2019-06

OPTIMIZATION ROUTINE FOR COMPRESSOR DESIGN USING COMMERCIAL SOFTWARE

Brantner, Martin

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OPTIMIZATION ROUTINE FOR COMPRESSOR DESIGN USING COMMERCIAL SOFTWARE

by

Martin Brantner

June 2019

Thesis Advisor: Anthony J. Gannon
Co-Advisor: Garth V. Hobson

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE OPTIMIZATION ROUTINE FOR COMPRESSOR DESIGN USING COMMERCIAL SOFTWARE	**5. FUNDING NUMBERS**
6. AUTHOR(S) Martin Brantner	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Previous research at Naval Postgraduate School produced a design procedure for generating gas compressor rotor models using commercial software. This design procedure was improved upon by parameterizing the blade profiles as quadratic functions of axial distance and chord location. An objective function for compressor performance using pressure ratio, efficiency, and massflow was formulated, which applied a holistic view of quantifying the value of a compressor design. An optimization algorithm was implemented within the procedure to seek an optimal design by generating a CAD model, performing computational fluid dynamics (CFD) analysis, and iterating the design function constants toward an optimal solution. The routine found an optimal compressor design; however, inconsistencies in the CFD solution data prevented the routine from finding the expected global maximum.

14. SUBJECT TERMS axial compressor, design tool, optimization	**15. NUMBER OF PAGES** 59
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

OPTIMIZATION ROUTINE FOR COMPRESSOR DESIGN USING COMMERCIAL SOFTWARE

Martin Brantner
Lieutenant, United States Navy
BA, University of Washington, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2019

Approved by: Anthony J. Gannon
Advisor

Garth V. Hobson
Co-Advisor

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Previous research at Naval Postgraduate School produced a design procedure for generating gas compressor rotor models using commercial software. This design procedure was improved upon by parameterizing the blade profiles as quadratic functions of axial distance and chord location. An objective function for compressor performance using pressure ratio, efficiency, and massflow was formulated, which applied a holistic view of quantifying the value of a compressor design. An optimization algorithm was implemented within the procedure to seek an optimal design by generating a CAD model, performing computational fluid dynamics (CFD) analysis, and iterating the design function constants toward an optimal solution. The routine found an optimal compressor design; however, inconsistencies in the CFD solution data prevented the routine from finding the expected global maximum.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
A. MOTIVATION....1
B. PREVIOUS WORK....2
C. OBJECTIVES....3

II. CHANGES TO DESIGN SPACE....5
A. INCREASING DEGREES OF FREEDOM....5
B. PARAMETERIZE DESIGN INPUTS....7

III. OPTIMIZATION PROCEDURE....11
A. OPTIMIZATION FUNCTION....11
B. OPTIMIZATION WORKFLOW....13
C. OPTIMIZATION PROCEDURE SETUP....14

IV. RESULTS....15
A. EXPECTED OPTIMIZATION RESULTS....15
B. OPTIMIZATION PROCEDURE RESULTS....17
C. DATA DISCREPANCIES....22
D. RESULTS REPEATABILITY....24

V. CONCLUSIONS AND RECOMMENDATIONS....29
A. CONCLUSIONS....29
B. RECOMMENDATIONS....29

APPENDIX A. MATLAB CODE....31
A. MAINOPTIM....31
B. ZFUN....31
C. CAMBER FUNCTION....32
D. DATAGENERATOR....33

APPENDIX B. OVERVIEW OF BLADE DESIGN PARAMETERS....35

APPENDIX C. CODE REPOSITORY....37

LIST OF REFERENCES....39

INITIAL DISTRIBUTION LIST....41

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Design Procedure Workflow. Adapted from [4].....2

Figure 2. Linear Interpolation. Adapted from [6]. 6

Figure 3. Physical Effect of Changing Constant "A" on the Compressor Blades.....9

Figure 4. Illustration of Optimization Function 12

Figure 5. Optimization Procedure. Adapted from [4]. 13

Figure 6. Recorded Data and Polyfit Data from Optimization Procedure 15

Figure 7. Refined Data and Polyfit Data from Optimization Procedure 17

Figure 8. Results of Optimizer Procedure 19

Figure 9. Optimization Function for Final Iteration.....20

Figure 10. Optimizer Procedure Blade Geometry.....21

Figure 11. Optimal Solution Design.....21

Figure 12. ZFun() Evaluations at A≈49.6.....23

Figure 13. Massrate Range with Respect to Zfun() Evaluation 24

Figure 14. Results of Second Optimizer Execution 26

Figure 15. ZFun() Evaluations at A≈44.4 27

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1 Initial Values for Camber Function 8

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CAD	computer aided design
CAM	computer aided manufacturing
CFD	computational fluid dynamics
CNC	computer numerical control
DOF	degree of freedom
EA	evolutionary algorithm
MATLAB	Matrix Laboratory
NLP	nonlinear programming
NPS	Naval Postgraduate School

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Thank you to Dr. Gannon for his immense patience throughout the length of this study.

Thank you to Dr. Frank Giraldo of the Applied Mathematics Department for fostering my interest in scientific computation, without which I would not have pursued this study.

Thank you to my wife for her support through the ebbs and flows of my stresses while pursuing my master's degree.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

From computer aided design (CAD) tools to assist with initial sketches to computer aided manufacturing (CAM) software creating toolpaths for computer numerical control (CNC) routers that manufacture complex parts; modern engineering design is nearly inconceivable without the use of computers in some manner. Leveraging increased computational power to rapidly iterate on design parameters holds the promise of creating final designs, which may have been previously unfeasible due to their manpower intensive nature. In addition, these methods could result in shorter timelines. Leveraging this power requires algorithms to automate the workflow and a well-defined optimization problem or goal-function to direct the automation towards an optimal solution.

Gas turbine compressors provide an excellent platform on which to apply an optimization routine within an automated workflow algorithm because the physical goals can be well modeled and the optimization goal can be well posed within a nonlinear programming (NLP) problem [1]. Furthermore, computational fluid dynamics (CFD) has achieved a high degree of sophistication, allowing the relevant goals to be estimated with relative accuracy. The estimation enables the programming of an optimization procedure that can then be used to search for an optimal design. Finally, the problem is well suited to bounding of the optimization problem because of the physical limitations of machining the resultant compressor.

The goals can be well formulated because compressors have a well-defined purpose of achieving a high-pressure ratio with high efficiency and a broad mass flow range. Higher pressure ratios reduce the required number of total stages, broader mass flow range increases stall margin and higher efficiency reflects better performance and thus reduced fuel usage [2]. Previous studies [3] have often focused on optimizing an individual component of pressure ratio, mass flow, or efficiency at times to the detriment of the other factors. Large power plants may be able to afford multiple stages, thus reducing the need for high stage pressure ratios. In contrast, a single use engine may prioritize efficiency at

the expense of stall margin. By formulating an optimization function, there is an opportunity to propose a holistic approach to quantifying the value of a compressor stage.

B. PREVIOUS WORK

Previous research has provided the groundwork for optimization. Sanger [4] reported in 1996 on combining CFD techniques to design a compressor rotor with minimal user input. This work was expanded in 2013 by Drayton [5], who created a design procedure using commercial software to create CAD models of turbine compressors. This procedure was successfully used to design, manufacture, and test a splittered rotor at NPS. A general workflow diagram is given in Figure 1. While CFD was used sequentially to produce an estimated speed line of the compressor, the procedure did not use automated modeling to improve upon the initial design.

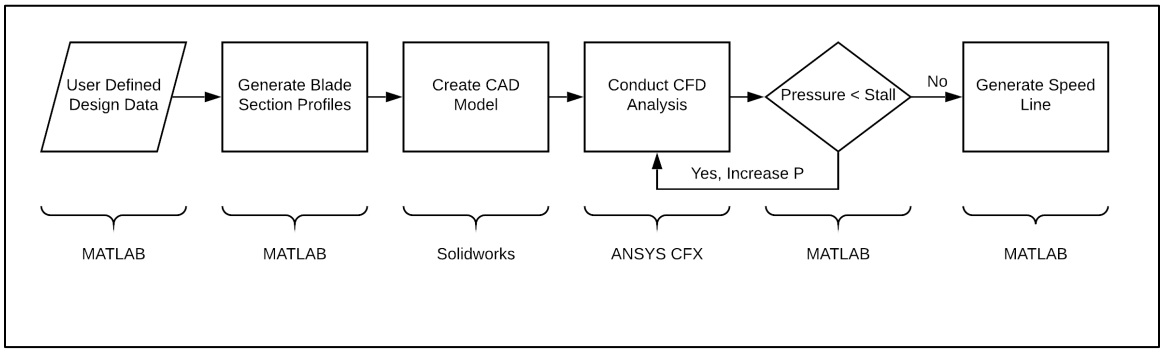


Figure 1. Design Procedure Workflow. Adapted from [4].

Optimization is a well-researched topic; however, application to compressor design is often limited by the computational power required. Application of an evolutionary algorithm (EA) to the design of a centrifugal compressor is described by Benini [5]. In that research, blade camber lines are described using 10-12 parameters derived from parameterization of Bezier curves. Blade thickness is described with a tip radius and a constant thickness elsewhere. A splitter blade is described as a derivation of the main blade. Thus, a model was produced using 14–16 parameters. The evolutionary algorithm sought

to maximize isentropic efficiency for fixed pressure ratios or mass flow rate and resulted in 400 hours of computation time and a proposed improved blade geometry.

C. OBJECTIVES

The object of this study is to improve upon the NPS design procedure [4] for axial splittered rotors by creating an optimization procedure framework using commercial software to search for an optimal design. Creation of the framework is based on four aims:

1. Expand the degrees of freedom within the design space as inputted to the MATLAB portion of the algorithm to increase the fidelity of the CAD models and thereby improve the procedure's ability to iterate on the design parameters.

2. Parametrize the design inputs for blade camber angle to be a function of blade section height and chord control point. The splitter blade camber is cast as a fraction of the main blade camber. Camber angle was selected as the single degree of freedom (DOF) optimization point because it has a direct effect on the work inputted to the airflow.

3. Formulate an optimization function for the NLP problem that modelled real world design goals using the output from Computational Fluid Dynamics (CFD) as input. The function holistically quantifies the value of a compressor design.

4. Use an optimization algorithm to call the function and iterate on design parameters to seek an optimal design. The resultant procedure provides the framework for future geometry development.

