates which failed to train did not affect the surrogate errors presented in this research because they could not be evaluated. The surrogates which produced unreasonably high errors, however, did affect the mean errors obtained among the tests. Due to this, the medians of the tests are also presented.

### 6.4.1 Surrogate Error Comparison

Surrogate error was measured by comparing the values emulated by the surrogates to a separate set of test data. This test data was based on HF analysis and created with the same design spaces as the training data. Each test set had 50 samples selected with Latin hypercube sampling [8]. These samples were chosen independently of those used in the HF training data. The
difference between the surrogate estimations and the test samples yield an error for every node on
the finite element model at every test sample. Double normalized root mean square error (double-
NRMSE) is used to obtain a single value for surrogate error. This metric, shown in Eqn. 6.4, is
the equivalent of normalized root mean square error over the set of test samples and over the set
of nodes on the finite element model. In this equation, $\sigma_{i,j}$ is the stress at node $i$ and test sample $j$
while $\hat{\sigma}$ is the same stress estimated by the surrogate. The maximum stress among all samples in
the test set is represented by $\sigma_{max}$, the number of HF nodes is $n$, and the number of samples in the
test set is $t$.

$$e = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{t} (\hat{\sigma}_{i,j} - \sigma_{i,j})^2}{nt\sigma_{max}^2}} \cdot 100\% \quad (6.4)$$

The error decreased exponentially with the number of LF samples, as shown with the ex-
ample in Fig. 6.10, and leveled off at some lower limit. This research defined “converged number
of LF samples” as the lowest number of LF samples which resulted in error with 10% of the final
value. This research defined “converged error” as the error at the converged number of LF sam-
ples. The converged errors and converged number of LF samples are shown as plus symbols in
Fig. 6.10. The ratio between this error and the error of the SF method is also presented as the error
ratio, Eqn. 6.5. Lower ratios mean better surrogates.

$$e_{ratio} = \frac{e}{e_{SF}} \cdot 100\% \quad (6.5)$$

The error ratios for the MF surrogate methods are shown in Tab. 6.2 and Tab. 6.3. The
shown error ratios in these tables are the means and medians for all of the tested 180 design spaces.
The first column contains the names of the different MF surrogates, the second column contains
the mean error ratio among the tested design spaces, and the third column contains the median
error ratio among the tested design spaces. Tab. 6.2 gives the error for the non-SPC MF methods.
The best of these methods, and the only one to have a mean reduction in error, was the method
presented by Benamara et al. with a mean error ratio of 78.3% and median error ratio of 74.5%. The
table also shows how the means of the multiplicative, Mifsud et al., and comprehensive methods
are very high. This was due to a small number of surrogates which failed to train with reasonable
accuracy. Errors greater than 100% mean that the MF method had error more than twice that of the
Figure 6.10: Double-NRMSE of surrogates with varying number of LF samples. The surrogates were trained on the Transonic Purdue Research Compressor design space with all of the design variables in Tab. 6.1 and 35 HF samples. Plus indicates the converged error.

Table 6.2: Error ratio of surrogates over all tested design spaces. These methods are those in the background section.

<table>
<thead>
<tr>
<th>Surrogate</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benamara et al.</td>
<td>78.3%</td>
<td>74.5%</td>
</tr>
<tr>
<td>Cai et al.</td>
<td>109.4%</td>
<td>82.4%</td>
</tr>
<tr>
<td>Additive</td>
<td>109.5%</td>
<td>80.6%</td>
</tr>
<tr>
<td>Multiplicative</td>
<td>680.5%</td>
<td>163.2%</td>
</tr>
<tr>
<td>Comprehensive</td>
<td>&gt; 1000%</td>
<td>77.2%</td>
</tr>
<tr>
<td>Mifsud et al.</td>
<td>&gt; 1000%</td>
<td>94.8%</td>
</tr>
</tbody>
</table>

SF surrogate. Exceptionally high mean errors were often due to a few tests which did not create surrogates with any reasonable accuracy. Where this was the case, comparing the medians also helps understand the error associated with each method. An SF RBF surrogate, like the one used by Bunnell et al., had a mean double-NRMSE of 2.8% and median error of 2.4% [62]. Using the method by Benamara et al. reduced the error, on average, to 78% of the single fidelity surrogate.

Tab. 6.3 gives the error for the SPC method. The table includes different secondary surrogates in the first column. Some of the methods in this table also had abnormally high means due to surrogates which failed to train with reasonable accuracy. The best SPC method was that which
Table 6.3: Error ratio of surrogates over all tested design spaces. These use SPC method, shown in Fig. 6.5 and Fig. 6.9, with various secondary surrogate methods.

<table>
<thead>
<tr>
<th>Secondary Surrogate</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-Kriging</td>
<td>44.9%</td>
<td>41.4%</td>
</tr>
<tr>
<td>Space-Mapping</td>
<td>60.3%</td>
<td>49.1%</td>
</tr>
<tr>
<td>Additive</td>
<td>100.6%</td>
<td>96.3%</td>
</tr>
<tr>
<td>Cai et al.</td>
<td>100.8%</td>
<td>96.4%</td>
</tr>
<tr>
<td>Comprehensive</td>
<td>&gt; 1000%</td>
<td>72.1%</td>
</tr>
<tr>
<td>Multiplicative</td>
<td>&gt; 1000%</td>
<td>&gt; 1000%</td>
</tr>
</tbody>
</table>

used Co-Kriging as the secondary surrogate. The SPC method with the Space-Mapping subsurrogate is also discussed because of the large number of failed surrogates based on Co-Kriging. When comparing both tables it can be seen that SPC method with the Co-Kriging and Space-Mapping secondary surrogates had the lowest mean and median error ratio among all of the tested methods. The Co-Kriging based SPC method had a mean error ratio of 45.5% and a median error of 41.7% through all of the tested design spaces. The Space-Mapping based SPC method had a mean error ratio of 61.6% and a median error of 49.4% through all of the tested design spaces.

Paired t-tests between the method by Benamara et al. and the SPC method showed a statistically significant difference in error ratio with a p-value of less than 0.0001. Comparing SPC with the Co-Kriging secondary surrogate and the method by Benamara et al. produced an estimated 28% difference in error ratio. Comparing SPC with the Space-Mapping secondary surrogate and the method by Benamara et al. produced an estimated 18% difference in error ratio. This difference is not only statistically significant, it is also practically significant. The SPC methods based on Co-Kriging and Space-Mapping both had a large number of tests which reduced the error by more than 50% of the SF surrogates. Tab. 6.4 shows that the SPC methods based on Co-Kriging and Space-Mapping reduced the error to below 50% on more than half of the tests. For comparison, the method by Benamara et al. only achieved that on 7% of the tests. These results show that using the SPC method provides a practically significant and statistically significant reduction in surrogate error. This is due to the shared principal component assumption which was shown and validated.
Figure 6.11: Double-NRMSE of MF surrogates compared to HF surrogates. The solid dark line is the estimated regression fit of the data. The slightly lighter, narrow band is the 95% confidence interval for the regression fit. The lightest, thickest band is sample 95% prediction confidence interval.
Table 6.4: Percent of tests in which the surrogates lowered the error by more than 50%.

<table>
<thead>
<tr>
<th>Surrogate</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPC with Co-Kriging</td>
<td>68%</td>
</tr>
<tr>
<td>SPC with Space-Mapping</td>
<td>65%</td>
</tr>
<tr>
<td>Benamara et al.</td>
<td>7%</td>
</tr>
</tbody>
</table>

The most significant factor which influenced each method’s error was the error of the SF model. Fig. 6.11 shows how the error of the SPC method with Co-Kriging and Space-Mapping secondary surrogates as well as the method by Benamara et al. change with respect to the HF surrogate error. The solid dark line is the estimated regression fit of the data. The light, shaded area is sample 95% confidence bound. Any new sample has a 95% probability of being within the shaded area. The SPC method with Space-Mapping as the secondary surrogate and the method by Benamara et al. both had linear relationships between HF error and the MF error. The slopes of those lines were 0.39 and 0.67 respectively, both with p-values of less than 0.0001. This means that the least squares regression estimates that the methods had error which was 39% and 67% of the HF error. The SPC method with Co-Kriging had a quadratic relationship to the HF error. The least squares estimate was 9.54 times the square of the HF, with a p-value of less than 0.0001. The estimates for the method by Benamara et al. and the SPC method based on Space-Mapping indicate that the mean of the Space-Mapping based method will always be less than the mean of the method by Benamara et al. However, the SPC method based on Co-Kriging had a lower mean error than that based on Space-Mapping when the HF method is below about 4% double-NRMSE. When the HF method has an error above 4% double-NRMSE, the error of the Co-Kriging based SPC method quickly increases. Fig. 6.11 also shows that the SPC method based on Space-Mapping has more scatter in the data at low SF error than the other methods. This means that, with SF error less than 1%, the reduction in error will not always be as good as the other methods.

6.4.2 Computational Cost

The computational cost to create LF samples was found to be much less than that of HF samples. In theory, the LF samples used in this study should be greater than 2,500 times faster to
Table 6.5: Computational cost of sample creation for different fidelities used in this research.

<table>
<thead>
<tr>
<th>Fidelity</th>
<th>Seconds</th>
<th>Ratio to HF</th>
</tr>
</thead>
<tbody>
<tr>
<td>HF</td>
<td>7.41</td>
<td>100.0%</td>
</tr>
<tr>
<td>LF</td>
<td>0.65</td>
<td>8.8%</td>
</tr>
<tr>
<td>LF with Mapped Mesh</td>
<td>3.03</td>
<td>40.9%</td>
</tr>
</tbody>
</table>

solve than the HF samples. This is based on the squared to cubic relationship between number of nodes and computational cost of FEA. In practice, the computational cost improvement is not as great due to the cost of starting and initializing the simulation software used to create the samples. The HF and LF samples used in this research were created on the same computer with 16GB of RAM running on a 3.70 GHz processor. The computational cost to create the samples was compared with the wall-clock time needed to create the samples, shown in Tab. 6.5. LF samples were created at an average rate of 0.65 seconds per sample while the HF samples were created at an average rate of 7.41 seconds per sample. Included in these measurements are the time to adjust the finite element mesh to match the new geometry and the time to solve the finite element analysis. This leads to LF samples which were about 8.8% of the computational cost of the HF samples. In other words, the LF samples were more than 11 times faster to solve than the HF samples. Lower computational cost of the LF samples leads to better cost saving and error reduction from MF models. The 8.8% found in this application is within the 10% or less range found by Park et al. to make MF surrogates reduce the computational cost [120]. The additional computational cost of adjusting the HF mesh and mapping the LF samples onto that mesh was about 2.38 seconds per sample. The total time for LF samples was 3.03 seconds per sample when using a common mesh. That brings the computational cost of LF samples to 40.9% of the HF samples, or about 2.5 times faster than the HF simulation. This makes them much less effective for MF modeling.

The next factor in determining computational cost was the number of LF samples needed to reach converged error. The SPC method with a Co-Kriging secondary surrogate required the lowest number of LF samples, with an average of 125 samples. The SPC method with a Space-mapping secondary surrogate required an average of 282 samples and the method by Benamara et al. also required an average of 282 samples. A paired t-test gave no statically significant difference in the number of LF samples needed to converge when using SPC with Space-Mapping or the method
Table 6.6: Computational cost to achieve a surrogate model with 5% double-NRMSE. The design space is the Transonic Purdue Research Compressor with all variables in Tab. 6.1.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (minutes)</th>
<th>Ratio to SF</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF</td>
<td>8.17</td>
<td>100%</td>
</tr>
<tr>
<td>Benamara</td>
<td>7.87</td>
<td>96%</td>
</tr>
<tr>
<td>SPC with Space-Mapping</td>
<td>1.85</td>
<td>23%</td>
</tr>
<tr>
<td>SPC with Co-Kriging</td>
<td>5.72</td>
<td>70%</td>
</tr>
</tbody>
</table>

Table 6.7: Double-NRMSE for a fixed computational cost of 10 minutes. The design space is the Transonic Purdue Research Compressor with all variables in Tab. 6.1.

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
<th>Ratio to SF</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF</td>
<td>4.61%</td>
<td>100%</td>
</tr>
<tr>
<td>Benamara</td>
<td>4.68%</td>
<td>102%</td>
</tr>
<tr>
<td>SPC with Space-Mapping</td>
<td>2.65%</td>
<td>57%</td>
</tr>
<tr>
<td>SPC with Co-Kriging</td>
<td>2.51%</td>
<td>54%</td>
</tr>
</tbody>
</table>

by Benamara et al. A paired t-test did show a statistically significant difference in the number of LF samples needed to converge when using SPC with Co-Kriging. Most tests converged with Co-Kriging with only 100 LF samples. Fewer samples leads to less computational cost.

