2016-03-01

An Automated Method for Optimizing Compressor Blade Tuning

Kurt Berlin Hinkle
Brigham Young University

An Automated Method for Optimizing Compressor Blade Tuning

Kurt Berlin Hinkle

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

Master of Science

Steven E. Gorrell, Chair
John L. Salmon
C. Greg Jensen

Department of Mechanical Engineering

Brigham Young University

March 2016

ABSTRACT

An Automated Method for Optimizing Compressor Blade Tuning

Kurt Berlin Hinkle
Department of Mechanical Engineering, BYU
Master of Science

Because blades in jet engine compressors are subject to dynamic loads based on the engine's speed, it is essential that the blades are properly "tuned" to avoid resonance at those frequencies to ensure safe operation of the engine. The tuning process can be time consuming for designers because there are many parameters controlling the geometry of the blade and, therefore, its resonance frequencies. Humans cannot easily optimize design spaces consisting of multiple variables, but optimization algorithms can effectively optimize a design space with any number of design variables.

Automated blade tuning can reduce design time while increasing the fidelity and robustness of the design. Using surrogate modeling techniques and gradient-free optimization algorithms, this thesis presents a method for automating the tuning process of an airfoil. Surrogate models are generated to relate airfoil geometry to the modal frequencies of the airfoil. These surrogates enable rapid exploration of the entire design space. The optimization algorithm uses a novel objective function that accounts for the contribution of every mode's value at a specific operating speed on a Campbell diagram. When the optimization converges on a solution, the new blade parameters are output to the designer for review. This optimization guarantees a feasible solution for tuning of a blade. With 21 geometric parameters controlling the shape of the blade, the geometry for an optimally tuned blade can be determined within 20 minutes.

Keywords: Airfoil tuning, blade tuning, IBR, blisk, bladed disk, surrogate model, radial basis function, optimization, genetic algorithm, particle swarm, NSGA-II, OMOPSO, Campbell diagram

ACKNOWLEDGMENTS

I want to express my gratitude to Kellianne for supporting me through the Mechanical Engineering program—for the countless nights, weekends, and holidays she spent alone while I was on campus.

I would also like to thank John Salmon, Steve Gorrell, Evan Selin, Ryan Heap, and Kurt Heinemann for their invaluable, technical contributions to this research.

Finally, I would like to express my gratitude to Greg Jensen for spawning my engineering career and Pratt & Whitney for funding my research and making my graduate experience an excellent financial choice.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Problem Statement

Considering the natural frequencies of an airfoil is an essential part of the development of compressors for jet engines [1]. When the compressor's rotors spin, their airfoils are subjected to consistent excitation as they experience changes in airflow due to upstream and downstream obstructions [2]. If the airfoil's natural frequencies occur near the frequencies of the dynamic loads, these excitations can cause resonance in the blade, which leads to premature failure of the airfoil [3].

Originally, the blades used for the rotors were manufactured separately from the rotor's drum and were mechanically fastened to the drum through a "dovetail" attachment. This method of attachment created problems with crack initiation and propagation where the blades attached to the rotor [4]. In the 1980s, the invention of integrated blade rotors (IBRs, bladed disks, or blisks) removed the need for a dovetail attachment because the rotor and blades were manufactured from one solid piece (see Figure 1.1). This innovation "reduce[d] the engine's part count, weight, and aerodynamic losses" as well as eliminated a source of crack initiation and propagation [5].

Though IBRs are the state of the art, the advantages they provide come with one main drawback: without the dovetail, IBRs offer little mechanical dampening of the blades [5]. This reduction in dampening increases the risk of the airfoils resonating to failure inside the compressor. To avoid this kind of catastrophic failure, "the vibration behaviour and of course the high cycle fatigue [HCF] strength of such structures must be analysed in detail" [6]. The vibration problem stemming from IBRs can be reduced in the design of the blades by tuning an airfoil. Tuning is the process of eliminating HCF failures by modifying blade geometry to alter the natural frequencies of the blade so that resonance cannot occur at the engine's primary operating speeds. By tuning a blade, a designer can minimize the resonance of the blade during operation of the engine, therefore prolonging the lifespan of the airfoil, the IBR, and the entire engine. Though tuning has been

Figure 1.1: Airfoils on the perimeter of an IBR. Courtesy of PM-Aerotec.

mentioned only in regards to IBRs due to their decreased mechanical dampening, it is essential for every airfoil to be tuned to avoid resonance.

Current practices for tuning involve designers modifying blade parameters and using a Campbell diagram to visualize how these parameters affect the modal frequencies of the blade. A Campbell diagram is a way to visualize the natural frequencies of a candidate blade based on an engine's speed. It also incorporates frequencies from engine orders, which is a term used for obstructions upstream or downstream of the airfoil. If one of the blade's natural frequencies is "too close" to an engine order, then resonance is possible; in fact, for a single Campbell diagram, there can be multiple locations where resonance is possible. Additional background information regarding Campbell diagrams as well as a sample Campbell diagram are provided in Section 2.1.

Tuning is an iterative process and having human interaction in this loop increases the design time for properly tuned airfoils, even with experienced designers. Therefore, to increase the reliability of IBRs and accelerate the design process of airfoils concurrently, it is desirable to develop an automated solution to optimize the blade geometry for tuning the airfoils on a bladed disk.

The geometry of an airfoil is one factor that affects its natural frequencies. These frequencies can be analyzed using holographic interferometry or through finite element analysis (FEA).

Each method has its drawbacks: holographic interferometry can only be performed on a manufactured blade, which is time-consuming and costly; regarding FEA, Heap et al. note, "...simulations can be very time consuming, with the computational time alone ranging from a few minutes to a number of days" [7]. However, by creating high-fidelity approximations of these FEA simulations, it is possible to significantly reduce the amount of time needed for computation instead of taking minutes, hours, or days to compute the results for one FEA, a surrogate model can be queried thousands of times per second [8]. In addition, these models create a continuous design space that can be explored by a human designer in real-time or by an optimization algorithm (see Section 2.3).

Though humans can effectively determine optimum values in design spaces that have one or two design variables, computers far surpass humans at efficiently determining the optimum values of more complicated design spaces. Since there are many more than three parameters used to define an airfoil and because there can be multiple locations of resonance on a Campbell diagram, it is essential to apply optimization algorithms to account for these factors simultaneously in order to determine global optima in these design spaces.

In order to develop an automated solution for tuning a blade, surrogate models can be used to map changes in the blade's geometric parameters to changes in the blade's natural frequencies. These natural frequencies can be compared to the frequencies of the engine orders at the same operating speed. An objective function can account for each of these comparisons simultaneously, and an optimization algorithm can be used to minimize the possibility of resonance.

1.2 Research Objectives and Scope

This research seeks to integrate the principles of surrogate modeling and optimization into the design loop of an airfoil. The primary and novel contributions from this thesis include the following:

1. Present a novel objective function for optimizing the design space created by the surrogate models
2. Use gradient-free optimization techniques to remove the designer from the iterative design loop

3. Automate the airfoil tuning process in order to reduce design time for airfoils and improve the robustness of the tuning process

In order to validate these contributions, the automated method is tested with two distinct sets of data, and an evaluation of the testing is provided to determine how well the method performed in each case study. This research is limited to the automation of the airfoil tuning process. Even though airfoil design is inherently multi-objective, this research does not include the integration of aerodynamic or structural design for an airfoil. This thesis focuses on the ideal tuning of the airfoils that is completed prior to considering manufacturing variability and robust optimization.

1.3 Thesis Outline

Chapter 2 provides background research for the major topics found in this thesis, namely, airfoil tuning, surrogate modeling, and optimization techniques. Chapter 3 provides a detailed description of the methodology and implementation of tuning process, both real-time manual tuning and automated tuning. It enumerates the six main steps of the automated process and provides justification for each of these processes. Chapter 4 documents the results of testing the method from Chapter 3 on two distinct sets of data. The first set consists of simplified data to prove the concept of the method; the second set consists of real data to prove that this method can be used in a legitimate design loop. Chapter 5 draws conclusions from the results presented in Chapter 4, and it suggests improvements that can be made to this method to make it even more useful to designers.

CHAPTER 2. BACKGROUND INFORMATION

2.1 Blade Tuning

Gas turbine engine compressor and turbine rotating components can be either one, solid piece—such as an IBR—or they can be an assembly consisting of a central hub with airfoils fastened around the perimeter of the hub. In each design, the airfoils experience excitation due to the consistent interactions between the rotor and stator blades. However, as noted in Section 1.1, IBRs have little mechanical dampening compared to assembled rotors.

Every airfoil has resonance frequencies that, if not accounted for in the design process, can cause catastrophic failure in the blade or even the entire engine. Although the problem of blade vibration cannot be prevented completely, it can be minimized during the design process by properly tuning the airfoils of the bladed disks. Even with an ideal, tuned blade, there will still be slight differences between each blade on the IBR due to manufacturing. These "mistuned bladed disks can have drastically larger forced response levels than the ideal, tuned design. The attendant increase in stresses can lead to premature high cycle fatigue (HCF) of the blades" [9]. In fact, high cycle fatigue is such a large issue that the U.S. Air Force estimated that "over 30% of all jet engine maintenance costs were due to HCF" [9]. For additional insight into mistuning, see Section 2.1.2. However, this thesis focuses primarily on the ideal tuning process perfomred during desing.

2.1.1 Ideal Tuning

When designing an airfoil, multiple factors must be considered including the aerodynamic, structural, and vibrational characteristics of the blade. An airfoil may have great aerodynamic properties, but the structural or vibrational properties of the blade may not be acceptable; likewise, an airfoil may be designed with excellent vibrational properties, but it may have terrible aerodynamic properties.

Because vibration is an integral part of blade design, when designing the airfoils for the compressor or turbine, it is essential to consider the modal frequencies of the blade in order to prevent failure due to resonance. As mentioned in Section 1.1, there are two different ways to analyze the vibrational properties of an airfoil, but only FEA methods can predict the modes of an airfoil during design. For example, Figure 2.1 illustrates mode shapes produced by ANSYS (an FEA software package) for a sample airfoil [10]. The blade was fixed in space along its left edge while the rest of the blade was permitted to move. The top-left image represents the first bending mode (1 Bending), while the top-center image represents the first torsional mode (1 Torsion). As one can see, more complicated modes can occur in the airfoil; these modes are typically seen at higher frequencies.

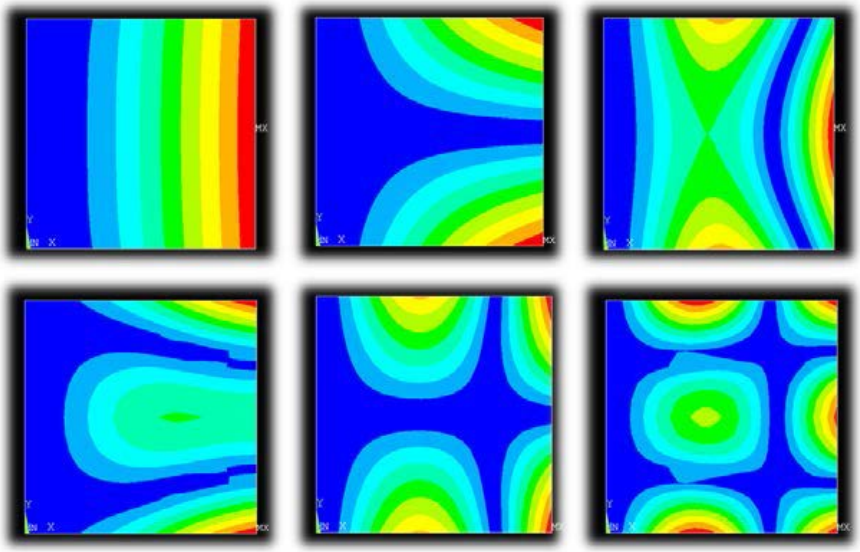

Figure 2.1: Sample mode shapes produced by finite element modal analysis.

A Campbell diagram is a common way to compare the modes of a blade to the engine orders at operating speeds [11]. Engine orders represent the number of obstructions upstream or downstream of the blade's rotor stage [12]. Examples of these obstructions include support struts, stator vanes, and rotor blades. The obstructions create dynamic loading on the considered airfoil, and that loading can cause resonance in the blade at various engine operating speeds. If a mode is "too close" to an engine order at an operating speed, resonance can occur and the blade may fail in HCF.

A sample Campbell diagram is shown in Figure 2.2. There are four modes displayed (1 Bending, 1 Torsion, 2 Bending, and Complex), and five engine orders (1E, 4E, 8E, 12E, 24E). One can observe from the diagram that there are several frequencies at which the candidate blade will resonate at operating speeds (idle, cruise, climb, and redline), specifically Points A and E. In this thesis, a mode at a certain operating speed will be referred to as a known "point" on a Campbell diagram. Since these points may cause resonance, the frequencies of Points A and E, the engine order frequencies, or the operating speeds must be changed. Because the obstructions upstream and downstream as well as the engine operating speeds are already defined, only the first option is considered in this thesis.

Points B and D may cause resonance because they appear close to engine orders 12E and 8E, respectively. This is where analysis is imperative—will these points cause resonance during operation? This analysis will be discussed in detail in Section 3.5. Last, Point C is several hundred Hertz above 8E and several hundred Hertz below 12E; therefore, it is not likely that Point C will be problematic. In practice, this threshold is set by the design engineer based on his or her company's standard practices.

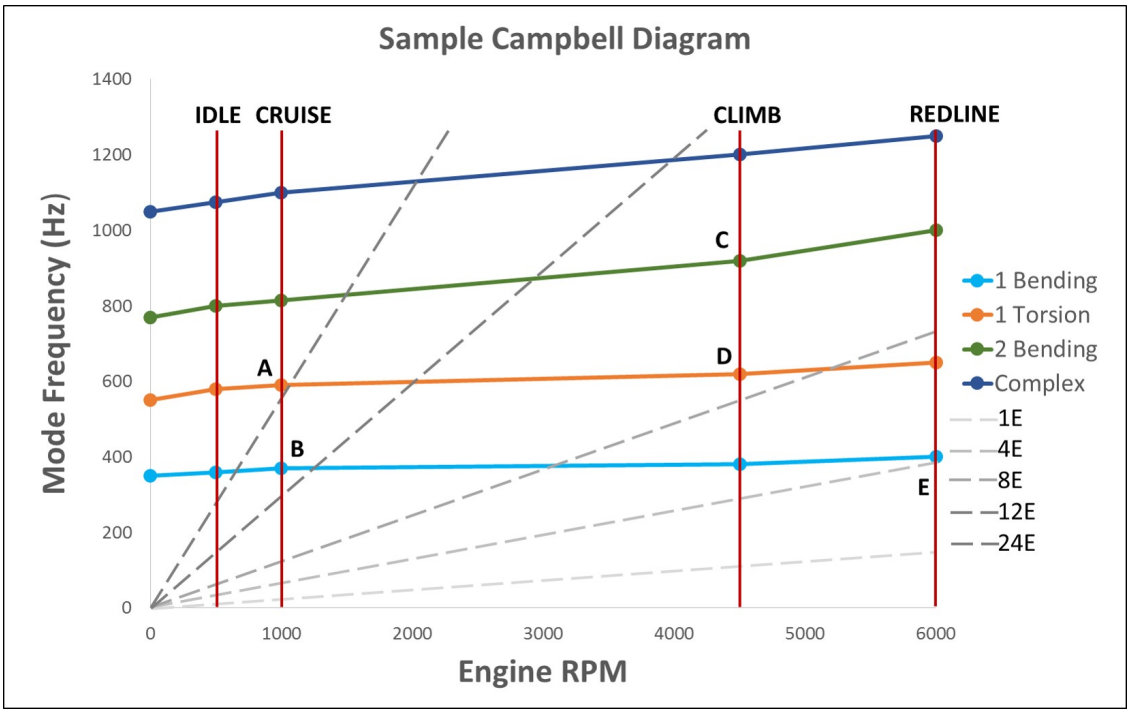

Figure 2.2: A sample Campbell diagram for analyzing the modes of an airfoil.

Tuning—varying the modal frequencies of the blade—can be accomplished in several ways, including varying the geometry of the blade, changing the number of blades on a rotor, using different types of airfoils on the rotor, or adjusting the upstream and downstream blade counts. However, this thesis will focus only on tuning a blade by varying its geometry.

There are many geometric parameters that influence the design of the blade. The parameters could consist of the chord, the thickness of the airfoil, blending radii, and blade twist. As each parameter is changed, the natural frequencies of the blade also change. It is important for the designer to know which geometric parameters can be changed so the necessary inputs for the optimization can be provided. For this thesis, surrogate models will be used to map these changes because of the advantages they provide in computational time and will be discussed in Section 2.2.

Due to the number of parameters influencing the geometry of the blade, it can be difficult for a human designer to completely explore the design space associated with the blade's parameters. Of course, an experienced designer could iteratively develop a design that satisfies the specifications for the airfoil. However, with such a complicated design space, it is extremely unlikely that the designer–without the help of optimization techniques–could determine the optimum solution that maximized the distance between each of the blade's modes and the closest intersection of engine orders and engine speeds.

2.1.2 Mistuning

When discussing the ideal process of tuning an airfoil during design, it is necessary to acknowledge the concept of "mistuning." Mistuning refers to the minute variations between airfoils due to manufacturing tolerances. These variations can be intentionally added to reduce the possibility of resonance; however, mistuning can also provide a greater possibility for resonance because each airfoil on a rotor is not identical. These small changes in airfoil geometry also vary the modal frequencies of the blade. If the change in modal frequencies is sufficiently large, the airfoil that was "ideally" tuned could resonate to failure. Though an important consideration in design and manufacturing, mistuning is outside the scope of this thesis.

2.2 Surrogate Models

Commonly known as surrogates, approximations, or metamodels, surrogate models can be thought of as multivariate regressions. It is an important aspect of this research because it provides a mapping between a blade's geometry and its modes at operating speeds (see Section 3.3). The modes are then used to construct a Campbell diagram. Visually, this can be seen in Figure 2.3.

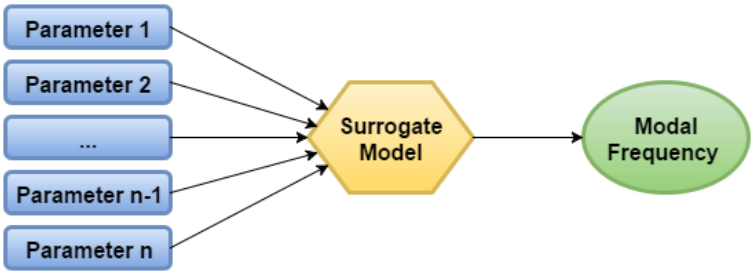

Figure 2.3: A visual representation of the mapping provided by a surrogate model.

As a simple comparison, a scatter plot shows a general trend between two variables—in this case, an independent variable (x) and a dependant variable ($f(x)$). Using this input and output data, several types of regressions can be generated from the data to model the trend: linear, quadratic, power, etc. It is clear from Figure 2.4 that some regressions more accurately predict the behavior of $f(x)$ than other types of regressions, but this, of course, is dependent on the nature of $f(x)$.

Similarly, surrogate models require input (independent variables) and output (dependent variables) data in order to be created. Because there are many different types of surrogate models, some surrogates more accurately predict outputs given the same input data. Surrogate models can provide a continuous approximation of the design space, meaning that for any input value between the minimum and maximum values in the training data, the surrogate will predict an output value.

The main advantage of surrogates comes with an increase in the number of independent variables. A single surrogate model can account for 'n' inputs in predicting a single output. This proves advantageous for this research: with a large number of blade parameters (inputs), it was desired to accurately predict the modal frequencies at operating speeds (outputs) based on changes to the parameters. Therefore, for each mode at each operating speed, a surrogate model was generated to map blade parameters to that specific point on the Campbell diagram. This can be

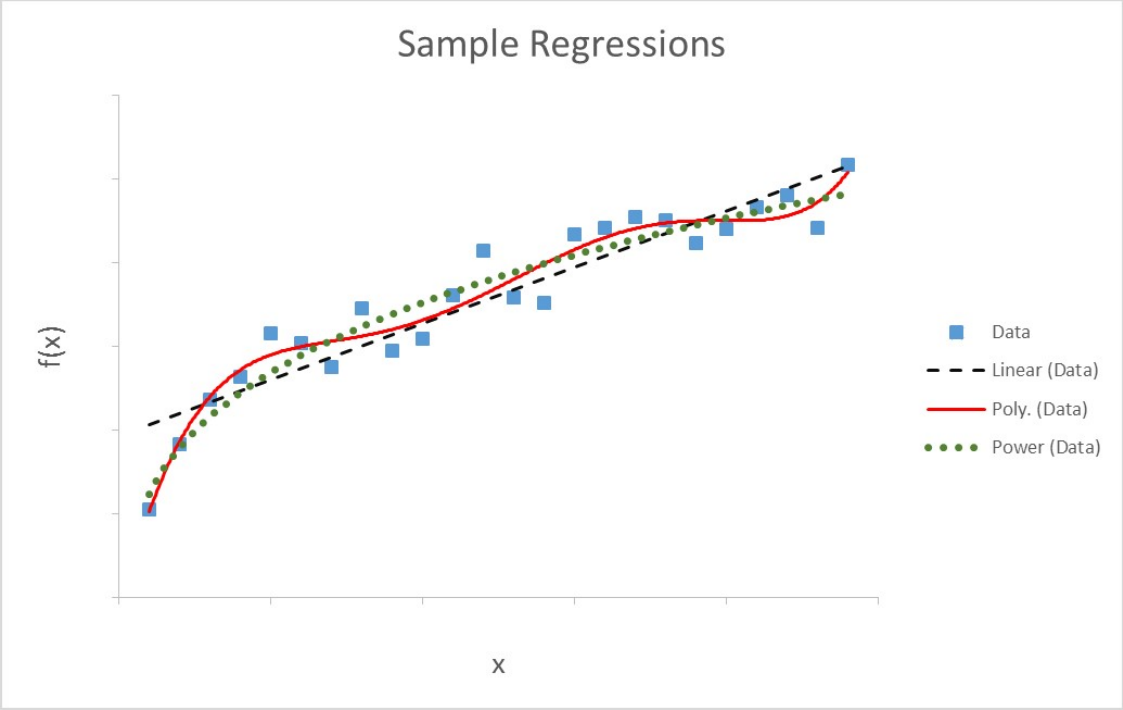

Figure 2.4: A linear, a polynomial, and a power regression obtained from an original set of data.

seen visually in Figure 2.5 where all the input parameters influence each of the surrogate models, and each of the surrogate models produces one output value for the Campbell diagram.

The initialization of a surrogate model takes anywhere from seconds to minutes depending on the approximation technique, the amount of training data, and the speed of the computer generating the model. With additional training data, the surrogate takes longer to initialize, but it generally provides a higher fidelity surrogate model.

Once the model is initialized, it can be queried in milliseconds. Furthermore, the surrogate does not have to be initialized multiple times, unless the training data changes. This proves advantageous as finite element and computational fluid dynamics analyses (FEA and CFD) can take hours or days to complete using a single set of input values. In contrast, querying an initialized surrogate model takes a fraction of a second. Based on these time savings and the accuracy of surrogate modeling, surrogates have become an integral part of engineering design [13].