THIS PAGE INTENTIONALLY LEFT BLANK

II. CHANGES TO DESIGN SPACE

A. INCREASING DEGREES OF FREEDOM

The previously developed design procedure [4] utilized a MATLAB script titled HardCodeBlade.m to provide the relevant design parameters to generate blade profiles. As developed by Drayton, the procedure utilized a cubic spline fit for camber angle using four control points. Following the original work, blade definitions were reduced to a series of three row, three column matrices. Each row of the matrix represented a blade section at 25%, 67.5%, and 110% of the inlet radius and each column represented 0%, 25%, and 100% of the chord length along the section at the three heights. 110% of inlet radius was selected as an outer limit to account for trimming the blades for tip clearance. Cubic spline was maintained for camber across the section chord and linear interpolation used between section heights to produce five blade section profiles

As implemented, this method had two restrictions. First, using Taylor series theory, the bounded error for centered point linear interpolation will be of order $(\Delta h)^2$, derived, as shown in Figure 2. Because the Δh for the blade sections is 42.5% of the inlet radius, the resulting minimum definition for a blade profile is 18% of the radius. Second, were higher order polynomials implemented to interpolate between defined points, the highest possible polynomial would be of degree 2 [6]. Because designing otherwise infeasible rotors is an implicit goal for the optimization procedure, an attempt was made to expand the design DOF by reducing the step size of the parameters. Increments of 10% were selected for section height (10% to 110%) which resulted in an 11x11 matrix and a minimum definition of 1%.

$$1: f(x_0+h) = f(x_0) + hf'(x_0) + {}^{h^2}\!/_{2}\, f''(x_0) + \ldots$$

$$2: f(x_0-h) = f(x_0) - hf'(x_0) + {}^{h^2}\!/_{2}\, f''(x_0) + \ldots$$

$$3: f(x_0+h) + f(x_0-h) = 2f(x_0) + h^2 f''(x_0) + \ldots$$

$$4: f(x_0) = \frac{f(x_0+h) + f(x_0-h)}{2} + \frac{-h^2 f''(x_0) + \ldots}{2}$$

Figure 2. Linear Interpolation. Adapted from [6].

The changes to HardCodeBlade.m necessitated changes to later portions of the code to accept the 11x11 matrix vice 3x3. The implementation of the revised HardCodeBlade.m was ultimately prevented by the mapping portion of the code. One challenge in creating the CAD models is the need to map the blade section geometry from a flat, 2-dimensional space onto a curved, 3-dimensional space. This mapping is necessary because of the 3-dimensional nature of the airflow into the compressor. Blade sections are designed using a 2-dimensional approximation of the flow and are then mapped onto a streamline approximation so that the fluid flow is nearly parallel to the blade section and the desired 2-dimensional flow characteristics are nearly achieved.

The streamline at the hub is approximated as the hub itself, which extends from centerline to 50% of the inlet radius per the previous procedure [4]. The revised HardCodeBlade.m defined multiple blade geometries within the volume of the hub. The corresponding streamlines for those geometries were all approximated at the hub, resulting in blade sections with different cross sections but nearly equal location on the hub surface. When those cross sections were drawn in SolidWorks, the resultant curves created a self-intersecting geometry that failed to produce a volume and thus a failed CAD model.

The problem of self-intersecting geometry was previously circumvented by defining one blade section on the hub (at 25% radius) and the subsequent point above the hub (at 67.5% radius). While multiple attempts were made to apply the revised HardCodeBlade.m file within the previous procedure, the benefits of design

parameterization ultimately outweighed the perceived benefits of expanding the number of blade section definitions and this aim was abandoned.

B. PARAMETERIZE DESIGN INPUTS

To reduce the complexity of the optimization problem and provide a demonstration of the proposed method, camber angle was selected as the sole DOF for blade profiles. Per Equation 1, reproduced from [2], the stage pressure ratio is directly related to the degree of turning (Δc_θ) of the flow. The degree of turning is determined by the change in camber angle from the leading edge to the trailing edge. Thus, the performance of the compressor is directly affected by adjusting the camber angle across the chord of the blade making it an ideal design parameter on which to iterate.

$$\frac{p_{03}}{p_{01}} = \left[1 + \eta_{st} \frac{U \Delta c_\theta}{c_p T_{01}}\right]^{\gamma/(\gamma-1)} \quad 1$$

Within the blade geometry, a camber angle of 0.0 is parallel with the machine axis and 90.0 is perpendicular, aligned with the compressor rotation. The camber angle of the chord was parameterized as a function of chord position and blade height as given in Equation 2. Within the design function, chord is defined by unit length with control points cast as a fraction of this at the 0.0, 0.25, and 1.0 locations. Blade height is defined, as a fraction of the case radius where 0.0 is the rotation axis and 1.0 is the radius. Blade elements are controlled at .25, 0.675, and 1.10. A quadratic equation was selected as a second order method is the highest interpolation achievable using three control points [6].

$$Camber(c,h) = A \cdot c^2 \cdot h^2 + B \cdot c^2 \cdot h + C \cdot c^2 + D \cdot c \cdot h^2 + E \cdot h^2 + F \cdot c \cdot h + G \cdot c + H \cdot h + I \quad 2$$

In Equation 2, the capital letters are constants supplied as input to the blade geometry MATLAB function. Because the values for “B” through “F” were maintained at 0.0 throughout testing, Equation 2 can be simplified, as shown in Equation 3. The initial values supplied to the optimization function are listed in Table 1.

$$Camber(c,h) = A \cdot c^2 \cdot h^2 + G \cdot c + H \cdot h + I \quad 3$$

“I” is the constant camber angle and controls all subsequent blade elements. 68 degrees was selected based on the design assumptions made for the Sanger rotor to achieve constant axial velocity through the compressor [7]. “G” controls change in camber solely with increase chord position. This value provides the degree of flow turning across the blade surface and is set to a negative value to reflect the relative rotational direction. “H” changes camber angle solely with increase in blade height from root to tip. “H” is set to a negative value to reduce camber at the blade tip, which accounts for the higher tip speed compared to the root of the blade.

Table 1 Initial Values for Camber Function

G	-37
H	-28
I	68

“A” describes the change in camber angle as a quadratic relation to blade height and chord position and was chosen as the singular DOF for the optimization algorithm. The physical effect of changing “A” values is shown in Figure 3. While a quadratic relationship was selected via an extension of interpolation, the second order also provides for a second derivative thus allowing a change in the curvature of the blade chord. This effect is best seen at the extremes of the “A” range in Figure 3 . At A = -44, the blade tip has a pronounced J-shape while at A = +44, the blade tip inverts at the trailing edge. The camber becomes greater than 90 degrees at A= +58 representing the upper limit of “A.” Additionally, “A” was selected because it does not affect the blade geometry at the leading edge (where c = 0.0) and minimally affects the geometry at the root (where h = 0.25); however, it dramatically affects the blade tip (where h = 1.1). This relationship allows the tip to be flattened compared to the root reflecting the fact that the tip will induce more work because of its greater velocity.

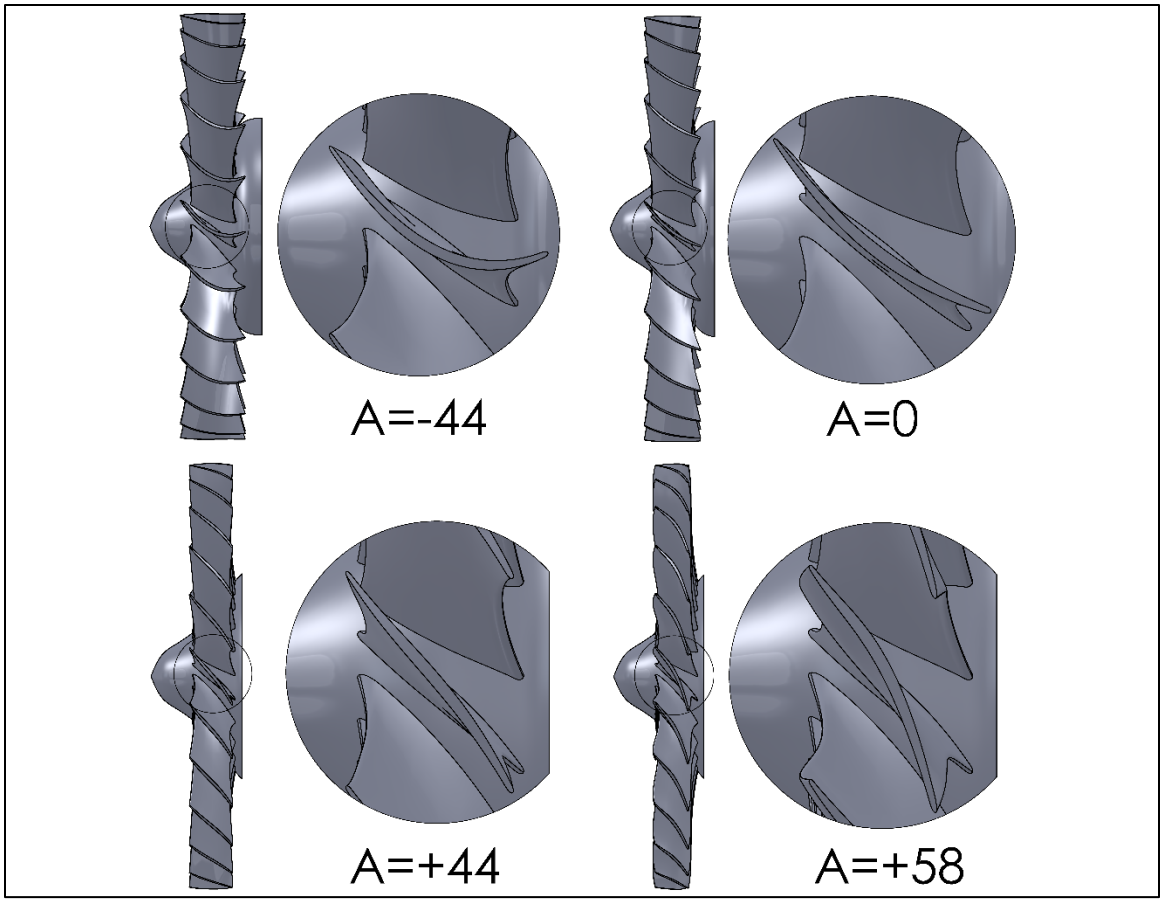


Figure 3. Physical Effect of Changing Constant “A” on the Compressor Blades

THIS PAGE INTENTIONALLY LEFT BLANK

III. OPTIMIZATION PROCEDURE

A. OPTIMIZATION FUNCTION

The optimization procedure requires an optimization function to quantify the value of any given compressor design. The optimization function is defined as $Z(P_r, \eta, \dot{m})$, as given in Equation 4. Equation 4 is the numeric integration by trapezoidal rule of the product of isentropic efficiency and pressure ratio with respect to the mass flow across the throttling test points.

$$Z = \sum_{i=1}^{n_o} \frac{(\eta_{i-1} * P_{r_{i-1}}) + (\eta_i * P_{r_i})}{2} \Delta_{\dot{m}_i} \qquad 4$$

The process of determining Z is illustrated in Figure 4. First, the pressure ratio is plotted versus the mass flow rate for three different back pressures. Second, the efficiencies are plotted against the mass rate at each back pressure. Third, the product of pressure ratio and efficiency is plotted against the mass rate for each back pressure. Finally, the product is integrated with respect to the mass rate to determine a scalar value, which represents the optimization value for the evaluated compressor. This is shown as the solid region in the figure.