Because the SPC method was found to be more accurate in this application, fewer samples are required to achieve a target level of error. The computational cost to achieve a surrogate with 5% double-NRMSE is given in Tab. 6.6 to show the benefit of using the SPC method. The table shows the time in minutes required to create enough samples for the 5% double-NRMSE in a design space with all nine design variables in Tab. 6.1. The first surrogate is an SF RBF based only on HF samples. Then, the table shows the method used by Benamara et al. because it was the most accurate of the methods discussed in the background. The SPC with Space-Mapping as the secondary surrogate required the least computational cost to achieve 5% error. For the same surrogate error the computational cost was reduced to 23% of the SF method. The method by
Figure 6.12: Double-NRMSE with varying computational cost in minutes. The design space is the Transonic Purdue Research Compressor with all variables in Tab. 6.1

Benamara et al. only reduced the cost to 96% of the SF method. The smaller cost reduction is from using a common grid and the higher error is due to not being able to produce as many samples in the allotted time. The SPC method with Co-Kriging, while it was shown to be the most accurate for the fewest amount of LF samples, requires more computational cost to train. This is due to the optimization iterations required in training which led to computational time which was only 70% of the SF method. The SPC method with Co-Kriging provided less of an improvement compared to SPC with Space-Mapping but more than that of Benamara et al.

Instead of fixing error, a more likely scenario would be creating surrogates with the lowest error for fixed computational cost. The level of error achieved with ten minutes of computational cost, chosen due to the times found in Tab. 6.6, is shown in Tab. 6.7. The table shows that, given ten minutes of computational cost, the method by Benamara et al. was no longer more accurate than the SF surrogate for this design space. The decrease in error given computational cost is shown in Fig. 6.12. For this design space, the SF surrogate error approached the error of the surrogate method by Benamara et al as computational cost increased. They had the same error at eight minutes of computational cost. The SPC method with the Space-Mapping secondary surrogate, however, maintained a significant decrease in error from the SF surrogate through the entire range. For one minute of computational cost the SPC method reduced error by 36% from the SF error. By ten minutes the error reduction was up to 43%. The SPC method with the Co-Kriging secondary
surrogate started with a much higher error because a large portion of the computational cost was spent on training the surrogate instead of creating more samples. As the allowed computational cost increased, a larger percent of the cost was spent on creating samples. This led to a much steeper decrease in error given a set amount of cost. By ten minutes of computational cost, the Co-Kriging based method produced had reached the same error as the Space-Mapping based method.

### 6.4.3 Replication with State-of-the-Art Blade

The SPC method with Space-Mapping and Co-Kriging secondary surrogates was also tested to a design space of PBS Rotor 4 to validate the method. The results shown thus far use the Transonic Purdue blade. It was used in this research due to its common usage in other methods based research and its relatively low computational cost for sample creation [62]. Such low computational cost allowed a large number of design spaces to be evaluated. PBS Rotor 4 is a fan blade which is more representative of blades being designed for current and future engines [102]. This HF finite element model had about 220,000 nodes while the LF finite element model had about 25,000 nodes. There were twelve tested design spaces which included the chord at three spans and the thickness-to-chord ratio at those same spans for a total of six variables. Each variable’s range was from 10% below to 10% above the nominal value. These variables and ranges are common for structural design exploration and optimization. There were between 25 to 50 HF samples in this test which were obtained at a computational cost of 365 seconds per sample, on average. Each LF sample was obtained at a computational cost of 73 seconds per sample, on average. Tab. 6.8 shows the error ratios of the SPC-method with Space-Mapping and co-kriging secondary surrogates. With between 100 to 500 LF samples to converge, the error was still reduced by a significant amount for this more advanced blade. The SPC method with the Space-Mapping and Co-Kriging secondary surrogates reduced error, on average, to 73% and 53% of the SF surrogate. Thus, the SPC method, was again able to improve the accuracy of the model. With the Co-Kriging secondary surrogate, the SPC method was again able to cut the error in half on many of the tests, but failed to train about a fifth of the tests. The mean SF error was 0.5%. Because it was below 1% error, the findings from the study on the Transonic Purdue blade suggest that the Co-Kriging based SPC method should be more accurate than that based on Space-Mapping. This shows that the results found using the Transonic Purdue blade may also be found on more advanced blade designs.
### 6.5 Conclusion

This research presented the shared principal component MF surrogate model in order to reduce the computational cost of creating surrogates with low error. This method assumed that principal components in the LF samples were similar to the principal components of the HF samples. This assumption was validated and allowed PCA to be performed on the LF and HF samples separately. Doing PCA separately saved additional computational cost by not requiring a common grid between fidelities. A secondary MF surrogate was used to emulate the HF scores from the mixed sample set of HF and LF scores. These emulated HF scores were then used with the HF PCs to emulate a result field.

The SPC method, with various secondary surrogates, was compared to other common and novel MF surrogates. The novelty of the method was based in not requiring a common grid between the fidelities and yet emulating the entire HF result field. The results showed that the SPC MF method decreased computational cost of creating surrogates and decreases the error of the surrogates more than other MF methods. Space-Mapping or Co-Kriging secondary surrogates had lower error than the other MF methods which were tested. They were able to decrease surrogate error by 60% and 45%, on average, for only a few hundred LF samples. By fixing a target error of 5% double-NRMSE the Space-Mapping based SPC method decreased the computational cost to 23% of the SF model. Because the non-SPC methods needed a common grid they were not as capable in error reduction for a fixed computational cost. The best non-SPC method only decreased computational cost to 96% while targeting 5% double-NRMSE.

The findings indicate that SPC with either Space-Mapping or Co-Kriging secondary surrogates should be used to emulate FEA. In an attempt to achieve the lowest error for lowest computational cost SPC with Co-Kriging and Space-Mapping should both be attempted for use as secondary surrogates. The findings suggest that Co-Kriging will likely be more accurate. If the

---

**Table 6.8: Error ratio of surrogates with the PBS Rotor 4 design space.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPC with Space-Mapping</td>
<td>73%</td>
<td>79%</td>
</tr>
<tr>
<td>SPC with Co-Kriging</td>
<td>53%</td>
<td>46%</td>
</tr>
</tbody>
</table>
Co-Kriging based method fails to train, as was found in about a fifth of the tests, or if the Space-Mapping based SPC model is more accurate, then the Space-Mapping based SPC model should be used. No additional computational cost will be accrued by checking the Space-Mapping based SPC model because training the Space-Mapping based method required negligible computational cost.

Using the presented SPC MF method for emulation of FEA can reduce the computational cost to create surrogates with low error. This helps overcome the challenge presented to engineers during optimization and design exploration with HF simulations. The lower error for less computational cost makes optimization and design exploration more effective. This in turn leads to better optima and better understood design spaces.
CHAPTER 7. CONCLUSIONS

New design methods must be implemented to meet the demand for improved gas turbine engines while keeping within desired structural limitations. To stay competitive and keep up with global demand, engine manufacturers must continue to innovate and improve upon existing turbomachinery designs. Improving these designs is non-trivial and requires new design space exploration methods. Two of the biggest challenges to exploring turbomachinery design spaces are the high computational cost of evaluating designs and knowing which designs to evaluate. The first challenge has been shown to not only effect the time it takes to obtain a good solution, but it also affects the quality of the final solution. Research found that a delay as low as 1.5 seconds between user input and visual indication of the results led to a 50% increase in the time required to accomplish the task [4]. Each claimed that delays increased both the time to find a satisfactory result and the difference of the result to some known optimum [4–7]. Engineering experience is often used to solve the second challenge, but this is difficult with new and state-of-the-art blade designs where the design spaces are unfamiliar. This dissertation addressed both of these challenges by implementing surrogate modeling and PCA-based methods to reduce the computational cost of evaluating structural designs and assist the designer in understanding how the design variables relate to the structural responses.

7.1 Research Contributions

Four research objectives were proposed in the introduction of this dissertation. The challenge of high computational cost to obtain a large number of design evaluations was addressed by the first research objective, given in Chapter 3. Surrogate methods were developed to make faster estimations than previous methods, of the response from a limited set of training samples. These were implemented for DSE of turbomachinery blades. The newly developed methods included the use of mesh morphing to create surrogates of finite element nodes and evaluating the errors of using
RBF with multiquadric kernels to explore compressor blade design spaces. Both node-based and PCA-based surrogates were used. Both were shown to be able to produce accurate surrogates with real-time evaluation. This research defined real-time as evaluations which required less than 0.017 seconds, or faster than the refresh rate of a 60Hz monitor. The research tested the effect of number of nodes, number of samples, number of design variables, and number of principal components on the emulation speeds. It was found that the emulation speed was dependent upon the number of nodes from the finite element model and the number of samples or number of principal components used to train the model. The number of design variables was not found to have an effect. The same factors tested for their effect on error. The surrogate error was found to be related to the number of samples used to train the model and the design variables in the design space. The number of nodes was not found to have an effect. The majority, about 90%, of the principal components could be discarded without an effect on error. The errors of the node-based and PCA-based surrogates were similar. The PCA-based surrogates, however, were able to emulate the response in half the computational cost to the nodal-surrogates for little to no decrease in surrogate error.

As presented in Chapter 3, the fast emulation of surrogates allow the designer to see and visually gain understanding of how those two parameters affect the full system response of the compressor blade. Designers may interact with the surrogates to visualize the structural trends in their designs or use the surrogates for optimization. Even where the solution is not solving real-time, the exploration of the design space is still much faster and therefore more feasible. By showing that the surrogates were within 10%, this work provides confidence to designers that these methods may be implemented to improve DSE in real applications.

The challenge of knowing which designs to evaluate addressed the second research objective, given in Chapter 4. Methods were developed to use PCA to explore the structural design space of turbomachinery blades. These methods utilized the information available through the PCA-based surrogates. A novel use of two-point correlation as well as the development of stress and geometry coupling was presented. These methods related the scores of the PCs to the design variables to indicate, for interactive DSE, which designs should be explored. Two-point correlation indicated this relationship by performing a cross-correlation between the scores of each PC and the values for each design variable. The variables which resulted in higher correlations were those which could be used to more effectively explore design spaces. Stress and geometry coupling
related these by viewing geometric portion PCs from coupled PCA. This provided understanding
to how blade geometry changes related to specific changes in stress. An iterative design explo-
ration workflow was presented to guide a designer through exploring the design space with these
methods. It was shown how PCA in DSE with two-point correlation or stress and geometry cou-
pling could be used to address the second challenge of knowing which designs to evaluate. Both
methods accomplished the same task with their own strengths and weaknesses. The stress and
geometry coupling method indicated a PC’s relationship to geometric changes even when those
changes aligned with more than one design variable. The two-point correlation method only in-
dicated relationships for PCs that related to only one design variable. The stress and geometry
coupling method also indicated which direction the design variable should be changed to increase
or decrease the PC score while the two-point correlation method did not. Stress and geometry
coupling, however, did not provide a PC’s relationship to design variables that did not change the
geometry. This means that two-point correlation should be used when operating conditions, such
as rotational speed and pressure, are used as variables. The two-point correlation method was used
in an example to show how it could be used, with the developed workflow, to obtain a desired result
quickly.

It was shown in Chapter 4 that using the information available through PCA could be used
to improve DSE. It contributed to the body of research by presenting methods to better understand
the structural responses and their relationship to the design variables. These methods provide a
better means to change the design variables such that the stress is better understood and controlled.
The improved understanding provided by these methods can help meet design objectives and de-
crease the computation cost needed to achieve them.

