EXAMENSARBETE INOM VEHICLE ENGINEERING,
AVANCERAD NIVÅ, 30 HP

STOCKHOLM, SVERIGE 2021

Development of a Tool for Inverse Aerodynamic Design and Optimisation of Turbomachinery Aerofoils

Utveckling av ett verktyg för invers aerodynamisk design och optimering av vingprofiler för turbomaskiner

BERKIN KURTULUS

KTH
SCHOOL OF ENGINEERING SCIENCES

Abstract

The automation of airfoil design process is an ongoing effort within the field of turbo-machinery design, with significant focus on developing new reliable and consistent methods that can meet the needs of the engineers. A wide variety of approaches has been in use for inverse airfoil design process which benefit from theoretical inverse design, statistical methods, empirical discoveries and many other ways to solve the design problem. This thesis work also develops a tool in Python to be used in airfoil aerodynamic design process that is simple, fast and accurate enough for initial design of turbo-machinery blades with focus on turbine airfoils used for operation in aircraft engines. To convey the decision-making process during development a simplified case is presented. The underlying considerations are discussed. Other available methods in the literature used for similar problems, are also evaluated and compared to demonstrate the advantages and limitations of the methods used within the tool. The inverse design problem is formulated as a multi-objective optimization problem to handle various different objectives that are relevant for aerodynamic design of turbo-machinery airfoils. Test runs are made and the results are discussed to assess how robust the tool is and how the current capabilities can be modified or extended. After the development process, the tool is verified to be a suitable option for real-life design optimization tasks and can be used as a building block for a much more comprehensive tool that may be developed in the future.

Sammanfattning

Automatisering av processen för design av vingprofiler kräver fortlöpande insatser inom området turbomaskindesign, med stort fokus på att utveckla nya tillförlitliga och konsekventa metoder som kan tillgodose ingenjörernas behov. Ett stort antal olika tillvägagångssätt har provats för omvänd design av vingprofiler såsom teoretisk invers design, statistiska metoder, empiriska upptäckter och många andra sätt att lösa designproblemet. Detta avhandlingsarbete är också ett lyckat försök att utveckla ett verktyg i Python som ska användas i den aerodynamiska designprocessen; det är enkelt, snabbt och noggrant för den initiala designen av turbomaskinblad med fokus på turbinblad som för användning i flygmotorer. För att förmedla beslutsprocessen under utvecklingen presenteras ett förenklat fall. De underliggande övervägandena diskuteras. Andra tillgängliga metoder i litteraturen som används för liknande problem utvärderas och jämförs för att visa fördelarna och begränsningarna med de metoder som används i verktyget. Det omvända designproblemet formuleras som ett multi-objektivt optimeringsproblem för att hantera olika mål som är relevanta för aerodynamisk design av turbomaskiner. Testkörningar görs och resultaten diskuteras för att bedöma hur robust verktyget är och hur de nuvarande funktionerna kan modifieras eller utökas. Efter utvecklingsprocessen verifieras verktyget som ett lämpligt alternativ för verkliga designoptimeringsuppgifter och kan användas som en byggsten för ett mycket mer omfattande verktyg som kan utvecklas i framtiden.

Acknowledgements

I want to express my gratitude toward my supervisors Eric Blidmark and Pieter Groth, at GKN Aerospace Sweden AB, for their continued assitance during my project. Also, I want to acknowledge the fact that GKN and Rotors department gave me the opportunity to work with them and have an accelerated learning experience within the industry as a candidate engineer. I want to thank everyone for broadening my perspective, not just technically, but also culturally.

For his support with my personal problems and academic guidance, I want to thank my supervisor at KTH, Prof. Raffaello Mariani. His efforts and support through tough times allowed me to focus on my thesis work.

I also want to acknowledge my parents for their everlasting support through many problems I faced during my Master's Degree. Without them, I would be living a very different life.

I also want to commemorate my friend Tunahan, the passing of whom, has altered how I live my life on a fundamental level ever since. I want to express my condolences to everyone who lives with the happy memories of lost ones and I hope that we will share a world where we can practice understanding and hospitality towards each other.

Contents

Nomenclature

χ	Geometric Design Parameter
$\delta*$	Displacement Thickness
$\mathbf{X}$	Vector of Optimization Input Parameters
$\mathbf{Y}$	Vector of Optimization Output Responses
$\text{gridpar}_{\text{i, j}}$	Grid parameter in i'th line at j'th position
θ	Momentum Thickness
c_P	Coefficient of pressure
D_{SS}	Diffusion rate at suction side
l	Length
P_0	Stagnation Pressure
X, X_i	Optimization Input Parameter
Y, Y_i	Optimization Output Response
2D	Two Dimensional
3D	Three Dimensional
CAD	Computer Aided Design
CAE	Computer Aided Engineering
Cf	Skin Friction Coefficient
CFD	Computational Fluid Dynamics
DoE	Design-of-experiments
EA	Evolutionary Algorithm
H	Shape Factor
Ma, M	Mach Number
PS, b	Pressure Side, bottom surface for turbine airfoils
s	Surface Arc Length
S2deg	Outlet Flow angle
sNorm	Normalized Surface Arc Length
SS, t	Suction Side, top surface for turbine airfoils

Chapter 1

Introduction

Similar to almost all other aerospace components, aero-engines and rocket engines are required to be more efficient, more sustainable, less costly and more reliable then their predecessors. For a given engine architecture, one way of improving the engine performance is to improve the designs of turbine and compressor components. For turbines, fans and compressors, the aerofoils are crucial components for the operation of these turbomachines and the improvement of aerofoil design directly relates to the performance of the component. The design process of aerofoils therefore requires attention to detail, which involves many aerodynamic, thermodynamic and structural considerations. Having an optimum blade design is the design philosophy in most cases, but traditionally this philosophy is realized by adjusting the geometry of the blade and then analyzing the blade to understand if the design is satisfactory and iterate over this process to find a better aerofoil. As one can imagine, this approach requires many iterations, time and know-how to find an airfoil that meets the constraints, and reaches an optimum for set objectives.

In recent years, to address this cumbersome process, many different techniques have emerged to handle the design process automatically, and with minimal designer interaction. Some of them involve inverse mathematical formulation of boundary layer equations [1], or Navier-Stokes equations with different end-wall boundary conditions [2]. Other methods, such as adjoint-based methods, involve a more complex formulation to solve the flow equations coupled with global optima search, like the discrete adjoint method[3]. However, a method that is simple enough for an initial aerodynamic design but comprehensive enough to be used in actual aircraft engines, is either commercially licensed as off-the-shelf codes which require extra effort to integrate into existing work-flow, or it is developed in-house by OEMs and component-designing

companies, which do not release the software for commercial use.

1.1 Research Question

The objective of this project is to develop a software/tool, that can be used for aerodynamic design of turbo-machinery aerofoils. More specifically, the user of the tool should be able to set a target pressure distribution around an airfoil alongside certain constraints, and the tool will adjust the airfoil geometry, in an attempt to achieve the given pressure distribution without violating the constraints. Furthermore, the aerofoil geometry should be optimized towards other objective functions. The design should focus on aerodynamic performance constraints and objectives, but possible other objective functions and constraints should also be inspected. The main goal is to construct an optimization loop as the tool which bridges the gap between aerofoil geometry generation tools (both in-house and commercially available), boundary layer analysis, CFD studies and result visualization. The milestones for the planned project is as follows:

- Literature review for suitable methods
- Familiarization with the available tools that will be integrated
- Implementing a way to change the airfoil geometry in the geometry generator tool
- Implementing a link between geometry generator and aerodynamic flow analysis
- Construction of an algorithm, as a loop, to achieve specified distributions at a single 2D section
- Possibly adding constraints for span-wise smoothness of aerodynamic and geometric parameters
- Possibly constructing of the design loop for 3D blade optimization towards aerodynamic objective functions

It should be noted that dealing with span-wise constraints and objectives is relevant specifically for optimizing the 3D blade design by using information generated at different streamlines simultaneously. These milestones are set as stretched targets and are not part of the mandatory work. It is attempted to integrate relevant functionality to the tool, but the fully automated capability is left as future work.

1.2 Context in Turbomachinery Design

In order to explain where the project fits in the actual engine design process, consider a simplified work-flow for engine design.

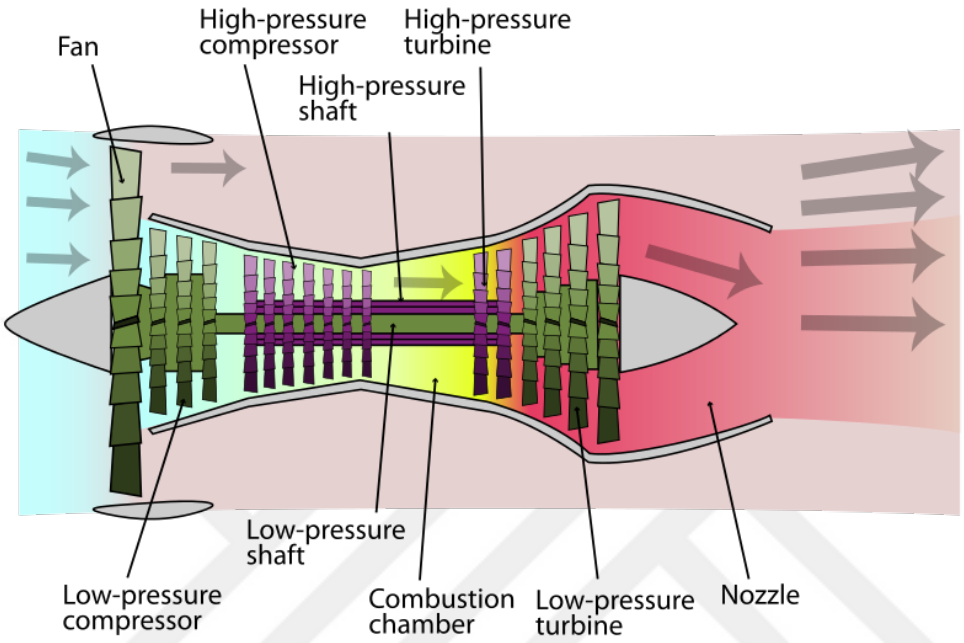


Figure 1.1: Simplified Operation of A Turbofan Engine [4]

In Figure 1.1, one can see the main components of a jet engine which includes the fan, two compressors, and two turbines. Initially, a performance analysis is done, so that the inlet and outlet conditions for each module is determined from engine performance requirements. This step treats each subcomponent as a black box, so that each sub-component should yield those conditions once their detailed design is finished. Once the inlet and exit conditions are determined, the components are subjected to a "through-flow / streamline" analysis. This is done to actually resolve the "black-box" and resolve the boundary conditions across each subcomponent. For rotary components like the turbines, compressors and fans, through-flow analysis computes flow variables along each streamline, for the inlet and exit location across each blade row in each stage. Therefore, after a successful turbine through-flow analysis, each blade row, whether it is a stator or rotor, can be designed individually. One can see this simplified process for a turbine, which is the main focus of this project, visualized as a flowchart as in Figure 1.2.