Pressure ratio is multiplied by isentropic efficiency to try to ensure the optimizer searches for high efficiency operation across the range of use. The product is integrated with respect to mass flow to maximize the operating region of the compressor prior to a choked condition. For the optimization procedure, the integration takes place across three data points representing three points of throttling of the compressor. This is implemented by performing the CFD analysis with increasing values of outlet pressure, specifically 0.00 atm, 0.01 atm and 0.02 atm. In an expanded method, the number of points could easily be increased up to the stall point.

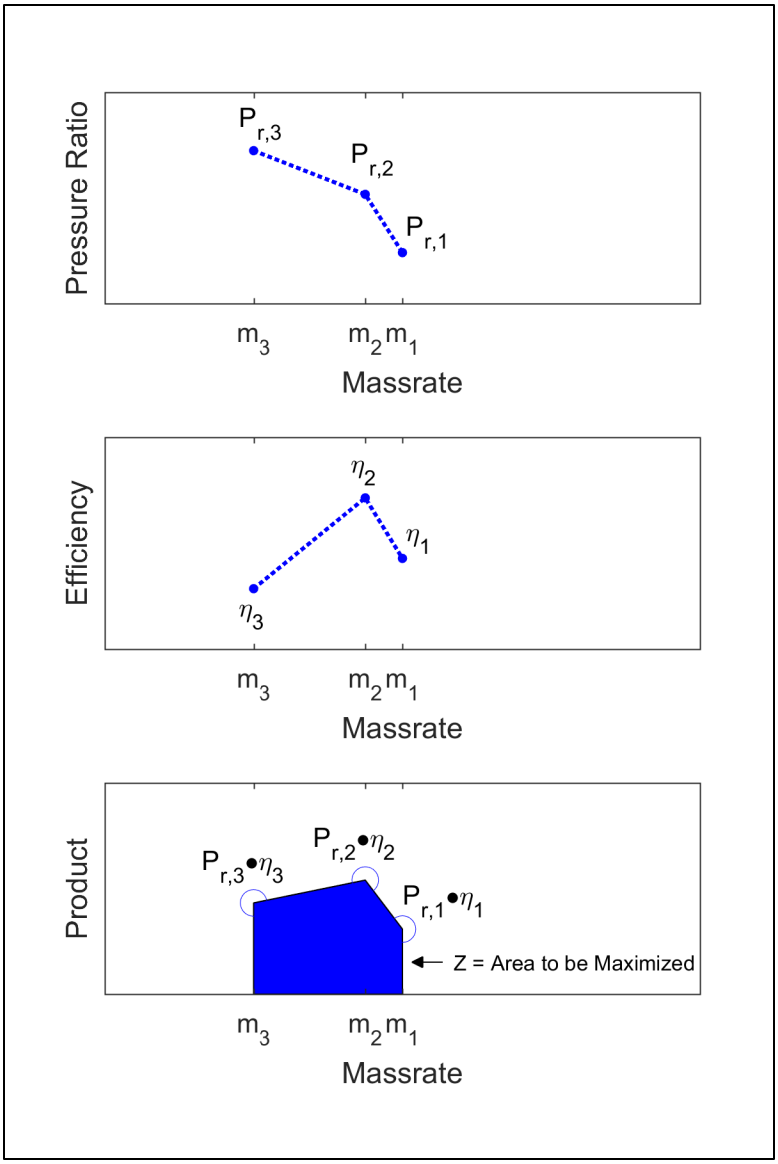


Figure 4. Illustration of Optimization Function

The optimization function was implemented in MATLAB using fminbnd() which is an implementation of the golden section search algorithm described by Brent [8]. The golden search algorithm is guaranteed to find a local minimum for a concave function by iteratively evaluating the function in the direction of reducing value and performing parabolic interpolation for the intermediates points. Fminbnd() was selected over alternative functions for several reasons. First, it provides for bounded optimization, which is necessary both to simplify the computational problem and the physical constraints of the

compressor. Second, by reducing the optimization problem to a single DOF, the capacity to use vectors and linear equalities available in other algorithms is unnecessary. Finally, fminbnd() does not attempt to compute a derivative. Because CFD is used as a blind source of data, it is unreasonable to expect a smooth derivative from the resultant optimization data, making a search for a zero derivative unreasonable. Other optimizers capable of dealing with more variables could be easily substituted in future work.

B. OPTIMIZATION WORKFLOW

The optimization function was incorporated into the previous procedure [4], as shown in Figure 5. The optimization procedure is implemented via MainOptim.m. MainOptim.m stores the lower and upper bounds, the options passed to fminbnd(), defines the optimization function as ZFun(), and calls fminbnd().

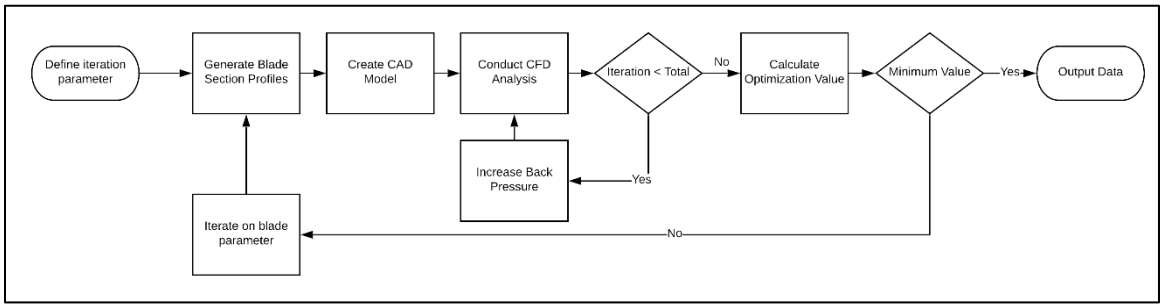


Figure 5. Optimization Procedure. Adapted from [4].

ZFun() is the function required by fminbnd() to compute an optimization value for the given design parameter. The function saves the current value of "A" to the MATLAB data structure DesignParam.mat that holds all of the constants used to design the blade sections. It then calls the scripts to produce the blade geometries, the air passage geometry, and the CAD model. It then calls a function, DataGenerator(), that iteratively runs the CFD analysis on the current compressor design. DataGenerator() saves the CFD solution for isentropic efficiency, pressure ratio, and mass flow rate into a data structure, RunContainer.mat. After DataGenerator() completes CFD at the three throttling pressures; ZFun() reads RunContainer.mat, computes the integral, multiplies by negative one and

returns the resultant value for fminbnd(). While the optimization problem seeks to maximize Equation 4, fminbnd() seeks the minimum value, necessitating the multiplication by negative one.

After fminbnd() receives the optimization value from ZFun(), it determines if a local minimum was reached. If not, fminbnd() iterates on the value of "A", and calls ZFun() on the iterative value.

C. OPTIMIZATION PROCEDURE SETUP

All design parameters are referenced as fractions of inlet radius, a value never explicitly defined. When the SolidWorks model is created, the ambiguity results in a radius of size 1.0 of the default unit, which is meters. While a 1.0m compressor is not suitable for manufacture and testing on the NPS test rig, the physical size of the compressor is inconsequential to validating the optimization procedure. Therefore, the 1.0m radius was maintained and subsequent parameters were selected accordingly. The rotational speed of the compressor was set to 1638 RPM corresponding to a tip speed of about Mach 0.5.

Prior to starting the optimization procedure, the prior CFD results file was reset to prevent skewing the CFD solution and to prevent file transfer errors. Previous experiments with the CFD solver demonstrated file transfer errors at higher iteration numbers (ranging from 200 to 500 time steps). While accepting the transfer error allowed the solver to continue, the necessitated human interaction invalidated the desired autonomy of the optimization procedure.

The optimization problem is posed as maximize $Z(P_r, \eta, \dot{m})$, by selection of "A", subject to "A" $\in$ [36, 58]. The upper bound of 58 is the smallest integer resulting in a camber angle at the trailing edge of the blade tip less than 90 degrees. Camber greater than 90 degrees is physically unrealistic and resulted in failed CAD modeling of the air wedge. The lower bound of 36 was selected because of the difficulty of achieving a convergent CFD solution at lower values of "A". Convergence for this problem was defined as a root mean square residual of 2.0e-4 for mass and momentum, which roughly corresponded with the observed steady state solution residual when iterations were unlimited.

IV. RESULTS

A. EXPECTED OPTIMIZATION RESULTS

To verify the results of the optimization procedure, ZFun() was manually called for values of "A" from 36 to 58 in 2.0 degree increments. The results of that testing are shown in Figure 6.