The challenge of high computational cost to make structural vibratory evaluations was
addressed by fulfilling the third research objective, given in Chapter 5. Novel methods were
developed to emulate the vibratory responses of turbomachinery blades for DSE. This required
the development of a computationally cheaper vibration analysis and a more accurate surrogate.
Chapters 3 and 4 contain surrogate and DSE methods to emulate static stress. Chapter 5, however,
contains surrogate methods to emulate vibratory responses. First, a simplified vibratory analy-
sis was developed to allow for enough samples to create a surrogate. This method, RANS-HMS,
used steady-state CFD to estimate the vibratory forcing from aerodynamic pressures. This analysis
method introduced 10% error but reduced the computational cost of each sample by 90%. It was also found that the surrogate methods developed for static stress, in Chapter 3, produced high error due to local spikes in vibratory response near resonance. The static stress surrogates were usually unable to capture spikes in response. When a sample used in the surrogate did capture the response the global error around that sample was higher. The surrogate error, around 85% on average, was seen as unacceptable for DSE. A surrogate method was developed which indirectly emulated the vibratory responses. A set of surrogates was created to emulate the values used in calculation of the vibratory response. This set of surrogates emulated the mode shape, mode frequency, mode stress, pressure distribution, and static stress. These values, unlike the vibratory stress, did not have spikes in their design spaces. Then, during DSE, these surrogates were used to estimate the vibratory response inputs. These estimations were used, on the fly, to calculate the vibratory stress and modified Goodman with HMS. The indirect emulation method produced average errors of 17% instead of the 85% obtained with surrogates trained directly on vibratory stress.

The developed methods presented in Chapter 5 improve DSE by allowing more designs to be evaluated in the design process. This developed methodology offers engineers and engine manufacturers the ability to find, test, and offer better designs with less chance of needing an expensive redesign. Such improvements can have long-term benefits for engine operators as they seek to increase performance while keeping the design within structural limits.

The DSE challenge of reducing the computational cost to make a large number of design evaluations was further addressed by completing the fourth research objective, given in Chapter 6. Surrogate methods were developed to emulate high-fidelity finite element analysis from a mixed set of high and low-fidelity finite element model samples. The high-fidelity samples had thousands of nodes and the low-fidelity samples had hundreds of nodes. The methods were applicable to both static and vibratory stress responses. Multi-fidelity surrogate methods were developed which further decreased the cost required to collect the training samples, reduce the surrogate error, or both. A novel PCA-based multi-fidelity surrogate was developed for DSE of FEA results. It was found that this method was able to reduce surrogate error by 50% for a fixed computational cost or reduce computational cost by 75% for a fixed amount of error.

Chapter 6 showed that using the SPC multi-fidelity method for emulation of FEA can reduce the computational cost, by 75%, to create surrogates with errors below 5%. The novelty of
the method was based in not requiring a common grid between the fidelities and yet emulating the entire high-fidelity result field. This helps overcome the challenge presented to engineers during optimization and design exploration with high-fidelity simulations. The lower error for less computational cost makes optimization and design exploration more effective. This in turn leads to better optima and better understood design spaces.

Together, the methods which were developed in this research were shown to decrease the cost of evaluating the structural responses of turbomachinery blade designs. This was done for both static and vibratory stresses. They also provided a method to help the designer understand which designs to explore. The work in this dissertation contributed by solving these two DSE challenges. This paves the way for better, and more thoroughly understood structural responses in turbomachinery blade designs. Better understanding and design are necessary as gas turbine engines continue to advance in the present and future.

7.2 Recommended Work

This section details a few areas of study which could continue the body of research. These areas include the study of different surrogates and techniques to improve DSE.

7.2.1 More Surrogate Methods

The surrogate models used in this research were based on radial basis functions (RBFs) with multiquadric kernels. This decision was based on previous research showing that RBF performed well across a wide range of test cases and the fast speed at which they could emulate the results [37, 40]. Other emulation techniques, however, may prove to have less emulation error, in the application of structural DSE of turbomachinery blades, with little to no increase in emulation time. Such techniques could include other RBF kernels or other surrogate models. They could also include ensembles of surrogates in which multiple types of surrogates are used simultaneously [121]. Their values are combined into a weighted average for final emulation. Another method to consider could be the way in which the FEA data was reduced. PCA was used in this research, however, other reductions methods could be used and tested for their emulation error and speed. Methods based on factor analysis or even nonlinear reduction techniques may provide faster...
or more accurate surrogates. Increase in surrogate accuracy is worth continued research due to the large impact improved engines will have on society.

### 7.2.2 More Robust Statistical Methods

Two-point correlation for DSE with PCA was shown to provide valuable insight into how variations in stress relate to the design variables. This method, however, only captures relationships which are linear. Two-point correlation would not detect a significant relationship if the relationship between the scores of PC and a design variable was parabolic. Future work could include the study of more robust statistical analysis to look for parabolic relationships as well. Doing so would help detect which variables should be checked and explored with interactive DSE.

### 7.2.3 Unsupervised Surrogate Training

Unsupervised learning is a phrase used in machine learning to describe training a model when the labels for the training set are unknown. For the surrogates used in this research, the design variables represent those labels. The surrogates rely upon some input to predict a response. Instead of using the design variables, PCA could be performed on a matrix, $Y_8$, comprised of just the node coordinates. This matrix would contain only the geometric information about the samples. PCA on this matrix would produce geometric PCs with their scores. The PCs could be used as the design variables with the scores for each sample used as the values for the design. The rest of the surrogate modeling could then proceed as described in this dissertation. This method was briefly implemented in the research and was shown to work. However, no research was performed to understand the emulation error and any possible surrogate improvements. This method could be beneficial when seeking to emulate stresses across some set of compressor blades created from manufacturing deviations.

### 7.2.4 Adaptive Sampling

This research selected samples for training surrogate models by using Latin hypercube DOE.s. One possible way to decrease the surrogate error would be to use adaptive sampling to select the samples [8]. This involves selecting an initial set of samples using only a portion, usually
around 50%, of the computational budget. Then, more samples are added after evaluating where new samples would best improve the surrogate mode. For most applications, this type of sampling was found to be necessary in order for multi-fidelity surrogates to decrease the error compared to single-fidelity surrogates [66, 115, 122]. The work in chapter 6 found that the developed multi-fidelity surrogates significantly decreased error without adaptive sampling. However, it is still possible that adaptive sampling could improve both multi-fidelity and single-fidelity surrogates for exploring static and vibratory stress of turbomachinery blades.

7.2.5 Multi-fidelity with Empirical Data

The multi-fidelity surrogates developed in this research emulated high-fidelity responses from a set of multi-fidelity data. Further research could be performed to develop a method which uses empirical data in the multi-fidelity surrogates. Empirical data is based on measurements from physical testing. A sparse set of empirical data, used as high-fidelity samples, could be mixed with finite element models, used as low-fidelity samples. A multi-fidelity surrogate could be used to emulate the empirical data with the assistance of the multi-fidelity data. This would allow more accurate estimation of designs response.

7.2.6 DSE of Other Parts

This research developed DSE methods which emulated result fields. The surrogate and PCA methods only required the values at the points in the field and were not dependent upon the type of simulation or part used to obtain the data. This means the developed methods could be applied for DSE of other parts and simulations. The findings of emulation speed in this research will be similar for any other application because the speeds were solely based on the use of the surrogate and not the types of simulations used to create the data. The error associated with other applications, however, will vary. This research used the application of blade alone compressor and fan models. The next step in this research could apply and expand the DSE method developed in this research to other parts analyzed with FEA or even CFD. One application could be DSE of cyclic analysis of compressor blades. This type of analysis has more accurate and complex mode
shapes for integrally bladed rotors. It accounts for inter-blade resonance and provides further insight into the structural responses of compressor blades.
REFERENCES


142


[34] Siemens, P. Star-ccm+ users manual. 15


design space exploration and optimization for cad models.” ACM Transactions on Graphics
(TOG), 36(4), p. 157. 17, 30, 31, 56, 60

niques under multiple modelling criteria.” Structural and Multidisciplinary Optimization,
23(1), pp. 1–13. 18, 33, 137

decomposition.” Master’s thesis, Brigham Young University. 18, 30, 58, 61, 115

time-accurate turbomachinery simulations using proper orthogonal decomposition.” AIAA
2013-3620. 18, 58, 115

axisymmetric hub profiling optimization using multi-fidelity non-intrusive pod surrogates.”
GT2017-65106, ASME. 18, 21, 22, 23, 58, 111, 115, 116

surrogate models in aircraft real-time and many-query aerodynamic analyses.” Progress in
Aerospace Sciences, 96, pp. 23–61. 20, 21, 22, 60


mode shape identification.” Master’s thesis, Brigham Young University. 20

blades using principal component analysis and support vector machines.” In Structural

[48] Ross, C. F., Patel, B. A., Slice, D. E., Strait, D. S., Dechow, P. C., Richmond, B. G.,
and Spencer, M. A., 2005. “Modeling masticatory muscle force in finite element analysis:
sensitivity analysis using principal coordinates analysis.” The Anatomical Record, 283(2),
pp. 288–299. 20, 21

V., 2018. “Common proper orthogonal decomposition-based spatiotemporal emulator for
design exploration.” AIAA Journal, 56(6), pp. 1–15. 20, 21

modynamics for hypersonic aerothermoelasticity.” AIAA journal, 50(1), pp. 74–84. 20


agement strategies in aircraft design.” In 18th AIAA/ISSMO Multidisciplinary Analysis and
Optimization Conference, AIAA 2017-4431. 21, 22


[79] Leng, Y., 2016. “Preliminary design tools in turbomachinery: Non-uniformly spaced blade rows, multistage interaction, unsteady radial waves, and propeller horizontal-axis turbine optimization.” PhD thesis, Purdue University. 31


148


149


APPENDIX A. IMPORTANT CODE

This appendix presents the surrogate modeling code necessary for methods presented in this research.

A.1 RBF and Node-based Surrogate

```python
import numpy as np
from scipy import linalg

import kernels
import norms

class Rbf(object):

    def __init__(self, x, y, **kwargs):
        """
        Radial Basis Function
        """

        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.

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

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.

""

self.x_t = x.T

epsilon = kwargs.pop('epsilon', 'epsilon')

if type(epsilon) == str:
    self.epsilon = getattr(kernels, 'calculate_' + epsilon)(self.x_t.T)
else:
    epsilon
self.kernel = getattr(kernels, kwargs.pop('kernel', 'multiquadric'))
self.norm = getattr(norms, kwargs.pop('norm', 'two_norm'))
smooth = kwargs.pop('smoothing', 0.)

r = self.norm(self.x_t[::, np.newaxis], self.x_t[::, np.newaxis, :])
self.psi = self.kernel(r, self.epsilon) - np.eye(x.shape[0]) * smooth
self.w = linalg.solve(self.psi, y).T

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.
    """
if len(x.shape) == 1:
    r = self.norm(x[:,np.newaxis],self.x_t)
    return np.matmul(self.w,self.kernel(r,self.epsilon))
else:
    r = self.norm(x.T[:,:,np.newaxis],self.x_t[:,np.newaxis,:])
    return np.matmul(self.kernel(r,self.epsilon),self.w.T)

A.2 PCA-based Surrogate

import numpy as np

import pcaModel as Pc
import rbf

class PrincipleSurrogateEmulator(object):

    def __init__(self, x, y, **kwargs):
        """
        PrincipleSurrogateEmulator

        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

**norm : str**

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

**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.

""

self._x = x

self._surrogate = None

self._num_pc = 0

self._pca_object = None

self._pc_comp = None

self.pc_scores = None

self._train_model(y, **kwargs)
```python
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.pc_scores = self._surrogate(x)
    if len(x.shape) == 1:
        return np.matmul(self._pc_comp, self.pc_scores) + self._pca_object.pca.mean_
    else:
        return np.matmul(self.pc_scores, self._pc_comp.T) + self._pca_object.pca.mean_
```

@property
def explained_variance(self):
    return self._pca_object.pca.explained_variance_ratio_

@property
def num_pc(self):
    return self._num_pc

@property
def training_pc_scores(self):
    return self._pca_object.transform

def get_component(self, component):
    ""
    The PCs, or eigenvectors, from the PCA.
    """
    Parameters
    ----------
    component : int
        The principle component to return. If 0 then return the mean
    Return
    -------
    np.ndarray: The PC normalized to the explained variance of the PC
    """
    if component == 0:
        return self._pca_object.pca.mean_
    else:
```
return self._pc_comp[:, component-1]
```

def get_correlation(self, norm_axis=None, order=1):
    """
    Determine the absolute value of the two-point correlation between
    the pc scores and design variables.
    """

    Parameters
    ----------
    norm_axis: int
        options = [None, 0, 1]
        If the option is None then no normalization is performed. If
        the option is 0 then the matrix normalized
        such that each PCs correlation sums to one. If the option is
        1 then the matrix if normalized such that each
        variables correlation sums to one.

    Return
    -------
    np.ndarray : 2D array of correlation values that relate PCs to design
    variables.
    Shape = [number of pcs, number of design variables]
    """

    scores = self._pca_object.transform
    correlation_mat = np.abs(np.matmul(scores.T, self._x))

    if norm_axis == 0:
        norm_vec = np.sum(correlation_mat, axis=0)
        correlation_mat = correlation_mat / norm_vec
    elif norm_axis == 1:
```