The blade can be designed with multiple approaches. One can consider

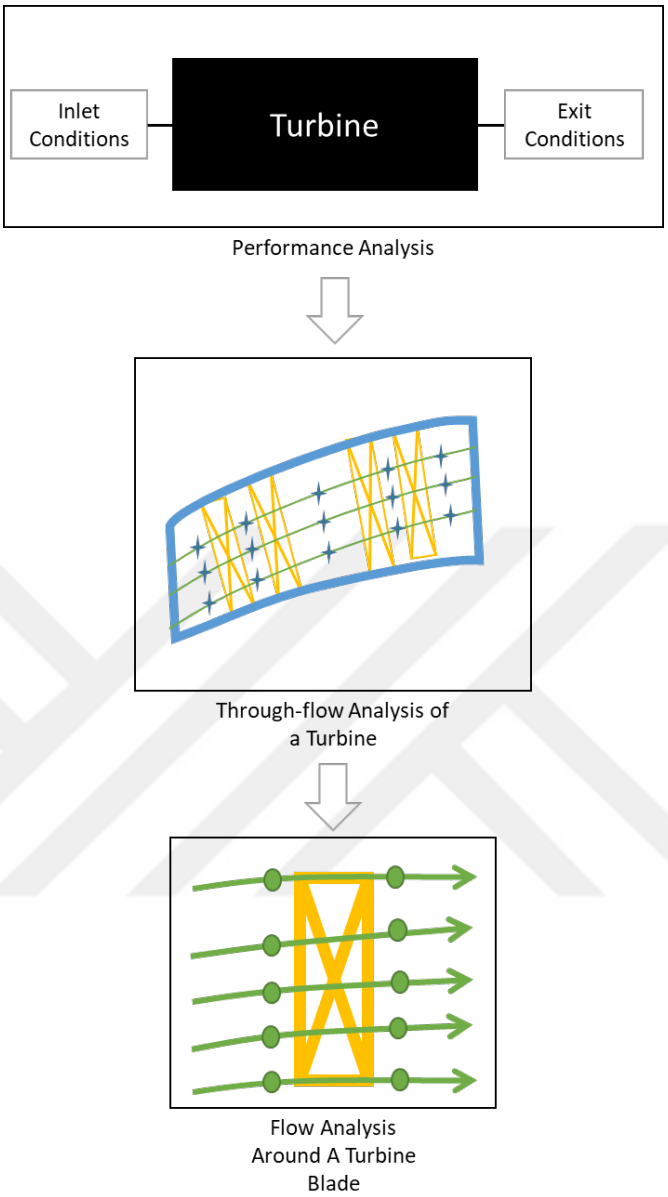


Figure 1.2: Simplified Design Process of A Turbine Blade

the full 3D blade shape, and perform 3D CFD calculations and use its results to form the outer geometry of a 3D blade. Another approach, would be to design specific 2D sections of the blade. Then, those 2D sections will be used to construct a 3D blade geometry, which can then be used for 3D flow analysis. A designer can also prefer to do 2D flow analysis at specific streamlines around the blade. Although fidelity will be lost for multi-dimensional effects, this way

the analysis will possibly have higher detail for boundary layer growth, and it will be cheaper in terms of computational cost.

It is clear that there are a multitude of possible options, and choosing which method to use, is not a straight-forward but an essential part of the design process. This master's degree project aims to present a solution to this problem, and it focuses on the development of the computational tool to enable its user to find an optimum 3D blade design by automating the design of 2D sections to meet user defined aerodynamic criteria.

Chapter 2

Background

2.1 Inverse Design

The inverse design problem is a fundamental problem that occurs in many different fields. For aerodynamics, the inverse problem generally refers to the methods of determining an aerodynamic configuration or shape that is determined to optimize certain aerodynamic objectives without violating certain aerodynamic constraints which will also satisfy the governing flow equations [5]. Conventionally, the aerodynamic design is realized by updating a reference design by considering the CFD results and experience, and iterating through this process. However, this process requires engineer's intervention and judgment skills to approximate a desired design, and it can fail in comprehensive design cases where there are many inter-dependent problems that should be addressed simultaneously. On the contrary, inverse design process enforces a target surface distribution, therefore inverse design methods require mathematically formulated solution procedures that will use the distribution information as an input to compute the aerodynamic shape. In simpler terms, the direct design is about updating the airfoil to reach a desired aerodynamic behavior, whereas inverse design is about determining a framework that can consistently determine the aerodynamic shape. One can also see the simplified relation as a summary in Figure 2.1.

For inverse design process in turbo-machinery, generally the blades of turbo-machine components are designed as a three-dimensional configuration by including the influence of adjacent blades and the finite blade length. Therefore, there are many flow-field and geometric considerations like energy losses, pressure drop, desired turning angle of the flow, turbulent effects, all of which may be included in a tool that is designed for doing airfoil inverse design com-

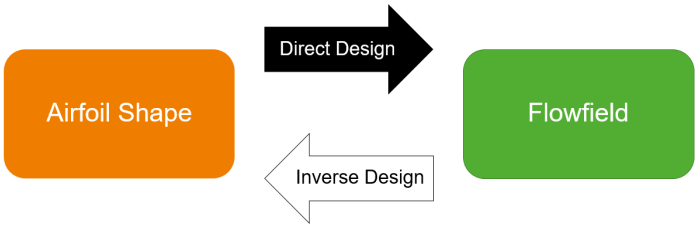


Figure 2.1: Simplified Representation of Aerodynamic Inverse Design

putations for turbomachines.

Such tools are most commonly based either on theoretical inverse design by reformulating flow-field equations [6] to determine the aerodynamic shape by forcing boundary conditions and pressure distributions, or based on formulation as an optimization problem for finding extrema of certain objective functions relevant to turbo-machine airfoils.

2.2 Simplification of Underlying Considerations within Project

Since the project is a software development project, and for a very technical case, in order to communicate the aims of project, the author would like to simplify the tasks with an example demonstration. For ease of understanding the following sections, consider the following analogous cases first.

2.2.1 Blade Geometry

First, consider a blade, that is identically shaped like a right cylinder, as in Figure 2.2. This cylinder is geometrically made up of two-dimensional circles. One can think of the base of the cylinder as the hub of the blade, the roof of the cylinder as the tip of the blade and the half-way circle as the midspan. The sections that are mentioned are the three "designed sections" for the blade. The blade shape, i.e. the "cylinder", is then formed by "pulling the circles towards each other" along the right angled line, which is analogous to the stacking-line. However, this cylinder does not necessarily have to be a right cylinder, and it can be an oblique cylinder or furthermore, a swept profile, just like in Figure 2.3. The relative positions of each circle is an independent attribute of the solid, since each designed section is still just a circle. This distinction is the difference between designed sections and the stacking line. In all of

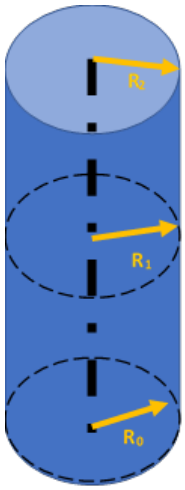


Figure 2.2: Right Cylinderical Sample Blade

these examples, the stacking occurs at the centroid of the circles, but another specifically defined location for all circles can be used to form the solid blade shape, like the points on each section that comprise the trailing edge of the blade. Also, one should note that, since each designed section is just a circle, the only parameter that controls each section is the radius of that circle.

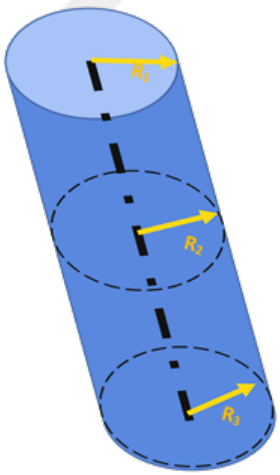


Figure 2.3: Oblique Cylinderical Sample Blade

Now, one might consider an elliptic prism, as in Figure 2.4. Instead of having circles as the designed sections, each ellipse is controlling the solid shape

now. So for each designed section, one has to know the semi-major and semi-minor axis length of the ellipses, so 2 parameters for each designed section, to form the solid body. This increase of parameters that control each designed section, is what the geometry parametrization is about. In these sample cases, the cylindrical shapes involve "circle" as their parametrization type, and the elliptic prism involves "ellipse" as its parametrization type. If there is a need to have a more intricate shape like an airfoil shape, it is highly probable that the number of parameters have to be more.

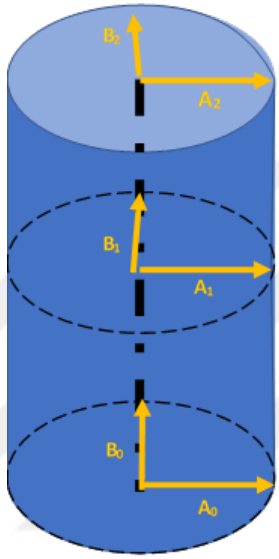


Figure 2.4: Sample Blade like an Elliptic Prism

In real life scenarios, the choice of the parametrization directly affects which airfoil shapes can be attained, and parametrizations that have much more different parameters are used. For instance, a parametrization that is widely available in the literature, PARSEC parametrization[7], has 11 basic parameters. In this particular parametrization, each parameter controls a physically interpretable part of the airfoil surface. However, in literature there are many other suggested parametrizations, some of which are not as readily intuitive. Some of the options will be discussed further in subsection 2.3.1.