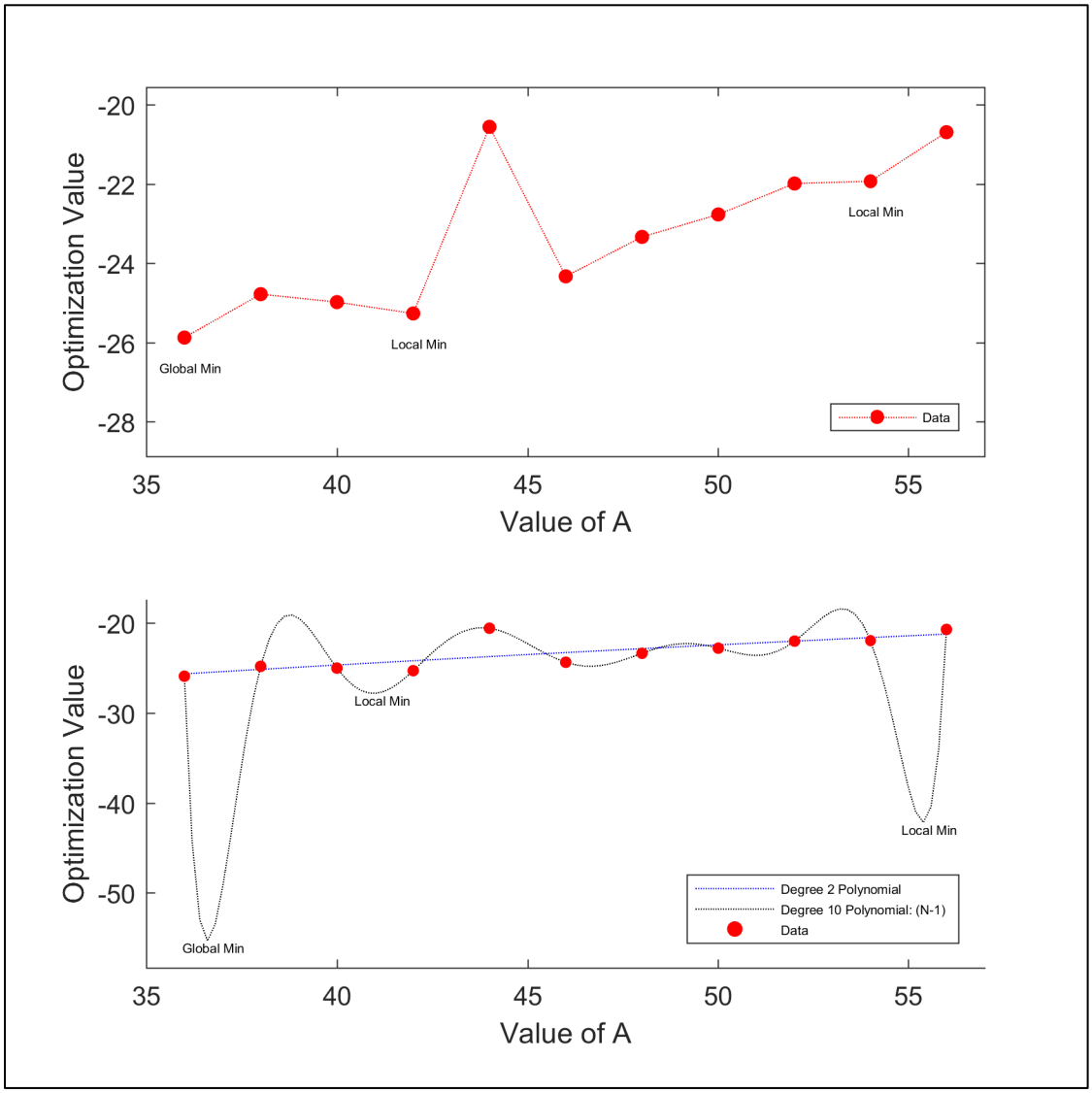


Figure 6. Recorded Data and Polyfit Data from Optimization Procedure

Polynomial fitting was performed with a degree (N-1) polynomial and a degree 2 polynomial representative of the parabolic interpolation performed by fminbnd(). As shown, using the quadratic interpolation, a global minimum should be expected at A≈36. However, the presumably erroneous data point at A=44 strongly skews the (N-1) polynomial interpolation, moving the global minimum to A≈37 with local minima at A≈55 and A≈41.

Removing the data point at A=44 produces the results shown in Figure 7. The degree two polynomial and the degree (N-1) polynomial now agree with respect to the global minimum value at A≈36 with local minima at A≈43 and A≈55. Additionally, the strength of the local minima now decreases with increasing "A" which correlates well with the recorded data. The discontinuity at A=44 reflects the difficulty in using CFD as a blind source of data because of the skewing effects of an inconsistent solution which could be perceived by fminbnd() as a valid result.

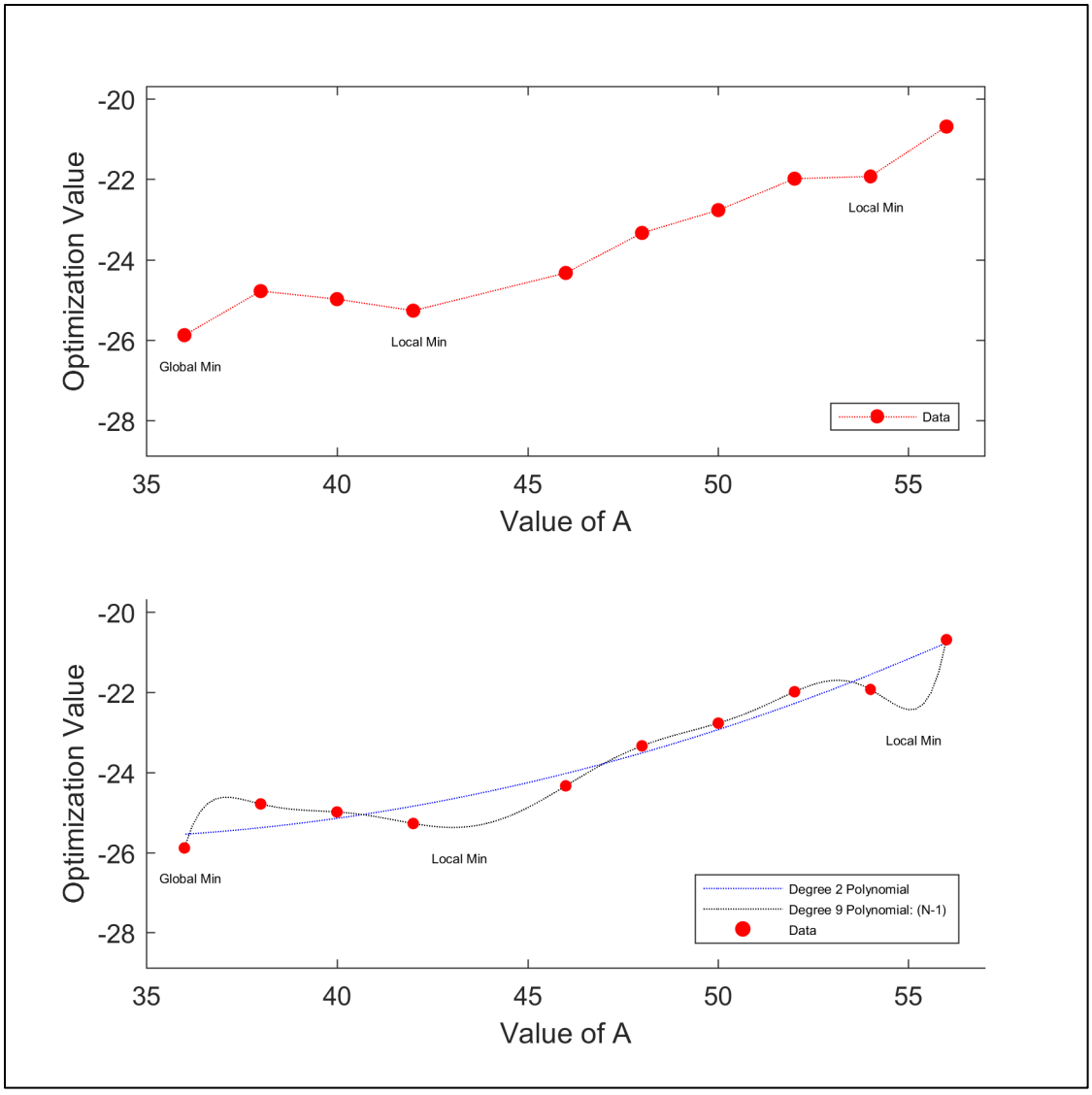


Figure 7. Refined Data and Polyfit Data from Optimization Procedure

B. OPTIMIZATION PROCEDURE RESULTS

The results for the optimization procedure are shown in Figure 8. The optimal solution found by fminbnd() was A= 49.597 with Z= -22.774 after 27 iterations and an initial value of A= 44.4 selected by fminbnd(). The initial and final blade profiles are shown in Figure 10.

As can be seen, there is a significantly discontinuous minimum at A= 49.597 where Z= -70.828. Unfortunately, this data point occurred at the second evaluation of ZFun() by

fminbnd(). The result was fminbnd() searching for a minimum near A≈49 despite the expected minimum occurring near A≈36. The second pane of Figure 8 shows the data if the erroneous evaluation is discarded. As can be seen, the minimum data point occurs at A=44.4 which was the first evaluation of ZFun() in both the recorded data and the refined data, and corresponds with the expected minimum. Unfortunately, the spread of ZFun() evaluations for small "A" increments precluded fminbnd() from seeking a minima near A=44.4. Also illustrated in the second pane is the spread of evaluations near A=49.6 and which is evaluated in Figure 12.