158
norm_vec = np.sum(correlation_mat, axis=1)
correlation_mat = (correlation_mat.T / norm_vec).T

return correlation_mat

def _train_model(self, y, **kwargs):
    
    """
    Train the pca-based emulator.
    """
    self._train_principle_component_analysis(y, kwargs.pop('limit', None))
    self._train_radial_basis_function(**kwargs)
    self.pc_scores = np.empty(self._num_pc, dtype=float)

def _train_principle_component_analysis(self, y, limit):
    """
    Perform principal component analysis. Set the principle component
    model in the pca object.
    """
    self._pca_object = Pc.PCModel(y, n_components=limit)
    self._num_pc = self._pca_object.transform.shape[1]
    self._pc_comp = np.ascontiguousarray(
        np.copy(self._pca_object.pca.components_).T, dtype=np.float64)

def _train_radial_basis_function(self, **kwargs):
    """
    Train the radial basis function. Set the surrogate model in the
    emulator object. Unique RBF weighting vectors
    created for each principle component.
    """
pc_scores = self._pca_object.transform

self._surrogate = rbf.Rbf(self._x, pc_scores, **kwargs)

A.3 PCA

import numpy as np
from sklearn.decomposition import PCA

class PCModel(object):
    def __init__(self, y, n_components=None):
        
        Perform principal component analysis

        Parameters
        __________

        y : np.ndarray
            Samples of the features on which to perform PCA.
            Shape = [number of samples, number of features]
        n_components: int, float
            number of principle components used in eigenvalue
decomposition.
            None: uses all available PC, Int: uses indicated #
of PC, Float: uses explained variance

        
        self.pca = PCA(n_components=n_components)
        self.transform = self.pca.fit_transform(y)
A.4 RBF Kernels

```python
import math
import numpy as np
from scipy.special import xlogy

import norms

def multiquadric(r, epsilon):
    return np.sqrt((1.0 / epsilon * r) ** 2. + 1.)

def multiquadric_song(r, sigma):
    return np.sqrt(r ** 2. + sigma ** 2.)

def inverse_multiquadric(r, epsilon):
    return 1.0 / np.sqrt((1.0 / epsilon * r) ** 2. + 1.)

def gaussian(r, epsilon):
    return np.exp(-(1.0 / epsilon * r) ** 2.)

def linear(r, _=None):
    return r

def cubic(r, _=None):
    return r ** 3
```
def quintic(r, _=None):
    return r ** 5

def thin_plate(r, _=None):
    return xlogy(r ** 2, r)

def gaussian_kriging(x1, x2, theta, p):
    return np.exp(-np.sum(theta*(np.moveaxis(np.abs(x1-x2),0,2)**p),axis=2))

def calculate_epsilon(x):
    edges = np.amax(x, axis=0) - np.amin(x, axis=0)
    edges = edges[np.nonzero(edges)]
    return np.power(np.prod(edges) / x.shape[0], 1.0 / edges.size)

def calculate_epsilon_cai(x):
    volume = 2.5
    distances = norms.two_norm(x.T[:,:,np.newaxis],x.T[:,np.newaxis,:])
    d = distances.max()
    m = float(x.shape[0])
    n = float(x.shape[1])
    sigma = d / math.sqrt(m) / math.pow(n, 1.0/m)
    rho = (np.exp(distances / math.pow(sigma, 2.0))).sum(axis=1)
    v = 1.0 / rho
    v *= volume / v.sum()
\[ \text{return np.power(v, 1.0 / d)} \]

def calculate_sigma(x):
    d = norms.two_norm(x.T[:,:,np.newaxis],x.T[:,np.newaxis,:]).max()
    \text{return d / math.sqrt(2.*float(x.shape[0]))} \\

A.5 Norms

import numpy as np

def two_norm(x1, x2):
    \text{return np.sqrt(((x1 - x2) ** 2).sum(axis=0))} \\

def one_norm(x1, x2):
    \text{return np.abs(x1 - x2).sum(axis=0)} \\

A.6 Indirect Vibratory Surrogates

import math
import numpy as np
import rbf
import pcse

class HMS(object):
    def __init__(self, x, mode_displace, load, mode_freq, load_freq,
coord, damp=0, mode_stress=None, **kwargs):

""
Harmonic Mode Superposition Emulator

Parameters
----------
x : np.ndarray
   Labels, or design variable values, which correspond to y.
   Shape = [number of samples, number of variables]
mode_displace : np.ndarray
   Samples of the mode displacements to emulate.
   Shape = [number of modes, number of samples, number of nodes*3]
load : np.ndarray
   Samples of the load to emulate.
   Shape = [number of samples, number of nodes].
mode_freq : np.ndarray
   Frequencies of the modes.
   Shape = [number of samples, number of modes]
load_freq : float
   Frequency of the loading function
damp : float
   Damping ratio of vibration
mode_stress : np.ndarray
   Samples of the mode stresses to emulate.
   Shape = [number of modes, number of samples, number of nodes].
coord : np.ndarray
   Coordinates of the model to emulate.
   Shape = [number of samples, number of nodes * 3]
**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

**norm : str**

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

**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.

""

self._mode_dis_x_surrogates = None
self._mode_dis_y_surrogates = None
self._mode_dis_z_surrogates = None
self._load_x_surrogate = None
self._load_y_surrogate = None
self._load_z_surrogate = None
self._mode_stress_surrogates = None
self._mode_dis_surrogates = None
self._mode_frequency_surrogate = None
self._coord_surrogate = None

165
self._static_surrogate = None

self._emulated_dis_x = None
self._emulated_dis_y = None
self._emulated_dis_z = None

self._emulated_load_x = None
self._emulated_load_y = None
self._emulated_load_z = None

self._emulated_freq = None
self._emulated_coord = None
self._emulated_static = None
self._emulated_results = None
self._return_value = None

self.num_modes = None

self.load_frequency = load_freq * 2 * math.pi
self.damp = damp

self._train_surrogates(x, mode_displace, load,
                       mode_freq * 2 * math.pi,
                       mode_stress, coord, **kwargs)

def _train_surrogates(self, x, mode_displacements, loads, mode_freq,
                      mode_stresses, coord, **kwargs):
    
    """
    Train the surrogates
    """
Parameters
---------

\( x : \text{np.ndarray} \)

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

\( \text{mode_displacements} : \text{np.ndarray} \)

Samples of the mode displacements to emulate.
Shape = \([\text{number of samples, number of nodes \times 3}]\).

\( \text{loads} : \text{np.ndarray} \)

Samples of the load to emulate.
Shape = \([\text{number of samples, number of nodes}]\).

\( \text{mode_freq} : \text{np.ndarray} \)

Frequencies of the modes

\( \text{mode_stresses} : \text{np.ndarray} \)

Samples of the mode stresses to emulate.
Shape = \([\text{number of modes, number of nodes}]\).

\( \text{coord} : \text{np.ndarray} \)

Coordinates of the model to emulate.
Shape = \([\text{number of samples, number of nodes \times 3}]\)

**kwargs:

\( \text{epsilon} : \text{str, float} \)

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.

\( \text{kernel} : \text{str} \)

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

\( \text{norm} : \text{str} \)
The name of the norm function, in the norm module, to use with
the radial basis function

**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.

```
num_modes, num_sam, num_nodes_3 = mode_displacements.shape
num_nodes = int(num_nodes_3 / 3)
reshaped_modes = mode_displacements.reshape((num_modes, num_sam,
                                          num_nodes, 3))

displacement_x = reshaped_modes[:, :, :, 0]
displacement_y = reshaped_modes[:, :, :, 1]
displacement_z = reshaped_modes[:, :, :, 2]

self._mode_dis_x_surrogates = [pcse(x, m, **kwargs) \n                                   for m in displacement_x]
self._mode_dis_y_surrogates = [pcse(x, m, **kwargs) \n                                   for m in displacement_y]
self._mode_dis_z_surrogates = [pcse(x, m, **kwargs) \n                                   for m in displacement_z]

load_x = loads[:, :, 0] * loads[:, :, 1]
load_y = loads[:, :, 0] * loads[:, :, 2]
load_z = loads[:, :, 0] * loads[:, :, 3]

self._load_x_surrogate = pcse(x, load_x, **kwargs)
```

168
self._load_y_surrogate = pcse(x, load_y, **kwargs)
self._load_z_surrogate = pcse(x, load_z, **kwargs)

self._mode_frequency_surrogate = rbf.Rbf(x, mode_freq.T, **kwargs)

if mode_stresses is not None:
    self._mode_stress_surrogates = [pcse(x, m, **kwargs) for m in mode_stresses]
else:
    self._mode_dis_surrogates = [pcse(x, m, **kwargs) for m in mode_displacements]

self._coord_surrogate = pcse.PrincipleSurrogateEmulator(x, coord)

self.num_modes = num_modes

self._emulated_dis_x = np.zeros((num_modes, num_nodes))
self._emulated_dis_y = np.zeros((num_modes, num_nodes))
self._emulated_dis_z = np.zeros((num_modes, num_nodes))

self._emulated_load_x = np.zeros(num_nodes)
self._emulated_load_y = np.zeros(num_nodes)
self._emulated_load_z = np.zeros(num_nodes)

self._emulated_freq = np.zeros(num_modes)
self._emulated_coord = np.zeros(num_nodes_3)
self._emulated_results = np.zeros(num_nodes)
self._return_value = np.zeros(num_nodes * 4)

def __call__(self, x):

169
""
Calculate HMS from emulated modes, loads, and frequencies

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 : calculated displacements or stresses from HMS.
   Shape = [number of evaluations, number of nodes] if multiple evaluations are supplied.
   Shape = [number of nodes, ] if only one evaluation is supplied.
""

for m in range(self.num_modes):
    self._emulated_dis_x[m] = self._mode_dis_x_surrogates[m](x)
    self._emulated_dis_y[m] = self._mode_dis_y_surrogates[m](x)
    self._emulated_dis_z[m] = self._mode_dis_z_surrogates[m](x)

self._emulated_load_x = self._load_x_surrogate(x)
self._emulated_load_y = self._load_y_surrogate(x)
self._emulated_load_z = self._load_z_surrogate(x)

self._emulated_freq = self._mode_frequency_surrogate(x)
p_x = harmonic_mode_superposition_p(self._emulated_dis_x,
        self._emulated_load_x)

p_y = harmonic_mode_superposition_p(self._emulated_dis_y,
        self._emulated_load_y)

p_z = harmonic_mode_superposition_p(self._emulated_dis_z,
        self._emulated_load_z)

qc, qs = harmonic_mode_superposition_q(self.load_frequency,
        self._emulated_freq,
        self.damp)

self._emulated_coord = self._coord_surrogate(x)

if self._mode_stress_surrogates is None:
    displace_x = np.sum(self._emulated_dis_x * p_x * qc, axis=0)
    displace_x += np.sum(self._emulated_dis_x * p_x * qs, axis=0)

    displace_y = np.sum(self._emulated_dis_y * p_x * qc, axis=0)
    displace_y += np.sum(self._emulated_dis_y * p_x * qs, axis=0)

    displace_z = np.sum(self._emulated_dis_z * p_x * qc, axis=0)
    displace_z += np.sum(self._emulated_dis_z * p_x * qs, axis=0)

    hms_displace = np.array([displace_x, displace_y, displace_z]).T

    self._emulated_results = np.linalg.norm(hms_displace, axis=1)
else:

    mode_stresses = np.array([m(x) for m in 
                            self._mode_stress_surrogates])

    p_qc_scale = np.sqrt((p_x*qc)**2 + (p_y*qc)**2 + (p_z*qc)**2)
    p_qs_scale = np.sqrt((p_x*qs)**2 + (p_y*qs)**2 + (p_z*qs)**2)

    self._emulated_results = np.sum(mode_stresses*p_qc_scale,axis=0)
    self._emulated_results += np.sum(mode_stresses*p_qs_scale,axis=0)

    displace_x = np.sum(self._emulated_dis_x * p_x * qc, axis=0)
    displace_x += np.sum(self._emulated_dis_x * p_x * qs, axis=0)

    displace_y = np.sum(self._emulated_dis_y * p_y * qc, axis=0)
    displace_y += np.sum(self._emulated_dis_y * p_y * qs, axis=0)

    displace_z = np.sum(self._emulated_dis_z * p_z * qc, axis=0)
    displace_z += np.sum(self._emulated_dis_z * p_z * qs, axis=0)

    hms_displace = np.array([[displace_x, displace_y, displace_z]]).T

    hms_displace = hms_displace.reshape(self._emulated_coord.shape)

    displaced_coord = hms_displace + self._emulated_coord

    return np.concatenate((self._emulated_results, displaced_coord,
                           self._emulated_coord), axis=0)
def harmonic_mode_superposition_p(mode, load):
    """
    Calculate p scales for HMS modes