2.2.2 Flow Study

In the previous section, the discussion was about how blade geometry is generated. The discussion in this section will be about how the blade can be analyzed and what kind of objectives may be relevant to an aerodynamic blade

design process.

Consider, a 3D aerofoil geometry, like the one in Figure 2.4. One can analyze the fluid flow around this aerofoil with different methods. One method would be to consider the full 3D behavior of fluid flow around the blade shape. However, this may be a very time-consuming activity, especially if the blade shape is subject to changes during the design process. Then, for every new blade shape, the flow study would have to be performed again to verify the aerodynamic design. Another approach would be doing a flow study on particular streamlines. Each chosen streamline can be used to do flow analysis around a 2D airfoil section that is formed by the intersection of the streamline and the blade. Whatever the choice is, it should be possible to represent the flow around certain sections of the blade, as a distribution of pressure coefficient or a Mach number distribution on the surface of the blade.

One should be aware of the fact that flow solver settings are directly influencing the computed results. And since those results would be used for evaluating how "good" the design is, the flow solver parameters may be considered as meta-parameters and they can be treated as additional parameters for the design. One such example parameter is one that controls the grid spacing. If there are more points in the mesh for the CFD analysis, then the results are thought to be of higher quality. This is because of the fact that, a finer mesh has the potential to capture different flow phenomena to a better accuracy. One such phenomenon is the boundary layer growth and laminar-to-turbulent transition. If the flow solver parameter that controls the grid spacing changes, it is very likely that the results for the aerofoil surfaces and the boundary layer parameters will be different. Therefore, the inclusion and capability to control flow solver settings is also relevant for inverse design, although it is also possible to assign all of these parameters beforehand as constants.

2.2.3 Objectives and Constraints

The design task for the blade must involve objectives and constraints that are numerically quantifiable. One can consider many different constraints and objectives, like a limit for exit flow angles or maximum allowed variation of flow variables along the span of the blade. One should note that, the nature of these objectives and constraints are very much dependent on how the flow analysis is done and what kind of results can be obtained from the analysis. For a 2D flow study case, constraints related to three-dimensional secondary flows like tip leakage or hub boundary layer cannot be considered. In a 3D flow solution, it comes down to post-processing efforts to set objectives and constraints on

a single chosen section. In any case, for the blade, the geometric design tasks should be quantifiable with different metrics, where the metrics will be used to determine how "good" the design is. For example, if achieving a specific outlet Mach number is an objective, one can quantify this as the minimization of discrepancy between the desired outlet Mach number and the actual outlet Mach number obtained from the flow study. Many other possible metrics, namely the objectives and constraints and their nature, will be discussed in more detail, in section 3.6.

2.2.4 Design Optimization

The inverse aerodynamic design process is about searching for better designs which are expected to achieve the requirements set by the designer. This inverse design process can actually be formulated as a design optimization problem. Then solving the inverse design problem is equivalent to solving the optimization problem and finding the "ideal" blade designs.

To find an "ideal" blade shape, certain established metrics must be used. The nature of these metrics in aerodynamic design optimization was briefly discussed in subsection 2.2.3. Consider the algorithm that finds the best possible designs, as a black-box operation for now. The aerodynamic turbomachinery airfoil design optimization process involves finding the best design parameters that is used within the parametrization for the designed sections to generate the 3D blade. The design parameters are best, when the resultant blade that they yield, gives the "best" metrics that represent / quantify the design tasks. So, for a set of input variables, a set of output variables are to be calculated. To simply represent this, one can express the mathematical relation as Equation 2.1

$$f(\mathbf{X}) = \mathbf{Y} \tag{2.1}$$

In this equation $(\mathbf{X})$ is the input vector, which comprises geometrical design parameters and additional metaparameters, and $(\mathbf{Y})$ comprises the responses for that design, which involve all the objective function values, as well as the values of the constraints. The operation f is an implicit function that represents all the steps that are necessary to convert input parameters into output variables.

To automate the inverse design process as a design optimization problem:

- A multitude of designs must be generated with different input parameter vectors $(\mathbf{X})$

- Operations that are represented as f have to be performed automatically for each design that is a candidate for the "best" possible design.
- The responses ($\mathbf{Y}$) must be calculated or obtained for each design case.
- A relation between input parameters and outputs must be formed by some means.

To find optimum designs by mathematical means, one has to consider the mathematical representation of the optimization problem:

$$\begin{aligned} \min_{\mathbf{X}} \quad & Y_1(\mathbf{X}), Y_2(\mathbf{X}), Y_3(\mathbf{X}), ..., Y_{N_{\text{objectives}}}(\mathbf{X}) \\ \text{s.t.} \quad & Y_j(\mathbf{X}) \leq \mathbb{C}_j,\ j \in \{k + N_{\text{objectives}} : k \in \{1, 2, 3, ..., N_{\text{constraints}}\}\} \end{aligned} \tag{2.2}$$

The solution of the mathematical problem in Equation 2.2, will give the best designs, more specifically the input parameter vectors $\mathbf{X}$ for the best designs. After all of these steps are performed, the best resultant designs will have been found by the optimizer, and the design optimization process will be complete. Note that the optimizer is the agent that establishes the relation between input parameters and outputs, and uses that relation to pick out best designs. A more in-depth discussion of how the optimizer establishes this relation, is available in section 3.7.2.

In real life scenarios, all of the objective functions need not be expressed as a minimization and not all constraints need to be expressed as lesser than or equal to a constraint value of $\mathbb{C}_j$. The problem can be mathematically reformulated, and by keeping track of this property of the objectives and constraints, the multi-objective design optimization problem can be solved. It is stated as the best designs -in plural form- deliberately since it is mathematically implausible to have a single outstanding solution that outperforms all other designs for all objective functions. It is more likely to have multiple different designs on the Pareto front, each with different design parameters, that have a trade-off between optimizing different objective functions. The concept of best-designs and their mathematical evaluation, are further discussed in subsection 2.3.3.

2.3 Literature Review

In order to have aerodynamic inverse design capability, geometry parametrization, flow analysis, and design optimization methods should be investigated. This investigation is done to understand the available options and to communicate the considerations for the project.

2.3.1 Geometry Parametrization Techniques

Any blade geometry for the design task has to be designed in a CAD-based environment, so that its geometric properties and behavior can be simulated. The nature of the project task involves mathematical representation of 2D airfoil section shapes, and how they can be constructed consistently. As previously explained in section 2.2, each airfoil section has to be parametrized to enable such consistent construction. In this section, some of the parametrization techniques that were considered, will be explained.

The parametrizations which will be discussed, are considered within the scope of inverse aerodynamic design. Ideally, the aerofoil design process should involve a geometry parametrization that has the minimum possible design parameters that can cover the largest possible different aerofoils and allow for a large design space. This is particularly relevant since the inverse aerofoil design process requires a search for an unknown design and this search is realized by searching for plausible combinations of design parameters. For parametrizations with excessively many parameters, this search of parameters would obviously take longer, which increases the computational cost of inverse design.

Note that this paper will not discuss how 3D geometries are mathematically generated from two-dimensional curves, as this is handled differently by different CAD tools. However, one can imagine such construction of 3D shapes and surfaces as a result of lofting or extruding 2D sections along a spline and that spline is chosen to be the stacking line.

PARSEC

Sobieczky [7] has established an airfoil parametrization method with wing airfoils in mind, but it is a well-known method for parametrization. According to the founders of the method, this parametrization provides curvature control of the airfoil surface with specific parameters within the parametrization technique. The method, illustrated in Figure 2.5, has geometrically interpretable parameters that are readily understandable. A very specific limitation of this parametrization for the turbomachinery community, is that the highly cambered airfoils are not very suitable for this parametrization, in addition to the leading and trailing edge being fitted with a circle and not an ellipse.

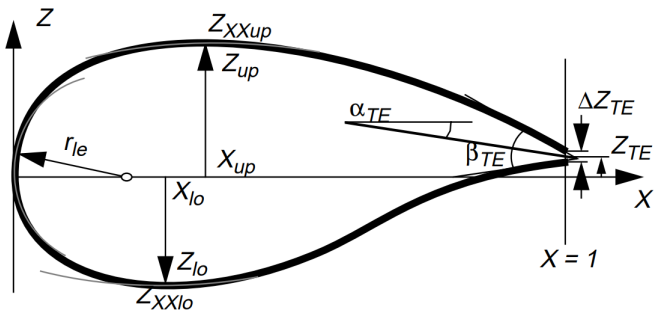


Figure 2.5: PARSEC Parametrization Technique [7]

Class-Shape Transformation

A geometry parametrization method, which is for parametrizing a class of solid body shapes, is discussed by Kulfan [8]. The two fundamental solid body classes this method can generate, are the wing airfoil type shapes and the body cross-section type shapes. The parametrization involves mathematical functions to describe the class of shapes and modify the shape of the body within that class. So as to parametrize an airfoil section, a coefficient is used to specify that the shape is an airfoil-like section, and separate shape functions (e.g. Bernstein polynomials) are used to control the surfaces of the airfoil-like shape. So, any change in the class function and any change of the shape functions will modify the solid body shape. This method yields parameters that are related to the polynomial coefficients and it enables choosing an arbitrarily high number of parameters to define the airfoil shape with the expense of having parameters whose effects cannot be interpreted intuitively from an engineering perspective.

Spline-based Methods

Forming airfoil shapes with splines in computer generated graphics and CAD modelling of surfaces is not a new idea. Rajnarayan, Ning, and Mehr [9] discusses possible advantages and disadvantages of Bézier splines, which are one of the most commonly used methods to generate consistently defined splines. The authors comment on the usefulness of B-splines, a specific class of Bézier splines, in the sense that they specifically enable "decoupled local shape variations" instead of causing global shape variations by changing a parameter, since it allows knot insertion to continuously refine the shape. However, in a research on design space coverage [10], the conclusion is that using B-spline is worse for approximating a large dataset of airfoils in comparison to Class-

shape transformation. Both methods need 25 design variables to cover 80% of all airfoils within the database they used for research. It is important to note that the geometric continuity and differentiability of the splines is an important consideration for airfoil parametrization, as these properties are highly influential on the flow characteristics of the airfoil.