Just as there are many types of regressions, there are also many types of surrogate models, including artificial neural network, radial basis function, kriging, polynomial response surface, Chebyshev, support vector machines, and space mapping models. In the chosen framework

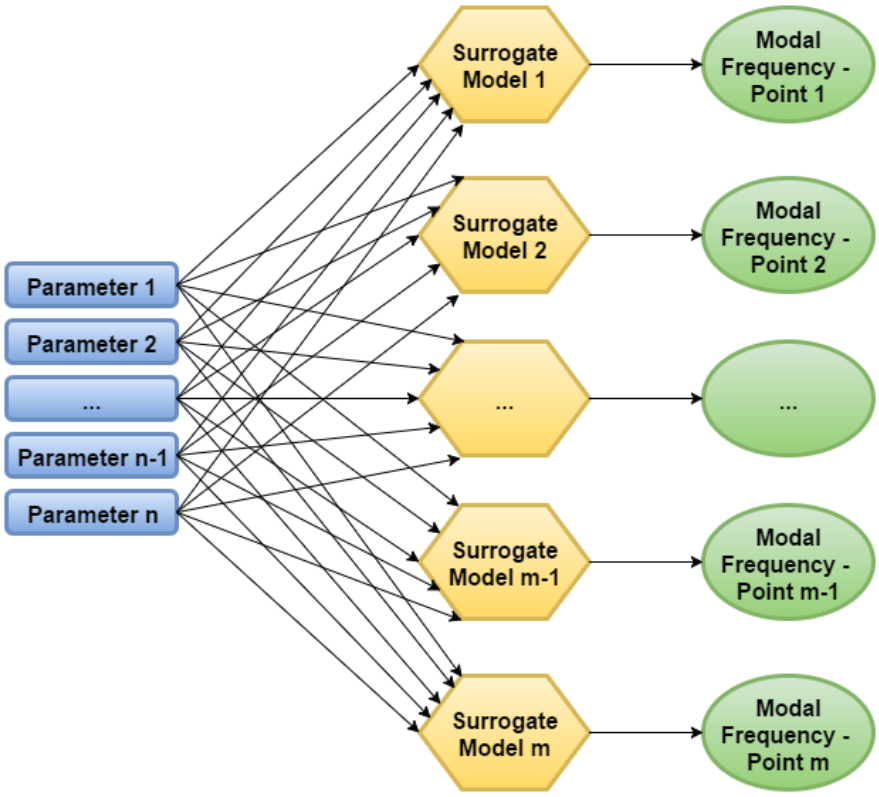

Figure 2.5: A visual representation of the mapping performed between the blade parameters, surrogate models, and points on the Campbell diagram.

for this research, SIMULIA's Isight multidisciplinary optimization software, only polynomial response surface, RBF/EBF, kriging, and Chebyshev modeling techniques are offered. Heap et. al. discuss the different modeling techniques, concluding that the radial and elliptical basis function (RBF/EBF) approximations provided the most robust models with the least amount of predicted error [7]. Therefore, it was determined that RBFs would work best in this research application.

2.2.1 Radial Basis Functions

Radial basis functions provide values given distances from the functions' origins. In essence, they provide a way to create "unknown functions from known data" which is to be expected in surrogate modeling [14]. RBFs output a value given a distance from the function's origin. In other words, the closer the candidate point is to the known data, the better the prediction will be. Table 2.1 provides examples of several radial basis functions. In these RBF examples, r is an input variable and b is a shape factor that is used to alter the distribution of the RBF.

Table 2.1: Several examples of radial basis functions, where r is an input variable and b is a shape factor.

Type	Equation
Gaussian	$e^{-\frac{r^2}{b^2}}$
Inverse Quadratic	$\frac{1}{b^2+r^2}$
Inverse Multiquadratic	$\frac{1}{\sqrt{b^2+r^2}}$

During design-time, Isight does not provide the user control over the type of RBF. However, in the Isight documentation, the creation of an RBF model is described in detail. Equation 2.1 is the equation listed in Isight's RBF documentation [15]:

$$F(x) = \sum_{j=1}^{N} \alpha_j g_j(x) + \alpha_{N+1} \tag{2.1}$$

where

$$g_j(x) \equiv \left[(x-x_j)^T (x-x_j)\right]^{\frac{MSP}{2}} \tag{2.2}$$

In this context, g_j are radial basis functions—the functions for determining how close x is to the known data points x_j—and $F(x)$ is the complete approximation function given an input parameter x. α_j are unknown expansion coefficients and MSP refers to the "model shape parameter" (a constant) of the RBF. The α array and model shape parameter are computed by Isight during initialization. The summation in Equation 2.1 provides a weighted contribution to the approximation from all of the training data used to initialize the surrogate model. The addition of α_{N+1} comes from an offset calculated when the model is initialized.

Isight provides the values for the α vector, the training data, and the MSP in the coefficients data file. Using this data alone, one can reproduce the RBF approximation outside of Isight. For example, Figure 2.6 provides an RBF approximation of the same data used in Figure 2.4. One will notice that the RBF approximation is significantly better at approximating the known data points as well as interpolating betweeen the training data.

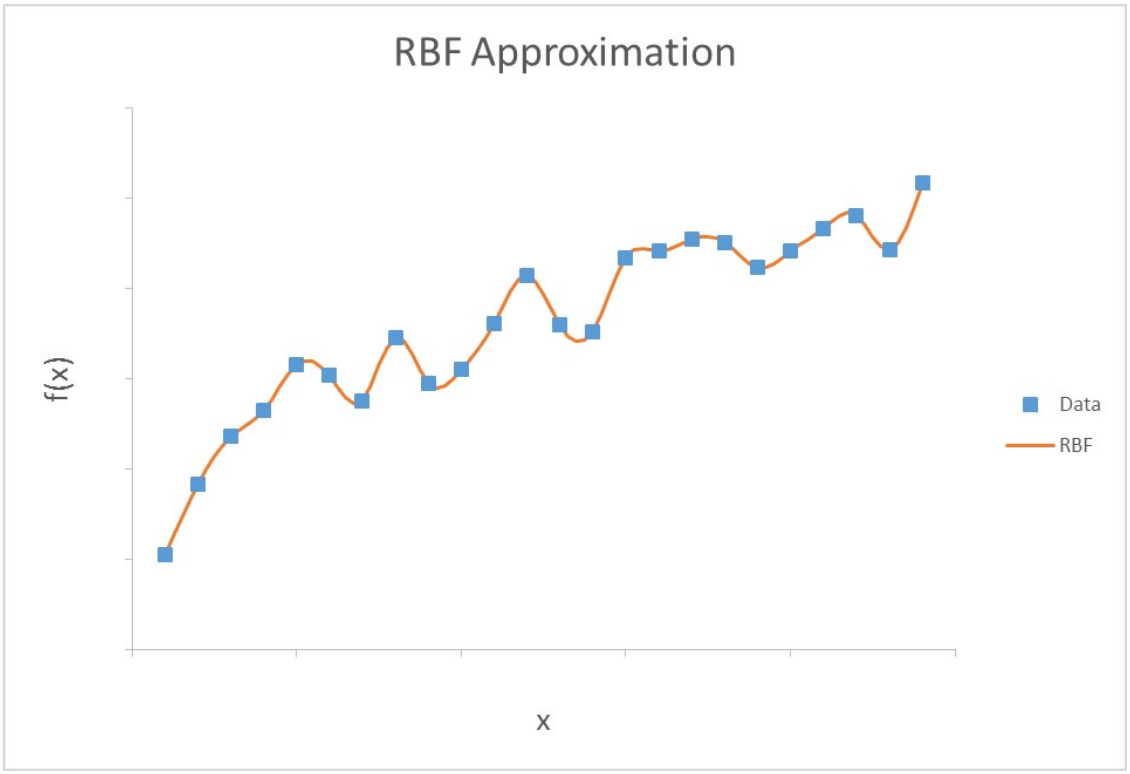

Figure 2.6: An RBF interpolation of the data used in Figure 2.4

2.2.2 Application in Research

One objective in this research was to make the software developed for this research modular; modularity makes object-oriented programs easy to manipulate and alter in the future, whereas monolithic programs—even though they can operate just as well—are extremely difficult to change because of their inherent dependencies. In an effort to make this software modular, Isight was used to create and validate the RBF surrogate models, but the coefficient data was exported from Isight so that the models could be recreated in any environment. For this tool, the "environment" used was a Java application.

In order to properly reconstruct Isight's RBF model using the coefficients data file, Microsoft Excel was used to recreate a sample surrogate model for Equation 2.3.

$$y(x) = x^2 + \frac{1}{2} \tag{2.3}$$

Although simple, Equation 2.3 is sufficiently complicated to test the basic abilities of Isight's RBF approximation. After creating a spreadsheet to recreate the model, a DOE (Latin Hypercube) was run in Isight to identify control input and output values to compare to the new Excel model. When the same inputs were used in the Excel model, the output values were identical to the output values from Isight, which can be seen in Table 2.2. Therefore, the reconstruction process was correct, and it was expanded to support more complicated approximations—ones that utilized additional input parameters. These approximations were also validated with an Isight DOE to ensure proper calculations.

Table 2.2: The outputs from the RBF created in Excel are identical to the outputs from the Isight RBF.

x	$y(x)$	Isight RBF	Excel RBF
0	0.5	0.5	0.5
0.03	0.5009	0.503	0.503
0.1	0.51	0.51	0.51
0.17	0.5289	0.52856	0.52856
0.2	0.54	0.54	0.54
0.205	0.542025	0.54207	0.54207
0.3	0.59	0.59	0.59
0.35	0.6225	0.62246	0.62246
0.4	0.66	0.66	0.66
0.44	0.6936	0.69362	0.69362
0.5	0.75	0.75	0.75
0.58	0.8364	0.83638	0.83638
0.6	0.86	0.86	0.86

2.3 Optimization Techniques

Keane states that the use of optimization techniques in aerospace engineering has become routine in industry [16]. Because many of the components designed for aerospace applications

(like airfoils for an IBR) offer tradeoffs between aerodynamic efficiency and structural robustness, often it necessary to consider multiple objectives when designing new components. However, this paper focuses solely on the blade tuning aspect of airfoil optimization, so only single-objective optimization will be discussed.

When optimizing a design space, there are two primary optimization techniques that can be used: gradient-based algorithms and gradient-free algorithms (genetic algorithms). In summary, gradient-based algorithms are much faster than genetic algorithms. As Zingg et al. state, "...proper exploitation of gradient information can significantly enhance the speed of convergence in comparison with a method that does not compute gradients" [17].

Despite the speed advantages offered by gradient-based optimization algorithms, these algorithms do not explore complicated design spaces as effectively as gradient-free algorithms. Zingg et al. continue, "The key disadvantages of gradient-based methods are precisely the strengths of genetic algorithms...[genetic algorithms] are tolerant of noise in the objective function and have no difficulty with categorical variables or topology. Furthermore, in principle, genetic algorithms find a global optimum" [17].

Determining the global optimum in the design space is the primary goal of this research; therefore, for this application, genetic algorithms are a better choice than gradient-based algorithms.

2.4 Optimization Algorithms

In this thesis, three gradient-free algorithms were utilized to determine the global optima of the two design spaces used: NSGA-II, a genetic algorithm; OMOPSO, a particle swarm optimizer; and CMA-ES, an evolutionary algorithm. Because of the strength of each algorithm and their unique ways of optimizing design spaces known to be complicated, these three algorithms were suitable choices for exploring the design spaces used in this research.

2.4.1 NSGA-II

NSGA-II is a "nondominated sorting genetic algorithm" developed to be a faster, more efficient algorithm than the contemporary evolutionary algorithms [18]. Originally developed by

Kalyanmoy Deb in 2000, the NSGA-II algorithm was designed to overcome three main problems faced by other algorithms, namely: the high computational complexity, a lack of elitism, and the need to specify a sharing parameter. Elitism is an additional factor in a genetic algorithm used to enhance the convergence properties of the algorithm. The sharing parameter is a metric chosen to calculate similarity between two members in a population. NSGA-II overcame all three of these problems, becoming an excellent choice in algorithms and is still "one of the most widely used MOEAs [Multi-Objective Evolutionary Algorithms]" [19].

Figure 2.7 is an example of NSGA-II's exploration path. One will note that for the first 1500 iterations, the algorithm explored many different locations of the design space, which can be seen by the large initial spread of objective values. However, after those iterations, the values of the objective converged, indicating that the algorithm began narrowing its exploration and used only the "best" locations of the design space to find the final objective value proposed.

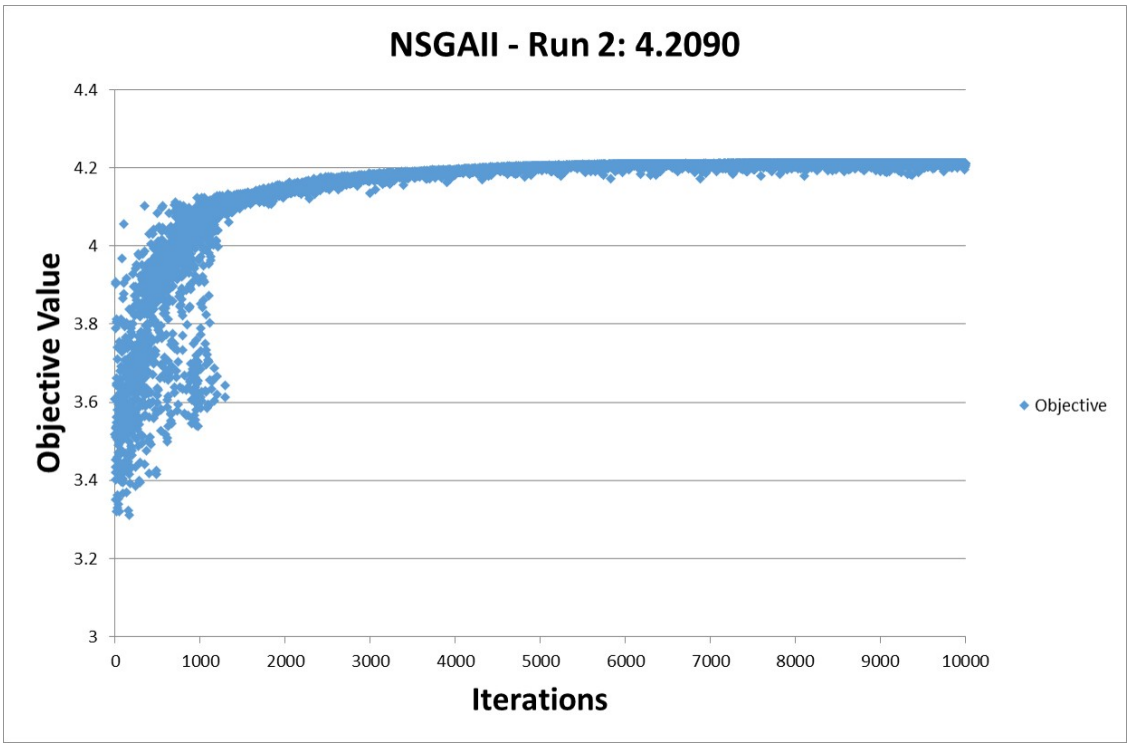

Figure 2.7: The exploration of NSGA-II in a sample design space.

2.4.2 OMOPSO

As a particle swarm optimization algorithm, the OMOPSO algorithm—a Multi-Objective Particle Swarm Optimizer—is a "population-based search algorithm based on the simulation of the social behavior of birds within a flock" [20]. This optimization technique allows the algorithm—even after thousands of iterations—to continue to explore the design space and escape local optima.

From Figure 2.8, one can immediately understand the difference in the particle swarm technique compared to NSGA-II. Within the first several hundred iterations, the algorithm found a competitive objective value and for the rest of the iterations, the majority of the algorithm's exploration revolves around that objective value, which is similar to the previous algorithm. In contrast to NSGA-II, OMOPSO continues to explore different locations of the design space, which is seen by the seemingly random data points with much lower objective values. This is an example of the "social behavior" of a particle swarm: many of the particles stay together while a few venture away from the main "flock."

Based on the research completed by Sierra and Coello, OMOPSO is superior to other particle swarm optimizations and is highly competitive with NSGA-II [20].

2.4.3 CMA-ES

The Evolutionary Strategy with Covariance Matrix Adaption (CMA-ES) algorithm is an evolutionary algorithm for "real-valued global optimization problems" [19]. Even though it was originally designed for small populations (and interpreted to be a robust local search strategy), it has been successfully used on a significant number of real world problems [21]. Furthermore, Hansen and Kern determined that on "seven of eight functions, the CMA-ES can precisely locate the global optimum" and that "the results were compared with other global search strategies, stated to achieve superior results...the CMA-ES outperforms these global searchers, typically by a factor of three..." [21].

To better understand the exploration techniques of CMA-ES, see Figure 2.9. Notice that for the first 200 iterations, the objective values found by each iteration are similar. However, as the iterations continue (iterations 200 to 3000), the values for the objective begin to spread out,

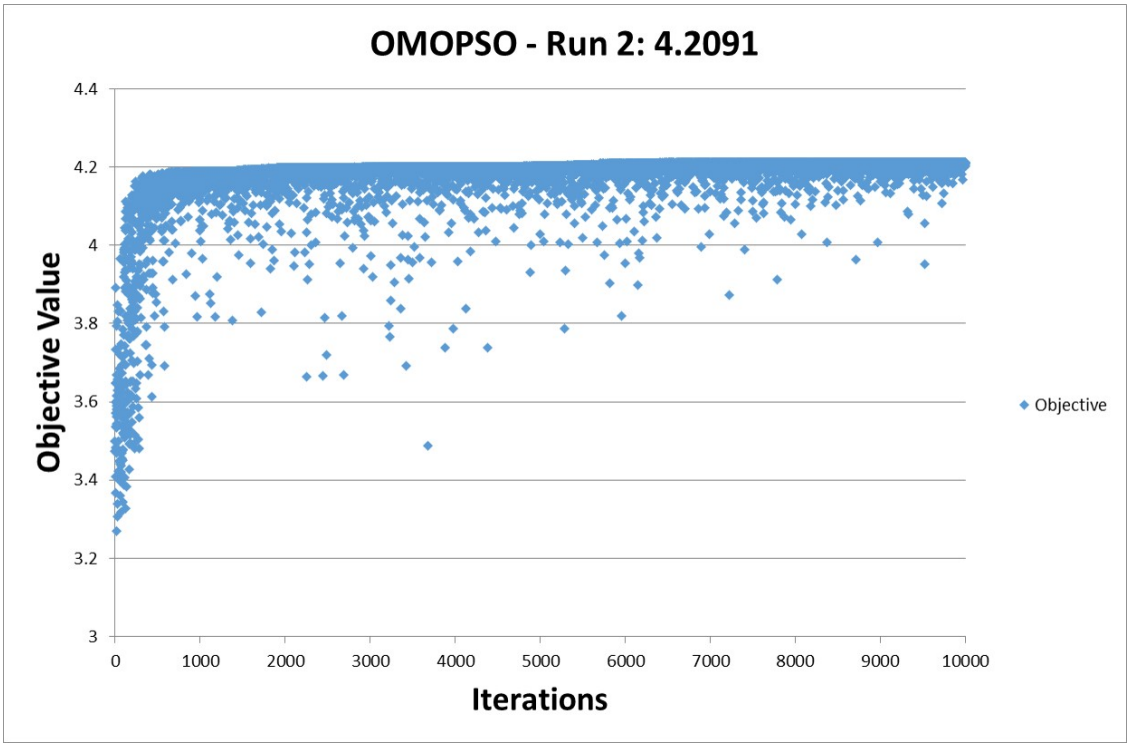

Figure 2.8: OMOPSO's exploration path through a sample design space.

illustrating how the algorithm is attempting to explore the complete the design space. After several thousand iterations, though, the objective value begins to converge in a similar manner to NSGA-II.

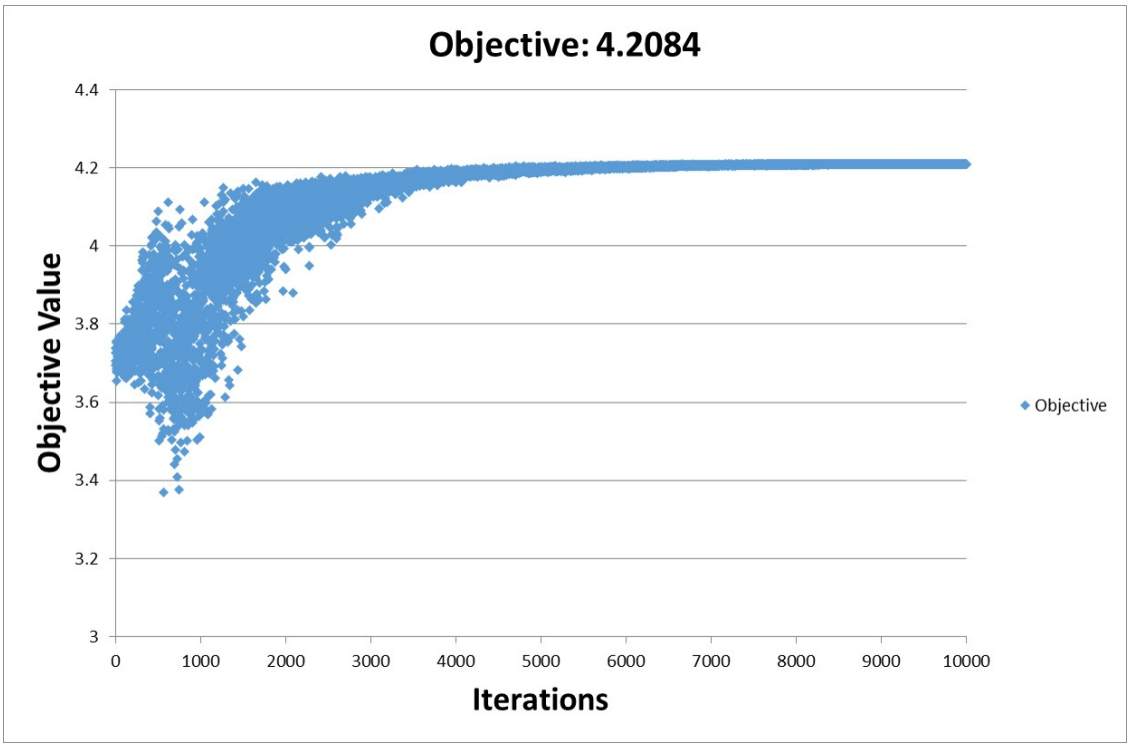

Figure 2.9: The exploration of CMA-ES.

CHAPTER 3. METHOD AND IMPLEMENTATION

3.1 Summary

Blade tuning is inherently an iterative process because it is only one consideration in the complete design of an airfoil. During the design process, the aerodynamic and vibrational properties of the blade must be analyzed. However, in this thesis, the focus is completely on the vibrational properties of the blade. It is assumed that the structural and aerodynamic analyses have already been performed and certain ranges have been given to each of the blade's parameter values. Therefore, it is the vibrational analyst's task to determine the best geometry of the blade using values within those ranges. Though this process is complicated, it can be broken down into six, straightforward steps:

1. Generate training data mapping inputs (blade geometry parameters) to outputs (modes at operating speeds)
2. Create surrogate models for each mode at each operating speed using the training data
3. Query the surrogate models to generate an initial Campbell diagram as a baseline for comparing later designs
4. Review the Campbell diagram for "good" and "bad" crossings
5. Tune the blade by adjusting the geometric parameters of the blade
6. Output the results of the tuning, including the final blade parameters and final Campbell diagram

Figure 3.1 illustrates these six steps and provides insight into the process of blade tuning by showing the iterative design loop used to tune an airfoil.

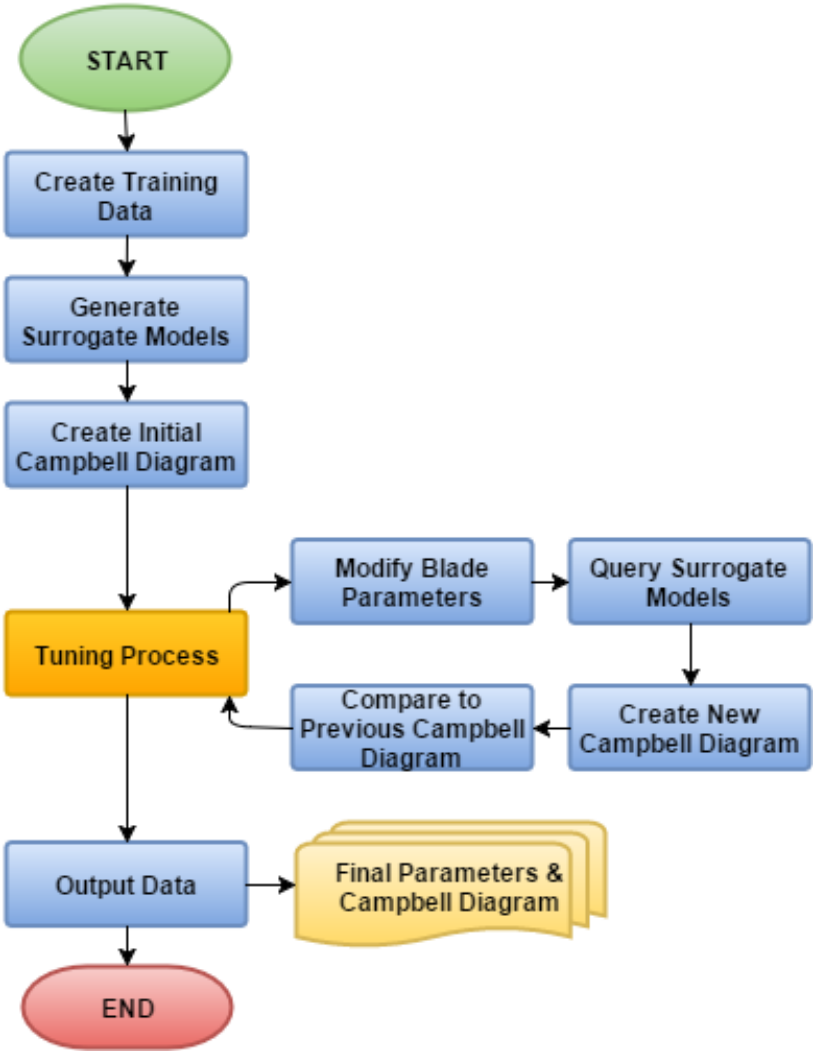

Figure 3.1: A flowchart illustrating the general blade tuning process.