    Parameters
    ----------
    mode : np.ndarray
        Mass normalized mode displacements for a specific direction
        Shape = [number of modes, number of nodes]
    load : np.ndarray
        Loads in the same direction
        Shape = [number of nodes]
    """
    p = np.dot(mode, load)
    return np.abs(p[:, np.newaxis])

def harmonic_mode_superposition_q(load_freq, mode_freq, damp):
    """
    Calculate q scales for HMS modes

    Parameters
    ----------
    load_freq : float
        Angular frequency of the load
    mode_freq : np.ndarray
        Angular frequencies for the modes
        Shape = [number of modes]
    damp : float
        """
Damping Ratio for the system

\[
\text{omegas} = \text{load_freq} / \text{mode_freq}
\]
\[
q = \frac{1}{\text{mode_freq}^2} / \left(1 - \text{omegas}^2\right)^2 + (2 \cdot \text{damp} \cdot \text{omegas})^2
\]
\[
\text{qc} = q \cdot \left(1 - \text{omegas}^2\right)
\]
\[
\text{qs} = q \cdot 2 \cdot \text{damp} \cdot \text{omegas}
\]

return np.abs(qc[:, np.newaxis]), np.abs(qs[:, np.newaxis])

class GoodmanEmulator(object):

    def __init__(self, alternating_emulator, steady_emulator, endurance, ultimate):
        
        Handles Goodman Calculations from Vibe and Steady Emulators

Parameters
----------

alternating_emulator : class instance
    Emulator which will calculate alternating stress. It should have a \_\_call\_\_ function which takes only the design variable inputs. It is also assumed that this emulator returns the displaced coordinates and nominal coordinates.

steady_emulator : class instance
    Emulator which will calculate steady stress. It should have a \_\_call\_\_ function which takes only the design variable inputs.

endurance : float
    Endurance strength for the part

ultimate : float
    Ultimate strength for the part
self._alternating_emulator = alternating_emulator
self._steady_emulator = steady_emulator
self._endurance = endurance
self._ultimate = ultimate

def __call__(self, x, alt_steady_stress=False):
    """
    Parameters
    ----------
    x : np.ndarray
        Labels, or design variable values, at which to emulate or
        predict the trained features.
        Shape = [number of variables, ] if only one evaluation.
    alt_steady_stress : bool
        Indication to return only the alternating and steady portions
        of stress instead of calculating goodman. The coordinates of
        the alternating emulator are discarded

    Return
    -------
    np.ndarray : Percent goodman for each node with displacements and
        coordinates if alt_steady_stress is false else vibe stress and
        steady stress only
        Shape = [number of nodes * 5] if alt_steady_stress is false else
        [[number of nodes], [number of nodes]]
    """
    alternating = self._alternating_emulator(x)
    steady = self._steady_emulator(x)
    num_nodes = steady.shape[0]
if alt_steady_stress:
    return alternating[::num_nodes], steady
else:
    goodman = percent_goodman(alternating[::num_nodes], steady,
                              self._endurance, self._ultimate)
    return np.concatenate((goodman,alternating[num_nodes:]),axis=0)

def percent_goodman(alternating, steady, endurance, ultimate):
    """
    Calculate percent goodman
    Parameters
    ---------
    alternating : np.ndarray
        Alternating stress
        Shape = [number of nodes]
    steady : np.ndarray
        Steady Stress
        Shape = [number of nodes]
    endurance : float
        Endurance strength. A material / geometric property
    ultimate : float
        Ultimate strength. A material property
    Return
    -------
    np.ndarray : Percent goodman values for each node
        Shape = [number of nodes, ]
goodman = endurance - (endurance / ultimate) * steady
return alternating / goodman

class MaxCyclicEmulator(object):

def __init__(self, alternating_emulator, steady_emulator):
    """
    Handles Max Cyclic Calculations from Vibe and Steady Emulators
    """
    self._alternating_emulator = alternating_emulator
    self._steady_emulator = steady_emulator

def __call__(self, x, alt_steady_stress=False):
    """
    Parameters
    """
    x : np.ndarray
Labels, or design variable values, at which to emulate or predict the trained features.

Shape = [number of variables, ] if only one evaluation.

alt_steady_stress : bool

Indication to return only the alternating and steady portions of stress instead of calculating max cyclic. The coordinates of the alternating emulator are discarded.

Return

---------

np.ndarray : Max cyclic stress for each node with displacements and coordinates if alt_steady_stress is false else vibe stress and steady stress only

Shape = [number of nodes * 5] if alt_steady_stress is false else [[number of nodes], [number of nodes]]

alternating = self._alternating_emulator(x)
steady = self._steady_emulator(x)
num_nodes = steady.shape[0]

if alt_steady_stress:
    return alternating[:num_nodes], steady
else:
    max_cyclic = alternating[:num_nodes] + steady
    return np.concatenate((max_cyclic, alternating[num_nodes:]), axis=0)

A.7 Multi-fidelity Surrogates

import numpy as np
from openmdao.surrogate_models.multifi_cokriging \

178
import MultiFiCoKriging as CoK
import pcaModel as Pc
import kernels
import norms
import rbf

class MultiFidelitySurrogateEmulator(object):

    def __init__(self, x, y, **kwargs):
        """MultiFidelitySurrogateEmulator

        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:
        fidelities : np.ndarray
            Fidelity of each sample in y.
            Shape = [number of samples, ]
        method : str
            Name of the multi-fidelity method to use.
            Options = ['song', 'bunnell', etc.]
        method_2 : str
            Name of the secondary method to use, is desired, to adjust
the PC scores in bunnell's method

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

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

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.

""

method = kwargs.pop('method', 'bunnell')
train_function = getattr(self, method + '_train')
self.emulation_function = getattr(self, method + '_predict')
self.em_input = {}

train_function(x, y, **kwargs)

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

\( x \) : \text{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.

Return
-------

\text{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.

```python
return self.emulation_function(x)
```