Other Methods

There are many other established airfoil parametrization methods. Castonguay and Nadarajah [11] tested four methods including Hicks-Henne Bump Functions alongside PARSEC and B-splines, to test the relevance of different geometry parametrizations to match target pressure distributions for aircraft wing airfoils using Navier-Stokes equations. Other parametrization methods such as Singular Value Decomposition [10] and domain-deformative methods like Radial Basis Function Domain Element [10] or methods based on Free-Form Deformation [12] first established by Sederberg and Parry [13], are used for different purposes for research that involves airfoil parametrization. These methods use other mathematical representations like matrices or control point coordinates to construct airfoil shapes, and it is observed that these parametrization techniques are not commonly used for turbo-machine airfoil optimization.

Within theoretical inverse design, mapping-based techniques like Conformal Mapping [14], Joukowski Transformation and Inverse Theodorsen Transformation are used to enable mapping of geometric shapes into a separate coordinate system which enables mathematical ease to solve governing flow equations.

2.3.2 Aerofoil Flow Field Characterization Methods

To formulate objective functions for aerodynamic design optimization, one first has to consider which flow variables can be used to establish metrics for comparison of different designs. Since flow analysis involves analysis of turbomachinery blade cascades, there are additional considerations for aerodynamic performance of the blade.

Distributed Flow Variables

The flow analysis around a turbomachinery blade involves certain distribution of aerodynamic and thermodynamic variables within the flowfield. These

variables include pressure, coefficient of pressure, Mach number among others. Within the flowfield, specifically on the solid-fluid interfaces, additional boundary layer parameters, like displacement thickness, momentum thickness, friction coefficient, can also be used to characterize the airfoil shape. As a predecessor method, Henderson [1] established an inverse boundary layer technique that can be used to compute an airfoil contour for given boundary layer parameters, upper surface coefficient of pressure distribution and thickness requirements. Today in almost all design scenarios, airfoils are designed with boundary layer development and flow variable distributions in mind.

Flow variables, that are changing within the flowfield, are distributed along the control volume that is being studied. However, in order to make sense of these distributed flow variables, they should be expressed at certain positions within the control volume. The position of such flow variables, can be expressed at a location on a streamline or at solid boundaries, which are the blade surfaces within the concept of turbomachinery aerofoils. These positions on the airfoil surfaces can be expressed as a function of axial or meridional position, axial chord, meridional or true chord, surface arc length. The study of Goel, Cofer, and Singh [15] uses the axial distance for the turbine airfoils to define Mach number distributions. However, other representative coordinate systems can be derived to specifically non-dimensionalize or normalize the blade surfaces. This alteration is more prevalent in studies where the airfoil parametrization is done as a coordinate transformation, as in conformal mapping.

The blade can be divided into a suction and pressure side by splitting the blade surface at two stagnation points within the flow, which are the leading and trailing edge points. Then, suction and pressure side can be individually represented as the surface arc length, defined from the leading edge. These lengths are also called suction-side length and pressure-side length. These arc lengths are a consistent way of defining the distribution of flow parameters on the blade surface, since there exists only one point on each blade surface that corresponds to a chosen arc length.

Aerodynamic Performance Indicators

There are quantities that are indicative of the aerodynamic performance of the turbomachinery component and they are formulated from the overall flow behavior, instead of being defined locally. For example, inlet and exit flow angles, absolute inlet and exit Mach number numbers, viscous and shock losses, are indicators of aerodynamic performance and they can be used directly to

assess the characteristics of a blade cascade. Dennis et al. [16] have included the total pressure loss and exit flow angle as criteria for the multi-objective optimization study for subject turbomachinery cascade.

Undesired Flow Phenomena

Airfoils for turbo-machinery applications are subject to many different operating conditions, and they are designed to prevent undesired flow phenomena that causes undesired effects for the operation of the turbo-machinery component.

One such flow phenomena for a turbine blade that is particularly important for this project, is the deceleration in the flow on the suction side of the blade before reaching the peak Mach number. This deceleration is known as pre-throat diffusion, and it is an undesired phenomenon, since it creates the possibility of the flow detaching from the surface or causing early laminar-to-turbulent boundary layer transition. Both of these phenomena are known to decrease the aerodynamic performance of the blade.

2.3.3 Optimization

The inverse aerodynamic design problem can be mathematically formulated as an optimization problem and the behavior of the solutions are directly relevant to the selection of optimal designs. In this section, mathematical optimization problems will be explained within the scope of aerodynamic design optimization.

Single-objective optimization

Single-objective optimization problems are simpler in the sense that only one metric is used to assess the fitness of candidate solutions. It is also common in engineering practice to represent a multiple objective problem as a single objective function by assigning the objective function to be the weighted sum of multiple objectives. This can also be an approach in airfoil design, where all specified objectives can be automatically aggregated into a single function [6, 17, 18], but it may not be consistently scalable for many different objectives, and it requires fine-tuning of weights necessitating additional attention in an automatic design tool.

Multi-objective optimization

If an optimization problem involves minimizing / maximizing multiple objective functions at the same time, it is generally impossible to achieve optimal designs that optimize all of the objective functions at the same time. There can be correlated objectives and special cases which falsify this statement, but in simple design challenges there is generally a trade-off while optimizing multiple objectives [19].

Pareto Optimality

Pareto frontier is a fundamental concept in multi-objective optimization problems. The so-called Pareto Front is a set of optimal solutions to the multi-objective problem. It is impossible to improve an objective without penalizing any other objectives for the problem [19]. Therefore, optimal designs are the optimal solutions to the optimization problem so that an optimal design cannot be improved for any of the objective functions without sacrificing the performance for any other objective that is calculated for the design. This concept of penalization of different objective functions across different solutions, can also be seen in Figure 2.6.

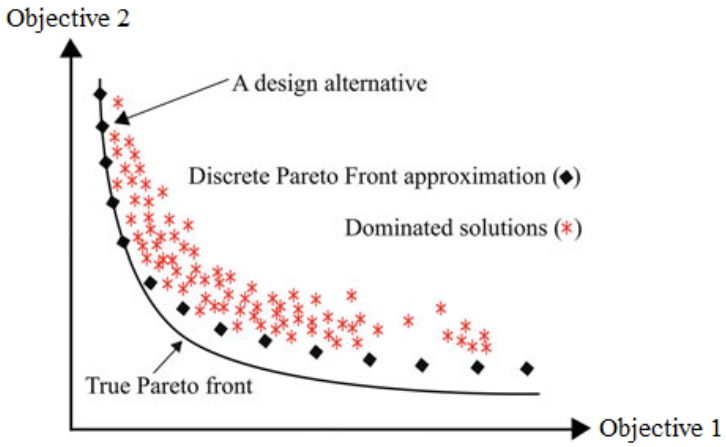


Figure 2.6: Representation of Pareto Front[20]

2.3.4 Optimization Methods

Multi-objective optimization algorithms in literature vary in scope, where the used method is chosen according to the nature of the optimization problem.

There are mathematically stable, iteratively convergent algorithms and there are heuristic algorithms that are used as alternative which can possibly find approximate solutions to the optimization problem without any guarantee for the convergence of the solution.

The iterative methods involve a mathematically formulated iterative process to compute the optimal solutions. These methods include gradient-based methods like Gradient Descent and Quasi-Newton methods.

The heuristic algorithms are more common in aerodynamic optimization problems and a wide variety of nature-inspired protocols are used to search for optimal solutions. These methods include:

- Evolutionary Algorithms
  - Genetic Algorithm (GA)
  - Evolutionary Strategies (ES)
  - Differential Evolution (DE) [21]
- Swarm Intelligence
  - Particle Swarm Optimization (PSO) [22]
  - Ant Colony Optimization (ACO) [23]
  - Artificial Bee Colony (ABC) [24]

Both of these category of algorithms are based on generating an initial set of "individuals" or "agents" that comprise the "population", and all of the "individuals" in this population are compared by calculating the objective functions and constraint values. Then these individuals are mixed, reduced and eliminated by different mechanisms, depending on the natural process the algorithm is formed after, to generate a new "generation" of population which supposedly should contain candidates for optimal individuals. For example, genetic algorithms achieve this generation process by reproduction, crossover and mutation in genes. The evolutionary strategies generate consecutive design populations by means of mutation and selection.

Within the scope of aerodynamic design, the individuals or agents in these algorithms are the designs that are realized by different parameters to construct them; and these parameters are changed as part of the algorithm for optimal design search. The algorithms have obvious differences and advantages, which is an ongoing research interest. For example, Safari and Lemu [25] compared the use of genetic algorithm and differential evolution for optimization of gas turbine blades in terms of computation cost, convergence rate and maximum

fitness and concluded that DE is noticeably less time-intensive and has better convergence characteristics, but GA provided designs that provide better fitness values for objective functions. As the continuation work, Safari and Lemu [26] compared different swarm intelligence algorithms with similar criteria.

There are other meta-heuristics that have gained popularity for aerodynamic optimization. The use of machine learning models like Artificial Neural Networks, and computer experiments as design-of-experiments (DoE) are higher level procedures to solve optimization problems. For these type of methods, a separate meta-model is established with training dataset. This model is then used to predict the optimal design solutions to the mathematical optimization problem with the aforementioned lower level heuristics.

2.3.5 Alternative Approaches for Inverse Design

For aerodynamic design optimization problems, there have been other novel ideas that have been presented as alternative frameworks and modifications to existing methods.

Embedded Multi-Fidelity Inverse Design Framework [27] is presented as an alternative to inverse design method based on parametrization of c_P profile, and using different flow solvers at different steps of the optimization framework. Specifically a low-fidelity CFD solver with inverse design capability is used to find an airfoil geometry. This airfoil is then analyzed with a high-fidelity flow solver and the optimizer is used to determine new c_P parametrization parameters until termination criteria are met.