3.2 Generate Training Data for Surrogate Models

In order to create a useful surrogate model, the model must be initialized with training data. Training data consists of known inputs and outputs for the function to be approximated. In this research, the training data maps geometric parameters from the airfoil to modal frequencies of the blade. The training data used for this research was provided by the research sponsor. The data was generated by performing an Optimal Latin Hypercube design of experiments (DOE) using finite element modal analysis on a parametric airfoil [22, 23]. For each run of the DOE, different combinations of parameter values (inputs) were supplied to create new airfoils which were analyzed with FEA to obtain new modal data (outputs).

It is possible to integrate the finite element modal analysis into the tuning process in order to determine "exact" modal changes due to changes in parameter values. Figure 3.2 illustrates how FEA can be integrated into the tuning process. Note that the tuning process presented in Figure 3.2 is nearly identical to the tuning process in Figure 3.1, but the use of surrogate models in Figure 3.1 has been replaced by FEA in Figure 3.2.

Though FEA provides exact modal changes, it also comes with an increased run-time. Heap et al. note that the use of surrogate models drastically increases the speed of optimization since the optimization must only query a continuous model instead of performing a new finite element analysis each time the parameters change [7]. In fact, surrogate models take only a few milliseconds to query in comparison to minutes or hours (depending on fidelity of the analysis) for each finite element analysis (see Section 2.2).

To remove modal analysis from the tuning process, a DOE can complete multiple analyses prior to tuning to create a set of training data. Then, a surrogate modeling technique can be used to approximate the values of the space between the known data points (described in Section 2.2.2). As an approximation of the finite element results, this is not as accurate as using FEA in the design loop; however, querying the model is orders of magnitude faster than determining a new finite element solution for every iteration in the tuning process.

In general, the accuracy of a surrogate model is dependent on the amount of training data used to initialize the model: the more data used, the better the accuracy. Taken to the limit of an infinite amount of training data, the surrogate model would be as accurate as the analyses used to generate the training data. However, generating an infinite data set is impossible, so for the finite data sets used in this research, the average error of each model ranged from 1.74% to 6.13%. For more information, Section 3.3 discusses error approximations for surrogate models. Also, Sections 4.1.1 and 4.2.1 discuss the accuracy of the surrogates used to approximate the data provided in the two data sets. Therefore, for this research, it was a conscious decision to sacrifice a small amount of accuracy for a drastic reduction in run-time.

3.3 Create Surrogate Models for each Mode

To create surrogate models, one can either generate the models using the necessary equations (see Section 2.2.1) or a commercial software package. For simplicity, this research used

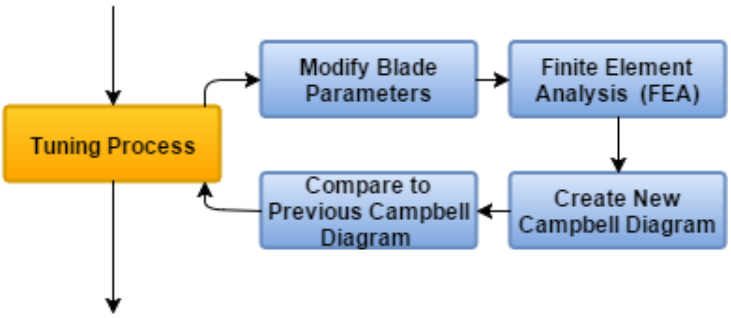

Figure 3.2: The general process for blade tuning incorporating finite element analysis into the design loop instead of surrogate modeling.

SIMULIA's multi-disciplinary optimization software (MDO), Isight, to produce surrogate models for mapping between airfoil geometry and the modal properties of the blade. The primary advantages that Isight provides over writing new code is that it has already been tested, debugged, and validated, and it has the ability to perform cross-validation error analyses on each of the surrogate models. This provides the designer insight into the fidelity of each approximation: the lower the predicted error, the higher the fidelity of the approximation. As described in the Isight documentation, "For cross-validation error analysis, a number of data points will be removed from the sampling data set, one at a time. For each of the removed points, the approximation coefficients will be re-calculated, and the exact and approximate output values will be compared. The removed point is then put back into the data set and the next point is removed. The points are selected randomly." [22]

Using Cross-Validation analysis, Isight can calculate the average (MEAN) error, the root mean square (RMS) error, and the maximum (MAX) error of each surrogate model. Each error is calculated according to following equations, where $f_{i,pred}$ is the predicted value of the surrogate, $f_{i,act}$ is the actual value, n is the number of points used to calculate each error, and $max(x)$ is a function that finds the maximum value from list of terms in its input x [22]:

$$MEAN = \frac{\sum_{i=1}^{n} \left| \frac{f_{i,pred} - f_{i,act}}{f_{i,act}} \right|}{n} \tag{3.1}$$

$$RMS = \frac{\sum_{i=1}^{n} \left| \frac{\sqrt{f_{i,pred}^2 - f_{i,act}^2}}{f_{i,act}} \right|}{n} \tag{3.2}$$

$$MAX = max\left(\left| \frac{f_{i,pred} - f_{i,act}}{f_{i,act}} \right| \right) \qquad for\ i = 1,2,...,n \tag{3.3}$$

For example, using the training data points and RBF surrogate used in Section 2.2.1, Isight predicted the errors listed in Table 3.1. Using this table, one can see that the average predicted error within the range of the training data is 2.13% and that the maximum predicted error for the range is 12.28%. This means that, on average, the surrogate will predict values within 2.13% of the actual function's data. Furthermore, the largest the error can be is 12.28%. The RMS error gives an error value between the MEAN and the MAX errors by weighting the values with larger differences (by squaring the terms first before subtracting). To visualize the approximation, see Figure 2.6. With only 24 points and a complicated function to approximate, a predicted average error of 2.13% is sufficient to use for approximating the original function.

Table 3.1: The predicted MEAN, RMS, and MAX errors for the RBF surrogate used in Section 2.2.2, based on Isight's cross-validation.

Properties	Values
Training Points	24
Inputs	1
Outputs	1
MEAN	2.13%
RMS	4.70%
MAX	12.28%

In addition to the native error analysis, Isight provides four native surrogate modeling techniques: response surface models, Kriging models, radial/elliptical basis function (RBF/EBF) models, and Chebyshev/Orthogonal Polynomial models. Based on the analysis presented by Heap et al., Isight's implementation of the RBF approximation is the best surrogate modeling technique to use due to its robustness and accuracy [7]. Therefore, RBF modeling was used to create the surrogates for this research.

One of the main goals of this research was to keep the project modular, so that each portion could be changed easily. Therefore, once the RBF models were created inside of Isight (see Section 2.2), the data representing the surrogate models were exported from Isight for external use: the exported data file provides all the necessary information for recreating the approximation in a different environment than Isight, including the α vector, the training data, and the MSP (described in Section 2.2.1). Thus, the designer is no longer limited to Isight's functionality for the next four steps in the automation process.

3.4 Query the Surrogate Models to Create a Campbell Diagram

In this research, querying the surrogate models requires several steps, including the selection of an appropriate programming language, the construction and validation of surrogate models in the chosen language, and the expansion of the hard-coded approximations so they are built from Isight coefficient data file. After this process is complete, the surrogates can be used to map between the blade parameters and points for the Campbell diagram, thus generating the modes at specific operating speeds.

3.4.1 Programming Language

In order to connect the steps of the blade tuning process, it was necessary to create a software tool to link the surrogate models to the Campbell diagram data, the Campbell Diagram data to the objective function, and the objective function to the surrogate models, which is summarized in the tuning process loop in Figure 3.1. To link the programs, a suitable programming language was essential. Although any object-oriented language would be capable of connecting the different components, for this thesis Java was chosen because of two open-source libraries that would allow the focus of this research to remain on automating this process:

- The JFreeChart framework provided a simple, yet effective, way of displaying the Campbell diagrams
- The Multiobjective Evolutionary Algorithms (MOEA) framework offered a full set of gradient-free optimization algorithms, including NSGA-II, CMA-ES, and OMOPSO (see Section 2.4)

Furthermore, Java was chosen because it is platform-independent, making this tool usable on both Windows and Linux operating systems.

3.4.2 Create Surrogates Using Java

Using the hard-coded process from Excel described in Section 2.2.2, an algorithm for generating RBF models in Java was developed using the same training data, coefficients data, and governing equations. This RBF approximation was validated in the same manner as the Excel model in Section 2.2.2; specifically, the predicted values from the Java RBF were compared to the predicted values from the Isight RBF.

The results of this comparison have been listed in Table 3.2, where the first column (x) defines the input values used in the function ($f(x)$) to be approximated. The second column ($f(x)$) calculates the actual output values. The Isight RBF column approximates $f(x)$ and one will notice that the some of the predicted values are slightly different from the actual values. This is expected since the Isight RBF is an approximation of the function. The fourth column illustrates that the values predicted by the Java RBF exactly match the values in the Isight RBF column, validating the algorithm for constructing the Java RBF. These values also match the values in the Excel RBF column of Table 2.2.

3.4.3 Parsing the Isight Coefficient Data File

Though hard-coding the coefficient values was effective for testing and verifying the Java RBF approximations, it is a short-sighted approach to making a software tool for analyzing any airfoil because the α coefficients and the training data vary for each surrogate model created. For example, suppose n training data points are used to initialize a surrogate model with m input parameters. Furthermore, once the model has been initialized, there must be $(n+1)$ α coefficients and 1 MSP value. Thus, for this surrogate model, $(n*m)+(n+1)+1$ values are required to hard-code the model. Furthermore, the number of hard-coded points is multiplied by the number of modes and the number of design speeds. Typically, there are at least 15 blade parameters, more than 300 training data points, at least 6 modes, and 4 operating speeds. Thus, in this example, it takes

Table 3.2: The outputs from the RBF created in Java exactly match the outputs from the Isight RBF.

x	$y(x)$	Isight RBF	Java RBF
0	0.5	0.5	0.5
0.03	0.5009	0.503	0.503
0.1	0.51	0.51	0.51
0.17	0.5289	0.52856	0.52856
0.2	0.54	0.54	0.54
0.205	0.542025	0.54207	0.54207
0.3	0.59	0.59	0.59
0.35	0.6225	0.62246	0.62246
0.4	0.66	0.66	0.66
0.44	0.6936	0.69362	0.69362
0.5	0.75	0.75	0.75
0.58	0.8364	0.83638	0.83638
0.6	0.86	0.86	0.86

115,248 hard-coded values to generate the surrogates to create a Campbell diagram. Therefore, hard-coding is infeasible for this tool.

A simpler way to generate the surrogate models in Java is to simply parse through the coefficient data files from Isight. A Java parser was written to extract the necessary information from each data file and hold it in memory during run-time. This information was used in the calculations from Equations 2.1 and 2.2 to generate new surrogate models each time the program is run.

Automating this process is important because it provides the foundational models for the entire tuning process. By parsing through the Isight data file and retrieving the relevant information automatically, any RBF approximation data file from Isight can be used to create the foundation of the entire automation process. Also, instead of setting up the Campbell diagram to be based on a specific number of blade parameters, it is based on 'n' parameters, making this a more robust and generic process.

3.4.4 Create the Campbell Diagram

A Campbell Diagram illustrates the relationship between the modal frequencies of an airfoil and the operating speeds for the jet engine. Therefore, in order to create a Campbell diagram, the engine speeds, the airfoil's modal frequencies, and the engine orders of the system must be known.

The operating speeds are independent variables and are specified in the initial design. The engine orders are of the form:

$$f_{eo}(\omega) = C\omega \tag{3.4}$$

where f_{eo} is the engine order's frequency, C is a scalar value that is set based on the number of obstructions associated with the engine order, and ω is the speed of the engine. Therefore, each engine order is simply a linear function of the engine speed with no offset frequency.

Each point on the Campbell diagram represents a certain mode at an operating speed. In this thesis, the points on the diagram are numbered by their mode first and then their speed. For example, the point corresponding to Mode 2 at Speed 1 is Point 2; likewise, Point 3 corresponds to Mode 3, Speed 1, and Point 11 corresponds to Mode 5, Speed 2. These points, and several others, are shown in Figure 3.3. Each point in a mode is connected by a line to approximate the behavior of the mode between the operating speeds. These connections are primarily for visual clarity because the values of the modal frequencies between the operating speeds are not important for tuning.

Each point on the diagram has its own surrogate model, so as the blade parameter values change, the frequencies of the points change in response. Suppose there are six modes and four design speeds, then there are 24 points on the Campbell diagram that need to be updated each time the blade geometry changes. An initial Campbell diagram can be created using these RBF approximations, operating speeds, and engine orders. A sample diagram is provided in Figure 3.3.

3.5 The Tuning Process

The tuning process is an inherently iterative process, which is depicted in Figure 3.1. By looking at the Campbell diagram for a given blade, one can discern if the airfoil is properly tuned—if the modal frequencies of the airfoil are near the engine order frequencies at an operating speed,

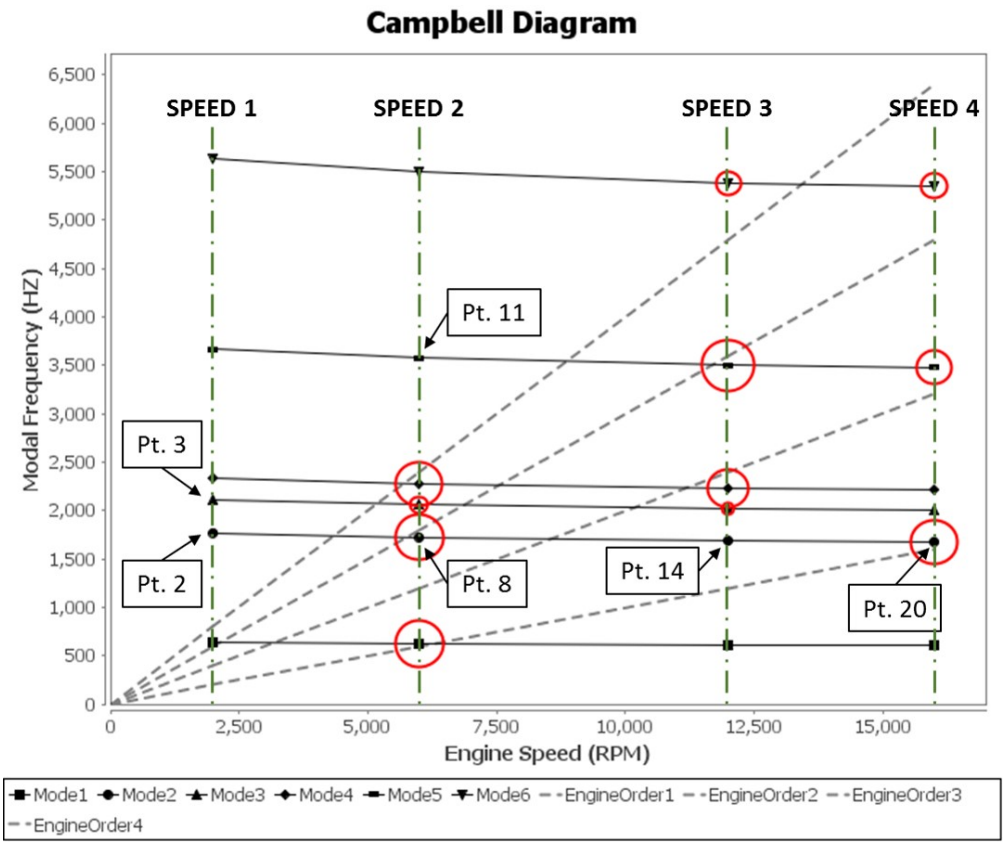

Figure 3.3: A Campbell diagram with six modes and four operating speeds. The circles indicate points of concern. The radii of the circles indicate the severity of the resonance.

resonance will occur. Therefore, in order to tune a blade, each point on the Campbell diagram should be "far enough" away from the closest engine order.

Consider Figure 3.3: let the frequency ratio (η) for Point i be defined by Equation 3.5

$$\eta_i = \left| \frac{f_i - f_{eo}}{f_{eo}} \right| \tag{3.5}$$

where f_i is the frequency of Point i and f_{eo} is the frequency of the nearest engine order to Point i. Also, define the fitness ratio (κ) to be a threshold set by the designer.

If $\eta_i < \kappa$, then Point i is too close to the nearest engine order, and, therefore, resonance is possible. The designer must alter the geometry of the airfoil to change this point's frequency to increase η_i. Once $\eta_i \geq \kappa$, the mode is "far enough" away and resonance will not occur.

When the blade parameters are altered, the surrogate models generate new values for the modal frequencies at operating speeds, and, therefore, new points on the Campbell diagram. These

new points are shown on an updated Campbell diagram and the designer reviews the diagram to decide if the updated design is better. The current standard in industry is to perform this review manually; however, having a human in the design loop slows the design process. In this thesis, this method of tuning will be referred to as manual tuning.

3.5.1 Speed Tolerances

It is important to note that operating speeds have certain tolerances associated with them. If these tolerances are incorporated into the Campbell diagram, they create vertical "bands" for the operating speeds instead of single, vertical lines. If accounted for, these bands would make η either more conservative (subtracting the tolerance from the operating speed) or less conservative (adding the tolerance to the operating speed). Consider Figure 3.4 which illustrates this situation by focusing on a mode at an operating speed on a Campbell diagram.

For a certain operating speed (the solid, red line), there is a tolerance "band" (the area between the dashed, red lines). Suppose that the candidate blade has a constant modal frequency (the green, horizontal line) within that band, and that the nearest engine order's frequency is linear (the dashed, blue line). Define f_k to be the frequency associated with a certain location k in Figure 3.4.

Without tolerances, $\eta_B = \left|\frac{f_B - f_A}{f_A}\right|$. With tolerances, $\eta_{B,-} = \left|\frac{f_D - f_C}{f_C}\right|$ or $\eta_{B,+} = \left|\frac{f_F - f_E}{f_E}\right|$. From observation, $\eta_{B,-} < \eta_B < \eta_{B,+}$, meaning that $\eta_{B,-}$ is more conservative than η_B and $\eta_{B,+}$ is less conservative than η_B.

One may also wonder why the Euclidean distance between locations C and B (the dotted, purple line) is not used for the numerator in Equation 3.5. Simply put, this would require the engine to operate at two different speeds simultaneously, namely the speed corresponding to location C and the speed corresponding to location B. Because this is impossible, the Euclidean distance is not used.

In this research, the tolerances on the operating speeds will be ignored because it is assumed that factors of safety to account for speed tolerances are already incorporated when the designer sets the value of κ. Therefore, the calculations for the η will follow Equation 3.5.

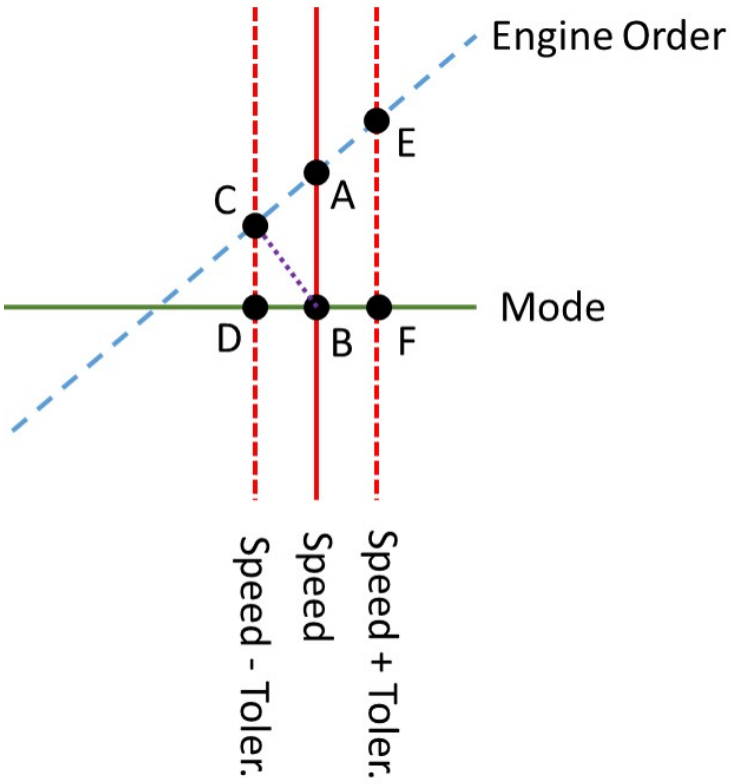

Figure 3.4: A simple illustration of the "band" created by incorporating tolerances for the operating speeds.

3.5.2 Manual Tuning

Manual tuning is difficult because the designer cannot independently test each parameter without simulating the process of a full-factorial DOE. Furthermore, the blade geometry parameters do not map one-to-one to points on the Campbell diagram. In other words, one geometric parameter can affect several points on the Campbell diagram. This proves problematic and is illustrated in Figure 3.3 at Point 20.

For Point 20 (Mode 2 at redline), the nearest engine order is too close, which is denoted by the circle. To fix this problem, the designer alters the blade geometry to increase the frequency of Point 20. It is important to note that changes in geometry generally affect the entire mode—or multiple modes—not just the point of interest. Therefore, by increasing the frequency of Point 20, Points 8 and 14 become problematic. Based on the characteristics of the blade, its modes, the engine orders, and κ, it is often impossible to move all of the points so that $\eta_i \geq \kappa$. Thus, it

is insufficient to simply "optimize" each point in series, but it is essential to consider all points simultaneously to find a truly optimal design. This can only be done through algorithmic tuning, which is presented in Section 3.5.3.

The flowchart of manual tuning was presented in Figure 3.1, where a designer examines the current Campbell diagram to determine which modes are too close to the engine orders, modifies blade parameters to move the problematic modes away from the engine orders, reviews the updated Campbell diagram, and either accepts or rejects the changes. An example of this process is shown in Figure 3.5: the designer begins with a candidate blade, examines its Campbell diagram, and determines there are problematic crossings; he alters the blade's geometry and reviews the updated Campbell diagram; if the design is better, he accepts the changes and continues to make improvements.

Manual tuning is useful for finding designs that are "good enough," but it cannot guarantee an optimally tuned airfoil—a human designer simply cannot explore the complete design space offered by, say, fifteen blade parameters. The designer can explore each parameter in series, but there is no way for him to effectively explore all the points simultaneously. However, an experienced designer can rapidly complete the manual tuning process to find an acceptably tuned airfoil—so if a "good enough" design is what matters, manual tuning can be completed fairly quickly.

3.5.3 Algorithmic Tuning

In contrast to manual tuning, algorithmic tuning is the novel process proposed in this thesis. It provides a way to consider all of the points on the Campbell diagram simultaneously using an objective function. Even though the general process for algorithmic tuning is very similar to manual tuning as shown in Figure 3.6, there is one primary difference: an optimization algorithm is used to analyze the tuning of the airfoil instead of a human designer. This provides significant advantages in two aspects: (1) the speed of the design process and (2) the quality of the design process. For example, if the surrogates on the Campbell diagram take a few milliseconds to query, within seconds after the optimization algorithm begins, hundreds of candidate designs have been analyzed. Also, by using a gradient-free algorithm to optimize the design space, thousands of candidate blades can be explored in minutes. In comparison to humans, computers are much more effective for effectively exploring n-dimensional design spaces.