def bunnell_train(self, x, y, **kwargs):

```python
"""
A multi fidelity system which assumes that the PCs describe about the same contours
"""

Parameters
----------

\( x \) : \text{np.ndarray}

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

\( y \) : \text{np.ndarray}
Samples of the features to emulate.
Shape = [number of samples, number of features].

fidelities = kwargs.pop('fidelities', None)
supplement_method = kwargs.pop('method_2', 'space_mapping')
coordinate_data = kwargs.pop('separate_coord', None)

fidelity_levels = list(set(fidelities))
fidelity_levels.sort(reverse=True)
num_fidelities = len(fidelity_levels)

x_by_fidelity = [x[np.where(fidelities == fidelity_level)[0]]
    for fidelity_level in fidelity_levels]
y_by_fidelity = []
c_by_fidelity = []
for fidelity_level in fidelity_levels:
    fidelity_ids = np.where(fidelities == fidelity_level)[0]
yf_list = list(y[fidelity_ids])
y_by_fidelity.append(np.array(yf_list, dtype=np.float))
    if coordinate_data is not None:
        cf_list = list(coordinate_data[fidelity_ids])
c_by_fidelity.append(np.array(cf_list, dtype=np.float))

num_samples = [ybf.shape[0] for ybf in y_by_fidelity]
num_features = [ybf.shape[1] for ybf in y_by_fidelity]

if all(nf == num_features[0] for nf in num_features):
    common_grid = True
else:
    False
q = min([min(num_samples), min(num_features)])

if coordinate_data is not None:
    if common_grid:
        coord_surrogate = rbf.Rbf(x, coordinate_data)
    else:
        coord_surrogate = rbf.Rbf(x_by_fidelity[0], c_by_fidelity[0])
    self.em_input['coord surrogate'] = coord_surrogate

y_hf = y_by_fidelity[0]
y_hf_pca_obj = Pc.PCModel(y_hf, n_components=q)
y_hf_components = np.ascontiguousarray(
    np.copy(y_hf_pca_obj.pca.components_).T, dtype=np.float64)
y_hf_scores = y_hf_pca_obj.transform

self.hf = y_hf_scores

scores_by_fidelity = [y_hf_scores]
for i in range(1, num_fidelities):
    y_f = y_by_fidelity[i]
pca_obj_f = Pc.PCModel(y_f, n_components=q)
y_f_comp = np.ascontiguousarray(
    np.copy(pca_obj_f.pca.components_).T, dtype=np.float64)
y_f_scores = pca_obj_f.transform

self.lf = y_f_scores

for j in range(q):
if common_grid:
    correlation = np.correlate(
        y_hf_components[:, j], y_f_comp[:, j])
else:
    hf_mag = y_hf_components[:, j].sum()
    f_mag = y_f_comp[:, j].sum()
    correlation = hf_mag*f_mag
    if correlation < 0:
        y_f_scores[:, j] *= -1.0

scores_by_fidelity.append(y_f_scores)

if supplement_method:
    train_func = getattr(self, supplement_method + '_train')
    predict_func = getattr(self, supplement_method + '_predict')

    f_by_fidelity = [fidelities[np.where(fidelities == f)[0]] \n        for f in fidelity_levels]
    f_by_f_array = np.concatenate(f_by_fidelity, axis=0)
    y_scores = np.concatenate(scores_by_fidelity, axis=0)
    x_by_f_array = np.concatenate(x_by_fidelity, axis=0)

    train_func(x_by_f_array, y_scores,
                   fidelities=f_by_f_array, **kwargs)

    def lf_surrogate(x_predict): return predict_func(x_predict)

else:
    x_lf = x_by_fidelity[-1]
y_lf_scores = scores_by_fidelity[-1]
lf_surrogate = rbf.Rbf(x_lf, y_lf_scores, **kwargs)

self.em_input['hf_comp'] = y_hf_components
self.em_input['hf_mean'] = y_hf_pca_obj.pca.mean_
self.em_input['lf_score_surrogate'] = lf_surrogate
self.em_input['pc_scores'] = y_hf_scores

def bunnell_predict(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.

    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.
    """

    self.em_input['pc_scores'] = self.em_input['lf_score_surrogate'](x)
if len(x.shape) == 1:
    if '_coord surrogate' in self.em_input:
        coord = self.em_input['_coord surrogate'](x)

        con = np.matmul(self.em_input['hf_comp'],
                        self.em_input['pc_scores']) + \  
                        self.em_input['hf_mean']

    return np.concatenate((con, coord), axis=0)
else:
    return np.matmul(self.em_input['hf_comp'],
                      self.em_input['pc_scores']) + \  
                      self.em_input['hf_mean']
else:
    if '_coord surrogate' in self.em_input:
        coord = self.em_input['_coord surrogate'](x)

        con = np.matmul(self.em_input['pc_scores'],
                        self.em_input['hf_comp'].T) + \  
                        self.em_input['hf_mean']

    return np.concatenate((con, coord), axis=1)
else:
    return np.matmul(self.em_input['pc_scores'],
                      self.em_input['hf_comp'].T) + \  
                      self.em_input['hf_mean']

def song_train(self, x, y, **kwargs):
    ""
    A two fidelity system following the method presented by Song in
"A radial basis function-based multi-fidelity surrogate model: exploring correlation between high-fidelity and low-fidelity models"

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].

epsilon_kwarg = kwargs.pop('epsilon', 'sigma')
fidelities = kwargs.pop('fidelities', None)
kernel_kwarg = kwargs.pop('kernel', 'multiquadric_song')
norm_kwarg = kwargs.pop('norm', 'two_norm')
smooth = kwargs.pop('smoothing', 0.)
coordinate_data = kwargs.pop('separate_coord', None)

if coordinate_data is not None:
    coord_surrogate = rbf.Rbf(x, coordinate_data)
    self.em_input['coord surrogate'] = coord_surrogate

if type(epsilon_kwarg) == str:
    epsilon = getattr(kernels, 'calculate_' + epsilon_kwarg)(x)
else:
    epsilon = epsilon_kwarg

kernel = getattr(kernels, kernel_kwarg)
norm = getattr(norms, norm_kwarg)

highest_fidelity = fidelities.max()
y_hf = y[np.where(fidelities == highest_fidelity)[0]]
x_hf = x[np.where(fidelities == highest_fidelity)[0]]
s_hf = x_hf.shape[0]

lowest_fidelity = fidelities.min()
y_lf = y[np.where(fidelities == lowest_fidelity)[0]]
x_lf = x[np.where(fidelities == lowest_fidelity)[0]]

n = y.shape[1]

r = norm(x_hf.T[..., :, np.newaxis], x_hf.T[..., np.newaxis, :])
psi = kernel(r, epsilon) - np.eye(s_hf) * smooth

lf_emulator = rbf.Rbf(x_lf, y_lf, smoothing=smooth,
                       kernel=kernel_kwarg, norm=norm_kwarg,
                       epsilon=epsilon_kwarg)

lf_at_hf = lf_emulator(x_hf)

c = np.concatenate((lf_at_hf.T[:, :, np.newaxis],
                     np.repeat(psi[np.newaxis, :, :], n, axis=0)), axis=2)
c_t = c.swapaxes(1, 2)
beta = np.matmul(c_t, np.matmul(np.linalg.inv(np.matmul(c, c_t)),
                                   y_hf.T[:, :, np.newaxis]))

self.em_input['lf_surrogate'] = lf_emulator
self.em_input['norm'] = norm
self.em_input['hf_x'] = x_hf.T
self.em_input['rho'] = beta[:, 0, 0]
self.em_input['omega'] = beta[:, 1:, 0]

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

    Parameters
    什什
    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.

    Return
    什什
    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.
    ""
y_lf = self.em_input['lf_surrogate'](x)
y_mf = y_lf*self.em_input['rho']
if len(x.shape) == 1:
    r = self.em_input['norm'](x[:, np.newaxis], self.em_input['hf_x'])
    if 'coord surrogate' in self.em_input:
coord = self.em_input['coord surrogate'](x)
con = y_mf + np.dot(self.em_input['omega'], r)
return np.concatenate((con, coord), axis=0)

else:
    return y_mf + np.dot(self.em_input['omega'], r)
else:
    r = self.em_input['norm'](x.T[:, :, np.newaxis],
                              self.em_input['hf_x'][:, np.newaxis, :])

if 'coord surrogate' in self.em_input:
    coord = self.em_input['coord surrogate'](x)
    con = y_mf + np.matmul(r, self.em_input['omega'].T)
    return np.concatenate((con, coord), axis=1)
else:
    return y_mf + np.matmul(r, self.em_input['omega'].T)

def benamara_train(self, x, y, **kwargs):
    """
    Two fidelity method as given by Benamara et al. in:
    "Multi-Fidelity Extension to Non-Intrusive Proper
    Orthogonal Decomposition Based Surrogates"
    and
    "LPC Blade and Non-Axisymmetric Hub Profiling Optimization
    Using Multi-Fidelity Non-Intrusive POD Surrogates"
    ""

    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].

coordinate_data = kwargs.pop('separate_coord', None)
fidelities = kwargs.pop('fidelities', None)
highest_fidelity = fidelities.max()
lowest_fidelity = fidelities.min()

if coordinate_data is not None:
    coord_surrogate = rbf.Rbf(x, coordinate_data)
    self.em_input['coord surrogate'] = coord_surrogate

y_hf = y[np.where(fidelities == highest_fidelity)[0]]
x_hf = x[np.where(fidelities == highest_fidelity)[0]]
y_hf_mean = y_hf.mean(axis=0)
y_hf_bar = y_hf - y_hf_mean

y_lf = y[np.where(fidelities == lowest_fidelity)[0]]
x_lf = x[np.where(fidelities == lowest_fidelity)[0]]
y_lf_mean = y_lf.mean(axis=0)
y_lf_bar = y_lf - y_lf_mean

q, r = np.linalg.qr(y_hf_bar.T)
q1, q2 = q[:, :1], q[:, 1:]
v, l, u = np.linalg.svd(np.matmul(q2.T, y_lf_bar.T))
psi = np.concatenate((q1, np.matmul(q2, v)), axis=1)
scores_lf = np.matmul(psi.T, y_lf_bar.T)
scores_hf = np.matmul(psi.T, y_hf_bar.T)
surrogate_lf = rbf.Rbf(x_lf, scores_lf.T, **kwargs)
scores_hf_estimated = surrogate_lf(x_hf)

scores_diff = scores_hf.T - scores_hf_estimated
surrogate_diff = rbf.Rbf(x_hf, scores_diff)

self.em_input['lf_surrogate'] = surrogate_lf
self.em_input['diff_surrogate'] = surrogate_diff
self.em_input['psi'] = psi.T
self.em_input['hf_mean'] = y_hf_mean

def benamara_predict(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.
    
    Return
    ------
    np.ndarray : emulated or predicted features.
        Shape = [number of evaluations, number of features]
        if multiple evaluations are supplied.
hf_scores = self.em_input['lf_surrogate'](x) + \
    self.em_input['diff_surrogate'](x)

if len(x.shape) == 1:
    if 'coord surrogate' in self.em_input:
        coord = self.em_input['coord surrogate'](x)
        con = np.matmul(hf_scores, self.em_input['psi']) + \
            self.em_input['hf_mean']
        return np.concatenate((con, coord), axis=0)
    else:
        return np.matmul(hf_scores, self.em_input['psi']) + \
            self.em_input['hf_mean']
else:
    if 'coord surrogate' in self.em_input:
        coord = self.em_input['coord surrogate'](x)
        con = np.matmul(hf_scores[np.newaxis:, :], \
            self.em_input['psi']) + \
            self.em_input['hf_mean']
        return np.concatenate((con, coord), axis=1)
    else:
        return np.matmul(hf_scores[np.newaxis:, :], \
            self.em_input['psi']) + \
            self.em_input['hf_mean']

def mifsud_train(self, x, y, **kwargs):
    """
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].

coordinate_data = kwargs.pop('separatecoord', None)
fidelities = kwargs.pop('fidelities', None)
fidelity_levels = list(set(fidelities))
fidelity_levels.sort(reverse=True)

if coordinate_data is not None:
    coord_surrogate = rbf.Rbf(x, coordinate_data)
    self.em_input['coord surrogate'] = coord_surrogate

y_by_fidelity = [y[np.where(fidelities == fidelity_level)[0]]
                 for fidelity_level in fidelity_levels]
x_by_fidelity = [x[np.where(fidelities == fidelity_level)[0]]
                 for fidelity_level in fidelity_levels]
f_by_fidelity = [fidelities[np.where(
                          fidelities == fidelity_level)[0]]
                 for fidelity_level in fidelity_levels]

y_sorted = np.concatenate(y_by_fidelity, axis=0)
a_mean = y_sorted.mean(axis=0)
a = y_sorted - a_mean
u, s, v = np.linalg.svd(a, full_matrices=False)
alpha = np.matmul(u, np.diag(s))

x_sorted = np.concatenate(x_by_fidelity, axis=0)
f_sorted = np.concatenate(f_by_fidelity, axis=0)[..., np.newaxis]
x_and_f = np.concatenate((x_sorted, f_sorted/f_sorted.max()), axis=1)
surrogate = rbf.Rbf(x_and_f, alpha)

self.em_input['a_mean'] = a_mean
self.em_input['v'] = v
self.em_input['surrogate'] = surrogate

def mifsud_predict(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.
    
    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.

if len(x.shape) == 1:
    surrogate_input = np.append(x, 1.0)
cat_axis = 0
else:
    f = np.ones(x.shape[0])
surrogate_input = np.concatenate((x, f), axis=1)
cat_axis = 1

alpha = self.em_input['surrogate'](surrogate_input)

if 'coord surrogate' in self.em_input:
    coord = self.em_input['coord surrogate'](x)
    con = np.matmul(alpha, self.em_input['v']) + self.em_input['a_mean']
    return np.concatenate((con, coord), axis=cat_axis)
else:
    return np.matmul(alpha, self.em_input['v']) + self.em_input['a_mean']

def co_kriging_train(self, x, y, **kwargs):

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].

""
coordinate_data = kwargs.pop('separate_coord', None)
fidelities = kwargs.pop('fidelities', None)
fidelity_levels = list(set(fidelities))
fidelity_levels.sort()

if coordinate_data is not None:
    coord_surrogate = rbf.Rbf(x, coordinate_data)
    self.em_input['coord surrogate'] = coord_surrogate

n_features = y.shape[1]

y_by_fidelity = [y[np.where(fidelities == fidelity_level)[0]]
                 for fidelity_level in fidelity_levels]
x_by_fidelity = [x[np.where(fidelities == fidelity_level)[0]]
                 for fidelity_level in fidelity_levels]

surrogates = [CoK() for f in range(n_features)]
for i in range(n_features):
    surrogates[i].fit(x_by_fidelity, [y_f[:, :, i][..., np.newaxis]
                                     for y_f in y_by_fidelity])

self.em_input['surrogates'] = surrogates
self.em_input['num_features'] = n_features

def co_kriging_predict(self, x):

197
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.

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.

if len(x.shape) == 1:
    con = np.empty(self.em_input['num_features'])
    for i in range(self.em_input['num_features']):
        con[i] = self.em_input['surrogates'][i].predict(
            x[np.newaxis, :], False).flatten()
else:
    con = np.empty((x.shape[0], self.em_input['num_features']))
    for i in range(self.em_input['num_features']):
        con[:, i] = self.em_input['surrogates'][i].predict(
            x, False).flatten()
if 'coord surrogate' in self.em_input:
    coord = self.em_input['coord surrogate'](x)
    cat_axis = 0 if len(x.shape) == 1 else 1
    return np.concatenate((con, coord), axis=cat_axis)
else:
    return con

def additive_train(self, x, y, **kwargs):
    ""
    \[y_{hf}(x) = y_{lf}(x) + z(x)\]
    ""
    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].
    ""
    coordinate_data = kwargs.pop('separate_coord', None)
    fidelities = kwargs.pop('fidelities', None)
    fidelity_levels = list(set(fidelities))
    fidelity_levels.sort(reverse=True)

    if coordinate_data is not None:
        coord_surrogate = rbf.Rbf(x, coordinate_data)
        self.em_input['coord surrogate'] = coord_surrogate

    y_by_fidelity = [y[np.where(fidelities == fidelity_level)[0]] for fidelity_level in fidelity_levels]
for fidelity_level in fidelity_levels
x_by_fidelity = [x[np.where(fidelities == fidelity_level)[0]] \nfor fidelity_level in fidelity_levels]

y_lf = y_by_fidelity[-1]
x_lf = x_by_fidelity[-1]
rbf_lf = rbf.Rbf(x_lf, y_lf, **kwargs)

y_hf = y_by_fidelity[0]
x_hf = x_by_fidelity[0]
y_lf_prediction = rbf_lf(x_hf)
y_diff = y_hf - y_lf_prediction
rbf_z = rbf.Rbf(x_hf, y_diff, **kwargs)

self.em_input['lf_surrogate'] = rbf_lf
self.em_input['z_surrogate'] = rbf_z