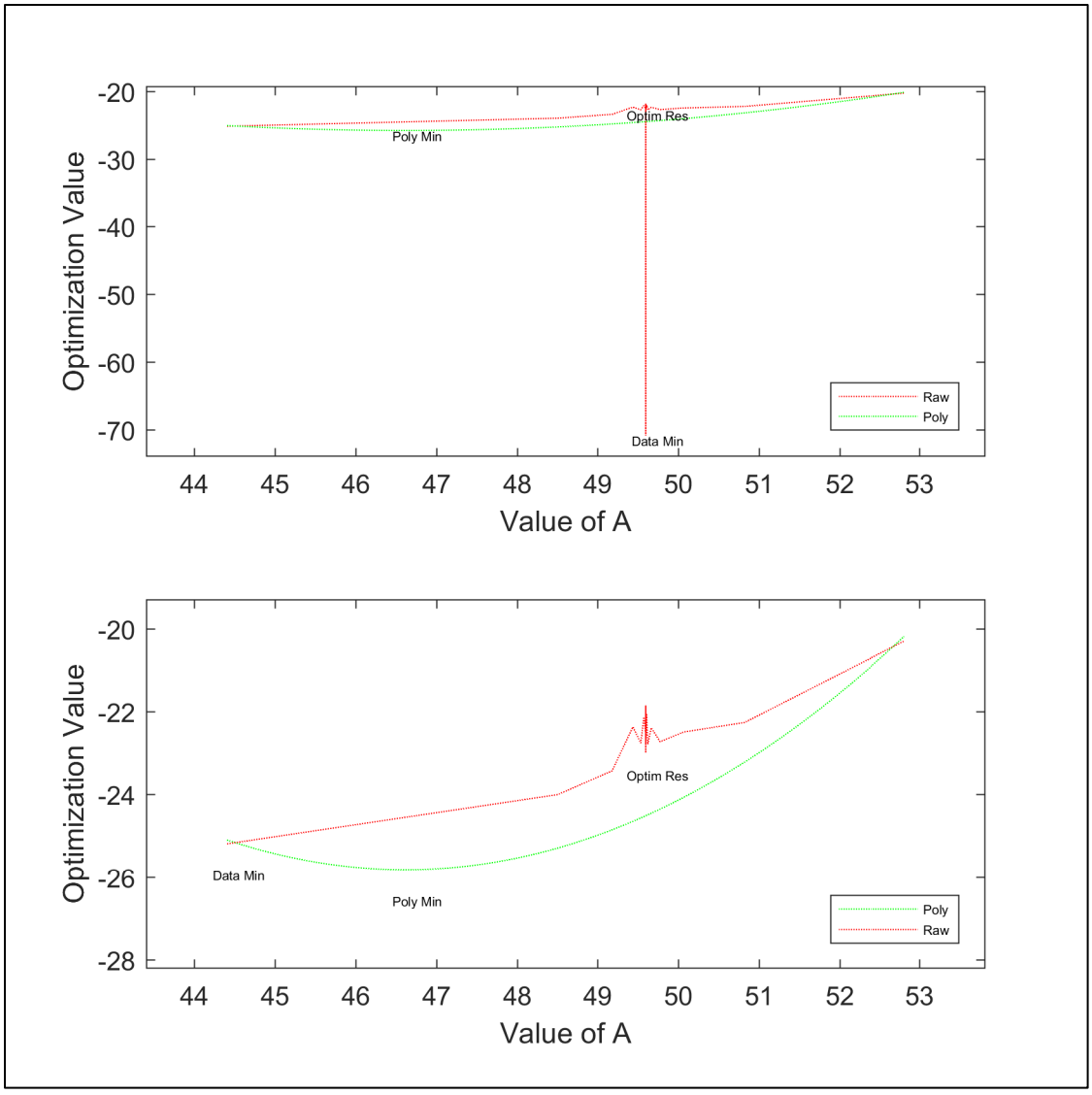


Figure 8. Results of Optimizer Procedure

The evaluation of ZFun() is illustrated in Figure 9. As can be seen, the optimal solution found by fminbnd() achieved a relatively low pressure ratio of 1.045 and a relatively low efficiency of 73.3%. This is representative of the fact that the underlying compressor geometry is not a particularly realistic design. However, the representative dynamics followed expectations. As throttle pressure was increased; the mass flow rate decreases, the pressure ratio increases, and efficiency decreases as the throttling approached the stall pressure. With an appropriate starting geometry and with an appropriate means of addressing erroneous data points, the underlying procedure could

then be used to achieve an optimal real-world compressor design for manufacture and validation.

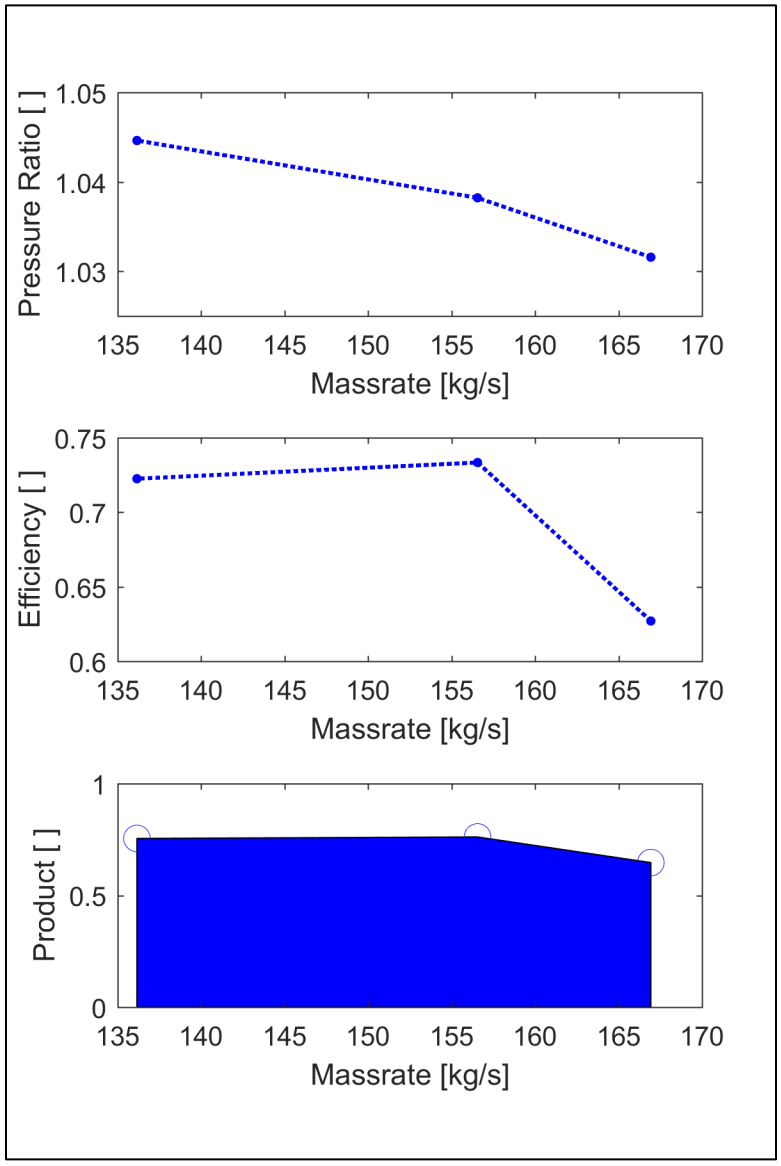


Figure 9. Optimization Function for Final Iteration

The difference in geometry compared to a realistic compressor is demonstrated by the initial and final blade geometries, as shown in Figure 10, and the optimal solution design, as shown in Figure 11. As can be seen the blade profiles are thinner than would be

expected for a compressor operating at Mach 0.5. Additionally, the blade length is much longer relative to the hub radius than would be expected for a single compressor stage.

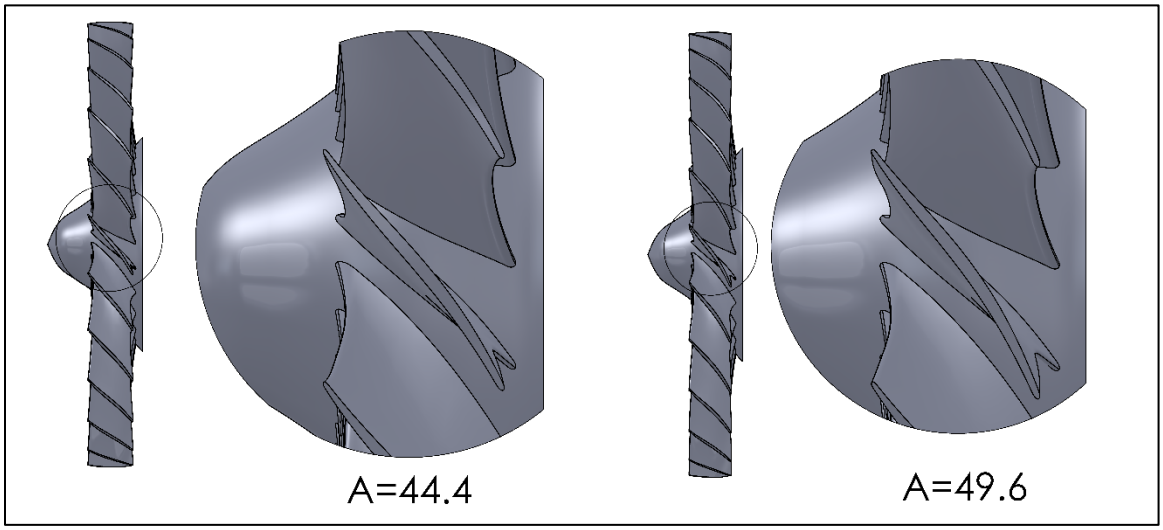


Figure 10. Optimizer Procedure Blade Geometry

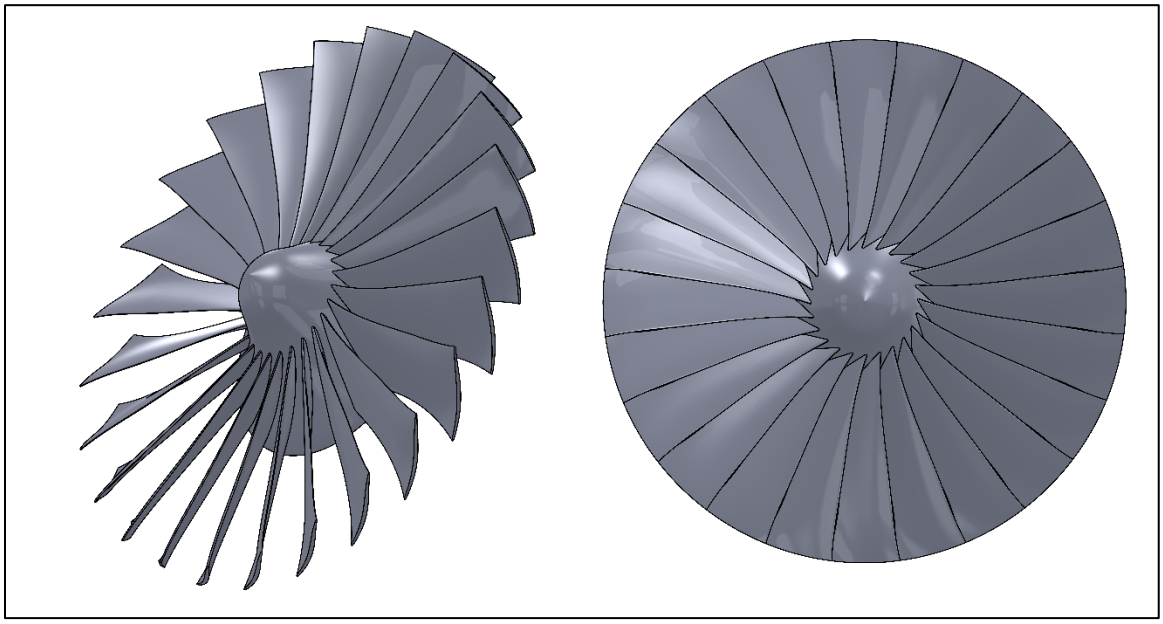


Figure 11. Optimal Solution Design

C. DATA DISCREPANCIES

The erroneous evaluation at A=49.597 is representative of the difficulty of using CFD as a blind source of data; a difficulty also illustrated in Figure 6 by the erroneous evaluation at A=44. Using the parabolic interpolation derived from the refined data set illustrated in Figure 7, the expected evaluation at A=49.6 would be Z=-23.0. This is near the final evaluation of Z=-22.8. However, fminbnd() performed 17 evaluations within $^{+}/_{-}$ 0.5 of A=49.6. Discarding the evaluation of Z=-70.828, fminbnd() performed 16 evaluations, plotted in Figure 12. As shown, within a range of $\Delta A=0.0493$, there was a variance of $\Delta Z=1.15$. The expected value of Z=-23.0 is located 2.04 standard deviations from the average value of Z= -22.4.