Discrete Adjoint Method [3] is a mathematical reformulation of the inverse aerodynamic design problem to incorporate the flow equations, mesh generation and the optimization calculations into the same mathematical procedure to calculate the optimal solutions.

Progressive Parametrization [28] is originally presented as a method for aircraft wings, but is worth noting. With progressive parametrization, the airfoil shape parametrization is modified as the optimization process goes on by adding more parameters for furher controlling the airfoil surface and expanding the design space as optimization proceeds.

Chapter 3

Methods

3.1 Explanation of Available Tools

The programming language Python is chosen for all scripting and integration purposes, since it is widely available and includes many robust libraries and synthax for mathematical calculations, data manipulation, data visualization, file operations and subprocess management. Python 3.x release is used. The tool was tested on Microsoft Windows with Anaconda distribution, that has a python interpreter for Python 3.6.0 release. However, the python scripts were programmed to be compatible with UNIX-like operating systems, with special attention to generating filepaths in platform-dependent manner and avoiding symbolic links wherever possible.

Geometry generator tool is an in-house software that can form the 3D blade shape. This in-house tool is preferred as it was programmed in a way that fits into the current design work-flow. This would enable the engineer to switch between automatic design or manual adjustments in a convenient and consistent manner. The tool is called Polly, and it is specific for turbomachinery blades. It has multiple airfoil parametrizations available, each of which is better for a different engine component. For sake of convenience, this tool may also be referred to as "GeometryGenerator" in-text, so as to remark the relevant discussion to be for blade geometry generation.

For the flow analysis, the method to analyze 2D airfoil sections (enclosed by splines that are formed by the intersection of stream-plane and the 3D blade shape) is chosen. In particular, the commercially available flow solver MISES v.2.68, developed by Drela and his colleagues [29], is used. This software is chosen for its ease of use in batch mode, tolerable computational cost, and its capability for boundary layer analysis on a blade cascade. It is a solver that

couples Euler equations with boundary layer equations and is specifically used for blade cascade design.

For optimization tasks and statistical metamodelling, another commercially available software optiSLang®v.7.4.0 by Dynardo is preferred. It is chosen in order to reduce the complexity of the programmed tool and to have a consistent optimization framework.

The tool is designed to handle many different scenarios, including different parametrization types, and different flow solver settings. However, due to time limitations for the project, the full 3D blade design capability is not implemented yet. The tool has been developed with 3D capabilities in mind to establish a basis that can be improved. These capabilities will be the main discussion of this chapter.

3.2 Geometry Generation

3.2.1 Generating Three-dimensional Blade Shapes

Designed Sections

The blades are designed by interpolating designed 2D sections with GeometryGenerator, designed along the span of the blade. The number of designed sections for each blade row depends on the expected variation of sections in the radial direction, and the number is typically between 3 and 7. In all design cases, it is very common to have a 3D blade shape that has been designed at least at its tip, its hub and its 50% span height. Additional designed sections can be used to further control the radial variation. The designed 2D airfoil sections can then be used along with the stacking line, to fully define the three-dimensional shape of the blade.

At different designed 2D sections, the same family of parameters can be used. The actual parameter values can be different from one designed section to the next. But, if all parameter values were to be identical for two different designed sections, those two sections should have geometrical shapes that are identical to each other. The blade shape can also be formed by considering different parametrization methods at different designed sections. However, the need of having smooth span-wise surfaces with the stacking line and other conditions may elicit unforeseen outcomes for the blade shape in the CAD environment. Nevertheless, this attitude may have benefits which have not been explored within the scope of this project.

The nature of forming the blade shape is an integral part of the project.

However, even from the aforementioned method, one can imagine that there is ambiguity in many concepts that are involved in this task.

Position of Designed Sections

There are different definitions of 50% line and many different ways to define the radial position of each airfoil section that yields a 3D blade. The mid-span can be defined as the geometric halved distance from hub or root of the blade, to the tip of the blade. This is a convenient definition, since it allows the section to be defined in relation to other geometric reference points, namely the hub line and the shroud line. Any blade section can be defined in a similar manner, according to the ratio of the relative distances, from the hub to the tip, of the designed section. However, if the designer has specific knowledge about the streamlines and their position in relation to the hub and the tip, all of the sections can be designed on streamlines instead of geometrically defined positions. This method has the advantage of the geometrical parameters having direct link to flow characteristics, but with the expense of relying on the prediction of where the streamlines are.

3.2.2 Chosen Approach

In order not to include the relation between streamline positions and designed sections into the optimization problem, fixed geometric ratios were used to find which sections of the blade should be designed. Also, as stated before, different parametrizations across different sections are not considered, and a fixed parametrization type across different designed sections are chosen in order not to clash with current capabilities of the GeometryGenerator tool.

3.3 Flow Analysis

Each three-dimensional blade is chosen to be analyzed with a two-dimensional flow solver MISES. This is achieved by using streamline positions in the 3D flow, and finding the 2D airfoil section which is formed by the intersection of any chosen 3D streamline and the 3D blade. Any 2D airfoil section that will be analyzed, is therefore perfectly coinciding with the 3D streamline that is used to slice the blade.

The flow solver has 4 modules that are used in succession, to analyze the 2D airfoil section and obtain the results. The procedure is very similar to other commercial flow solver. First, the grid / mesh is generated which is the

discretization of the domain of the flow problem. In MISES, the flow study is done within the flow channel of two adjacent blades within the blade row. A non-optimal sample mesh can also be seen in Figure 3.1. After the mesh is generated, the boundary conditions are declared and flow analysis can then be performed. With the calculated results, the results are dumped into 3 different text files and plots are saved.

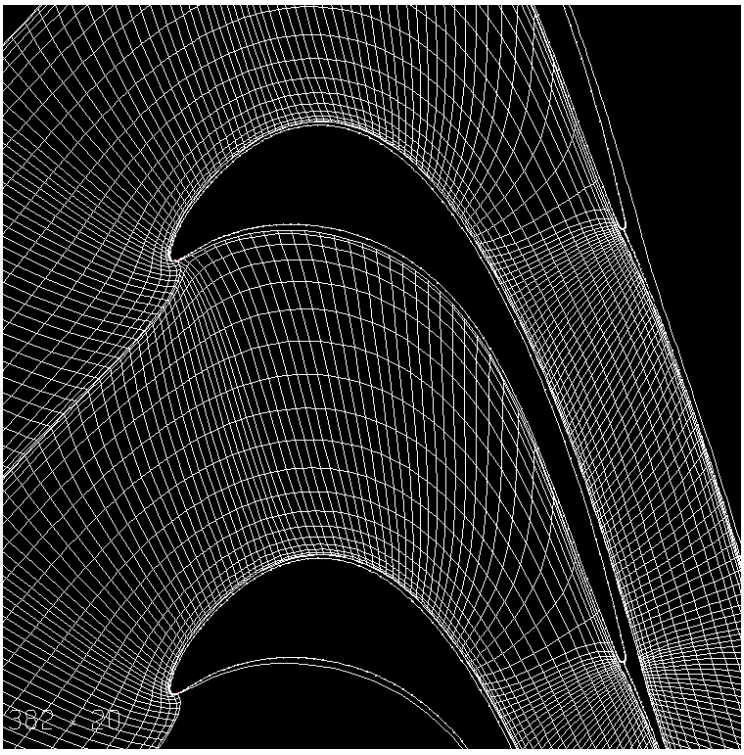


Figure 3.1: An example mesh in MISES

There are many available settings within the whole process. Mesh generation settings and iteration count are just a few examples. It is important to detect more critical settings for the flow analysis, so that the process can be automated, and errors can be handled within tool runtime. There are two major problems that can be repeatedly encountered:

- For certain airfoil shapes, the mesh cannot be generated due to limitations within MISES.
- Flow solution does not converge.

If the grid cannot be generated, it is difficult to detect how to change grid generation settings to successfully create it, at least within runtime. Therefore,

if the grid cannot be initialized, the error is allowed to propagate, and it is handled after the flow convergence check.

Even if the grid is successfully generated, the flow solver may still not converge. For each flow analysis, MISES requires the user to enter the number of iterations. If the flow solver converges, then MISES automatically returns. However, if it does not converge, the total number of iterations are forcefully performed by MISES without any means to stop. In many cases, if the solution does not converge initially, with more iterations convergence can be reached. However, in some cases, the solution diverges during runtime and the flow solver wastes time and memory to reach the specified number of iterations, with no potential to converge. To prevent this phenomenon, the flow study is programmed to work slightly differently.

Instead of specifying 250 iterations consecutively without any interruption, 25 iterations are specified. After 25 iterations, the log of flow solver is read and flow solver convergence state is checked. If it is detected that the flow solver did not converge, another 25 iterations are done. This goes on until the maximum allowed iterations of 250 is reached, and this limit is hard-coded for the current version of the tool. If at any point, the convergence check yields the conclusion that flow study diverged, after the 25 iteration block that MISES forcefully does, the flow solver is terminated and the analysis is considered as failure. If the flow solution does not converge after 250 iterations, it is also considered as failure. If the grid cannot be initialized, this error is allowed to propagate without preventing the execution of the flow solver. Since the mesh generation is inconclusive, MISES cannot be executed. After 10 failed execution attempts, the tool detects failed convergence and the design is treated as failure. In any case, the flow analysis convergence is a fundamental consideration when the project task involves automation.

3.4 Relation Between Designed 2D Sections and Analyzed 2D Sections

Any 3D blade within the scope of this project, is designed by designing multiple 2D sections that are contours of the 3D blade shape at specific span heights, and each of these designed sections are parametrized with a set of design parameters. This process was explained in subsection 3.2.1.

The flow analysis is done for a 2D airfoil section with MISES, however, is not necessarily the same section as the 2D designed airfoil section. The flow analysis is always done around a 2D airfoil section which is on a streamline

within the 3D flow. However, any streamline does not necessarily completely overlap with a 2D designed section of the 3D blade. If the designed 2D sections are designed on streamlines, then this complete coincidence is valid. In cases where designed sections are not defined with streamline positions but positioned relative to the hub and tip, there will be a discrepancy between the designed 2D section and the analyzed 2D section.