def additive_predict(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.
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.

    """
    if 'coord surrogate' in self.em_input:
        coord = self.em_input['coord surrogate'](x)
        con = self.em_input['lf_surrogate'](x) + \n             self.em_input['z_surrogate'](x)
        cat_axis = 0 if len(x.shape) == 1 else 1
        return np.concatenate((con, coord), axis=cat_axis)
    else:
        return self.em_input['lf_surrogate'](x) + \n            self.em_input['z_surrogate'](x)

def multiplicative_train(self, x, y, **kwargs):
    """
    yhf(x) = rho(x) + ylf(x)

    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].
coordinate_data = kwargs.pop('separate_coord', None)
fidelities = kwargs.pop('fidelities', None)
fidelity_levels = list(set(fidelities))
fidelity_levels.sort(reverse=True)

if coordinate_data is not None:
    coord_surrogate = rbf.Rbf(x, coordinate_data)
    self.em_input['coord surrogate'] = coord_surrogate

y_by_fidelity = [y[np.where(fidelities == fidelity_level)[0]]
    for fidelity_level in fidelity_levels]
x_by_fidelity = [x[np.where(fidelities == fidelity_level)[0]]
    for fidelity_level in fidelity_levels]

y_lf = y_by_fidelity[-1]
x_lf = x_by_fidelity[-1]
rbf_lf = rbf.Rbf(x_lf, y_lf, **kwargs)

y_hf = y_by_fidelity[0]
x_hf = x_by_fidelity[0]
y_lf_prediction = rbf_lf(x_hf)
y_scale = y_hf / y_lf_prediction
rbf_rho = rbf.Rbf(x_hf, y_scale, **kwargs)

self.em_input['lf_surrogate'] = rbf_lf
self.em_input['rho_surrogate'] = rbf_rho

def multiplicative_predict(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.

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.

if 'coord surrogate' in self.em_input:
    coord = self.em_input['coord surrogate'](x)
    con = self.em_input['rho_surrogate'](x) * \n        self.em_input['lf_surrogate'](x)
    cat_axis = 0 if len(x.shape) == 1 else 1
    return np.concatenate((con, coord), axis=cat_axis)
else:
    return self.em_input['rho_surrogate'](x) * \n        self.em_input['lf_surrogate'](x)

def space_mapping_train(self, x, y, **kwargs):
    """
yhf(x) = F(ylf(x))

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].

coordinate_data = kwargs.pop('separate_coord', None)
fidelities = kwargs.pop('fidelities', None)

fidelity_levels = list(set(fidelities))
fidelity_levels.sort(reverse=True)

if coordinate_data is not None:
    coord_surrogate = rbf.Rbf(x, coordinate_data)
    self.em_input['coord surrogate'] = coord_surrogate

y_by_fidelity = [y[np.where(fidelities == fidelity_level)[0]]
                 for fidelity_level in fidelity_levels]
x_by_fidelity = [x[np.where(fidelities == fidelity_level)[0]]
                 for fidelity_level in fidelity_levels]

y_lf = y_by_fidelity[-1]
x_lf = x_by_fidelity[-1]
rbf_lf = rbf.Rbf(x_lf, y_lf, **kwargs)

y_hf = y_by_fidelity[0]
x_hf = x_by_fidelity[0]
y_lf_prediction = rbf_lf(x_hf)
rbf_hf = rbf.Rbf(y_lf_prediction, y_hf, **kwargs)

self.em_input['lf_surrogate'] = rbf_lf
self.em_input['hf_surrogate'] = rbf_hf

def space_mapping_predict(self, x):
    """
    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.

    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.
    """
lf = self.em_input['lf_surrogate'](x)
if 'coord surrogate' in self.em_input:
    coord = self.em_input['coord surrogate'](x)
    con = self.em_input['hf_surrogate'](lf)
cat_axis = 0 if len(x.shape) == 1 else 1
return np.concatenate((con, coord), axis=cat_axis)

else:
    return self.em_input['hf_surrogate'](lf)

def cai_train(self, x, y, **kwargs):
    ""
    A Multi-Fidelity system which assumes that uses the methods by
    Cai et al. in
    'Adaptive Radial-Basis-Function-Based Multifidelity
    Metamodelling for Expensive Black-Box Problems'
    ""

    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].
    ""

    epsilon_kwarg = kwargs.pop('epsilon', 'epsilon')
    fidelities = kwargs.pop('fidelities', None)
    kernel_kwarg = kwargs.pop('kernel', 'multiquadric')
    norm_kwarg = kwargs.pop('norm', 'two_norm')
    coordinate_data = kwargs.pop('separate_coord', None)

    if coordinate_data is not None:
        coord_surrogate = rbf.Rbf(x, coordinate_data)
        self.em_input['coord surrogate'] = coord_surrogate
epsilon = getattr(kernels, 'calculate_' + epsilon_kwarg) \
    if type(epsilon_kwarg) == str else epsilon_kwarg
kernel = getattr(kernels, kernel_kwarg)
norm = getattr(norms, norm_kwarg)

s = x.shape[0]
fidelity_levels = list(set(fidelities))
fidelity_levels.sort(reverse=True)
num_fidelities = len(fidelity_levels)

y_by_fidelity = [y[np.where(fidelities == fidelity_level)[0]] \
    for fidelity_level in fidelity_levels]
x_by_fidelity = [x[np.where(fidelities == fidelity_level)[0]] \
    for fidelity_level in fidelity_levels]

epsilons = []
for f in range(num_fidelities):

    x_f = x_by_fidelity[f]
    epsilon_f = epsilon(x_f)

    if type(epsilon_f) == np.float64:
        s_f = x_f.shape[0]
        epsilons += [epsilon_f]*s_f
    else:
        epsilons += list(epsilon_f)

epsilons = np.array(epsilons)

h = np.zeros((s, s), dtype=np.float)
index = 0
for f in range(num_fidelities):

    x_f = x_by_fidelity[f]
    s_f = x_f.shape[0]

    epsilon_f = epsilons[index:index+s_f]
    r = norm(x_f.T[..., :, np.newaxis], x_f.T[..., np.newaxis, :])

    h_f = kernel(r, epsilon_f)
    h[index:index+s_f, index:index+s_f] = h_f

    index += s_f

h_star = np.zeros((s, s), dtype=np.float)
index = 0
for f in range(num_fidelities):

    x_f = x_by_fidelity[f]
    x_ff = np.concatenate(x_by_fidelity[f:], axis=0)
    s_f = x_f.shape[0]

    epsilon_f = epsilons[index:]
    r = norm(x_f.T[..., :, np.newaxis], x_ff.T[..., np.newaxis, :])

    h_f = kernel(r, epsilon_f)
    h_star[index:index+s_f, index:] = h_f

    index += s_f
y = np.concatenate(y_by_fidelity, axis=0)
x = np.concatenate(x_by_fidelity, axis=0)

omega_1 = np.matmul(h.T, h_star)
omega_2 = np.matmul(np.linalg.inv(omega_1), h.T)
omega = np.matmul(omega_2, y)

self.em_input['omega'] = omega.T
self.em_input['x'] = x.T
self.em_input['epsilon'] = epsilons
self.em_input['norm'] = norm
self.em_input['kernel'] = kernel

def cai_predict(self, x):
    
    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.

    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.
if len(x.shape) == 1:
    r = self.em_input['norm'](x[:, np.newaxis], self.em_input['x'])
    cat_axis = 0
    results = np.matmul(self.em_input['omega'],
                        self.em_input['kernel'](r, self.em_input['epsilon']())
    )
else:
    r = self.em_input['norm'](x.T[:, :, np.newaxis],
                               self.em_input['x'][:, np.newaxis, :])
    cat_axis = 1
    results = np.matmul(
                      self.em_input['kernel'](r, self.em_input['epsilon']()),
                      self.em_input['omega'].T)

if 'coord surrogate' in self.em_input:
    coord = self.em_input['coord surrogate'](x)
    return np.concatenate((results, coord), axis=cat_axis)
else:
    return results

def low_fidelity_train(self, x, y, **kwargs):
    
    Emulate with only the Low Fidelity Samples

    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].

fidelities = kwargs.pop('fidelities', None)
""
fidelity_levels = list(set(fidelities))
fidelity_levels.sort(reverse=True)
""

y_by_fidelity = [y[np.where(fidelities == fidelity_level)[0]]
    for fidelity_level in fidelity_levels]
x_by_fidelity = [x[np.where(fidelities == fidelity_level)[0]]
    for fidelity_level in fidelity_levels]

y_lf = y_by_fidelity[-1]
x_lf = x_by_fidelity[-1]
rbf_lf = rbf.Rbf(x_lf, y_lf, **kwargs)

self.em_input['lf_surrogate'] = rbf_lf

def low_fidelity_predict(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.
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.

"""
if 'coord surrogate' in self.em_input:
    coord = self.em_input['coord surrogate'](x)
    con = self.em_input['lf_surrogate'](x)
    cat_axis = 0 if len(x.shape) == 1 else 1
    return np.concatenate((con, coord), axis=cat_axis)
else:
    return self.em_input['lf_surrogate'](x)

def high_fidelity_train(self, x, y, **kwargs):
    ""
    Emulate with only the high fidelity samples

    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].
    """
    fidelities = kwargs.pop('fidelities', None)
fidelity_levels = list(set(fidelities))
fidelity_levels.sort(reverse=True)

y_by_fidelity = [y[np.where(fidelities == fidelity_level)[0]]
    for fidelity_level in fidelity_levels]
x_by_fidelity = [x[np.where(fidelities == fidelity_level)[0]]
    for fidelity_level in fidelity_levels]

y_hf = y_by_fidelity[0]
x_hf = x_by_fidelity[0]
rbf_hf = rbf.Rbf(x_hf, y_hf, **kwargs)
self.em_input['hf_surrogate'] = rbf_hf

def high_fidelity_predict(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.
    
    Return
    -------
    np.ndarray : emulated or predicted features.
"""

if 'coord surrogate' in self.em_input:
    coord = self.em_input['coord surrogate'](x)
    con = self.em_input['hf_surrogate'](x)
    cat_axis = 0 if len(x.shape) == 1 else 1
    return np.concatenate((con, coord), axis=cat_axis)
else:
    return self.em_input['hf_surrogate'](x)

"""
APPENDIX B. TRAINING DATA MODEL DEVELOPMENT

This appendix presents the steps which were necessary to create the training data and surrogates for design space exploration (DSE). These include the workflows which were needed for both static stress and vibratory stress.

B.1 Static Stress

This section details the creation of computer-aided design (CAD) models, finite element analysis (FEA) models, analysis workflows, and surrogate creation workflows needed to emulate static stress.

B.1.1 Parametric CAD Model

The first step was to create a parametric CAD model. While this step is necessary for any part explored with DSE, this appendix walks through the steps used to create a parametric CAD model of a turbomachinery blade. Each CAD package has slight variation to the terminology of the CAD modeling functions and methods. Consequentially, general explanations are given where possible and Siemens NX terminology are given where specifics are needed. Most turbomachinery blade geometries are defined by a series of airfoil cross-sections, like that of the Purdue blade shown in Fig. B.1. For the parametrization used in this work, the cross-sections were translated, rotated, and scaled. Sweep, lean, average radius, and blade height parameters used translations. Angle parameters used rotations. Blade chord and thickness-to-chord-ratio parameters used scalar adjustments. The amount that each cross-section was adjusted was defined by a set of expressions in the CAD software. Some of the expressions used for the Purdue blade are shown in Fig. B.2. This figures shows the defined angle at all of the cross-sections. The number in each expression name indicates the percent span of the cross-section it controls. This example shows numerical entries for the formula of angle at the tip and root, span 100 and 0 respectively. The angle formulae
of the spans in between the tip and root were defined as a relationship between the blade tip angle and the cross-sections percent span. Similar sets of expressions were made to control the cross-sections with the other design variables used in this research. A solid model was then created from the set of cross-sections. A through-curves sweep was used to create a solid model which passed through each of the cross-sections. Updating the expression formulae updated the solid geometry to a new design. When using a blade-alone model with no fillet, this completed the parametric CAD model.

Other options could include modeling the blade disk, adding a fillet, etc. The PBS Rotor 4 airfoils included the fillet. A small portion of the disk was modeled by extruding bottom surface down about an inch. When the fillet is not defined, a disk may be modeled and joined to the blade. A fillet may be created between the disk and blade. Most turbomachinery models also round the blade tip so that all points on the tip surface are at a constant radius from the axis of rotation. This was done by creating a horizontal line at the outer radius of the blade. The line was swept into a cylinder surface. The regions of the blade beyond that surface were removed. Whatever the additional options may be, these part features may also be parametrized and then united to make a complete solid model. When adding a fillet, it may be advantageous to slice the solid model into separate bodies around the fillet. This, seen in Fig. B.3, may help when creating the finite element model. The figures shows the fillet, light blue, split from the blade, dark purple, and the small part of the disk, green, which was modeled.
B.1.2 FEA Model

Like the CAD section, the general process is presented, but this section discusses specifics to meshing in NX pre/post and FEA in ANSYS where necessary. First, a mesh must be created from the CAD model. The element type should be chosen based on the application. Linear elements solve more quickly, but are less accurate where deflections are large. Cube elements are
Figure B.4: Coarse linear cube mesh for Purdue blade

generally more accurate and should be used where possible. Triangular elements are more easy to use when modeling complex geometry, but they present the possibility of numerically artificial stress concentrators. The static stress finite element models of the Purdue blade used linear cube elements in this research. A coarse representation of the mesh is shown in Fig. B.4. PBS Rotor 4 had a higher possibility of large deflections than the Purdue blade because of its higher aspect ratio. This led to the use of nonlinear elements. Cube elements were used on the blade and triangular elements were used on the fillet and disk. The use of different elements was allowed by splitting the CAD model for the different parts of geometry. The separate meshes were connected with mesh interfaces. The size of the elements on the mesh depends on the available computational budget and the stage of the design process. Samples collected early in the design process can have more coarse meshes because the early stages are usually dedicated to concept exploration. Samples collected later in the design process, where detailed design and analysis is performed, require finer meshes. This research used meshes with between 500 and 1,000,000 nodes on the Purdue blade and 220,000 nodes on PBS Rotor 4. If the final analysis only requires the nodes for the surface of the part, as in the case of interactive DSE, a surface node group should be defined.

Material properties, boundary conditions, and loads were used to create a completed finite element model. This research used $Ti - 6Al - 4V$, a material common to blades used in research and the material from which the real blades were created. A fixed boundary condition was applied at the blade root for blade alone models and at the bottom of the disk when the disk was modeled. The blades were loaded with a uniform pressure distribution on the blades pressure surface. This research used the blade’s experimental reports to determine magnitude of the pressure distribution. This was a 20 psi pressure load on the pressure surface of the blades. Lastly, a rotational load,