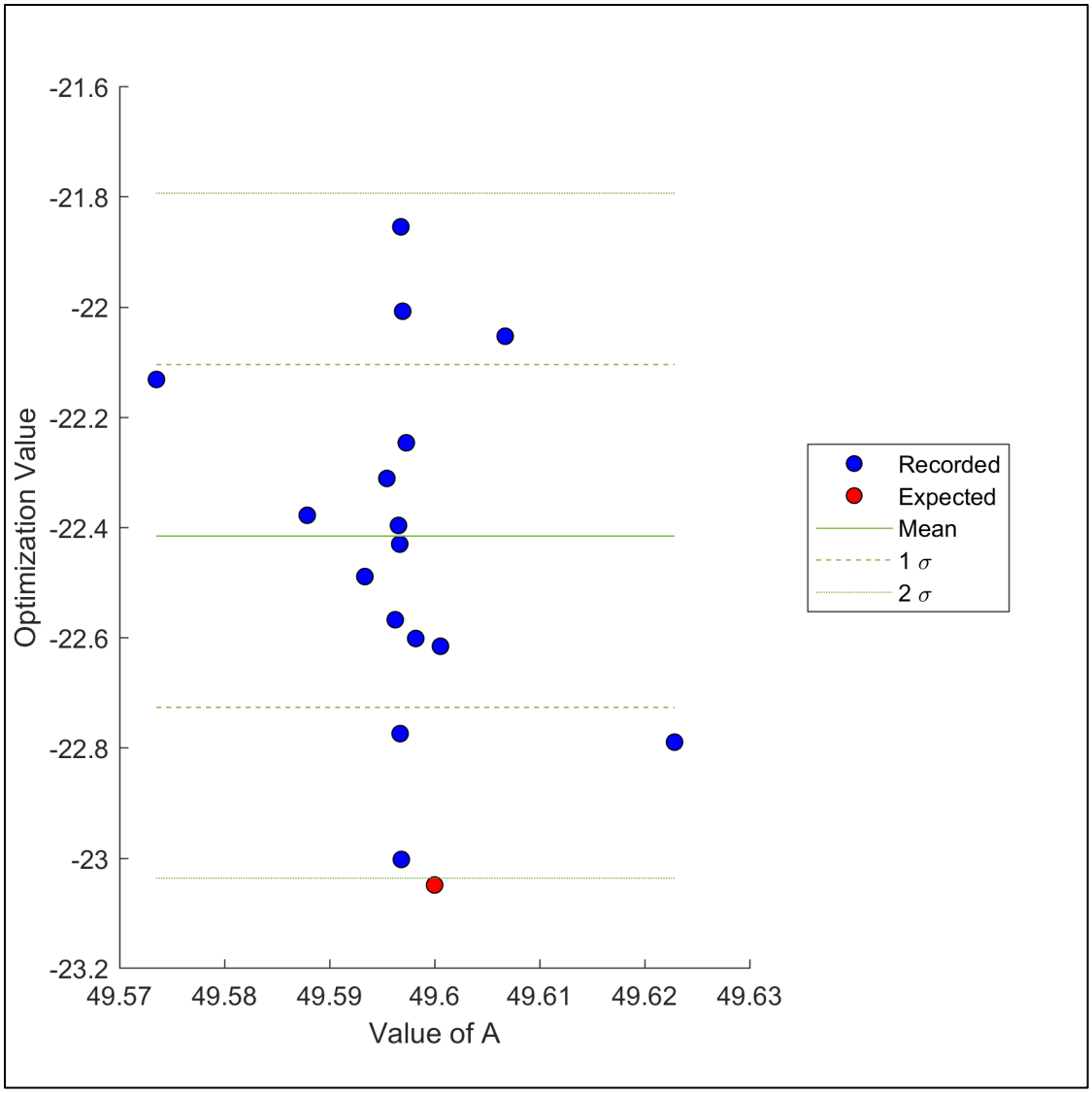


Figure 12. ZFun() Evaluations at A≈49.6

Analyzing the speedline data from the ZFun() evaluations shows the cause of erroneous ZFun() evaluation at the second iteration. Figure 13 shows the CFD solution data for mass flow rate as a function of the iteration number found during the optimizer routine. As can be seen, the second iteration had a significantly broader mass flow rate range than the other 26 iterations. The first iteration had a range consistent with iterations 3 - 26; however, it was located much higher than the later iterations. Iteration three had a smaller range than the following iterations; however, this may have been realistic because iteration

three used A=52.807 which is higher than the other iterations. The mass flow rate discrepancy is likely the result of a non-converged CFD solution.

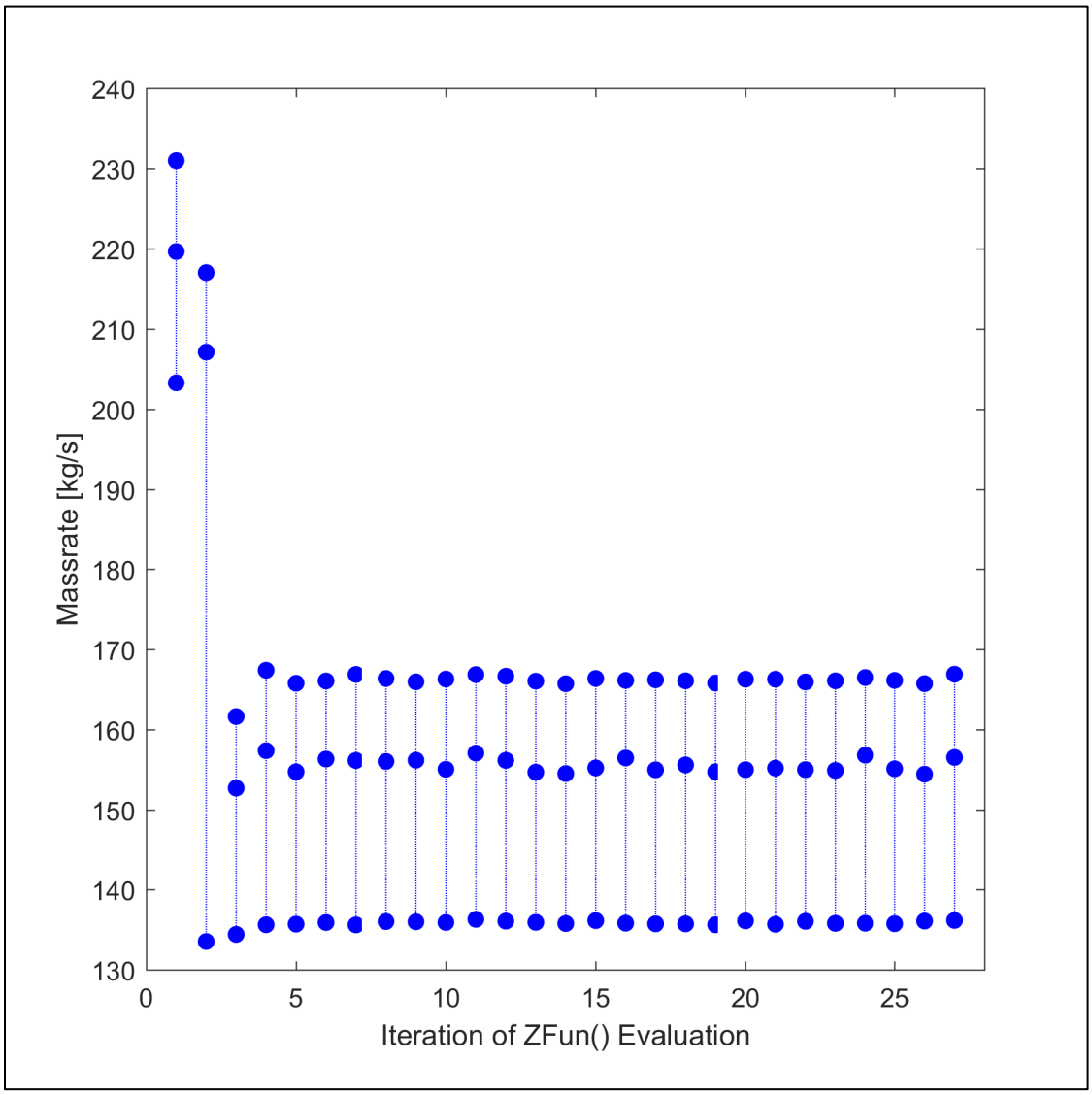


Figure 13. Massrate Range with Respect to Zfun() Evaluation

D. RESULTS REPEATABILITY

The optimization procedure was repeated for the same optimization problem to test for repeatability and in an attempt to correct the inconsistent data from ZFun() evaluations. For the second optimization execution: the CFD solution data was reset, ZFun() was

manually called for “A” values of 44 and 46, and the CFD solution was retained prior to starting the procedure. Because the CFD solver uses previous results data as an initial solution, retaining the solutions for A=44 and A=46 was expected to improve consistency and avoid the discrepancy illustrated in Figure 13.

The results for the second execution of the optimization procedure are shown in Figure 14. As can be seen, the polynomial fitting supports the optimal design at A≈36; however, a minimum optimization value of Z=-26.592 was found at A=44.403. Unfortunately, this was the first iteration of ZFun() and drove fminbnd() to seek an optimum at A≈44.4. After 22 iterations, fminbnd() determined the minimum to be at A=44.403 with Z= -23.520.