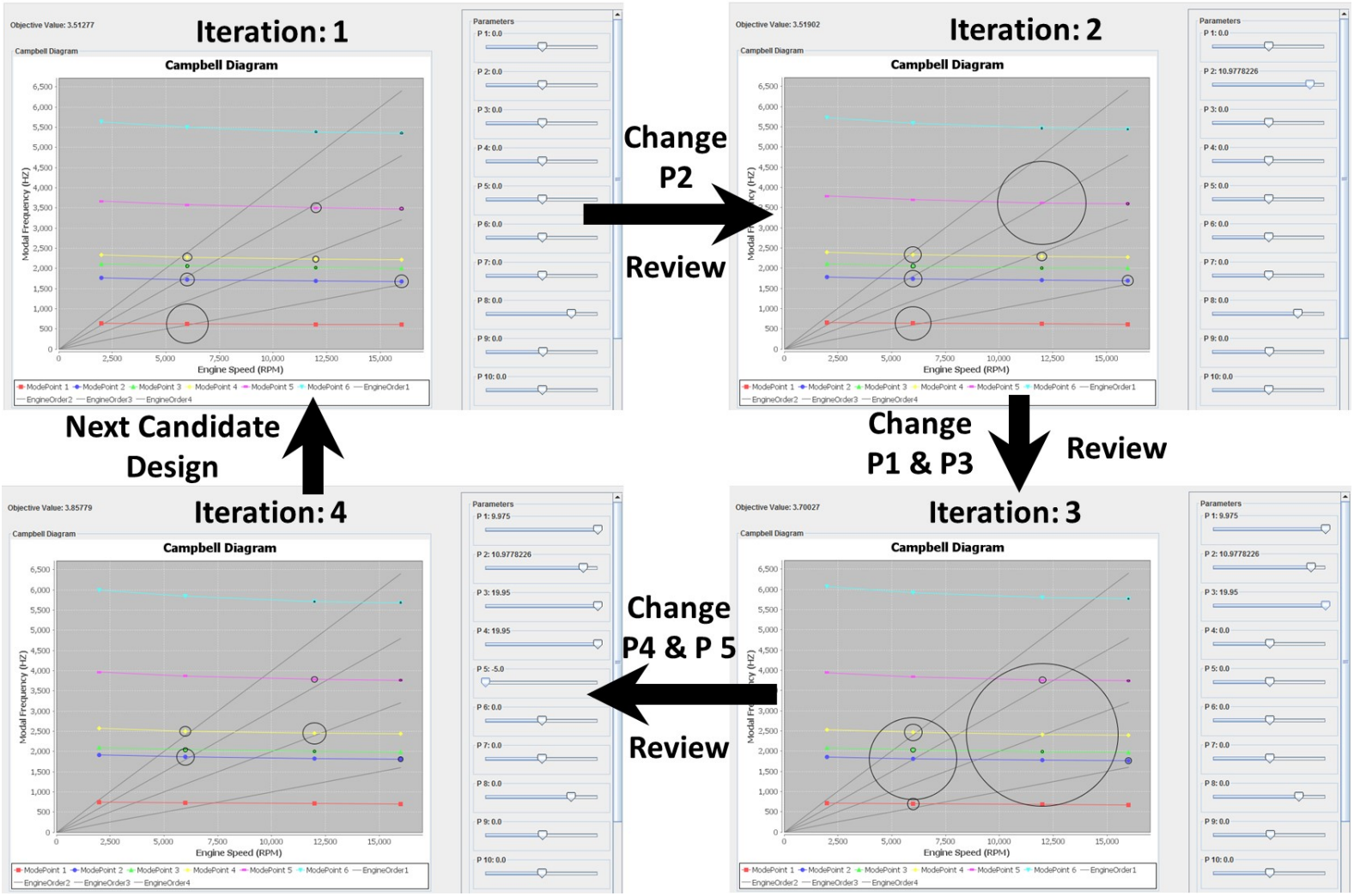

Figure 3.5: A summary of the manual tuning process: blade parameters are changed, the updated Campbell diagram is reviewed, additional blade parameters are modified, the updated diagram is reviewed, and so on.

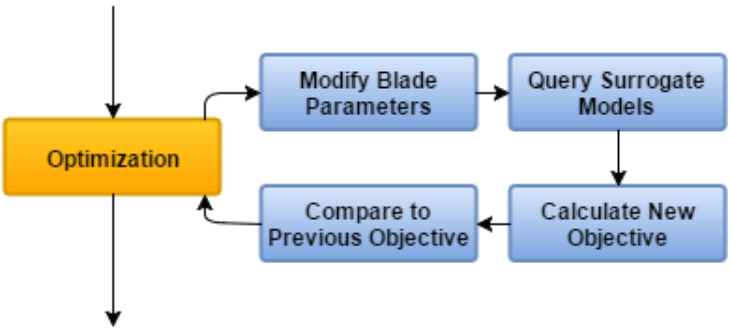

Figure 3.6: The algorithmic tuning process.

Because the algorithm cannot "see" the updated Campbell diagram to compare it to the previous Campbell diagram, it is necessary to quantify each design in order to decide whether a new design is better than the previous, best design, thus optimizing the tuning process. This quantification is referred to as an objective function, and, as mentioned in Section 3.5.2, this objective function is able to account for every point on the Campbell diagram simultaneously. The objective is used to determine the blade design with the least possibility for resonance.

Objective Function

To consider all of the Campbell diagram points simultaneously, each point must contribute to the objective function—the function to be optimized. Equation 3.6 defines the objective function:

$$Max : z = \sum_{i=1}^{b} w_i \eta_i + \sum_{j=1}^{n-b} w_j \kappa \tag{3.6}$$

where z is the function to maximize, $w_{i,j}$ are weights for each point, b is the number of points where $\eta < \kappa$, n is the total number of points on the Campbell diagram, η_i is the frequency ratio for Point i, and κ is the fitness ratio. The code used to calculate the objective function for this research is provided in Listing 3.1.

Some points on a Campbell diagram are more important than others to consider, but, for simplicity in this thesis, each point's contribution is equal in weight; however, Equation 3.6 and Listing 3.1 account for unique weights (see Section 5.5).

Listing 3.1: The code used in this research to calculate the objective function

```
public double calculateObjectiveFunctionValue()
{
    double z = 0.0;
    for (ModePoint mode : modePointList)
    {
        counter++;
        if (!mode.isGood())
        {
            z += mode.calculateWeight(1.0) *
                    Math.abs(mode.getFrequencyRatio());
        }
        else
        {
            z += mode.calculateWeight(1.0) * fitnessRatio;
        }
    }
    return g;
}
```

The objective function is constructed by first separating the points of the Campbell diagram into two categories: (1) Points with $\eta \geq \kappa$ and (2) Points with $\eta < \kappa$ (see Section 3.5). All the points in the first category contribute the same amount to the objective function—there should be no advantage for those points to be pushed further away from the nearest engine order. Therefore, each point in Category 1 should contribute a constant value to the objective function. In contrast, the points in Category 2 should contribute values less than the constant value. For this thesis, η was used as the contribution to the objective until $\eta = \kappa$. Then κ was used as the constant value contributed. This concept is illustrated in Figure 3.7 for a fitness ratio set at 0.2.

Consider, for example, Point 8 in Figure 3.3. Point 8 has a modal frequency of 1700 Hz and the nearest engine order has a frequency of 1750 Hz at the same speed. Therefore, $\eta_8 = |(1700 - 1750)/1750| = |-0.0286|$ where the negative indicates that the nearest engine order is located above Point 8 on the Campbell diagram. Suppose $\kappa = 0.2$. Because $\eta_8 < \kappa$ ($|-0.0286| < 0.2$), Point 8 contributes 0.0286 to the objective. In contrast, consider Point 11 ($\eta_{11} = 0.479$). Because $\eta_{11} > \kappa$, Point 11 contributes κ (0.2) to the objective instead of η_{11} (0.479).

Using this objective function, the theoretical maximum value for the objective (z_{max}) is defined in Equation 3.7, where n is the total number of points on the Campbell diagram. For the

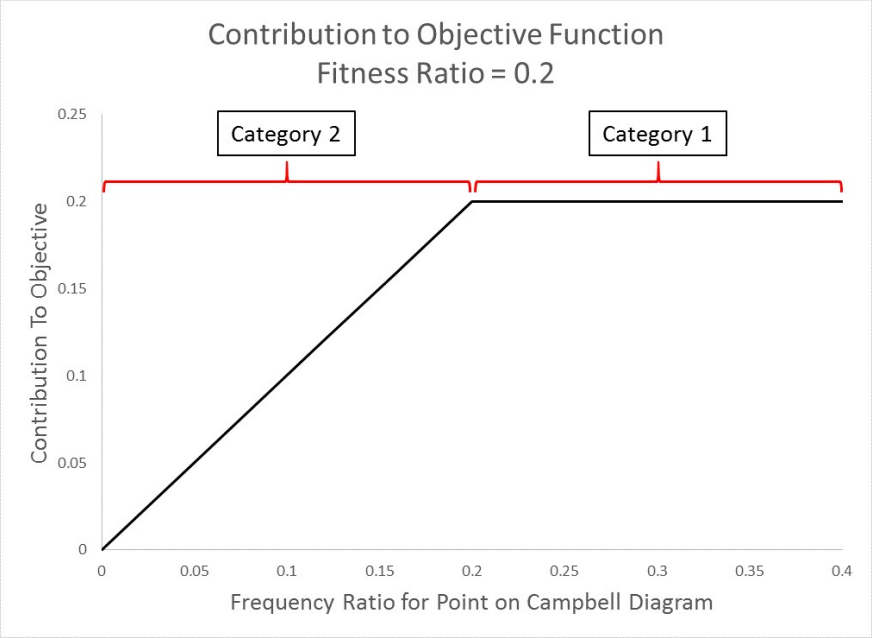

Figure 3.7: An illustration of each point's contribution to the objective function. Once a point is "far enough" away, it contributes κ to the objective function.

Campbell diagram in Figure 3.3, there are 24 points with $\kappa = 0.2$. Therefore, $z_{max} = 4.8$ for this design space.

$$z_{max} = \kappa n \tag{3.7}$$

3.5.4 Graphical User Interface (GUI)

In order to manually tune an airfoil, a designer must be able to make changes to the blade's geometry and visualize how those changes affect the modal frequencies of the blade; then, if necessary, the designer can continue to make changes to the blade. An easy way to implement this iterative process is to create a graphical user interface (GUI) to display a Campbell diagram to the designer as well as a way to change each parameter value. For this research, the Java library JFreeChart was used to create a Campbell diagram, and a custom version of Java's JSlider class was used to allow the designer to easily manipulate the airfoil's parameters. This GUI can be seen in Figure 3.8.

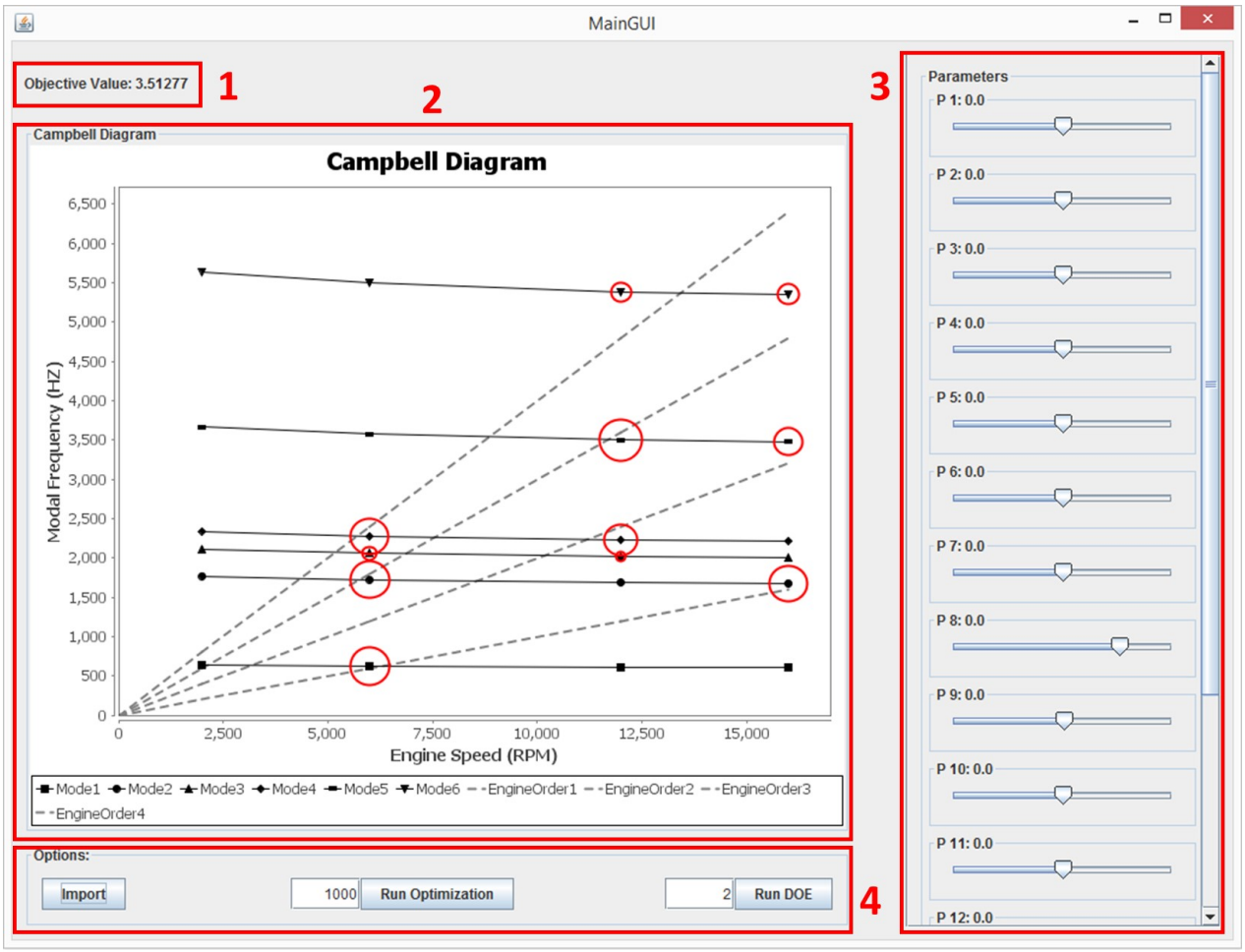

Figure 3.8: An overview of the GUI developed for this research.

Before the GUI launches, the program parses through the Isight coefficients data files for the candidate airfoil. Using this information, the RBF surrogates are generated and held in memory so they can be queried repeatedly throughout the design process (noted in Section 3.4.3). Once all of the surrogates have been generated, the information necessary for the initial Campbell diagram is calculated. Last, the GUI is displayed to the user. The GUI provides a front-end to this software tool for the designer to use to manipulate the blade parameters and visualize the updates as the Campbell diagram is updated.

The GUI can be separated into several integral panels, which have been labeled with numerical identifiers in Figure 3.8.

(1) Objective Panel

The Objective panel is the simplest of the four panels, containing only a label and a value for the objective function (see Figure 3.8). This value is calculated in real-time as the blade parameters are changed, according to the process defined in Section 3.5.3. In essence, the value provides a metric for the tuning quality of the airfoil.

(2) Campbell Diagram Panel

The Campbell Diagram panel contains a Campbell diagram created based on the current values of the parameters values listed in the Parameter Sliders panel. This diagram is updated in real-time to provide an effective and accurate method for manually tuning the airfoil. As one can see in Figure 3.8, circles have been added to each point on the Campbell diagram to indicate the potential each point has for resonance. The radius (R_i) of each circle is calculated according to Equation 3.8, based on a scaling value (C) to make the circle visible on the Campbell diagram, the frequency ratio (η_i), and the fitness ratio (κ) discussed in Section 3.5. If the frequency ratio of a certain point is greater than the fitness ratio, no circle appears on the Campbell diagram. Otherwise, the size of the circle denotes the severity of the crossing. Like the rest of the Campbell diagram, the circles update in real-time as the parameter values are changed.

$$R_i = C(\kappa - \eta_i) \tag{3.8}$$

(3) Parameter Sliders Panel

The Parameter Sliders panel, shown in Figure 3.8, lists the parameters used to create the surrogate models for the Campbell diagram. Because Java's `JSlider` objects only allow `int` values to be stored, a custom class, `DoubleSlider`, was created to store `double` values. As an inherited class, a `DoubleSlider` object is able to utilize the primary functionlity of a JSlider—the registered listeners, display properties, etc—while overriding the properties to `get` and `set` its current value. This customization allows the designer to make incremental changes to the parameters values instead of only being able to change them by integer values.

(4) Options Panel

The Options panel consists of three buttons:

1. Import—This button imports parameter values from a text file. These parameters are used to update the slider values, update the displayed Campbell diagram, and calculate a new objective value for the Objective panel.

2. Run Optimization—This button begins the optimization process described in Section 3.5.3. The user can specify the convergence criteria (the maximum number of iterations) by changing the value to the left of the button. As seen in Figure 3.8, the default number of iterations is 1,000. The specific algorithm (NSGA-II, CMA-ES, or OMOPSO) must be defined in the source code before running the program.

3. Run DOE—This button starts running a full-factorial design of experiments on the design space. It uses the user-defined number (default: 2) to determine how many different values to use when testing each parameter. For example, a value of 3 would test each parameter at its minimum, middle, and maximum value. This is a "brute force" exploration method, and, for a complicated design space (over 15 parameters), this can take more than 10 hours to complete. The DOE process is multithreaded for parallel processing.

3.6 Output Data

The last step in Figure 3.1 is to output the final parameters for the airfoil. For manual tuning, this is accomplished by simply viewing the values provided by the parameter sliders and examining the Campbell diagram. For algorithmic tuning, when the optimization completes, it produces a text file listing the parameters for the best objective at each iteration as well as the final, optimum value. For example, if the algorithm was permitted 10,000 iterations, there would be 10,000 lines in the output file, plus the final, optimum value. Figure 3.9 displays a portion of the end of the optimization output file. The parameter sliders, Campbell diagram, and objective value automatically update to this final, optimum value when the optimization terminates. The optimization values from Figure 3.9 were used in Figure 3.10 to display the optimized Campbell diagram.

Iteration	η_1	η_2	η_3	η_4	η_5	...	η_m	Objective
9990	1.17167	1.634757	1.616987	2.478484	4.282093	...	-0.10725	4.197192
9991	1.173721	1.637789	1.612762	2.500515	4.325838	...	-0.10419	4.209005
9992	1.174329	1.642432	1.618526	2.495926	4.29284	...	-0.10323	4.20201
9993	1.173134	1.637505	1.612501	2.501005	4.324548	...	-0.10419	4.209001
9994	1.173748	1.637804	1.612771	2.500456	4.325866	...	-0.10416	4.208984
9995	1.175142	1.639139	1.613213	2.499979	4.322875	...	-0.10519	4.207517
9996	1.171852	1.635113	1.611475	2.499002	4.328836	...	-0.10302	4.206847
9997	1.173865	1.637884	1.612778	2.500462	4.326035	...	-0.10418	4.208992
9998	1.17404	1.637984	1.612764	2.500411	4.326363	...	-0.10411	4.208861
9999	1.173849	1.637873	1.612746	2.500496	4.326039	...	-0.10417	4.209
10000	1.171595	1.632808	1.609668	2.496056	4.337453	...	-0.10093	4.203384

Results:	P_1	P_2	P_3	P_4	P_5	...	P_n	Objective
	-0.66208	14.99926	19.95	19.28681	-4.99965	...	48.00345	4.209018

Figure 3.9: A portion of the end of the optimization output file. Labels have been added for clarity.

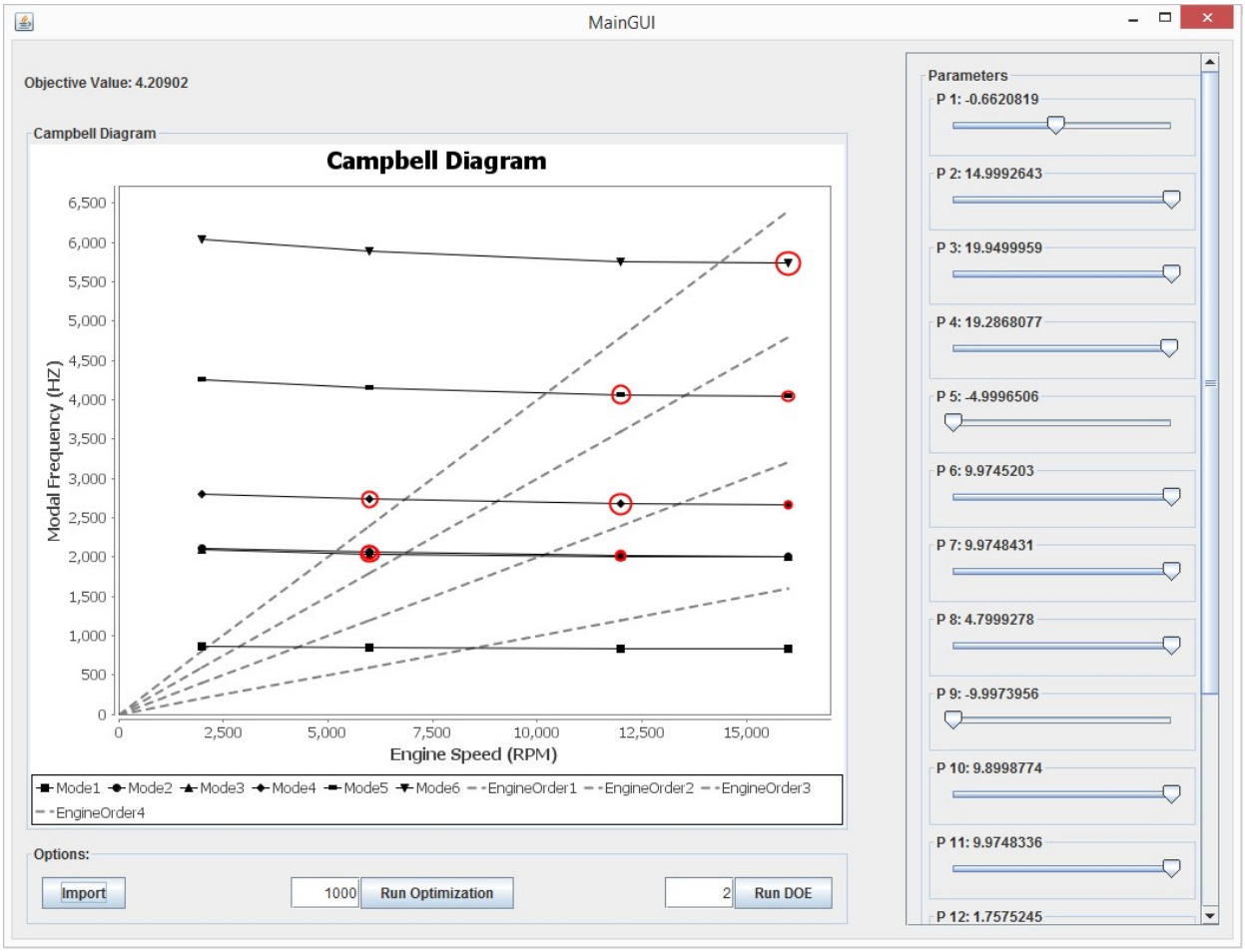

Figure 3.10: The GUI after the tuning process.

CHAPTER 4. RESULTS

Two sets of data were used to validate this tuning method: the first data set (Data Set 1) consisted of a reduced-order design space compared to the second data set (Data Set 2). To clarify differences between the two sets, Data Set 1 was used to prove the concept for the optimization, whereas Data Set 2 was used to prove that the method worked for tuning real airfoils. The exact parameters will not be specified for these data sets because they are not necessary for evaluating the results of this method—the optimization algorithms do not need to know the physical meaning of each parameter. They only require the number of design parameters and the range of values for each of those parameters. For example, labeling a parameter "chord" in the optimization is as useful as naming it "P." In this thesis, all parameters will be referred to as "P" with a corresponding digit (ex: P1). Again, the mapping between the geometry and the natural frequencies is what is important, not the name of each parameter.

For the same reason, the exact modes (1 Bending, 1 Torsion, etc), engine orders, and operating speeds will not be provided for the data sets used in this research. In fact, the objective function is calculated using ratios of the natural frequencies and engine orders so knowing the names of the modes and engine orders is superfluous. Therefore, in this thesis, only the *relationships* between the blade parameters, the modal frequencies, the engine orders, and the operating speeds matter because the algorithms do not use any other information. Of course, once the algorithm has converged, the parameter values can easily be interpreted by the designer.

Validation of the method for each set of data consisted of the following steps:

1. Evaluate the accuracy of the surrogate models
2. Complete a series of optimizations using two gradient-free algorithms
3. Compare results of each optimization run
4. Examine results of the optimizations to determine if the optimal design is feasible

5. Complete a full-factorial design of experiments (DOE) to explore the design space to validate the solutions found by the optimizations

6. Use the data from the DOE to perform a sensitivity analysis on each of the blade parameters

4.1 Data Set 1

The reduced-order set of data used 15 parameters to control blade geometry and had six modes with frequencies at four operating speeds. Therefore, there were 15 input parameters controlling 24 output points on the Campbell diagram. Four engine orders (see Section 2.1) were used for the Campbell diagram.