determined by the operating speed of the engine, was applied. The Purdue blade has an operating speed of 20,000 rpm while PBS Rotor 4 has an operating speed of 20,200 rpm.

B.1.3 Analysis Workflow

To perform static FEA for a selected design, the following steps are performed.

1. Update CAD model parameters
2. Update Mesh
3. Solve FEA
4. Write Solution

First, the expressions in the CAD model were updated to match the design to be analyzed. This updated the geometry of the solid model. Next, the finite element mesh was then updated to match the new geometry. If the mesh must be congruent, in which the nodes remain in the same location relative to the geometry of the part, mesh morphing may be performed. Mesh morphing adjusts the existing nodes to match the new geometry [123]. This, however, comes with a decrease in mesh quality when the geometric changes are large. If mesh congruency is not required the part may simply be re-meshed. This research used mesh morphing after testing the mesh quality over the design space. Next, linear static FEA was performed in ANSYS. The von Mises stress and x, y, and z coordinates for all nodes were written to a file. If the nodes are to be reconstructed for visualization then the nodes that make up each elements are also written to a file.

B.1.4 Training Set Workflow

DSE requires collecting data for a set of designs. A design of experiments (DOE) was used to select which designs should be analyzed. This required specifying which design variables should be explored and how many samples were to be analyzed. This research used Latin hypercube sampling to space the samples evenly through the design space. The selected designs variable values were written to a text file. The analysis steps in B.1.3 were performed for each selected sample.
This research used an automated workflow to perform the analysis steps and collect the data for a set of samples. Python was used to create the DOE and control the analysis workflow. The CAD models were updated and the meshes were automatically morphed using the NX application programming interface (API). The code which accessed the NX API was written in Java. Python was used to call the compiled Java files. The samples were collected faster when a single call to the NX API performed these steps for all of the designs. The updated finite element models were all written to files for the FEA simulation to solve. Python was used to call the ANSYS API to solve FEA. The ANSYS API was accessed with code files written in APDL. These files contained instruction on solving the model and writing the data.

B.1.5 Creating Surrogate Models

The creation of the surrogate models requires the data from FEA and their corresponding design variable values. A Python script was used to read the von Mises stress and x, y, and z coordinate data along with the design variable values for each design. The design variable values were normalized such that each variables maximum was 1.0 and minimum was 0.0. This helped improve the accuracy of the surrogate model. The stress and coordinate data were arranged in a single matrix as described in Ch. 3. The rows were the samples and the columns were the values at the nodes. The surrogates were trained on this matrix and the design variable values as shown in App. A.

B.2 Vibratory Responses

This section details the creation of CAD, computational fluid dynamics (CFD) and FEA models needed to obtain training data for vibratory responses. It also covers the analysis and surrogate workflows needed to emulate the vibratory responses.

B.2.1 Parametric CAD Model

Two separate CAD models were used in the vibratory analysis. The CAD model used with the structural analysis mesh was the same as that described for the analysis of static stress. The second model was used to control the geometry for the fluid CFD mesh.
Figure B.5: PBS Rotor 4 CAD model of fluid domain.

The fluid CAD model was developed from the structural CAD model. The surfaces of the parametrized solid model were used to create a sheet model of the blade. Updating the expressions would then update the sheet model. The sheet models of the fluid inlet, fluid outlet, hub, shroud, and sides were also included in this CAD model. The side surfaces were developed from the turbo-wizard in Star-CCM+. All of the surfaces were sewn to create the fluid domain of the rotor, shown in Fig. B.5.

B.2.2 CFD Model

The CFD model developed for this model was based on the steady-state model discussed in the thesis by Peterson. [111]. The appendix of Peterson’s thesis details the steps of creating the CFD model. A coarse representation of the CFD mesh is shown in Fig. B.6. The model included a single blade passage of the rotor domain and the stator domain. The figure shows the blades and the right side repeating surfaces. The rotor and stator domains each contained about six million cells.
CFD Verification

The CFD model and mesh parameters which were used in this research were based on those used and developed by Peterson, Soderquist, and List [61, 90, 111]. Verification of the model is based on their works. The domain and mesh parameters used in this dissertation were based on the values presented and suggested by their works. Also, a fully domain and grid-independent solution is not always necessary for DSE. During early phases of the design process CFD meshes with lower number of cells may be used to explore the design space. Because this dissertation was focused on design methods, the models and verification provided by Peterson, Soderquist, and List were accepted for this research. This research only tested number of iterations for convergence. Fig. B.7 shows the convergence of the corrected mass flow, total pressure ratio, and adiabatic efficiency for the nominal PBS Rotor 4 design. Each of these changed less than 0.01% after 15,000 iterations. Fig. B.8 shows the residuals for the same design. The continued downward trend indicates model convergence.
Figure B.7: PBS Rotor 4 corrected mass flow, pressure ratio, and adiabatic efficiency convergence through the simulation iterations.
Figure B.8: PBS Rotor 4 residuals through the simulation iterations.

**CFD Validation**

The following figures serve to validate the Reynolds-Averaged Navier-Stokes (RANS) CFD model against the higher-fidelity unsteady Reynolds-Average Navier-Stokes (URANS) model and experimental data. As with CFD verification, validation does not seek to show an exact match. DSE may be performed at any stage of the design process. Computationally less expensive simulations produce more error but allow for more analyses. Thus, this verification section seeks to show similar trends, but not exact matches to the experimental solution. Fig. B.9 shows the performance map for the nominal PBS Rotor 4 design. The static outlet pressure of 5000 psi led to the simulation with the best adiabatic efficiency. Outlet pressures lower than this moved the simulation towards choke, the right area in the figure where pressure ratio and effectiveness drop without a change in corrected mass flow. Outlet pressures higher than this moved the simulation towards compressor stall, the left area of the figure. All CFD simulations in this chapter used the 5000 psi outlet pressure found to be peak efficiency of the nominal design.

The experimental results and URANS simulations were used to validate the RANS simulation results. The mass flow, pressure ratio, and adiabatic efficiency for the nominal geometry are compared to the experimental values found in the test report [59]. This comparison is shown in Fig. B.9. As shown, the values are not exactly the same as the empirical testing. The RANS
Figure B.9: PBS Rotor 4 performance map.
simulation shows a maximum mass flow of about 28.2 kg/s at choked flow. This value is about 1% higher than the experimentally obtained value and less than 1% lower than the URANS value. The lowest mass flow achieved was about 27 kg/s before compressor stall at a maximum pressure ratio of about 1.98. This pressure ratio is about 2% lower than the experimental obtained values and the URANS values. The maximum adiabatic efficiency shown is about 86.5%, 3% lower than the experimental value and less than 1% lower than the URANS value. Fig. B.10 through Fig. B.12 show the static pressure along the hub, the total pressure and total temperature at the inlet, and the total pressure and total temperature at the outlet. These figures show the data for RANS simulation and the experimental data. The close values between the experimental data and the RANS simulations also serve to validate the RANS simulation. These comparisons serve to understand the error of the RANS model when compared to the higher fidelity URANS model and experimental test. This validates the model when the discussed errors are acceptable for the fidelity of analysis desired with DSE.

Figure B.10: PBS Rotor 4 stage static pressure along the axial direction. The simulation values are averaged circumferentially. The experimental values are documented in the testing report [59].
Figure B.11: PBS Rotor 4 stage inlet total pressure and total temperature in the radial direction. The simulation values are averaged circumferentially. The experimental values are documented in the testing report [59].
Figure B.12: PBS Rotor 4 stage exit total pressure and total temperature in the radial direction. The simulation values are averaged circumferentially. The experimental values are documented in the testing report [59].

B.2.3 FEA Model

The same finite element model used for the static stress was also used for vibratory responses. The one exception is that the pressure load applied to the blade surface of the vibratory finite element model was obtained from CFD.

B.2.4 Analysis Workflow

To perform vibratory stress analysis for a selected design, the following steps were performed.

1. Update fluid CAD model parameters
2. Export CAD model as IGES
3. Import IGES into CFD model
4. Update CFD mesh

5. Solve RANS

6. Write blade surface pressures to file

7. Update structural CAD model parameters

8. Update FEA mesh

9. Solve static FEA

10. Solve modal analysis FEA

First, the design variable values were used to update the fluid CAD model. Then, the sheet
body surfaces of this CAD model were exported as an IGES file. This file was imported into
the CFD model and split into surfaces. The fluid region boundaries were updated to reference
the new surfaces. Then, the mesh was updated to match the new fluid region and the RANS
simulation was solved. After the simulation converged, the static absolute pressures and element
normals on the surface of the blade were compiled into a table and written to a file. Convergence
was checked with corrected mass flow, total pressure ratio, adiabatic efficiency, and residuals.
Some samples failed to converge due to the design of the blade inducing blade stall. This caused
numerical instabilities which ended the simulation. A stalled sample meant that such the analyzed
design would be infeasible and analysis was no longer continued. The loss of samples was deemed
acceptable because they were located in regions of the design space which were unprofitable to
explore. The designs which caused blade stall were noted so that DSE could avoid those regions
of the design space. If the sample was feasible, the structural CAD model was updated to match
the design variables. The FEA mesh was morphed to match the new geometry. The pressures from
the CFD simulation were applied to the surface of the blade mesh. The finite element model was
written to a file. Static FEA was performed to obtain the static stresses and modal analysis FEA
was performed to obtain the mode shapes and mode frequencies. Harmonic mode superposition
(HMS) was eventually used to calculate the vibratory responses but this was performed during
emulation and not in the collection of the samples.
B.2.5 Training Set Workflow

DSE requires collecting data for a set of designs. A design of experiments (DOE) is used to select which designs should be analyzed. This required specifying which design variables should be explored and how many samples were to be analyzed. This research used Latin hypercube sampling to space the samples evenly through the design space. The selected designs variable values were written to a text file. The analysis steps in B.2.4 were performed for each selected sample.

This research used an automated workflow to collect the data for a set of samples. Python was used to create the DOE and control the analysis workflow. The CAD models were updated and the IGES files of the fluid domain were exported using the NX application programming interface (API). The code which accessed the NX API was written in Java. Python was used to call the compiled Java files. The samples were collected faster when a single call to the NX API performed these steps for all of the designs. The set of IGES files were manually copied to the supercomputer. A job-array was submitted to solve CFD for all of the designs. This job-array called java files which accessed the StarCCM+ API. The API imported the IGES, split the IGES into surfaces, updated the fluid region boundaries to the new surfaces, updated the meshes, solved RANS, and output the tables of pressures on the surface of the blade. Each of the designs were manually checked for convergence as they finished. Those that were not converged, but whose residuals indicated likely convergence with more iterations, were run for more time.

Once the pressures for all of the converged simulations were collected they were manually transferred back to a local computer. Python was then used again to automate the rest of the processes. The NX API was called using a Java file to update the FEA mesh to the new structural geometry. It also applied the pressures from CFD to the blade surface. The updated finite element models were all written to files for the FEA simulation to solve. The ANSYS API was accessed with code files written in APDL. These files contained instruction on solving the model and writing the data.
B.2.6 Creating Surrogate Models

The creation of the surrogate models required the data from FEA and their corresponding design variable values. The vibratory surrogates indirectly emulated the vibratory stress and modified Goodman values. A Python script was used to read the nodal data sets. These were static von Mises stress, coordinates, modal von Mises stresses, modal displacement, mode frequencies, and pressures on the surface of the blade. The Python script also read the design variable values for each design. The design variable values were normalized such that each variables maximum was 1.0 and minimum was 0.0. This helped improve the accuracy of the surrogate model. Surrogate models were created on each of the nodal data sets. Emulation of vibratory stress was performed, as described in Ch. 5, by emulating these sets and calculating the vibratory responses on the fly.