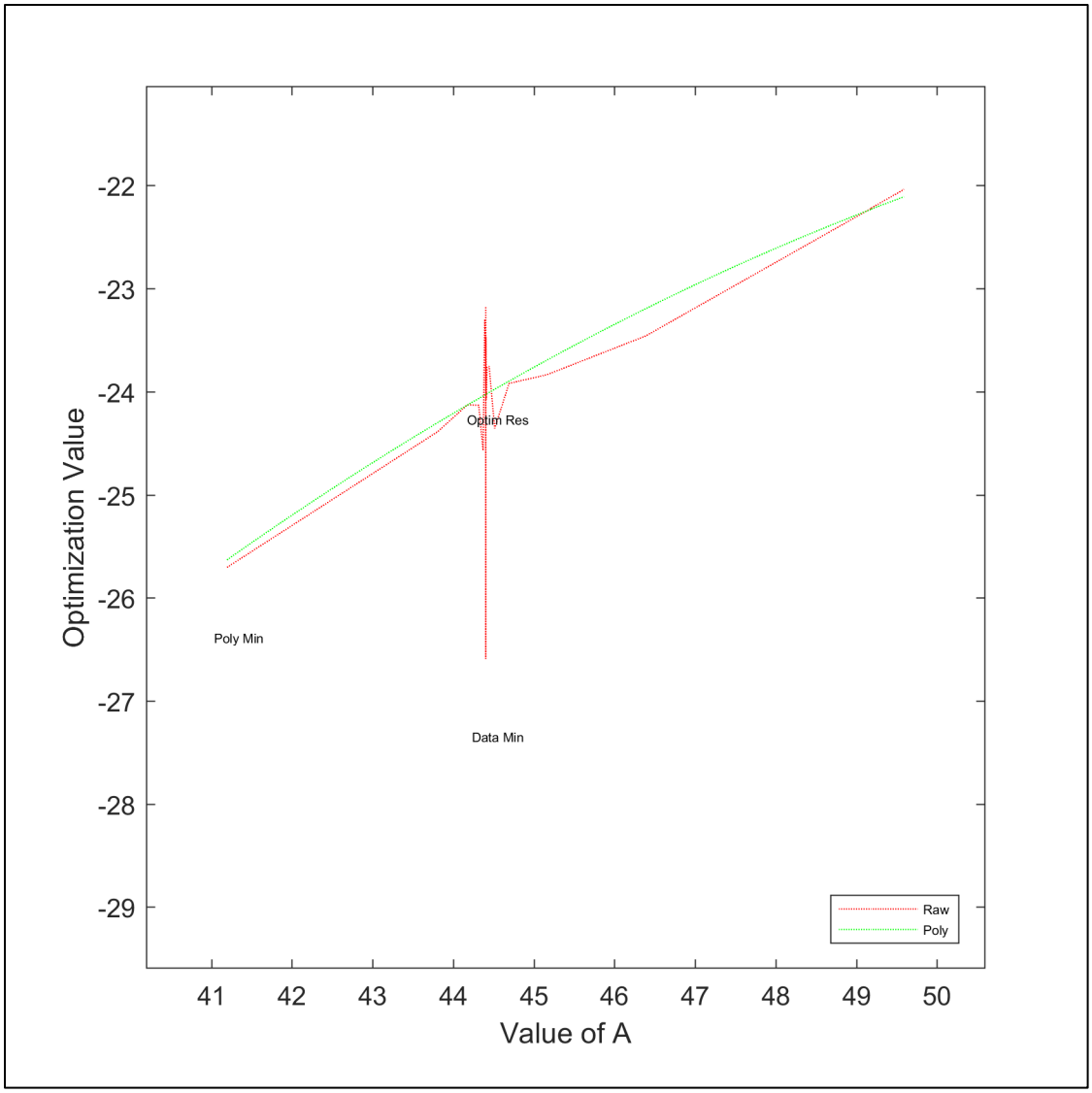


Figure 14. Results of Second Optimizer Execution

Performing the same analysis as for the first execution, the parabolic interpolation produces an expected evaluation at A=44.4 of Z=-24.4. Fminbnd() performed 17 evaluations within $^{+}/_{-}$ 0.5 of A=44.4, plotted in Figure 15. The expected value of Z=-24.4 was well within one standard deviation and the first iteration of Z=-26.6 was well outside two standard deviations. Together, the ZFun() evaluations support the process of preloading the CFD solution data; however, more than two iterations will be necessary to increase the accuracy of the resulting data set.

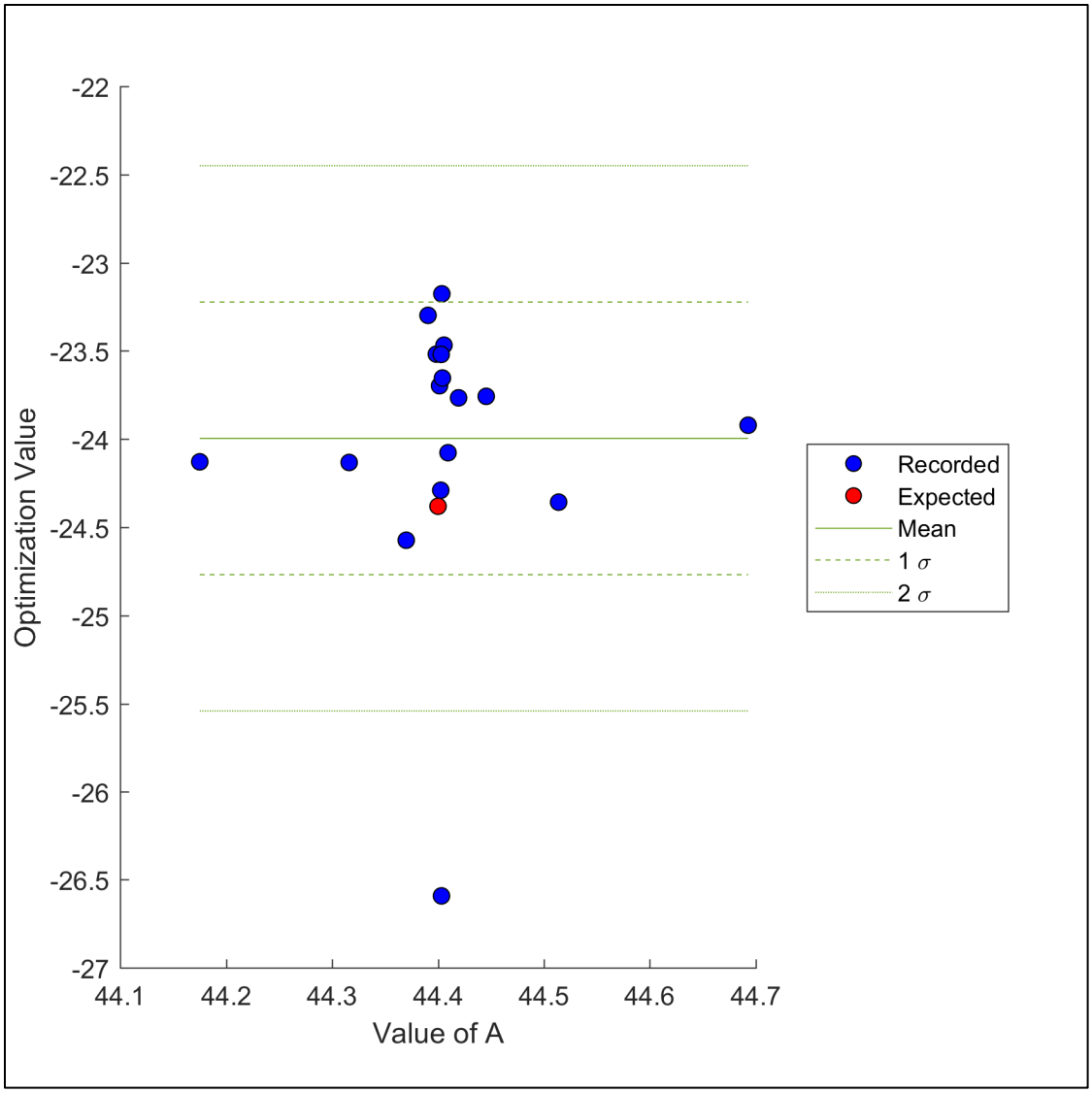


Figure 15. ZFun() Evaluations at A≈44.4

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

This study sought to implement an optimization procedure for designing compressor stages using commercial software within the previously developed design procedure [4] via four aims. The aim of increasing the degrees of freedom within the design parameters was unsuccessful because of the difficulties associated with creating the SolidWorks model. The aims of creating a holistic optimization function, parameterizing the design inputs, and creating an optimization procedure were successful. While the optimization procedure successfully found an optimal solution to the problem, the optimal design would not be suitable for manufacture because the underlying design parameters do not represent a realistic compressor. However, the framework for the optimization procedure demonstrated the necessary file transfer and function calls to allow for future development of a realistic compressor.

While the optimization procedure found an optimal solution, the inconsistent CFD data complicated using CFD as a blind source of data for the optimization function. This complication was present in two parts. First, fminbnd() has no method for screening inconsistent data received from ZFun() if the CFD solution was non-converged. Second, for converged solutions, CFD produced data with broadly varying optimization values for slightly varied values of "A." The limitation of fminbnd() from screening inconsistent CFD data prevented the procedure from finding the expected global optimum. While a CFD solution with reduced residuals may reduce the discrepancy between expected values and determined values, it would be unlikely to reduce the standard deviation of evaluations performed at nearby values for "A" to a level where fminbnd() would not find erroneous minima. Ultimately, a more robust optimization algorithm capable of filtering erroneous data and escaping local minima is necessary to identify a globally optimum design.

B. RECOMMENDATIONS

The next iteration of the NPS design procedure should replace fminbnd() with a more robust algorithm. Such an algorithm could use a Kalman filter or another method to

remove erroneous data from affecting the optimization search. Alternatively, the algorithm could use parabolic fitting for the entire data set as opposed to the nearest points to identify the global minimum. Furthermore, a new algorithm could provide the capacity for multiple DOF on bounded intervals without computing a derivative. While "A" was selected because of it produced large variations of the camber angle at the tip, it does effect the camber for all blade heights and an optimal real compressor would likely require iterations on all of the constants listed in Equation 2. After a more robust algorithm is implemented, the ultimate goal would be to use the optimization procedure to produce a novel, viable compressor that could be manufactured and tested.

APPENDIX A. MATLAB CODE

A. MAINOPTIM

MainOptim() is the MATLAB script which runs the optimization procedure.

```
%% Prepare environment
clear all %Clear variable space
diary Log_Optim.out %Begin Diary Log

%% Initialize container structure
OptimRun.a = []; %Vector of "A" values
OptimRun.z = []; %Vector of "Z" values
OptimRun.Speed = []; %Structure of speedline values
save OptimRun.mat OptimRun %Save and close
clear OptimRun

%% Define optimization parameters
fun = @(x)ZFun(x); %Optimization function
lb = 36; %Lower bound
ub = 58; %Upper bound
options = optimset('MaxFunEvals', 50); %Algorithm options

%% Call optimization algorithm
[x,fval,exitflag,output] = fminbnd(fun,lb,ub,options)

%% Close diary
diary off
```

B. ZFUN

ZFUN() is the MATLAB function which quantifies the value of a compressor design.

```
function [Value] = ZFun(Iteration)
%ZFun is the function used by fminbnd to quantify design value
% It receives the iterative "A" value and returns the "Z" optimization
% value

%% Prepare environment
fprintf("\nBeginning design iteration.\nA value: %.3f \n", Iteration)

%DesignParam is a MATLAB structure that holds all of the design parameters
%used to parametrically define blade section profiles
clear DesignParam
load('DesignParam.mat');
DesignParam.A = Iteration;
save DesignParam.mat DesignParam;

%% Create compressor CAD Model
```

```
Blade3pt(DesignParam);
Passage();
GeomGen();

%% Run CFD
DataGenerator();

%% Calculate the "Z" Value
%RunContainer is used by DataGenerator() to hold the individual CFD
%solution values
load('RunContainer');
temp      =      [RunContainer.MassFlow,      RunContainer.PressureRatio,
RunContainer.Efficiency];
temp = sortrows(temp);
%Perform trapezoidal integration. Value is the optimization value of the
%design
Value = -1 * trapz(temp(:,1), (temp(:,2) .* temp(:,3)));
fprintf("\nCompleted design iteration. \nZ value: %.12f  \n", Value)

%% Save the iterative values to the master container
load OptimRun.mat
OptimRun.a=cat(1,OptimRun.a, Iteration);
OptimRun.z=cat(1,OptimRun.z, Value);
OptimRun.Speed = cat(1, OptimRun.Speed, RunContainer);
save OptimRun.mat OptimRun
clear OptimRun
end
```