3.5 Representation of Input Parameters For Designs

3.5.1 Representation of Geometric Design Parameters

The geometrical parameters for parametrized design sections do not have to be input parameters for the optimization loop, directly. Different representations of geometric design parameters within the input vector are implemented to the tool. These implemented representations are relevant in different design scenarios and are useful when searching for best designs. Any particular parameter value for parametrization of the designed sections will be denoted as χ and its representation in the input vector will be denoted as $\mathbf{X}_k$

Replacing Old Parameter Values with New Values

One can consider each design parameter for a 2D section, and use this parameter as is. In this case, the value of the the parameter is directly used within the design input vector $\mathbf{X}$. This is the most common way of representing geometric design parameters as an input parameter, and this operation can be mathematically expressed as in Equation 3.1.

$$\begin{aligned} \chi_{\text{new}} &= \chi_{\text{old}} \\ X_k &= \chi_{\text{new}} \end{aligned} \tag{3.1}$$

Changing Old Parameter Values

One can also consider the changes in design parameters, instead of the actual value of the design parameters within the parametrization. This change can be expressed in multiple ways. For the project, two different methods have been implemented. Percent increase in a parameter, is denoted as p, and it is formulated as in Equation 3.2.

$$\begin{aligned} \chi_{\text{new}} &= \chi_{\text{old}} * (1 + p/100) \\ X_k &= p \end{aligned} \tag{3.2}$$

According to this representation, a 5% increase in a geometric parameter will be stored as $X_k = 5$ and not by $X_k = (1 + 0.05) * \chi_{\text{old}}$. This kind of a representation is thought to be suitable for parameters in any parametrization, where the parameter is a non-dimensional number, like a length-to-length ratio.

Delta change in a parameter, is denoted by δ and it is formulated as in Equation 3.3

$$\begin{aligned} \chi_{\text{new}} &= \chi_{\text{old}} + \delta \\ X_k &= \delta \end{aligned} \tag{3.3}$$

According to this representation, a delta increase of -3 in a geometric parameter will be stored as $X_k = -3$ and not by $X_k = \chi_{\text{old}} - 3$. This kind of a representation is thought to be suitable for parameters in any parametrization, where the parameter is a non-intuitive quantity.

It is important to state that, even though the reference value of the design parameter is removed from the input vector, the original value still affects the design for both of the representations that involve changes. This reference value of the parameter within the parametrization comes from the reference design. This reference design is none other than the initial blade design "guess", which contains information about the parametrization method of the designed sections and the actual values of parameters, which are used in conjunction to generate the 3D blade shape.

The representation of design parameters as deviations from their reference values, is a viable option, especially for optimization. If the initial design guess is not arbitrarily generated, but found by engineering expertise, it is logical to predict that the best designs (from the optimum design search) will have design parameter values that are close to the parameter values of the initial design guess. In such a scenario, the optimum design search can be formulated to use the delta change or percent change representation. This enables the user to search for best designs, by specifying the maximum allowed change in the parameters without having the need to understand the parameters' plausible ranges.

Examples of Aforementioned Representations

Consider the example in Figure 2.4. To simplify the discussion further, only consider the designed section at the mid-span. Within this ellipse parametrization, the lengths A_1 and B_1 are the only two geometric design parameters for the mid-span section. Now, assume that B_1 value is constant, and the best designs are searched by changing A_1 only. The value of this parameter is always relevant and always used to generate the 3D blade shape. However, the discussion is about possible representations of this search for A_1.

- A_1 can be used directly as an optimization input parameter. This type of representation of the optimization parameter from a geometrical design parameter is a replacement process. So if A_1 is 1 unit long, the user can search for optimum designs for this design parameter by a optimization input range like $A_1 \in [0.5, 1.5]$ and $\mathbf{X} = \langle A_1 \rangle$.
- A_1 can also be searched by representing the search as a delta change. Consider a maximum allowable change of 0.5 units for A_1. This will yield an interval of $A_1 \in [0.5, 1.5]$, which is the same interval as before. However, this time, the optimization input parameter is δ and it is used for the search of A_1. So now, the input parameter for the optimization framework is $\delta \in [-0.5, +0.5]$ and $\mathbf{X} = \langle \delta \rangle$.
- A_1 can also be searched by representing the search as a percent change. Consider a maximum allowable change of 50% for A_1. This will yield an interval of $A_1 \in [0.5, 1.5]$, which is again the same interval. However, this time, the optimization input parameter is p and it is used for the search of A_1. So now, the input parameter for the optimization framework is $p \in [-50, +50]$ and $\mathbf{X} = \langle p \rangle$.

3.5.2 Representation of Other Parameters

There are additional parameters which can be included in the airfoil design optimization problems, which are not geometric design parameters. These other parameters can include anything that influences the selection of best designs as the solution to the problem. Within the scope of mathematical optimization and machine learning, these parameters are considered to be modelling parameters that can be tuned to achieve a fitting representation of the relations between the optimization inputs and output responses. Specifically for machine learning methods, these additional parameters are considered as an optimization input parameter which are named as hyperparameters, and they can

be separately optimized by using mathematical optimization methods. This process is called meta-optimization in machine learning and it has many similarities with the underlying considerations of airfoil design optimization. In airfoil optimization problems, one can include many other parameters that influence the selection of optimal designs and these parameters can be cumulatively separated from design parameters while they are modified within the optimal design search. Since, there is no specific naming convention for such parameters in inverse design methodology, they will be referred to as as meta-parameters; which is a naming convention in similar problems; insinuating that these parameters are not used to construct the optimized blade geometry, but they still influence the determination of the optimal blade geometry.

The concept of so-called meta-parameters for each design case, is another important consideration for the inverse design problem. There are possible metaparameters within the optimization, as for example, the statistical tolerances, degrees of polynomials used for statistical fitting, and sampling settings. However, due to the limitation of available tools, these optimization meta-parameters cannot be dynamically modified within runtime, which prevents their use in the input parameter vector to optimization. For this project, metaparameters are chosen to be the flow solver settings. The four MISES setting files that the tool reads, explained in subsection 3.7.1, contain all the metaparameters that can be accessed within runtime, and they can be assigned as metaparameters. However, in all of those files, there are many settings that are irrelevant and should not be included as an optimization input. However, there are some settings which have a significant contribution. Such metaparameters that are worth explicitly mentioning, are the mesh generation settings. More specifically, the spacing ratios for the mesh around the suction side and the pressure side of the airfoil can have a significant effect on flow analysis results. This in turn, affects the selection of best designs within the optimization framework.

In order to accommodate all such (numerically representable) flow solver settings, a representation similar to Equation 3.1 was chosen. So, if the user wants to specify new values to any such setting, the new numerical value will be used to overwrite the corresponding flow solver setting file. After all such modifications are done to all corresponding setting files, flow solver MISES is used with the new setting files to analyze sections. This representation is exemplified in Equation 3.4

$$\text{gridpar}_{6,3} = 0.9 \tag{3.4}$$

In (3.4), the changes in gridpar.xxx file is demonstrated. This file is one of

the setting files that the flow solver MISES requires. The encoded information in this equation is to change the setting, that is at 6th line and 3rd position, to value to 0.9. This new value is replaced in the gridpar.xxx file, which changes the grid spacing and affects the flow analysis results. This metaparameter for the blade design evaluation, can also be assigned as an optimization parameter as $X_j = \text{gridpar}_{6,3}$

3.6 Representation Of Tasks and Responses

The representation of input parameters ($\mathbf{X}$) in Equation 2.1, was discussed in section 3.5. In this section, how output responses of a blade design ($\mathbf{Y}$) can be represented. These responses include the values of objective functions and the constraints.

Objectives and constraints are cumulatively treated as responses, since both of them are used to evaluate how desirable a design is. These responses are the "metrics" that were mentioned previously. The constraints are used to filter the blade designs that are unfeasible. The values of the objective functions are used to rank the blade designs, that are strictly feasible. In this section, all of the built-in types of responses that are compatible with the tool, along with their explanation, will be mentioned.

3.6.1 Default Constraint of Flow Study Convergence

The only condition where a design is consistently considered as a failed design, no matter what user-specified constraints are, is when the flow solver MISES does not converge. If there is a failure with the grid generation, or if the solver diverges within runtime, or if the solver reaches maximum allowed iterations, the flow study is considered as failed. Correspondingly, the 2D aerofoil section which is analyzed and the 2D designed aerofoil section that is used to generate the blade, is considered as failed design and scrapped. The aforementioned cases are considered as failure, since the flow analysis results can be considered invalid for these cases and cannot be used, at least from an engineering standpoint. One should note that any failed design because of flow study failure is still included in the overall optimization problem, since it contains valuable information about correlations between input parameter combinations and flow analysis failure. The flow solver module within MISES is called ISES, and the convergence state for ISES is encoded in the output file, with 3 different numbers, as shown in Table 3.1

Table 3.1: Encoding Flow Solver Convergence as an Output Response for the Output File

Encoded Number	Explanation
-1	It is detected that the flow solution has diverged, with no possibility of convergence.
0	The flow solution did not converge after maximum allowed iterations with flow solver. OR The grid generation has failed with chosen design parameters.
1	The flow solution converged within default flow solver tolerance, before reaching maximum allowed iterations

3.6.2 Single-valued responses

There is a wide variety of flow variables that are computed during the flow analysis as a result of the blade geometry and the boundary conditions. MISES has modules that dump the results into separate files. The information that can be obtained from all the files include:

- Flow-field within the flow channel between two adjacent blades in the blade row
- Boundary layer parameters on the blade surface
- Flow variables at the inlet and outlet stations which are not specified as boundary conditions.
- Flow variables on the leading and trailing edge of the 2D airfoil section, which is analyzed
- Flow variables that relate to turbo-machinery performance, like turbulent losses.

Because of the available information in MISES output files, boundary layer information is being used but flow-field information could not be treated. This is mainly because MISES, when doing flow solutions, manipulates the problem domain. Depending on the airfoil section that is analyzed, blunt or sharp trailing edges are removed and treated differently depending on the input airfoil geometry file, and mesh is displaced to replicate boundary layer growth, which makes it very difficult to transfer the flow-field information onto the airfoil surface.