4.1.1 Accuracy of Surrogate Models

A surrogate model was generated for each of the 24 points using 384 sets of training data—the total amount of training data created and provided by the research sponsor (see Section 3.2). There is no special significance to the number "384"—it is simply the number of runs completed by the Optimal Latin Hypercube DOE. However, in general, an increased amount of training data yields more accurate surrogate models (see Section 2.2).

The 24 surrogates were independently analyzed using Isight's native Cross-Validation method, which consists of randomly removing data points from the training data, calculating new approximations using the reduced training data sets, and comparing the original approximation to the new approximations (see Section 3.3).

Using Cross-Validation analysis, the average mean (MEAN) error, the average root mean square (RMS) error, and average maximum (MAX) error for each surrogate model was calculated. The results are summarized in Table 4.1. With 15 design variables and only 384 training data points, the calculated MEAN, RMS, and MAX errors for the simplified data are extremely small, with the MAX error (of the 24 models) being around 4.74%. Saravanamuttoo et al. state, "the natural frequencies must be correct within a few percent to be useful" [12]. Based on this criteria, the approximations were used to create the design space for optimization.

Table 4.1: A table enumerating the MEAN, RMS, and MAX errors for the surrogate models used in each data set

Data Set	Training Data	Inputs	Outputs	MEAN	RMS	MAX
1	384	15	24	1.74%	2.25%	4.74%
2	365-1247	21	52	6.13%	7.93%	17.03%

4.1.2 Optimization Runs

Because Data Set 1 was only used to prove the concept for the proposed method, only five optimization runs were completed with each gradient-free algorithm. For each optimization, the algorithm was permitted 10,000 iterations to converge to the optimum value of the design space. NSGA-II and OMOPSO completed each optimization within two minutes. Due to the stochastic nature of gradient-free algorithms, the final values for the objective function did not match exactly, even for different runs of the same algorithm. However, with a complicated design space and only 10,000 iterations allowed, this is not surprising because the algorithms' "random" exploration of the design space does not always enable them to explore the space completely. The results of the optimizations are presented in Table 4.2 where it is seen that NSGA-II and OMOPSO found similar objective values.

Table 4.2: This table compares the five objective values found by the two algorithms in five different optimization runs

Algorithm	z_{max}	$z_{initial}$	Objective Value				
			Run 1	Run 2	Run 3	Run 4	Run 5
NSGA-II	4.8	3.5128	4.1511	4.2090	4.2057	4.1543	4.1170
OMOPSO	4.8	3.5128	4.1591	4.2091	4.2091	4.1235	4.1230

For perspective on the values in Table 4.2, Equation 3.7 defines the theoretical maximum objective value for this design space: $z_{max} = (24)(0.2) = 4.8$. However, as stated in Section 3.5.2, it is not always possible to move every point so that $\eta \geq \kappa$. Therefore, z_{max} is a ceiling value, used only for comparison. $z_{initial}$ is the objective value when each parameter value is set to its baseline

value. Figure 3.3 is the Campbell diagram for these baseline parameter values, and it has been duplicated in Figure 4.1 for the reader's convenience.

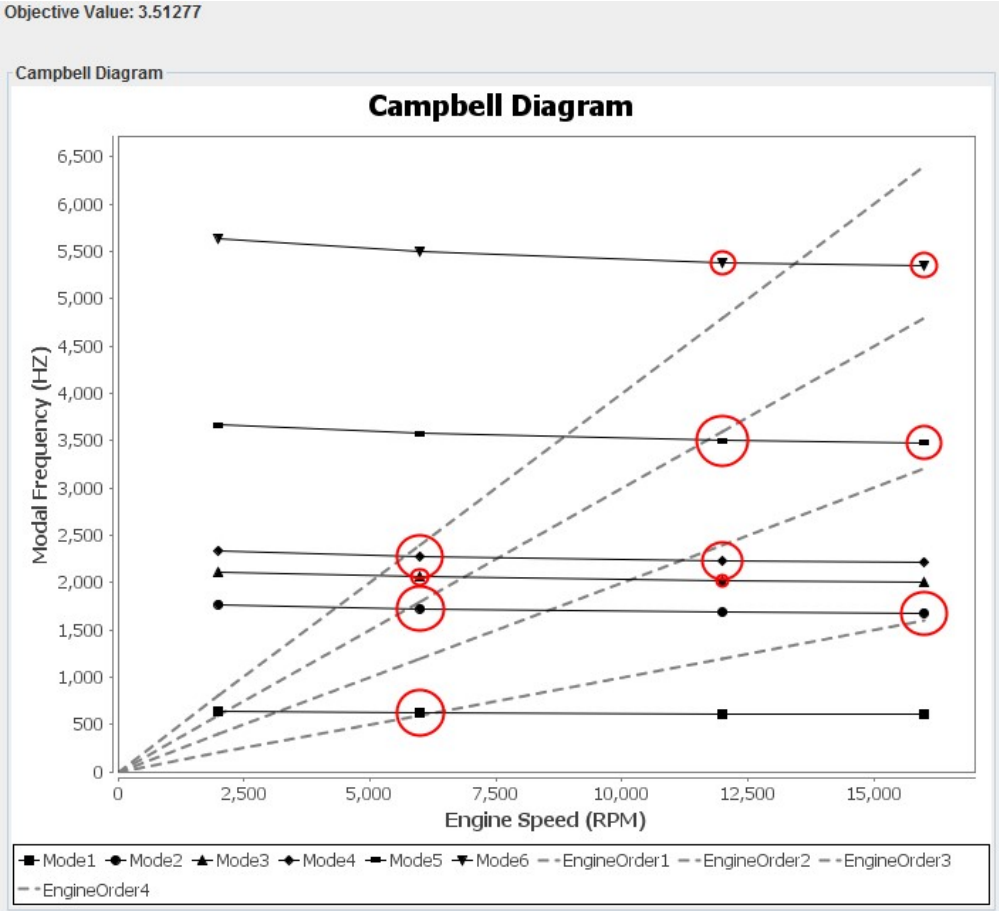

Figure 4.1: The Campbell diagram for Data Set 1 with baseline values, where $z_{initial} = 3.5128$.

Table 4.2 indicates that NSGA-II and OMOPSO found several groups of optimum values: the runs had objective values near 4.12, 4.15, and 4.20. Beyond the second decimal place, the fluctuations in each objective value can be attributed to the convergence criteria of the algorithms. For example, $4.2090 \approx 4.2057 \approx 4.2091$. This was verified by comparing the Campbell diagrams proposed by each run within each of the three objective groups. Even though 4.12, 4.15, and 4.20 are numerically similar, their proposed parameter values generate completely different Campbell diagrams.

The three categories of optima show that there are local optima in the design space that are sufficiently robust to prevent the gradient-free algorithms from consistently finding the global

optimum of the design space. Based on these objective values, it was necessary to explore the validity of these optimization runs.

Consider the objective values near 4.20 in Table 4.2—NSGA-II Runs 2 & 3 and OMOPSO Runs 2 & 3. When plotted, the parameter values found by each algorithm are nearly identical, which can be seen in Figure 4.2. The blade parameters (P1-P15) are listed along the horizontal axis; the normalized values of the parameters are plotted vertically, above the corresponding variable. Lines linking the parameter values have been added for clarity in showing the parameters provided by the same optimization run. For the four optimizations, the parameter values for P2-P3, P5-P8, P9, and P11 varied by less than 1%. The values for P4, P10, P12, and P14-P15 had less than 10% variation, while the values for P1 and P13 varied by less than 20%. Therefore, it can be concluded that for NSGA-II Runs 2 & 3 and OMOPSO Runs 2 & 3 found the same location in design space.

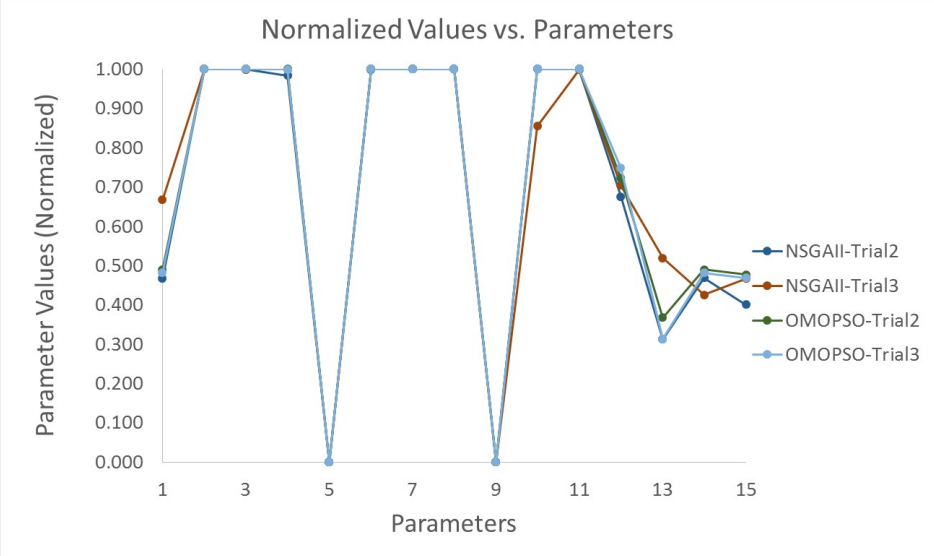

Figure 4.2: This figure illustrates that for objective values over 4.20, the algorithms identified the same location of the design space and that many of the parameters polarized to their minimum or maximum values.

These variations in parameter values are minimal, though, which was seen when comparing the Campbell diagrams associated with each of these optimizations—the Campbell diagrams were indistinguishable from each other. The Campbell diagram for NSGA-II Run 2 is included in Figure 4.3 for reference.

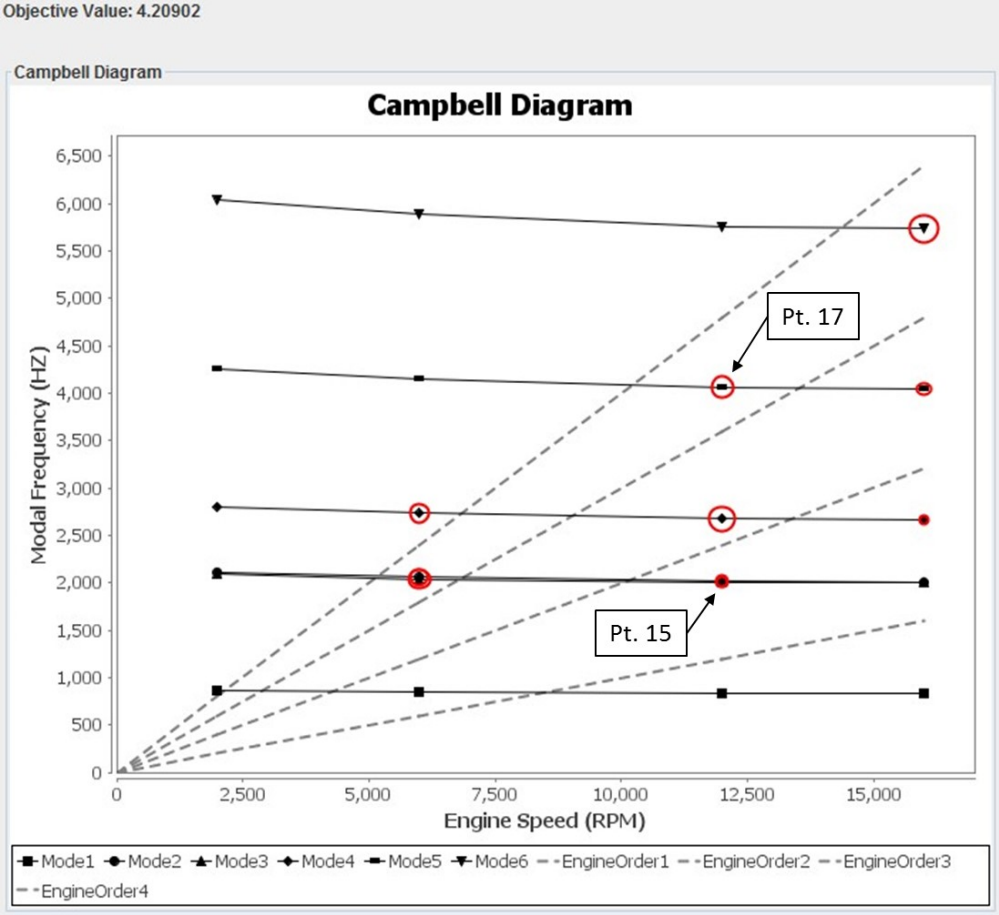

Figure 4.3: The Campbell diagram for NSGA-II Run 2 after the optimization completed. Compare with the original diagram in Figure 4.1.

A similar analysis was performed for the optimization runs with objective values near 4.15—NSGA-II Runs 1 & 4 and OMOPSO Run 1—to determine if these runs converged to one location in the design space or multiple locations in the design space with similar objective values. Many of the parameter values proposed by these runs were not similar, and they are shown in Figure 4.4. With the exception of P1, each of the parameter values varied by more than 10% of their ranges. In fact, P2, P4, and P6 had 100% variation within their ranges, and P5, P8, P10-P11, and P14 varied by more than 30% within their ranges. Thus, it can be concluded that, for these runs, the algorithms found different locations in the design space with similar objective values. This means the algorithms found several local optima.

Similar conclusions were drawn from an analysis performed on the runs with objective values near 4.12. To illustrate some of the differences between the different optima proposed by

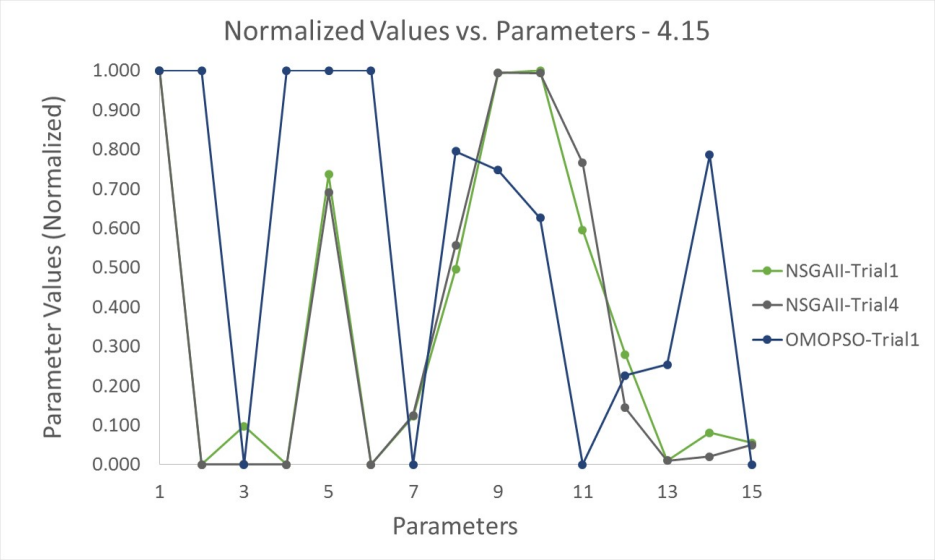

Figure 4.4: This figure illustrates that for objective values equal to 4.15, the algorithms identified different locations of the design space.

the algorithms, the Campbell diagram for NSGA-II Run 5 ($z = 4.12$) is shown in Figure 4.5. In comparison to Figure 4.3, the frequency of Modes 1, 2, 4, and 5 decreased by approximately 500 Hz whereas Modes 3 and 6 remain unchanged. With the significant reduction in frequencies, there are still several problematic points, especially the one from Mode 5 at the third operating speed (Point 17). Resonance is possible for Point 17 in Figure 4.3, but the distance between these points and the closest engine orders has been maximized properly—it is directly between Engine Orders 3 and 4. In contrast, Point 17 from Figure 4.5 is located on the nearest engine order.

4.1.3 Feasibility of Optimum

Based on the results in Table 4.2, every design was feasible, but it appeared that for the highest objective values ($z > 4.20$), the optimization algorithms were driving to the limits of each design parameter. This can be seen in Figure 4.2 where P2-P4, P6-P8, P10, and P11 consistently reached their maximum values while P5 and P9 reached their minimum values.

The purpose of using Data Set 1 was to prove that this method could be used to automate the blade tuning process. The charts in Figure 4.6 prove that the optimizations worked: as the number of iterations increased, the objective function continued to increase and approach the final, optimum value. Note the initially large (and subsequently smaller) distribution in the objective

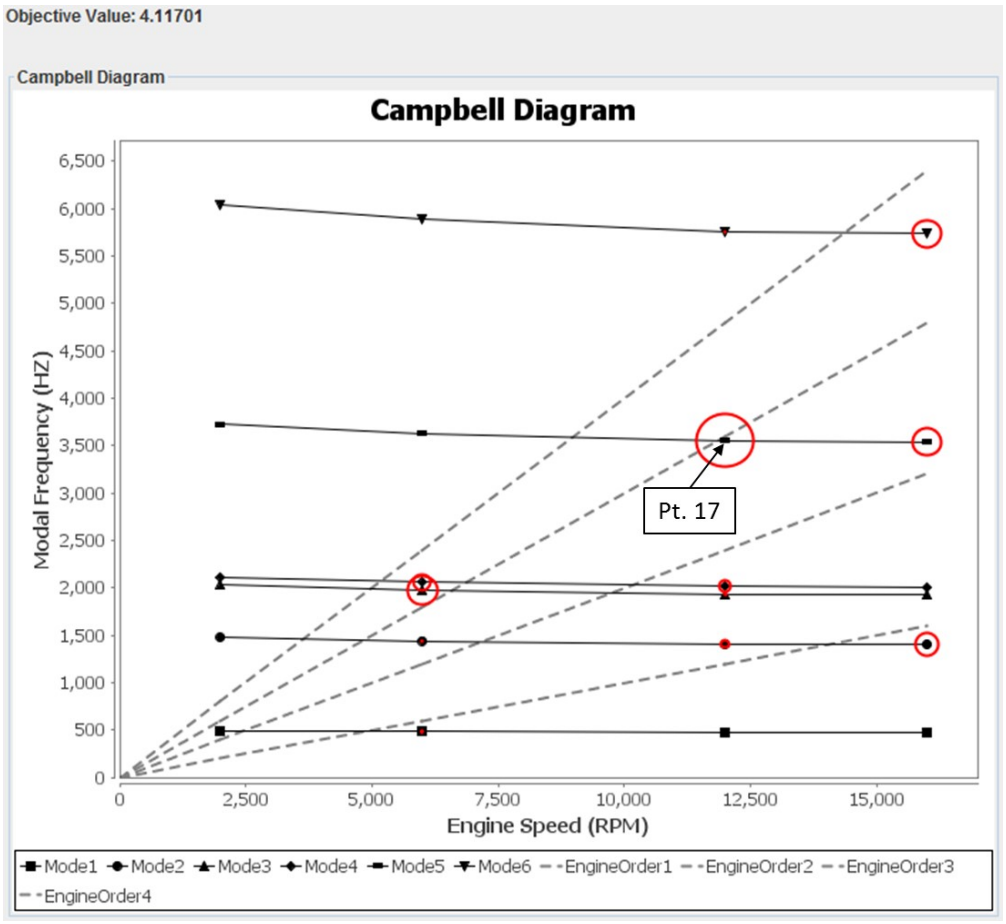

Figure 4.5: The Campbell diagram for NSGA-II Run 5 after the optimization completed. Compare with the original diagram in Figure 4.1 and the Campbell diagram for NSGA-II Run 2 in Figure 4.3.

values for NSGA-II and OMOPSO. As the iterations progress, the optimizations converge on a single design, but they continue to explore other areas of the design space.

The Campbell diagram in Figure 4.3 illustrates the concept that the optimization process worked. When compared to Figure 4.1, one will immediately notice the significant reduction in circles for each diagram. These circles were added around points on the Campbell diagram to easily visualize locations of potential resonance—the larger the circle, the more problematic the point. Note that the number of problem points has been reduced from 11 to 10, and that the radii of the circles have been significantly reduced (see Figure 4.3). The radii are calculated according to Equation 3.8. Suppose resonance may occur at Point *i*. As Point *i*'s contribution to the objective

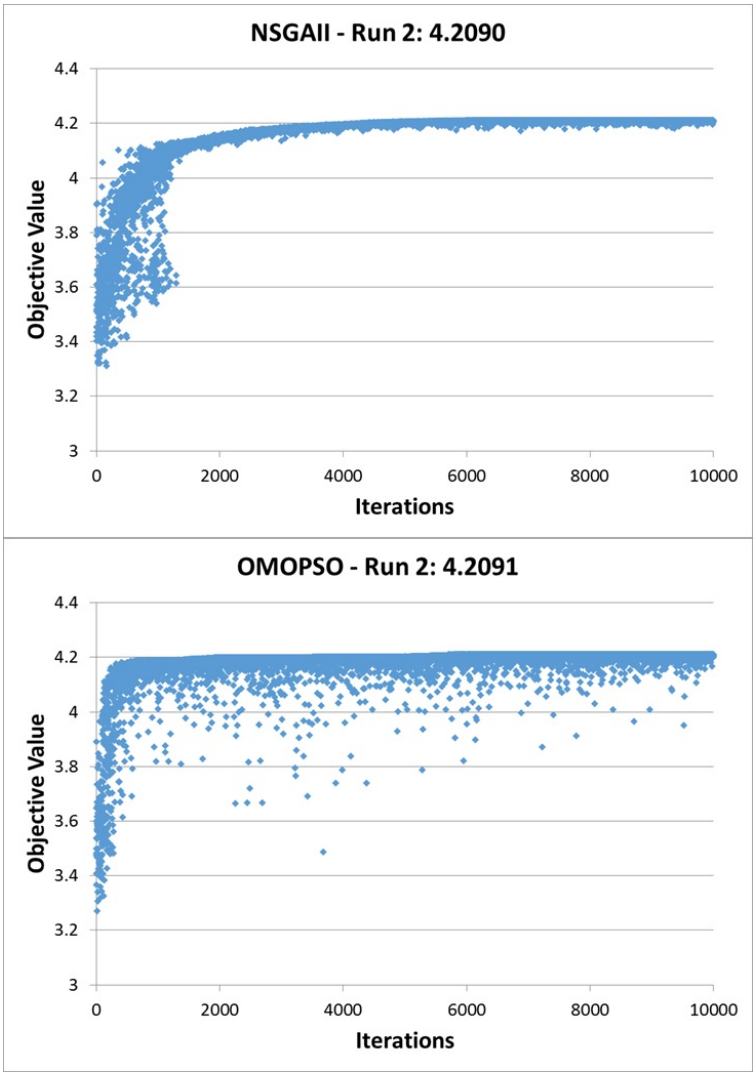

Figure 4.6: The objective function continually improves with the number of iterations in the optimization.

increases—meaning η_i increases—the radius (R_i) of the circle concentric with Point i decreases until $R_i = 0$, which occurs when $\eta_i \geq \kappa$.

Table 4.3 enumerates the contribution of each point on the Campbell diagram to the objective function. It is interesting to note that although many of the η values (η_7, η_{18}, and η_{20}) improved as expected, some of the contributions (like η_9 and η_{22}) actually decreased in value. This decrease occurred because the optimization accounts for all of the points simultaneously: if a decrease in one η yields significant improvement in several other ηs, then the optimization accepts the improvement. In fact, $\eta_{22} \geq \kappa$ initially, but its optimized value decreased its value so that

$\eta_{22} < \kappa$. Despite this reduced contribution to the objective function, the overall improvement of the optimized design is better.

Table 4.3: This table lists the initial and final values of the objective contributions for Data Set 1.

Point	Initial Contribution	Optimized Contribution	Improvement
1	0.2	0.2	0.0%
2	0.2	0.2	0.0%
3	0.2	0.2	0.0%
4	0.2	0.2	0.0%
5	0.2	0.2	0.0%
6	0.2	0.2	0.0%
7	0.042795	0.2	367.3%
8	0.042521	0.142763	235.7%
9	0.142816	0.131842	-7.7%
10	0.050652	0.137635	171.7%
11	0.2	0.2	0.0%
12	0.2	0.2	0.0%
13	0.2	0.2	0.0%
14	0.2	0.161375	-19.3%
15	0.161266	0.163096	1.1%
16	0.071086	0.113150	59.2%
17	0.028987	0.128620	343.7%
18	0.120586	0.2	65.9%
19	0.2	0.2	0.0%
20	0.049163	0.2	306.8%
21	0.2	0.2	0.0%
22	0.2	0.169029	-15.5%
23	0.087496	0.157335	79.8%
24	0.115400	0.104177	-9.7%
Objective	3.5128	4.2090	19.8%

The objective function quantifies this collective improvement, indicating that the value has increased from 3.5128 to 4.2090, an improvement of 19.8%. However, for 13 of the 24 points, $\eta_i \geq \kappa$ initially. With this offset $((13)(0.2) = 2.6)$ removed, the optimization improved the design by 76.2%. This illustrates that the blade has been tuned and resonance has been minimized.