C. CAMBER FUNCTION

StaggerFun() is the sub-function which was added to Blade3pt to parametrically define the camber angle for the blade section.

```
function Blade = StaggerFun(Blade, DesignParam)
%StaggerFun defines the camber angle across the blade.
% It receives the Blade structure and the DesignParam structure and
returns
% the Blade structure with the camber matrices
for j=1:Blade.CtrlHts %For each section height
    for i=1:Blade.CtrlPts %For each chord point
        %Save the camber angle for the main blade. Main blade is
designated
        %by the "1" in the third index.
        Blade.Stagger(j,i,1) = ...
            DesignParam.A*(Blade.Heights(j)^2)*(Blade.Controls(i)^2) ...
            + DesignParam.B*Blade.Controls(i) ...
            + DesignParam.C*(Blade.Controls(i)^2) ...
            + DesignParam.D*Blade.Heights(j) ...
            + DesignParam.E*(Blade.Heights(j)^2) ...
            + DesignParam.F;
    end %Chord Point
end %Section height
```

D. DATAGENERATOR

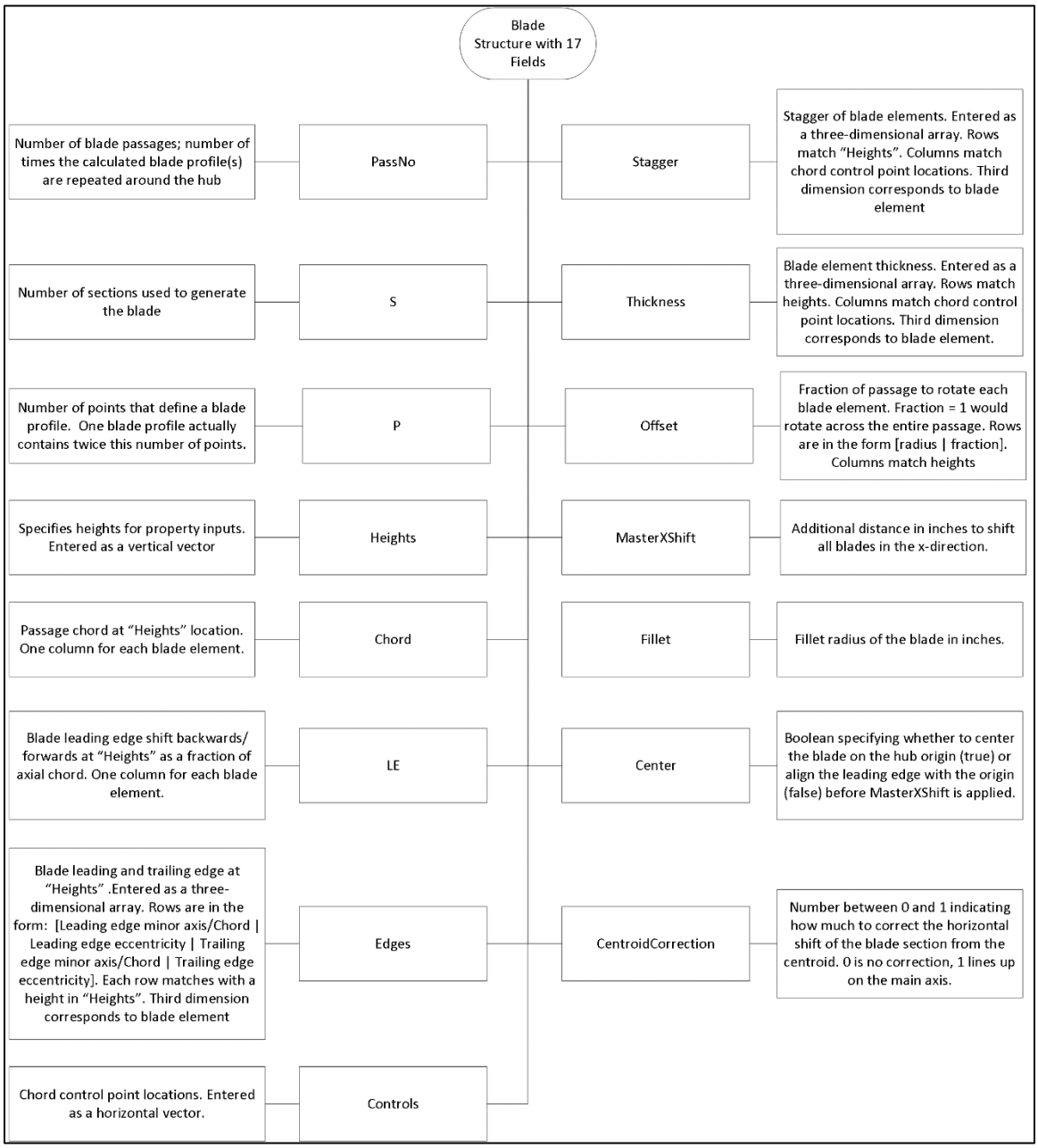
DataGenerator() is the MATLAB script which performs the CFD analysis on the compressor.

```
function RunContainer = DataGenerator()
%Matlab script to create the data points for optimization

%%Prepare the environment
clear all
close all
fprintf("\nGenerating optimization data points\n")

%% Define the working path
filePath = [pwd '\'];
% File location of ANSYS
FileProgLoc    = ...
    '"C:\Program Files\ANSYS Inc\v181\Framework\bin\Win64\RunWB2"';
% Location of template Workbench project
FileProjLoc    = ['"' pwd '\WorkingProject.wbpj' '"'];

%% Define the CFD parameters
% V = 343 [m/s] = 20,580 [m/min], r = 1.0 [m]
% W = 0.5*(V/r) = 10,290 [Rad/min] = 1637.7 [Rot/min]
%Rotor RPM (Negative for our particular orientation)
AngularVelocity = -1638;
%Number of passages (so 12 for current 24 bladded splittered rotor)
PassNo          = 12;
P_min           = .00; %Min Pressure
P_max           = .02; %Max pressure
Del_Atm         = .01; %Pressure step
Runs            = 1+floor(P_max/Del_Atm); %Number of runs

%% Intialize container for CFD data
RunContainer = struct('PressureRatio', zeros(Runs, 1), ...
    'Efficiency', zeros(Runs, 1), 'MassFlow', zeros(Runs, 1));

%% Perform CFD
for i = 0:(Runs-1) %For the number of runs
    %If this is the first run for the compressor, use the python script
    % which runs CFD with mesh updating
    if i == 0
        FileScriptLoc = ['"' pwd '\UpdateProject.py' '"'];
    %If this is not the first run, run CFD with the existing mesh
    else
        FileScriptLoc = ['"' pwd '\UpdateProject_NoMesh.py' '"'];
    end
    %Define the outlet pressure [atm]
    OutletPressure  = P_min + i*Del_Atm;

    % Input parameters to adjust RPM and outlet pressure are written to
a
    % .dat text file which is read by ANSYS
```

```
    try
        file = fopen([filePath 'InputParams.dat'], 'wt');
        fprintf(file, '%f\n%f', OutletPressure, AngularVelocity);
        % Only catch exceptions so that the file can properly be closed
if
        % there is an error
    catch err
        fclose(file);
        rethrow(err)
    end
    fclose(file);

    %Display status
    index=i+1;
    fprintf("\nStarting run %i of %i .\n", index, Runs)
    fprintf("Back pressure is %.3f atm.\n", OutletPressure)

    % Runs the ANSYS Workbench script
    eval(['! ' FileProgLoc ' -F ' FileProjLoc ' -R ' FileScriptLoc ' -
X'])

    pause(5) %Time for system processes to complete
    fprintf("\nCompleted run %i of %i .\n", index, Runs)

    %% Read the ANSYS output and save to the run container.
    Outputs = ReadAnsysData([filePath 'SavedOutput.dat']);
    RunContainer.PressureRatio(index) = ...
        Outputs.pTotalOut / Outputs.pTotalIn;
    RunContainer.Efficiency(index)    = Outputs.effTT;
    RunContainer.MassFlow(index)      = Outputs.mFlowOut * PassNo;
    clear outputs;
    save RunContainer.mat RunContainer
    fprintf("\nRunContainer index %i updated\n", index)
end %CFD runs

%Display status
fprintf("\nData points generated\n")
end
```

APPENDIX B. OVERVIEW OF BLADE DESIGN PARAMETERS

Blade3pt is a MATLAB script that produces the data structure used subsequently to define the blade section profiles. The data structure has 17 parameters, as described below.



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. CODE REPOSITORY

The original code used for this study is maintained at the Turbopropulsion Laboratory at NPS. To access it, contact Dr. Anthony Gannon at ajgannon@nps.edu.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] A. D. Belegundu and T. R. Chandrupatla, O*ptimization Concepts and Applications in Engineering,* Cambridge, United Kingdon: Cambridge University Press, 2011.

[2] P. G. Hill and C. R. Peterson, *Mechanics and Thermodynamics of Propulsion*, Reading: Addison-Wesley, 1992.

[3] M. Schnoes, C. Vos, and E. Nicke, "Design optimization of a multi-stage axial compressor using throughflow and a database of optimal airfoils," *J. Gloal Power Propulsion Soc.,* vol. 2, pp. 516-528, Oct, 2018.

[4] S. Drayton, "Design, Test, and Evaluation of a Transonic Axial Compressor Rotor with Splitter Blades," Ph. D. dissertation, Dept. of Mech. & Aero. Eng., NPS, Monterey, CA, USA, 2013. [Online]. Available: https://calhoun.nps.edu/handle/10945/37616.

[5] E. Benini, "Optimal Navier-Stokes Design of Compressor Impellers Using Evolutionary Computation," *Int. J. of Computational Fluid Dynamics ,* vol. 14, no. 5, pp. 357-369, Oct, 2003.

[6] U. M. Ascher and C. Greif, *A First Course in Numerical Methods*, Philadelphia: Society for Industrial and Applied Mathematics, 2011.

[7] N. L. Sanger, "Design Methodology for the NPS Transonic Compressor," *TPL Technical Note 99-01,* unpublished.

[8] R. P. Brent, *Algorithms for Minimization Without Derivatives*, Englewood Cliffs: Prentice-Hall, 1973.

[9] N. L. Sanger, "Design of a Low Aspect Ratio Transonic Compressor Stage Using CFD Techniques," *J. of Turbomachinery,* vol. 118, pp. 479-491, Jul, 1996.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California