Other information is manipulated so that the user can have access to many of the features of the flow solution. All of such features can be assigned either as an objective or a constraint.

Flow variables

The flow variables are the output variables of the analysis results. They include exit flow angle, loss coefficients, absolute exit Mach number and among others. The naming convention for the flow variables are compatible with the names that flow solver MISES is using. The flow variables include indicative quantities of the aerodynamic performance of the blade cascade, which is analyzed.

The whole list of flow variables that the user can assign as a response, is as follows:

- S1deg
- S2deg
- Sexit
- Re1
- Ncrit
- Turb
- M1(ref)
- M2(ref)
- V2/V1
- P2/P1
- Po2/Po1
- Poa2/Poa1
- P1/Po1
- P2/Po1
- P2/Poa1
- P2/Poa2
- rho2/rho1
- rho1/rho1a
- rhoo1/rho1a
- M1(abs)
- alpha1
- M2(abs)
- alpha2
- Loss_Omega
- Loss_visc
- Loss_shock
- Loss_Zeta
- Xtr1
- Xtr2

One should also note that some of these variables are actual aerodynamic performance indicators, which are very suitable to be assigned as response of a design within the scope of inverse aerodynamic design.

Formulated variables

By using the result files, one can also formulate those variables that are relevant to the flow study and the performance of the blade design. The formulated variables are indicated in Equation 3.5.

$$\text{Ma}_{\text{SS,max}} = \max(M_{\text{suction side}}(x))$$

$$\text{x\_at\_MaSSmax} = x \mid \text{Ma}_{\text{SS}}(x) = \text{Ma}_{\text{SS,max}}$$

$$D_{SS} = \frac{\text{Ma}_{\text{SS,max}} - \text{Ma}_{\text{trailing edge}}}{\text{Ma}_{\text{SS,max}}} \tag{3.5}$$

$$\text{Rate\_Parameter} = \frac{D_{SS}}{\Delta l_{\text{uncovered}}/l_{SS}}$$

For the definition of D_{SS}, the suction side diffusion rate, the Mach number at the trailing edge and the peak Mach number at the suction side of the airfoil are included. For the definition of rate parameter, the length of the suction side (l_{SS}) and uncovered length of airfoil surface ($\Delta l_{\text{uncovered}}$) are used, both of which are measured as the arc length in curvilinear coordinates on the airfoil surface. The uncovered length is defined as the arc length on an airfoil, from the position of the throat to the trailing edge of the airfoil.

3.6.3 Distribution Matching Tasks

The inverse design process involves control of flow variables' distributions along the blade surfaces. Certain flow variables, like the coefficient of pressure or the Mach number, can be described as a distribution along the blade surfaces. If the design task involves generating a blade design, which yields a flow variable distribution that exactly matches with a target distribution, the mis-match between the target and predicted distributions has to be quantified.

This quantification is established in many different ways in literature. Cardamone [17] has proposed a consistent way to characterize a resultant flow distribution by calculating many different points in a flow velocity profile, as can be visualized in Figure 3.2. The suggested calculation involves detection of extrema values (maximum and minimum), and also inflection points of velocity values for both the suction and pressure side of an airfoil, which is done to calculate different diffusion factors to be used for optimization. For optimization purpose within the research of Cardamone [17], a single objective function, which is a weighted sum of multiple objective functions including diffusion rates and losses, is used to search for the optimal design. This approach might be useful for design cases where there is a strict requirement for flow acceleration and deceleration. However, it may also be useful to formu-

late other objective functions from the calculated points that are not diffusion factors. In such a case, only the discrepancy between these calculated points on calculated flow distributions (with flow analysis) and a target velocity profile can be used to formulate an objective function, which is optimized to find the optimal design.

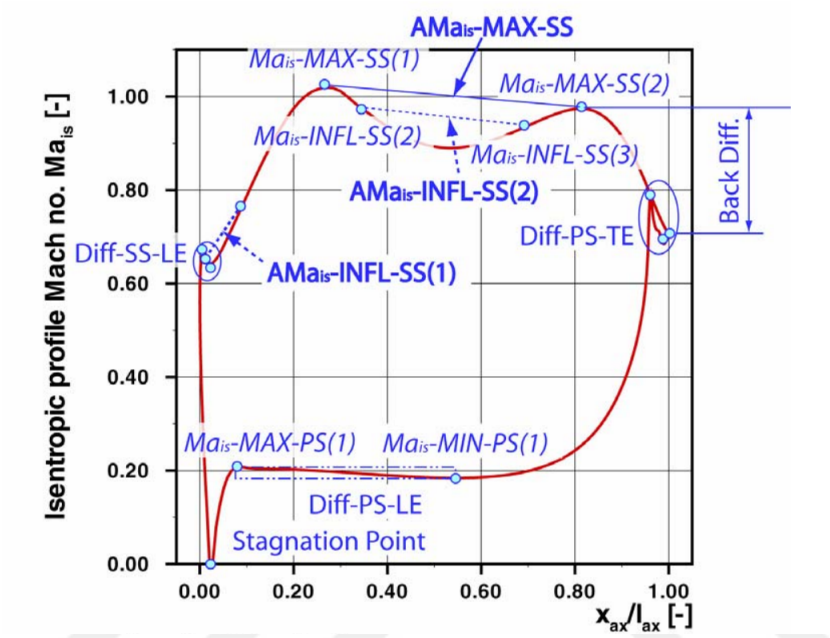


Figure 3.2: Calculated Points in Velocity Distribution to Characterize Distribution[17]

Amoiralis and Nikolos [30] have achieved the mismatch quantification by computing the area between a target distribution and the distribution yielded by the flow analysis, when testing the effectiveness of different geometry parametrizations. The optimization objective, then, is to reduce the total area between the two curves, which represent the distribution in Figure 3.3. The method used in this project is very close to this representation. However, the objective function is formulated as minimization of sum of squared errors, where the error is calculated as the difference in value between the target distribution and the resultant distribution from flow analysis, instead of calculating an integral. This is done mainly because of difficulties in consistently formulating integrals for a discrete data set.

Since all distributions can only be stored as discrete data points, there needs to be a consistent way to calculate the sum of squared errors. If there is no sampling points for the target and the resultant distribution corresponding to the

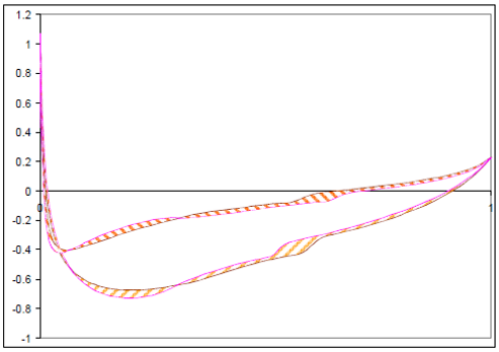


Figure 3.3: Minimizing the area between target and computed distributions [30]

same position on the blade surface, the error would be undefined at those positions. Because of this reason, the target distributions that the user specifies, are requested from the user as a set of profiling points, which is converted to a continuous function. This target function is formed by linearly interpolating between specified profiling points with coordinates at x-axis and y-axis, which results in a continuous distribution over the range that the user specifies. The linear interpolation enables independent definition of profiling points, so that if a profiling point is added downstream, the target distribution function does not change upstream, which would have been the case if a polynomial interpolation were used. One can observe this target distribution function generation process in Figure 3.4.

The analysis results are then fitted with a first order interpolating spline. This spline is piece-wise linear and it can also be seen in Figure 3.4 .This spline is then sampled at 201 points which divide x-axis limits into 200 intervals of equal length. With this new generated continuous target function, the error can be computed for re-sampled 201 points against this continuous target distribution. It is formulated as in Equation 3.6. This sum of squared errors will be the metric that represents the accuracy of the design, for matching the specified target distribution.

$$\text{Total Error for Distribution Mismatch} = \sum_{i=0}^{200} (f_{\text{target}}(x) - y_i)^2 \tag{3.6}$$

For specification of distributions, it is generally recommended to use normalized surface length for the surface to specify the location of the flow variables. For suction and pressure side of the airfoils, the axial coordinate (x)

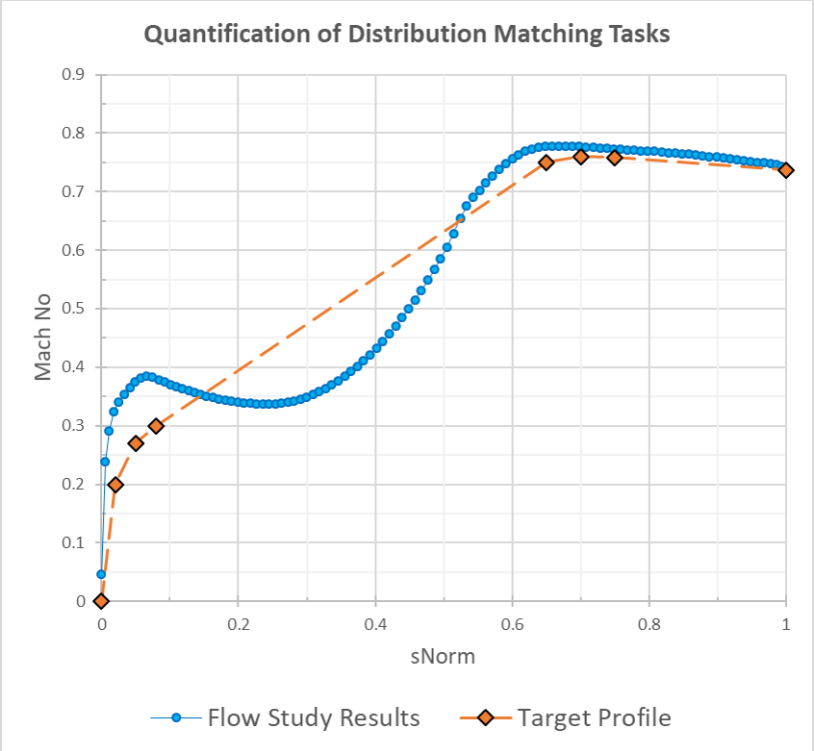


Figure 3.4: Difference between
Constructed Target Distribution and Resultant Distribution

cannot be used without special treatment. This is due to the fact that, towards the leading and trailing edge, depending on the blade metal angles, there are certain axial coordinate values that indicate two different positions on the blade surface. So by specifying an axial position, the user would be ambiguous to the tool, since the tool cannot interpret which of the positions the user prefers.