4.1.4 Full-Factorial DOE

The differences in the optima indicated that the design space was complicated. To provide further evidence that the algorithms had found the global optimum at $z = 4.20$, a full-factorial DOE was used to explore the design space thoroughly. Based on the parameter values from the optimization runs, many had tended to move towards their limits. Therefore, the full-factorial DOE was run using only minimum and maximum values for each parameter—this will be referred to as a "2-level" DOE because each parameter was tested at two values. With 15 parameters being tested independently, this resulted in about 33,000 (2^{15}) evaluations. Using a multi-threaded full-factorial DOE developed for this validation testing, the DOE completed in less than one minute. However, the DOE's maximum objective function value was far below the maximum objective values found by NSGAII and OMOPSO. From this, it was determined that the 2-level DOE did not have enough resolution to completely explore the design space.

Using a 3-level DOE (minimum, middle, and maximum values), there were over 14.3 million (3^{15}) evaluations which took approximately 9 hours to complete. The best evaluations—the evaluations with the highest objective values—were compared to the results from the optimizations. No objective values exceeded 4.21. Only thirteen evaluations from the DOE had objective values between 4.20 and 4.21, while thousands of iterations had values near 4.12 and 4.15. The 13 evaluations had roughly the same parameter values, meaning they were all located in a similar area of the design space. Though the 3-level DOE does not guarantee the location of the global optimum, it provides evidence that the algorithms may have found the global optimum. For more conclusive evidence, a higher-level DOE was necessary. However, running a 4-level DOE was infeasible due to time constraints—it would take approximately 28 days to complete the 4-level DOE.

4.1.5 Sensitivity Analysis

Understanding the impact of each blade parameter on the modal frequencies of the blade is important, especially for manual tuning, because it provides the designer insight into which parameters should be adjusted to get the most change from the smallest amount of variation. Using the data from the 3-level DOE, a sensitivity analysis was performed to determine which blade

parameters were most influential on the objective. This analysis was performed using the following steps:

1. Insert all of the 3-level DOE evaluations (over 14 million records) into a database.
2. Select a simple random sample of 10,000 records for each parameter at each value (minimum, middle, and maximum value).
3. Statistically analyze the 45 data sets (15 parameters at 3 different values) to determine which parameters impacted the objective function—and therefore, the natural frequencies of the blade—the most.

As an example, Figure 4.7 shows three box plots representing the distributions of the objective values when holding P2 at its minimum (left), middle (center), and maximum (right) values. A box plot, or box-and-whisker plot, is a simple way of viewing a large amount of data. First, the data is ordered from least to greatest value. The "box" contains the middle 50% (25%-75%) of the data and the "whiskers" contain the other 50% (0%-25% and 75%-100%). The line in the center of the box indicates the the median value of the data. The data points outside the whiskers indicate statistical outliers.

From Figure 4.7, note that when P2 is held at its maximum value, the largest objective value ($z = 4.18$) is greater than the largest objective values for P2 at its minimum value ($z = 4.12$) and middle value ($z = 4.11$). Therefore, to achieve the greatest objective value, P2 should be held at its maximum value. One will see from Figure 4.2 that this is true for all of the optimizations where $z > 4.20$.

In contrast, the box plots for P5 held at its three values are shown in Figure 4.8. P5 offers the greatest objective value when it is held at its minimum value, even though P5-max offers the greatest value within the whiskers of the box plot. Typically, the outliers shown on a box plot are removed in order to find the true mean of the data; however, for this analysis, the true mean of the objective is not important. The focus of these plots should remain on the range of the objective value, specifically the greatest objective value offered by each parameter, including the outliers. Therefore, to reach the highest objective value, P5 should be held at its minimum value. This can be confirmed in Figure 4.2, where P5 is held at its minimum value for all four optimizations.

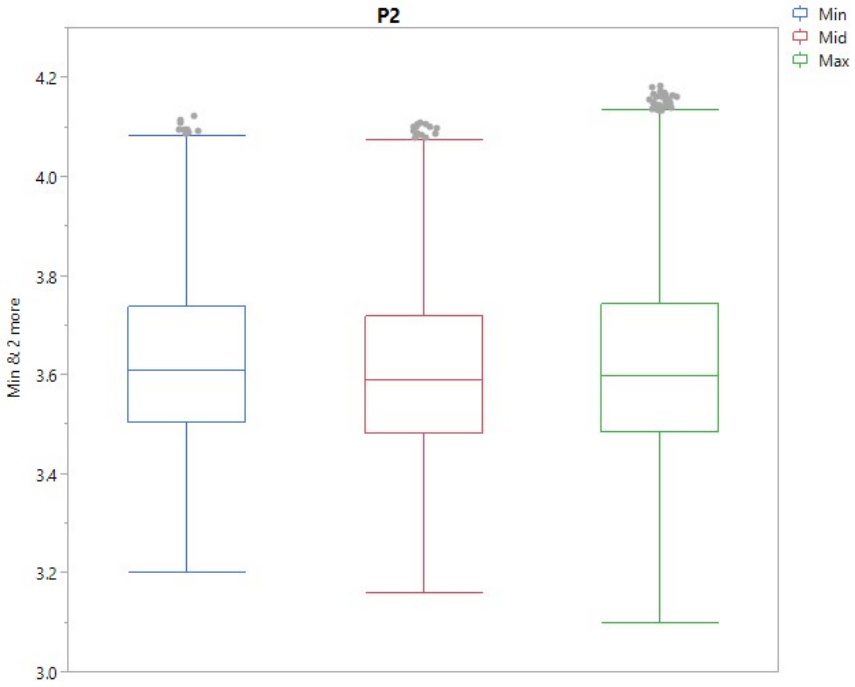

Figure 4.7: Three box plots representing distributions of the objective values while holding P2 at its minimum (left), middle (center), and maximum (right) values.

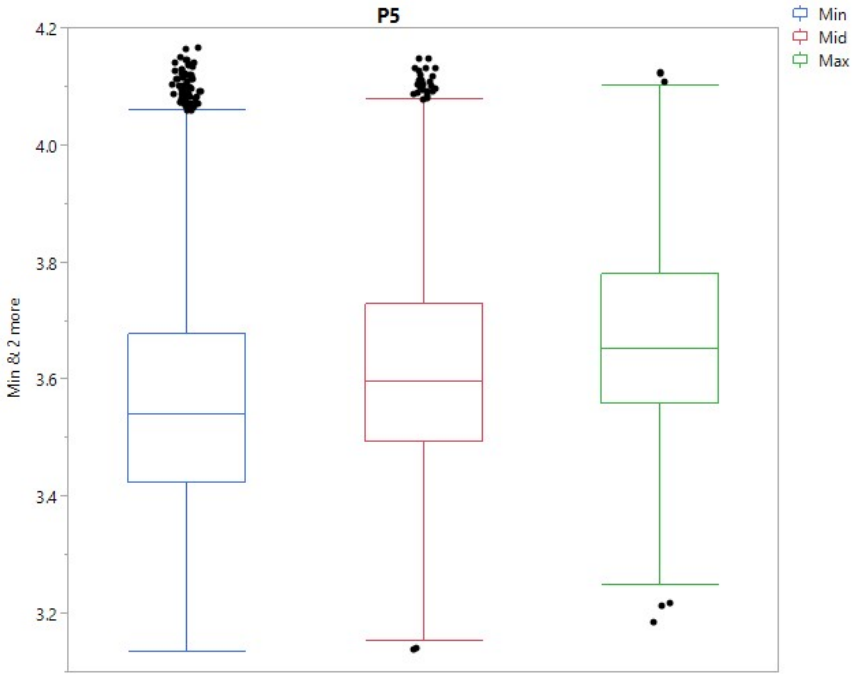

Figure 4.8: The three box plots for the distributions of the objective values for P5 at its minimum (left), middle (center), and maximum (right) values.

Sensitivity analyses were carried out for all 15 parameters, confirming the results from the optimizations with the highest objective values. These analyses even explained why some of the parameters (P12-P15) did not have consistent values for those optimizations: the highest object values proposed by the parameters being held at their three values were nearly identical. Therefore, it can be concluded that these parameters had very little effect on the objective function, explaining why their values were inconsistent with each optimization. For clarity, P13's box plots are presented in Figure 4.9 where the maximum objective values for all three values are nearly identical, unlike the distributions in Figures 4.7 and 4.8.

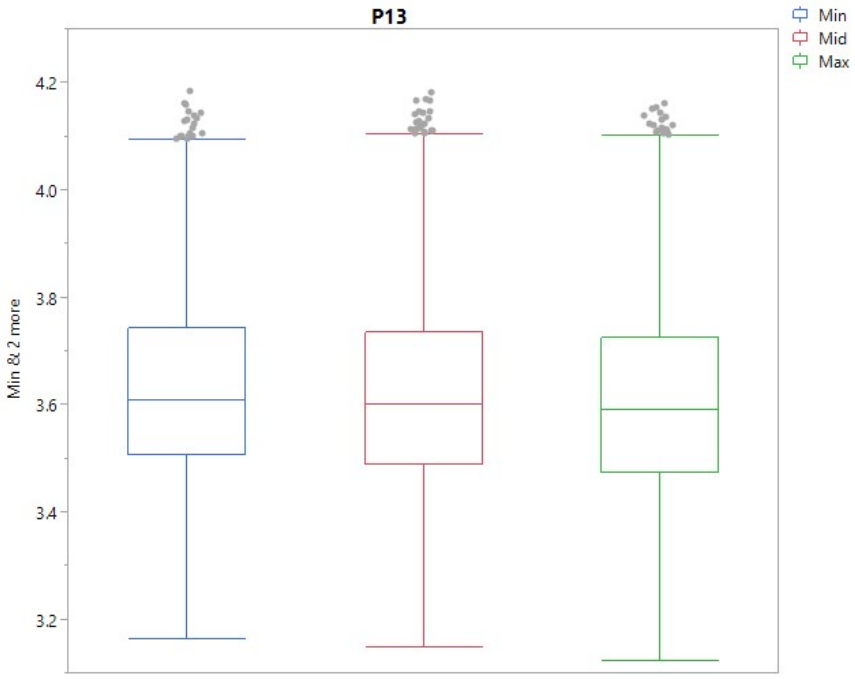

Figure 4.9: The three box plots for the distributions of the objective values for P13 at its minimum (left), middle (center), and maximum (right) values. Note that the maximum objective values are nearly identical for all three distributions.

4.2 Data Set 2

The second data set was used to test the validity of the automated tuning method. This data consisted of 21 blade parameters and 13 modes at four different speeds. The data also contained nine engine orders. With more than twice as many modes and engine orders, and an additional six

parameters, this design space was significantly more complicated than the design space from Data Set 1. The analysis for Data Set 2 followed a similar approach to the analysis of the previous data set.

4.2.1 Accuracy of Surrogate Models

Originally, 1250 sets of training data were provided by the research sponsor, but many of the sets did not have output values for every set of inputs. This typically occurs at higher frequencies since modes often change shape or are difficult to identify with FEA. Therefore, the training data was processed so that only inputs with a corresponding output could be used to train each surrogate. For example, if only 365 out of the 1250 sets had output values for Point *i*, then the *i*th surrogate model was trained with 365 sets of data; likewise, if 845 out of 1250 sets had output values for Point *j*, then the *j*th surrogate model was trained with 845 sets of data.

The 52 surrogate models (13 modes at four operating speeds) were initialized using between 365 and 1247 sets of training data. These surrogates were analyzed using Isight's Cross-Validation method mentioned in Section 4.1.1, and the results of the analysis have been summarized in Table 4.1. The increase in error can be attributed to less training data for several of the surrogates and added complexity due to the increased number of blade parameters (from 15 to 21). Though the accuracy of the surrogates for Data Set 2 is lower than the accuracy of Data Set 1, the surrogates were still used because they were the best that could be generated in Isight from the training data. The purpose of this research is to test the automated method of tuning, not the creation of surrogates within Isight. For higher fidelity models, additional training data could be used to initialize the surrogates.

4.2.2 Optimization Runs

In order to validate this method using real blade data, 20 optimizations were performed on Data Set 2 using each optimization algorithm. The algorithms were permitted 10,000 iterations for convergence. Because of the additional complexity in the real data, the algorithms completed their iterations in 20 minutes, approximately 10 times longer than the optimizations in Data Set 1. A summary of the results is presented in Table 4.4, where one can see that the means of the two

algorithms are similar and have similar standard deviations. Though it may seem disconcerting that the algorithms do not find exactly the same point each time they optimize the design space, this is just a result of using gradient-free algorithms: the stochastic nature of the algorithms prevents them from converging like gradient-based algorithms [17].

Table 4.4: The means and standard deviations of each algorithm. The OMOPSO algorithm had one statistical outlier, which was removed for analysis.

Algorithm	z_{max}	$z_{initial}$	Runs	Mean	σ
NSGA-II	10.4	8.5739	20	9.0085	0.00286
OMOPSO (Original)	10.4	8.5739	20	9.0102	0.00502
OMOPSO (Modified)	10.4	8.5739	19	9.0116	0.00305

In the 20 OMOPSO optimizations, there was one statistical outlier with an objective value of 8.972 (Run 7). When it was removed from the data set, the standard deviations of the NSGA-II and OMOPSO algorithms were similar (Table 4.4, OMOPSO (Modified)). Upon investigation, Run 7's objective value predicted a significantly different location of the design space compared to the other 19 runs. This comparison was performed by analyzing the Campbell diagrams and parameter values predicted by Run 7 compared to each of the other runs. Without this outlier, the OMOPSO algorithm was able to consistently predict an optimum value in each of its runs.

One of the primary differences between the optimization of Data Set 2 and Data Set 1 is that the optima proposed by NSGA-II and OMOPSO for Data Set 2 had many values for the parameters between their minimum and maximum values, unlike Data Set 1 whose optimum parameters typically polarized to either the minimum or maximum values for each parameter. Figure 4.10 shows the normalized parameter values for several runs of the optimization algorithms. Some of the normalized parameter values have large spreads, but the objective value remains constant. This indicates that these blade parameters have little effect on modal frequencies of the blade. For example, compare P4 to P6: the optimized value for P4 remained constant, while the values for P6 varied by 90% of its range. This spread in values indicates that small changes in P4 greatly affected the value of the objective function, while large variations in P6 did not significantly change the objective value. This is true for many of the blade parameters above P11, as shown by the large

spread in their optimized values. More detail regarding these observations is provided in Section 4.2.5.

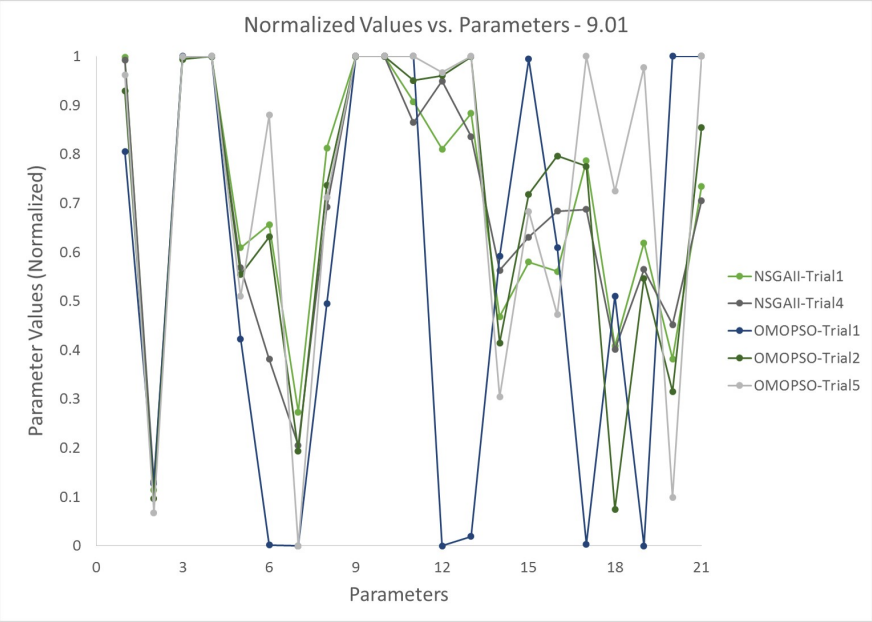

Figure 4.10: With approximately equal objective values, the algorithms identified similar locations of the design space. Many of the parameters did not polarize to their minimum or maximum values.

For this design space, the optima found by the two algorithms were similar in objective values, with mean values that were only 0.015% different. Though many of the optimized parameter values varied from run to run (see Figure 4.10), the Campbell diagrams associated with each optimization were nearly identical to each other. This allows the designer to immediately recognize that certain blade parameters are more useful for tuning than other parameters because they have a greater influence on the natural frequencies of the blade. Suppose a parameter defines the blending radius for where the blade is fastened to the rotor's hub, and another parameter defines the length of the chord. The "chord" parameter has a much greater influence on the modal frequencies of the blade than the blending radius parameter. Applied to this research, that means that variations in the chord parameter will impact the objective more significantly than variations in the blending radius.

In fact, with no prior knowledge of the real parameter names, upon completing these optimizations and examining the normalized parameter value plots, the parameters that were or were not influential on the objective could be easily identified. The research sponsor confirmed that

P6–P7 and P12–P21 were parameters that have little influence on the natural frequencies of the blade. Therefore, for the blade in this data set, the parameters that should be adjusted for tuning are P1–P5 and P8–P11. Once these have been set, the other parameters can be adjusted to make small improvements to the blade's tuning.

4.2.3 Feasibility of Optimum

By optimizing the design space generated from Data Set 2 and determining that the outputs are feasible and significantly better tuned than the original, candidate airfoil (based on the objective values and Campbell diagram), it can be concluded that this method works consistently and is an automated method for tuning a real compressor blade.

For a visual comparison, the Campbell diagrams in Figures 4.11 and 4.12 illustrate the difference between the initial blade and the final, optimized blade. The frequencies of modes 1 and 10-13, increased to eliminate possible resonance, reducing the number of problem points from 16 to 12, and eliminating 25% of the points of concern. Modes 5–9 also increased in frequency, but were not able to achieve frequencies where $\eta_i \geq \kappa$. The final objective values consistently improved the initial objective by 5.1% (see Table 4.4). Though this seems to be minimal improvement, remember that 36 of the 52 points on the Campbell diagram initially had values of $\eta > \kappa$. When this offset is subtracted from the objective (both final and initial) values, the optimized objective value shows a 28.1% improvement over the initial design. This illustrates that the optimization worked as anticipated and produced a better design given the original criteria.

Table 4.5 has been included to understand how the points on the Campbell diagram changed. For clarity, the 36 points that initially had values of $\eta > \kappa$ were omitted. Notice that many of the contributions of the points increased (Points 5–6, 19–20, 33–36, etc), moving them farther away from the nearest engine order. However, similar to Data Set 1, the contributions of Points 32 and 45–47 decreased, specifically that $\eta_{45} \geq \kappa$ initially but, after the optimization, η_{45} was reduced to value lower than κ. Therefore, it contributed less to the objective. As noted in Section 4.1.3, this phenomenon occurred because the optimization accounted for all of the points on the Campbell diagram simultaneously instead of trying to optimize each point in series.

Table 4.5: This table lists the initial and final values of the objective contributions for Data Set 2 that changed during the optimization.

Point	Initial Contribution	Optimized Contribution	Improvement
5	0.024269	0.068963	184.2%
6	0.068556	0.159800	133.1%
19	0.014065	0.067967	383.3%
20	0.115648	0.133476	15.4%
27	0.193937	0.2	3.1%
32	0.164661	0.093916	-43.0%
33	0.004772	0.015364	222.0%
34	0.035014	0.041137	17.5%
35	0.062421	0.162969	161.1%
36	0.199071	0.2	0.5%
40	0.134203	0.2	49.0%
45	0.2	0.139961	-30.0%
46	0.014665	0.003482	-76.3%
47	0.013062	0.010910	-16.5%
48	0.009289	0.117896	1169.2%
49	0.139023	0.2	43.9%
50	0.181246	0.2	10.3%
Total	1.5739	2.015838	28.1%

4.2.4 Full-Factorial DOE

To provide evidence that the global optimum was found, a full-factorial DOE was used to explore the design space comprehensively. For a 2-level DOE with 21 parameters, there were over 2 million evaluations (2^{21}), with a run-time of 12 hours. With the 2-level DOE, the highest value of the objective function was determined to be 8.959. Because this value was significantly lower than the values found by the optimization algorithms, it was necessary to re-run the DOE with greater resolution. However, the predicted run-time for 3-level DOE was 6.8 years, making it infeasible.

When reviewing the parameter values found by the algorithms, the values tended not to polarize to their extremes as they had done in the previous design space (see Section 4.1.3). This can be seen in Figure 4.10 where the normalized values for each parameter have been shown. Based on this observation, it is understandable why the 2-level DOE provided an objective value that was much lower than the values provided by the algorithms.

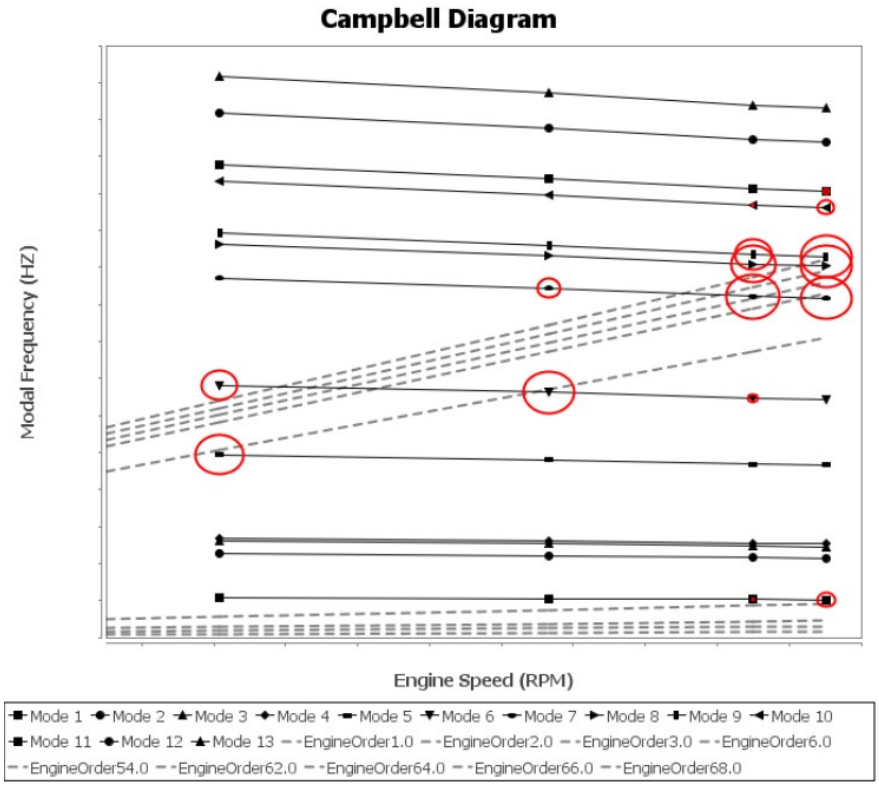

Figure 4.11: The initial Campbell diagram for Data Set 2. The circles highlight points that will cause resonance.

With only the 2-level DOE, it cannot be concluded that the global optimum was reached by the algorithms, but the analysis indicates that the algorithms consistently found the same optimum value in the design space despite their different parameter values. Therefore, the algorithms either found the global optimum or an extremely robust local optimum. Knowing the quality of algorithms and based on evidence from Data Set 1, it is likely that the algorithms found the global optimum since all of the runs found the same objective value and the Campbell diagrams were identical. Also, with 40 independent runs and with the stochastic exploration of each optimization, if a higher objective value existed, it is likely that the one of the optimizations would have discovered it. Because no alternative was discovered, $z = 9.01$ may be the global optimum for Data Set 2.

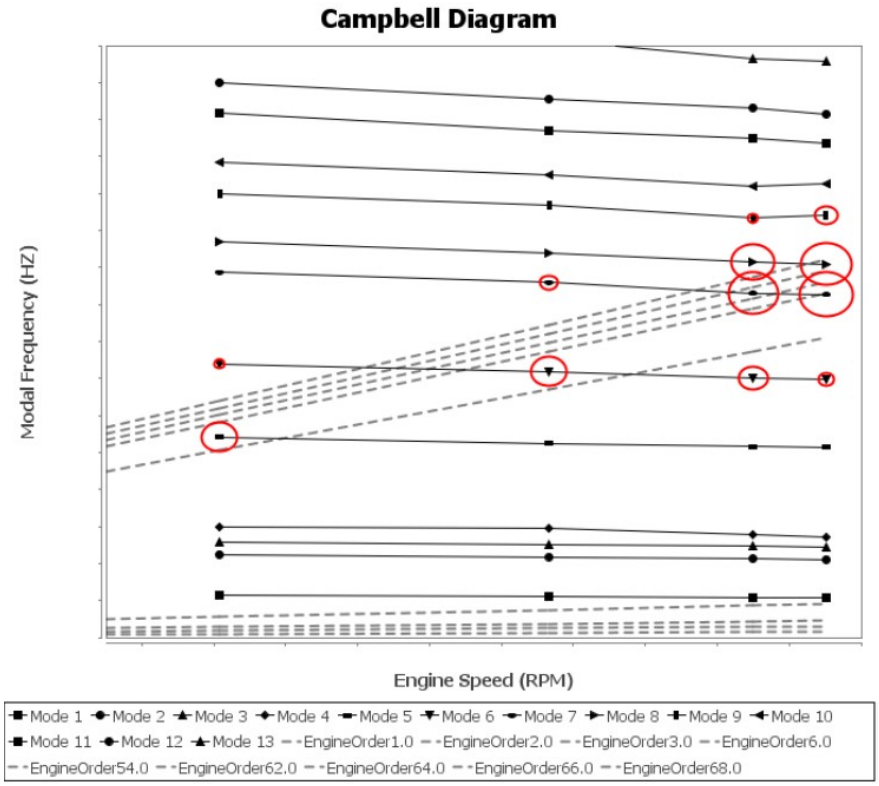

Figure 4.12: The final Campbell diagram for Data Set 2. The circles highlight points of concern.

4.2.5 Sensitivity Analysis

Because it was infeasible to run a 3-level full-factorial DOE, the data obtained from the 2-level DOE was used to perform a sensitivity analysis on each parameter. A similar process to the one mentioned in Section 4.1.5 was followed. Even though many of the parameters proposed by the optimizations tended toward values within their ranges (instead of minimizing or maximizing like the parameters in Data Set 1), the sensitivity analysis was useful for confirming the values found by the parameters that optimized to either their minimum or maximum values. For example, from Figure 4.10 one can easily determine that the proposed values for P4, P9, and P10 were all at their maximum values. The sensitivity analysis for P4 has been included in Figure 4.13 to confirm the value proposed by the optimizations, even though it only proves that the maximum value of P4 is better for the objective than the minimum value of P4. The sensitivity analysis cannot draw any other conclusions because no other values in P4's range were tested.

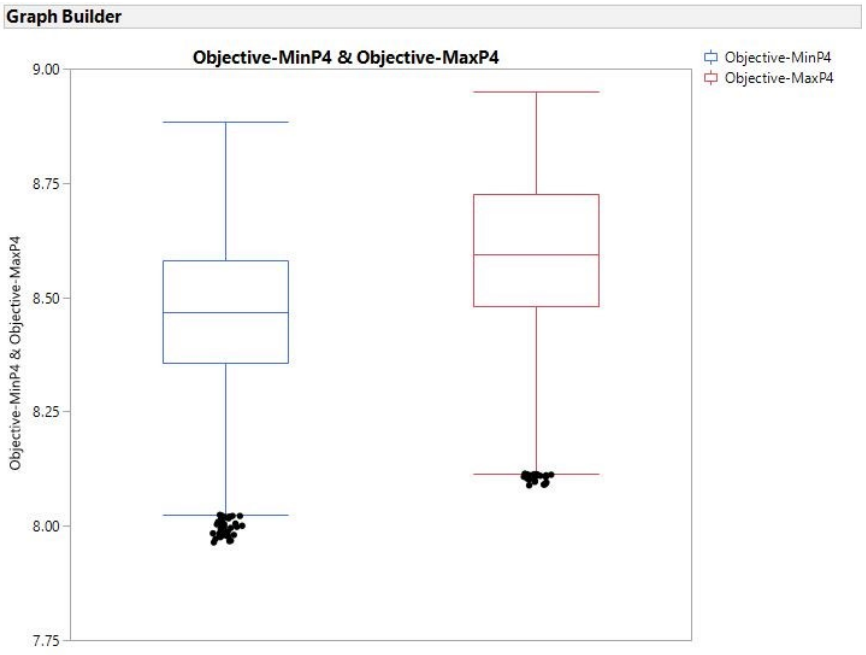

Figure 4.13: The two box plots for the distributions of the objective values for P4 at its minimum (left) and maximum (right) values.

Similar charts were constructed for all of the other parameters to confirm the polarized parameter values and to explain the behavior of many of the parameters above P12. Figures for these charts will not be included because they provide the same conclusions presented in Section 4.1.5. Specifically, the sensitivity analysis provided further confirmation of the optimization results, and that the parameters with the greatest influence on the objective should be adjusted (during manual tuning) before the parameters with very little influence on the objective.

4.3 Further Analysis

This section presents trends and additional information discovered during the exploration of the design spaces generated by the two data sets. It will focus primarily on different values of κ, a comparison of the optimization algorithms used, and insight into optimized results where $\eta_i < \kappa$.

4.3.1 Various Fitness Ratios

The optimizations performed on both data sets used $\kappa = 0.2$. It was desired to understand how varying κ would affect the results of the optimizations. Therefore, two alternate values of κ (0.1 and 0.4) were used to explore the effects of κ on Data Set 2.

First, κ was set to 0.1 and three optimizations were performed using OMOPSO. The objective values from each of the optimizations were nearly identical, but the parameter values varied widely, indicating that the optimization found different locations of the design space with the same objective values. The normalized parameter values are listed in Figure 4.14 to visualize the difference in parameter values for these three optimizations.

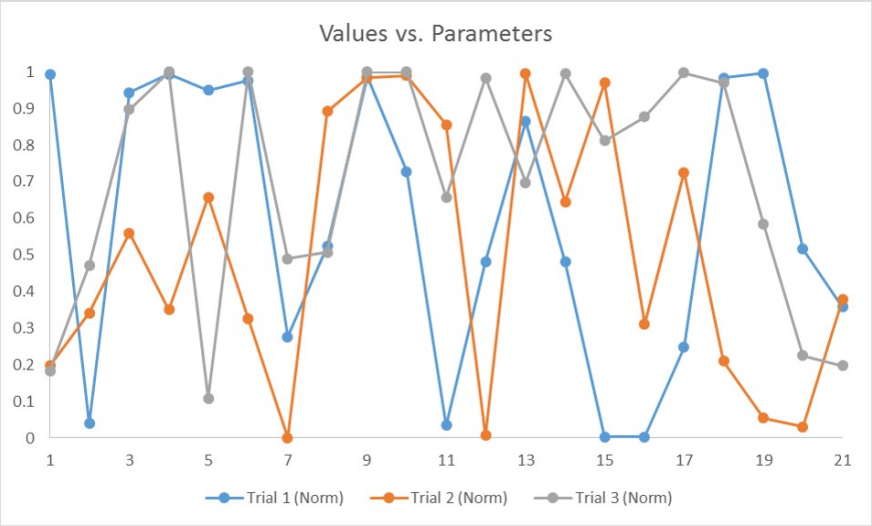

Figure 4.14: The normalized parameters for the three optimizations run on Data Set 2 with $\kappa = 0.1$

Even though the optimizations found different optima in the design space, it makes sense with a lower value of κ. The lower threshold allows for greater flexibility in the optimization because $\eta \geq \kappa$ for more points on the Campbell diagram. Furthermore, because the points have larger ranges to fluctuate before negatively affecting the objective function, the parameter values can be varied more to find additional optima. For example, suppose $\eta_i = 0.15$ for a certain point on the Campbell diagram. For $\kappa = 0.2$, Point i contributes η_i—a varying amount—to the objective; in comparison, for $\kappa = 0.1$, Point i contributes κ and continues to contribute κ to the objective as long as $\eta_i > \kappa$. This allows for much more flexibility in the parameter values when κ is set at a lower

value. This concept is seen in Figure 4.15, where there are nine problem points initially (top), whereas there are 16 in Figure 4.11, meaning that $\eta \geq \kappa$ for all but nine of the points. Therefore, as long as the other 43 points maintain their η values, only nine points contribute varying amounts to the objective function. As seen in Figure 4.15, the final Campbell diagram (bottom) could not completely eliminate the resonance from these points, but the nine points were moved to locations to give the least possible resonance. In fact, only six points remain problematic.

To further explore this concept, three optimizations were run on Data Set 2 with $\kappa = 0.4$ and are shown in Figure 4.16. Notice that the objective values and the parameter values are more similar than the parameter values in Figure 4.14. This is expected because a larger value of κ allows for less fluctuation in the parameter values to reach the same objective value. In other words, there are less local optima in this design space because of the higher threshold.

4.3.2 Comparison of Algorithms

For both data sets, the performance of NSGA-II and OMOPSO was remarkably similar. This is interesting because the two algorithms utilize completely different methods for optimization. As discussed in Section 2.4, NSGA-II is a genetic algorithm while OMOPSO is a particle swarm algorithm. The difference in the exploration techniques is visually illustrated in Figure 4.17. Note that OMOPSO continues to randomly explore well outside the general trend throughout the entire optimization; in contrast, NSGA-II hones in on a certain, optimum value after several hundred iterations and continues its convergence in that specific area. Even with their different techniques, the two algorithms converged to very similar optimum values for both design spaces.

Even though these two algorithms converged to the same value, it does not mean that all gradient-free algorithms are equally effective. A third algorithm was tested—a covariance matrix adaptation evolution strategy (CMA-ES) algorithm—to prove this point (see Section 2.4.3).

For Data Set 1, CMA-ES performed comparably to the other two algorithms. It took 20 seconds longer to complete each optimization, but the values it converged to were very similar to the objective values produced by NSGA-II and OMOPSO, including all three categories of optimum values (see Section 4.1.2). However, when optimizing Data Set 2's design space, CMA-ES provided vastly different objective values for each run. Figure 4.18 displays the average "rise time" for the three algorithms. NSGA-II and OMOPSO are similar, but CMA-ES converged to a

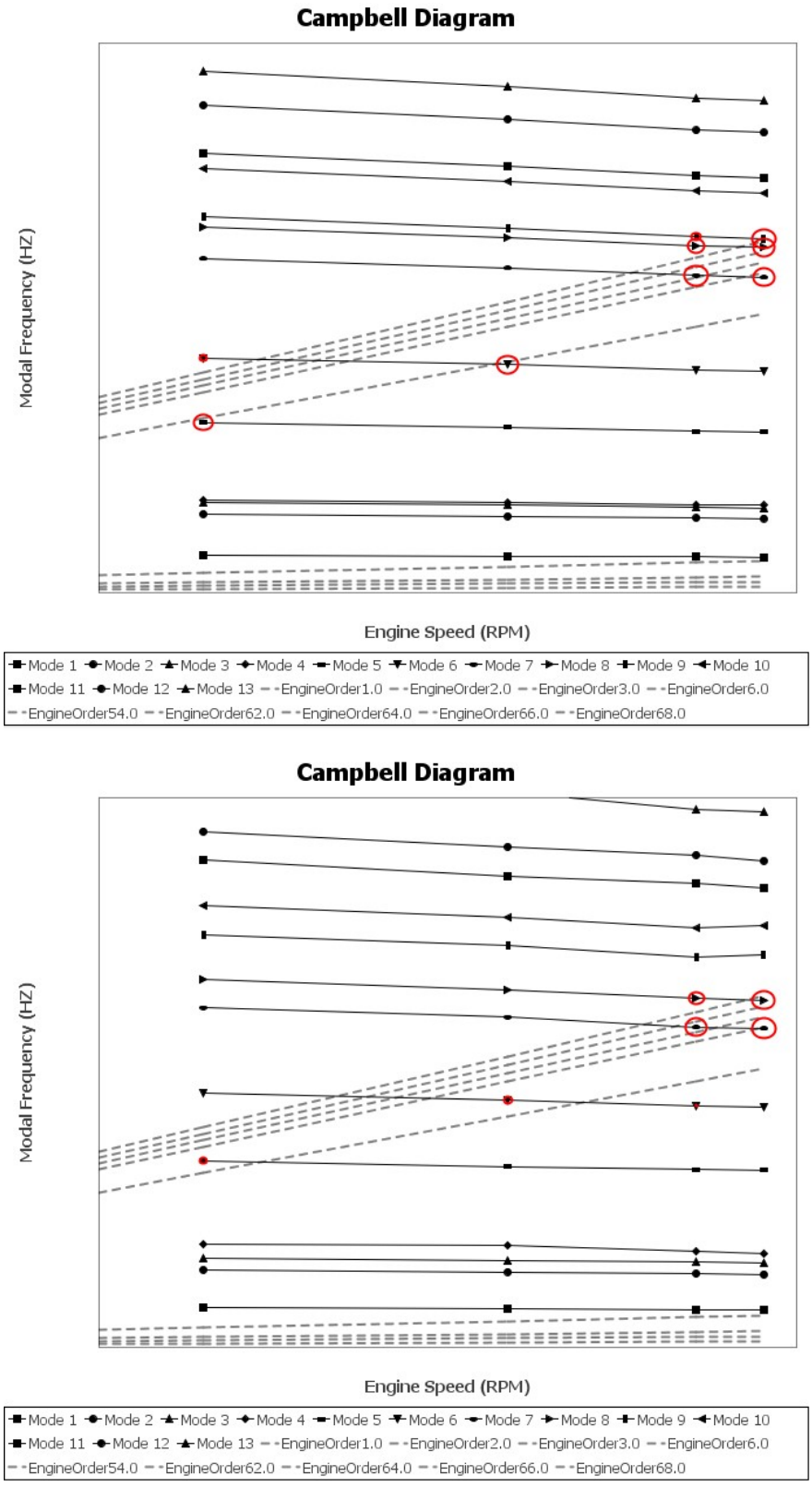

Figure 4.15: The initial (top) Campbell diagram and the final (bottom) Campbell diagram when $\kappa = 0.1$.

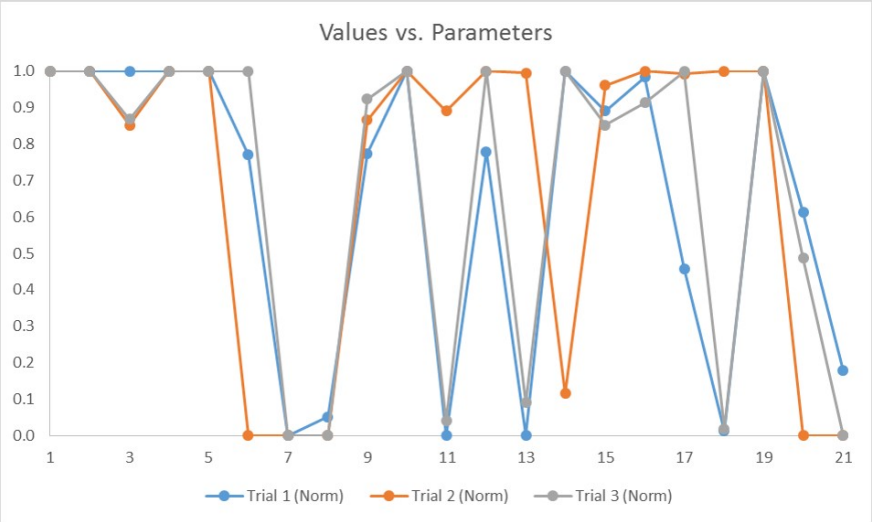

Figure 4.16: The normalized parameters for the three optimizations run on Data Set 2 with $\kappa = 0.4$. Compare with Figure 4.14 to see that these parameter values were much more similar.

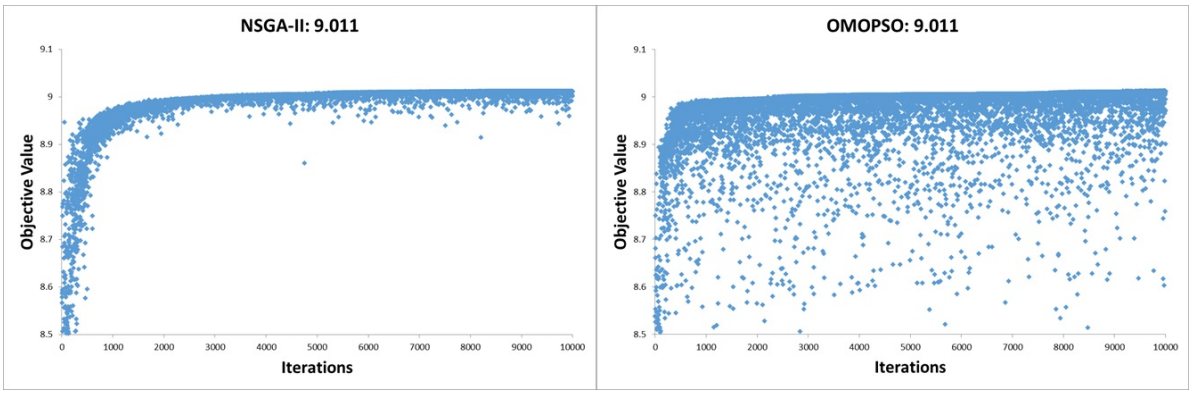

Figure 4.17: A visual comparison of the search techniques of NSGA-II (left) and OMOPSO (right).

significantly lower objective value. Furthermore, with 10,000 iterations, CMA-ES took 6 minutes longer to reach its "optimum" design.

The performance of CMA-ES may be attributed to a lack of tuning in the population size (λ) parameter of the algorithm. Hansen et al. suggest, "Tuning (that is increasing) the population size considerably improves the performance [of CMA-ES], compared to [its] performance with default population size" [21]. Furthermore, because CMA-ES was developed originally for robust local searches, it is not as effective as NSGA-II or OMOPSO at searching complete design spaces, though CMA-ES has proven to find global optima in several real world problems [21].

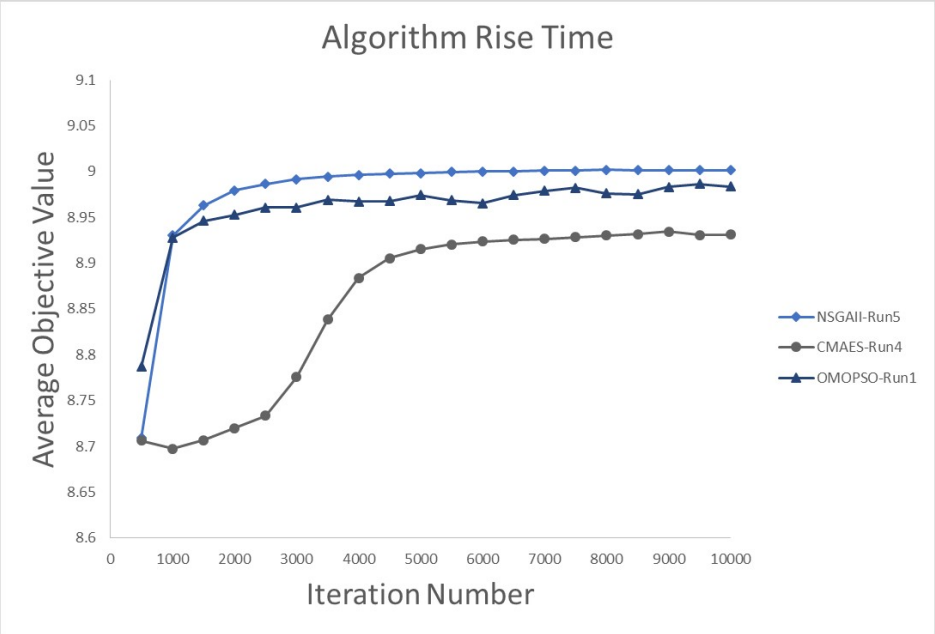

Figure 4.18: This figure illustrates the performance of NSGA-II, OMOPSO, and CMA-ES.

4.3.3 Unique Weighting

Notice in Figures 4.3 and 4.12 that although $\eta \geq \kappa$ for most of the points, there remained several points that remained too close to the nearest engine orders. These problematic points are indicated by the red circles centered on the points. These values of η remained smaller than κ because it is often impossible to manipulate a single point on the Campbell diagram (using only blade geometry) without affecting other points. In fact, from evidence obtained from the research sponsor, most points within a single mode either increase or decrease in frequency together. Also, some geometric parameters affect multiple modes. Therefore, even though not all of the points reach locations where $\eta \geq \kappa$, the optimization seeks the greatest good for the greatest number of points.

Weighting Single Points

Figure 4.19 illustrates how NSGA-II Run 2's optimization looked for, and tested, Point 15's value of η when it exceeded the threshold κ, but it determined that the best choice, when accounting for all of the points simultaneously, was for η_{15} to be less than κ (as seen in Figure 4.3).

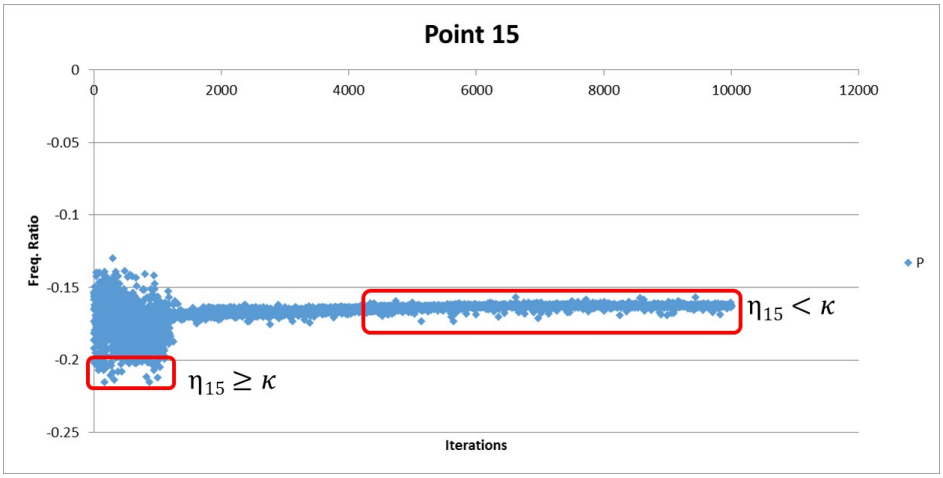

Figure 4.19: Notice that the algorithm attempted to find an optimum where $\eta_{15} \geq \kappa$. However, these solutions were rejected in favor of a better solution when factoring in all the points on the Campbell diagram.

Now suppose that the designer knew that Point 15 was more likely to cause resonance than all of the other points. Using the same objective function, the designer could add a unique weight to Point 15's contribution to the objective to prioritize η_{15} above all other ηs in the optimization (see Section 3.5.3). This forces the algorithm to make $\eta_{15} \geq \kappa$.

To show that this weighting functions properly, a relative weight of 200,000 was given to Point 15 so that it contributed 200,000 times more to the objective function than any other point on the Campbell diagram. When the blade parameters are set to their baseline value, the Campbell diagram looks identical to Figure 4.1, but now $z = 32256.5253$ because of the weight added to Point 15. Specifically note that $\eta_{15} < \kappa$ initially.

This weighted objective function was then optimized using NSGA-II and permitted 10,000 iterations to converge to an optimized design. This convergence is shown in Figure 4.21, where it is seen that the algorithm quickly (within 1,000 iterations) found an area of the design space where $\eta_{15} \geq \kappa$. From iteration 1,000 to 10,000 it continued to explore the design space, while only varying the value of η_{15} by ± 0.01.

The optimization converged to a solution that maintained $\eta_{15} \geq \kappa$ and is shown in Figure 4.22. Notice that $z = 40003.9455$, which makes sense because Point 15 contributes its weight times κ (0.2 for this case) to the objective, accounting for 40,000 units of the objective value $((200,000)(0.2) = 40,000)$ as shown in Section 3.5.3. Once Point 15's contribution is scaled down

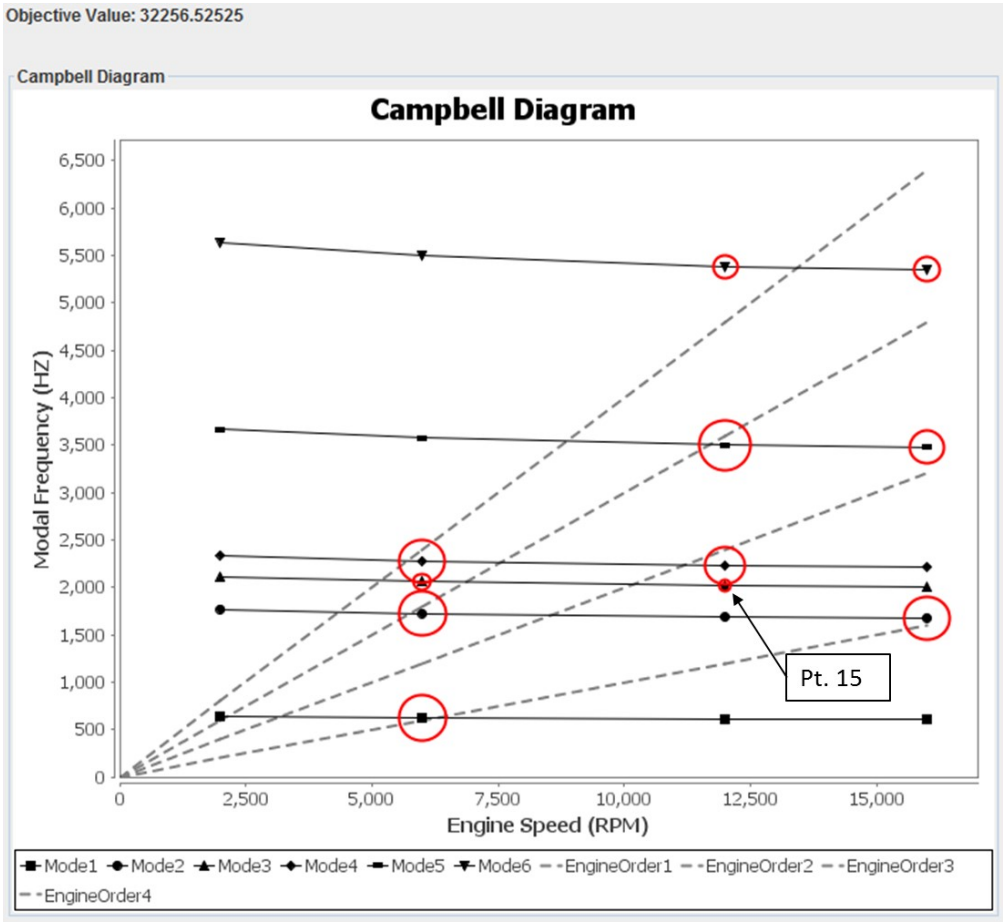

Figure 4.20: The initial Campbell diagram for a weighted optimization on Data Set 1 showing that $\eta_{15} < \kappa$ initially. Compare with the optimization in Figure 4.1 to see the diagrams are identical, but the objective values are different.

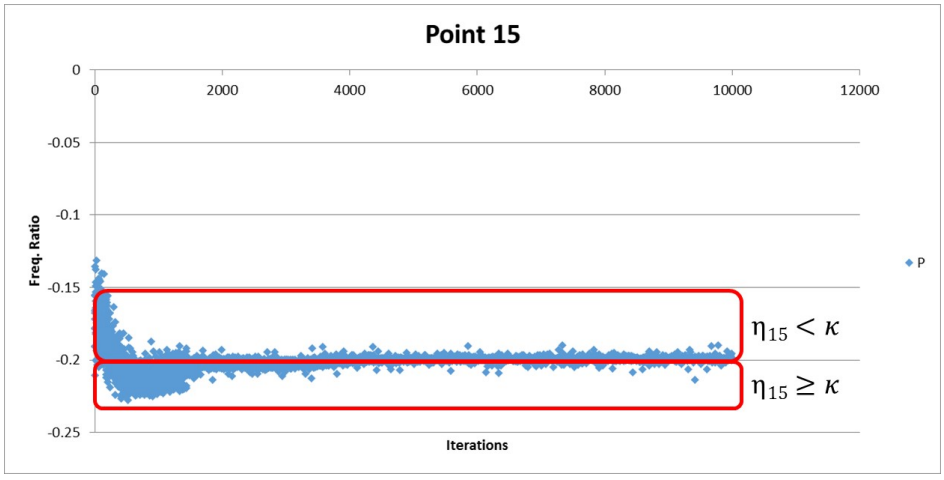

Figure 4.21: This figure shows how NSGA-II quickly (within 1,000 iterations) found a location in the design space where $\eta_{15} \geq \kappa$ and continued to explore while maintaining that value of η_{15}.

to 0.2 instead of $40,000$—yielding $z = 4.1456$—it can be compared more easily to the previous objective values for Data Set 1. As expected, it is not the global optimum, but the weighted optimization still improves the tuning of the blade compared to its initial design, offering an 18.0% improvement (collectively) over the baseline design. When the offset (2.6) is removed for the points with $\eta_i \geq \kappa$, the improvement offered by the weighted optimization is 69.3%. This proves that the weighting scheme in the objective function works for weighting one point on a Campbell diagram.

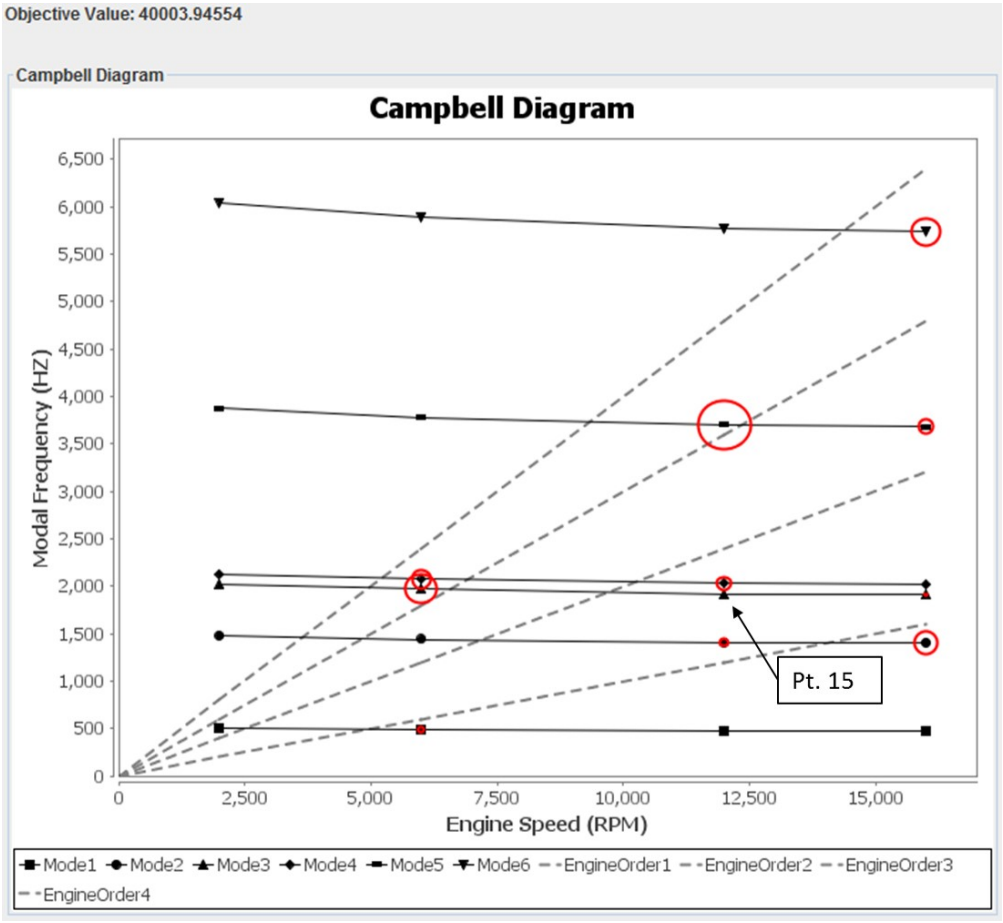

Figure 4.22: The Campbell diagram for a weighted optimization on Data Set 1 showing that $\eta_{15} \geq \kappa$. Compare with the optimization in Figure 4.3.

Weighting Multiple Points

Consider the situation where multiple points are weighted. If the weighted points cannot be varied independently from each other, then the frequency ratios for these points may remain less than the threshold value. This is unlike Point 15 in the previous example where the optimization forced Point 15 to move so that $\eta_{15} \geq \kappa$.

Figure 4.23 shows the baseline Campbell diagram when weighting Points 14, 15, 16, 20, 21, and 22 to be 10,000 times more important than the other points. Notice, once again, that the initial diagram is identical to Figures 4.1 and 4.20, but the objective value is different.

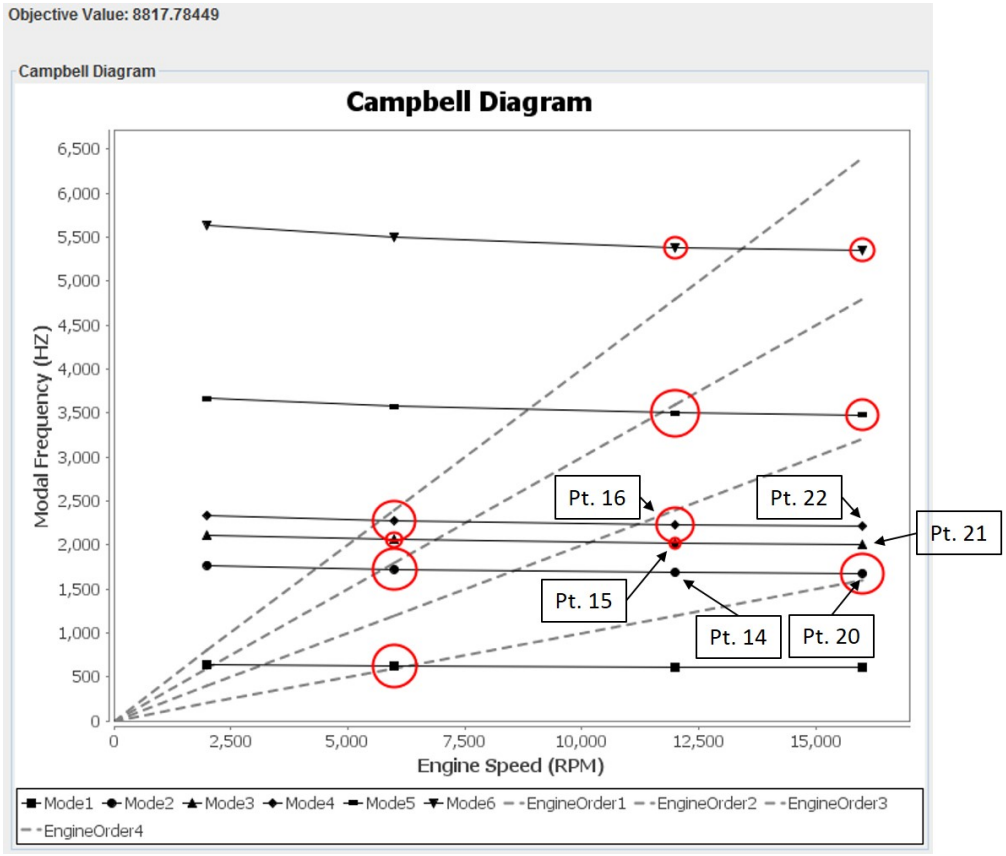

Figure 4.23: The initial Campbell diagram when weighting multiple points (Points 14, 15, 16, 20, 21, and 22) to be 10,000 times more important than the other points.

After completing 10,000 iterations, NSGA-II determined the optimized Campbell diagram retained values of η where $\eta_i < \kappa$ for some of the weighted points. These can be seen in Figure 4.24, where Points 14 and 20 remain too close to the nearest engine order. This is to be expected

since the points cannot move independently of each other. Note that because of the relatively small contributions (in comparison to the weighted points) of Points 9, 10, 17, 18, 23, and 24, these unweighted points remain near the closest engine orders.

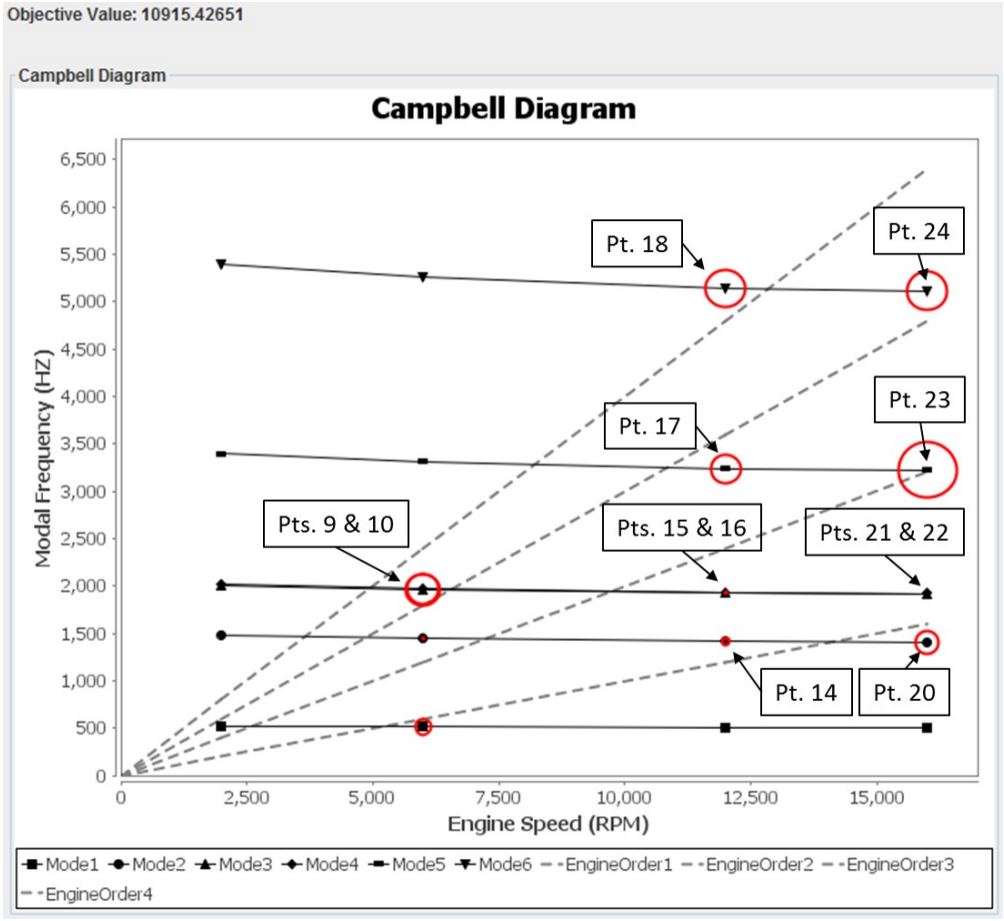

Figure 4.24: The optimized Campbell diagram when weighting multiple points (Points 14, 15, 16, 20, 21, and 22).

From the Campbell diagram, it can be concluded that the optimization accounted for the unique weights and the corresponding points: the contributions for the weighted points have been provided in Table 4.6. It is interesting to note that η_{14} negatively improved during the optimization, despite it having a unique weight. However, when considering that the other five points had the same weight, it is understandable since the drastic improvements in the contributions of the other points outweighed the slight reduction in Point 14's contribution. In fact, the overall improvement in the objective value was 23.8%. Furthermore, when the contributions that remained constant were

removed, the overall improvement was 43.6%. Therefore, the optimization successfully accounted for multiple points with unique weights.

Table 4.6: This table lists the initial and final values of the objective contributions for Data Set 1 with weighted points. Only these weighted points have been included.

Point	Initial Contribution	Optimized Contribution	Improvement
14	2000	1777.7	-11.1%
15	1612.7	1964.0	21.8%
16	710.9	1964.0	176.2%
20	491.6	1207.4	145.6%
21	2000	2000	0.0%
22	2000	2000	0.0%
Total	8817.784	10915.430	23.8%

CHAPTER 5. CONCLUSION

After a brief summary of the research presented in this thesis, this chapter discusses the primary findings of this research, including evaluations of the objective function, the optimization algorithms, and, the general method for algorithmic tuning. Also, suggestions are provided for future work involving this automated blade tuning process, including additions required before integrating this research into industry's airfoil design loops.

5.1 Summary

Airfoil tuning is an essential part of the design process for developing compressor blades for jet and gas turbine engines. This tuning ensures greater safety during operation of the engine because the dynamic loading on the blade will not cause the blade to resonate and fail prematurely. This thesis presents a method for tuning airfoils manually and a novel method that automates the tuning process using surrogate models and optimization techniques. Through the automated method presented in this paper, thousands of candidate blade designs can be explored in minutes. This accelerates the design process, saving time and money for companies in industry, while improving the robustness of the design.

5.2 Objective Function

This thesis presents a novel objective function that simultaneously accounts for all modes at all operating speeds on a Campbell diagram. This objective function provides an advantage over manual tuning because it quantifies the tradeoffs between different sets of blade parameter values. The objective function was tested and validated using two sets of data to create different design spaces. Data Set 1 used 15 blade parameters to map to 24 points on its Campbell diagram; Data Set 2 used 21 blade parameters to map to 52 points on its Campbell diagram. These design spaces

determined the effectiveness and robustness of the objective function because, in each design space, the objective could simultaneously account for all of the points on the Campbell diagrams.

5.3 Optimization Algorithms

NSGA-II and OMOPSO, two gradient-free optimization algorithms, were used to optimize the objective function for each set of data. For Data Set 1, each algorithm performed five optimizations of the design space. Strong evidence is provided in Section 4.1.4 to conclude that the design space was properly optimized by both algorithms to minimize resonance for the final, optimized airfoil.

Additional testing was performed on Data Set 2—data for a real airfoil—to prove the method could be used in a legitimate design process. The design space was optimized 20 times with each algorithm. Evidence is presented in Section 4.2.3 that these algorithms found a robust optimum consistently. This optimum could be the global optimum of the design space since 39 of 40 optimizations found the same location of the design space. The remaining optimization found a local optimum with a lower objective value than the other 39. A full-factorial DOE with sufficient resolution could not be used to validate the solutions found by NSGA-II and OMOPSO due to a predicted run-time of 6.8 years.

5.4 Algorithmic Tuning

Based on these results, the automated method presented is an improved method for tuning an airfoil, because it thoroughly explores the blade's design space to optimally tune the blade to minimize resonance. This is far more reliable than a human designer tuning the same blade, given the same amount of time.

As expected, with additional complication from an increase in input parameters, points on the Campbell diagram, and engine orders, the amount of time to optimize the design space increases. For Data Set 1 (15 inputs, 24 outputs, and 4 engine orders), both NSGA-II and OMOPSO can complete an optimization (10,000 iterations) in 2 minutes. In contrast, for Data Set 2 (21 inputs, 52 outputs, and 9 engine orders), the algorithms complete the optimization (10,000 iterations) in 20 minutes.

5.5 Future Work

This thesis lays the foundation for additional research in this area. Primarily, this thesis presents only a single solution for each optimization run; airfoil design is inherently a multi-objective optimization problem. Different factors like aerodynamic properties must also be incorporated into the optimization to make it truly useful to a designer. Thus, either a weighting scheme or a multi-objective optimization should be incorporated into this method to find more feasible and useful designs.

Furthermore, as stated in by Heap et al., as more DOE runs are included, the error of an RBF surrogate model decreases [7]. In order to determine better results from the optimization, surrogate models with greater accuracy are required. Incorporating more DOE runs into the initialization of the surrogates would improve the results of this method.

Last, this thesis presents a method for determining the theoretical global optimum in a design space. However, at its core, theoretical optimization is flawed in that it "leads to designs that cannot be called 'optimal,' but instead are potentially high-risk solutions that can have a high probability of failing in use" [24]. Furthermore, constrained optimal designs may include binding constraints that the optimization algorithm has driven towards. These designs are close to infeasible and, therefore, unreal solutions. This forces the designer to focus on more than an optimal design—namely a robust, optimal design: a design that is resistant to variation in design variables. This makes a single-objective problem (like the one presented in this thesis) a multi-objective optimization in that the designer wants to optimize the objective function while minimizing the standard deviation of the objective function. This can also be applied to multi-objective problems by incorporating another objective: the robustness of the candidate design.

REFERENCES

[1] Cowles, B., 1996. "High cycle fatigue in aircraft gas turbines: an industry perspective." *International Journal of Fracture,* **80**(2-3), pp. 147–163. 1

[2] Wagner, J. T., 1978. Vibratory tuning of rotatable blades for elastic fluid machines, Aug. 22 US Patent 4,108,573. 1

[3] Meher-Homji, C. B., and Gabriles, G., 1998. "Gas turbine blade failures: causes, avoidance, and troubleshooting." In *27th Turbomachinery Symposium, Houston, TX, Sept*, pp. 20–24. 1

[4] Marsh, S., 2013. "Preventing fretting fatigue in blade dovetail roots by modifying geometry of contact surfaces." *Power Transmission Engineering,* **28**, pp. 45–49. 1

[5] Younossi, O., Arena, M. V., Moore, R. M., Lorell, M., and Mason, J., 2002. Military jet engine acquisition: Technology basics and cost-estimating methodology Tech. rep., DTIC Document. 1

[6] Honisch, P., Strehlau, U., and Kuhhorn, A., 2012. "Modelling of industrial blade integrated disks (blisks) with regard to mistuning." In *Proceedings of ISMA*. 1

[7] Heap, R. C., Hepworth, A. I., and Jensen, C. G., 2015. "Real-time visualization of finite element models using surrogate modeling methods." *Journal of Computing and Information Science in Engineering,* **15**(1), p. 011007. 3, 11, 23, 25, 79

[8] Gano, S. E., Kim, H., and Brown, D. E., 2006. "Comparison of three surrogate modeling techniques: Datascape, kriging, and second order regression." In *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA-2006-7048, Portsmouth, Virginia*. 3

[9] Castanier, M. P., and Pierre, C., 2006. "Modeling and analysis of mistuned bladed disk vibration: current status and emerging directions." *Journal of Propulsion and Power,* **22**(2), pp. 384–396. 5

[10] Selin, E. D., 2012. "Application of parametric nurbs geometry to mode shape identification and the modal assurance criterion." Master's thesis. 6

[11] Sever, I. A., 2004. "Experimental validation of turbomachinery blade vibration predictions." PhD thesis, University of London. 6

[12] Saravanamuttoo, H. I. H., Rogers, G. F. C., and Cohen, H., 2001. *Gas turbine theory*. Pearson Education. 6, 44

[13] Zhu, P., Zhang, Y., and Chen, G., 2011. "Metamodeling development for reliability-based design optimization of automotive body structure." *Computers in Industry,* **62**(7), pp. 729–741. 10

[14] Schaback, R., 2007. "A practical guide to radial basis functions." *Electronic Resource*. 11

[15] Kansa, E. J., 1999. "Motivation for using radial basis functions to solve pdes." *RN,* **64**(1), p. 1. 12

[16] Keane, A. J., 2009. "Comparison of several optimization strategies for robust turbine blade design." *Journal of Propulsion and Power,* **25**(5), pp. 1092–1099. 14

[17] Zingg, D. W., Nemec, M., and Pulliam, T. H., 2008. "A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization." *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique,* **17**(1-2), pp. 103–126. 15, 58

[18] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T., 2000. "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii." *Lecture notes in computer science,* **1917**, pp. 849–858. 15

[19] Hadka, D., 2015. *MOEA Framework User Guide*., 2.7 ed. MOEA available at http://www.moeaframework.org/. 16, 17

[20] Sierra, M. R., and Coello, C. A. C., 2005. "Improving pso-based multi-objective optimization using crowding, mutation and-dominance." In *Evolutionary Multi-Criterion Optimization*, Springer, pp. 505–519. 17

[21] Hansen, N., and Kern, S., 2004. "Evaluating the cma evolution strategy on multimodal test functions." In *Parallel problem solving from nature-PPSN VIII*, Springer, pp. 282–291. 17, 68

[22] Isight, S., 2009. "Isight component guide." *North Carolina, USA: Engineous Software*. 22, 24

[23] Park, J.-S., 1994. "Optimal latin-hypercube designs for computer experiments." *Journal of statistical planning and inference,* **39**(1), pp. 95–111. 22

[24] Van der Velden, A., and Koch, P., 2010. "Isight design optimization methodologies." *ASM Handbook,* **22**. 79