For specifying locations on any specified blade surface, any of the following variables can be used (for x-axis):

- Surface arc-length defined from leading edge (s)
- Surface arc-length covered from leading edge, normalized by the corresponding surface total arc-length (sNorm)
  For chosen suction or pressure side, sNorm $\in [0, 1]$

For specifying flow variable distribution on any specified blade surface, any of the following variables can be used (for y-axis):

- Shape factor (H)
- Absolute Mach number (Ma)
- Skin friction coefficient (Cf)
- Displacement Thickness (δ^*)
- Momentum Thickness (θ)

All of these variables are defined on the airfoil surface and all of them are defined within the scope of boundary layer theory. The displacement thickness and momentum thickness is defined in the normal direction from a point in the grid, and it is necessarily on the discretized airfoil shape that acts as a solid boundary for the flow solver. The shape factor is then defined as the ratio of displacement thickness to the momentum thickness.

3.7 Software/Tool Structure

3.7.1 Evaluation of Each Design

The tool requires an input file that contains information about the project details, declaration of design tasks, and which input parameters should be used. This input file should be located in the working directory and when this input file is passed as a command prompt / shell / bash argument, it reads the input file and reads certain files at a specific directory.

There is a set of files that should be available for the tool to perform the required tasks. More specifically, 5 files are required in addition to the input file. As mentioned before, the input file also contains information about the directory of these files. Each of these files contains fundamentally different information. One should be aware that the tool must also have read and copy permissions for each of these files. The list of required files and the information they contain, is as follows:

- An initial geometric blade design “guess”, which is used as the base / reference design. It has the extension that can be interpreted by GeometryGenerator.
- The position of the stream line in a pre-defined coordinate system, will be used to intersect with the blade and generate a 2D airfoil section. It is contained in a prefix stream.xxx file for MISES

- Incomplete section geometry details will be automatically calculated, for the 2D airfoil profile. The profile is the result of slicing the blade, as explained in the previous bulletpoint and it is contained in a prefix blade.xxx file for MISES. The results of the geometry calculations are added to this file.
- Default choice for the grid parameter settings, which will be used to automatically generate the grid / mesh, within the flow solver. It is contained within gridpar.xxx file for MISES.
- Flow solver settings that are used to set boundary conditions and running modes for the flow solver. It is contained within ises.xxx file for MISES.

One should note that stream.xxx, ises.xxx and an incomplete blade.xxx files are generated by the software that does through-flow analysis. However, two files that contain the reference design and the grid settings, should be generated by some other means before the tool is executed.

These 5 files are specific for the project, and they are not modified for different designs. Only the input file is different for different designs and they include the changes in the reference design and the changes in files for MISES. How the changes in the blade designs are encoded and how the changes in MISES files are encoded, were explained in section 3.5

Once the metrics are computed and post-processed by the tool, the output responses are calculated and written into a separate output file in the working directory. Once a new design is generated and evaluated, this working directory should contain one design input file, one design output file, and one folder that contains generated files to calculate output responses for the design.

The representative flowchart for the script that performs these steps is available in Figure 3.5. Note that, the script that performs aforementioned operations is called Foilmagic, and it was programmed in Python.

3.7.2 Optimization of Reference Design

To find the best designs, a separate optimization framework needs to be established. The best designs are searched by formulating the search as a multi-objective optimization problem, which was previously explained in subsection 2.3.3 and all of the optimization-related calculations are left to the optimizer, optiSLang.

The default configuration of the tool, includes two suggested optimization schemes. Within each of these two template optimization schemes, two optimization methods exist. Both of the optimization methods will be discussed

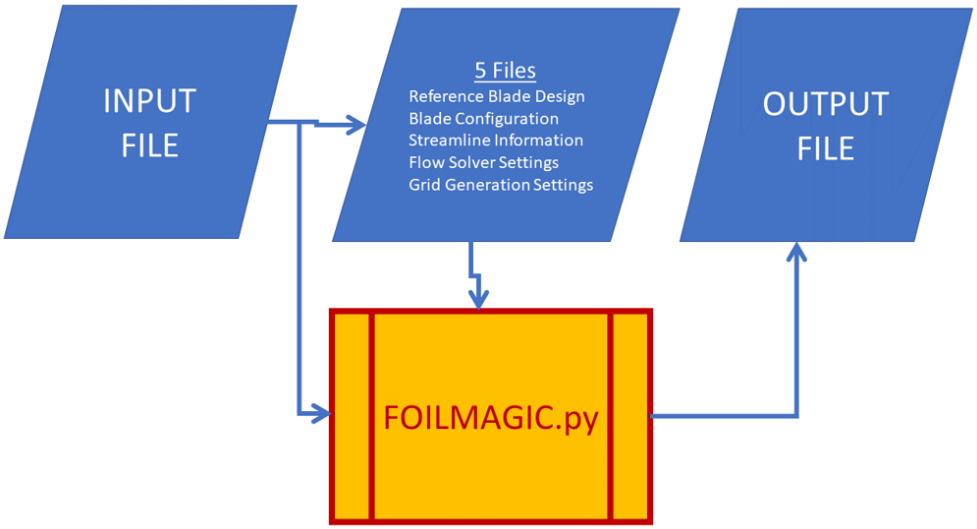


Figure 3.5: Flowchart for Designing and Analyzing Blades with Foilmagic

in 3.7.2 and both of the optimization schemes will be explained in subsection 3.7.2. However, since neither of the two optimization frameworks are strictly necessary for optimization calculations that yield the best designs, one can treat each framework as a suggestion within the tool, rather than a limiting built-in feature. Both of the suggested template solver chains are configured within optiSLang, and the template project is named as Foilfinder. Within this template project, the user of the tool can directly pick one solver chain out of the two pre-configured chains. In both cases, this tool in optiSLang is repeatedly calling Foilmagic, the script explained in subsection 3.7.1, for each new design that it creates. With successful execution of Foilmagic for each design, the output responses of that particular design is evaluated and written into a separate file. This relation between optiSLang and Foilmagic script can be seen in Appendix A.1.

OptiSLang generates different designs by changing the values of the geometric design parameters and metaparameters in the input file template, and reads all of the output files Foilmagic generates for each input file. However, in order for optiSLang to successfully read Foilmagic output files, an output file template also has to be given to Foilfinder project in optiSLang. This relation is illustrated and is available in Figure 3.6.

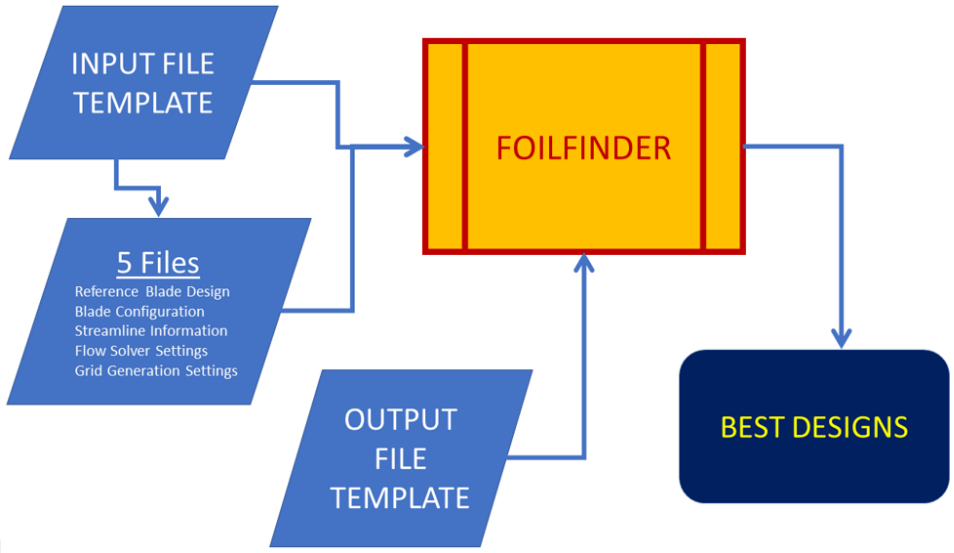


Figure 3.6: Flowchart for Finding Optimum Blade Shapes with Foilfinder

Optimization Methods within the Tool

The optimizer that was chosen (optiSLang) contains a collection of many diferent methods that can be used in the scope of single-objective and multi-objective optimization. The tool is reconfigured within optiSLang using two separate methods, within the collection that the optimizer offers. Specifically, a Design-of-experiments study with Latin Hypercube Sampling and Evolutionary Algorithm were chosen and configured as a part of the inverse design tool. These two methods were explained in subsection 2.3.4. In this section, these methods will be discussed in the scope of inverse aerodynamic design.

Latin Hypercube Sampling can be used for a design-of-experiments study to inspect the relation between input parameters and the output responses and to calculate a metamodel to represent the relation between input parameters and each response. The results are not filtered by any constraints, so that the study can calculate response surfaces for this metamodel, which can then be used for additional decisions. This may include reduction of input parameters from the study or estimating output responses for a given design. It should be noted that Latin Hypercube Sampling is not an optimization method, but a statistical sampling method for computational experiments which yields a statistical metamodel. This meta-model can be used for optimization, which is the way it is treated within the tool.

Evolutionary algorithm can be used to search for optimal designs, while each intermediate design is generated to potentially approximate one of many

possible best designs. For this search default settings for mutation, crossover, initial population size, selection type in optiSLang were chosen. At the time of the development of the tool, which specific Evolutionary Algorithm optiSLang was using is not well-defined in resources from Dynardo. It is believed that the specific algorithm is developed by Dynardo and it is based on Evolutionary Strategy. Evolutionary Algorithm is used to search for optimum designs, generating the Pareto Front as the design population grows. A population-based algorithm is beneficial for aerodynamic inverse design, as it is computationally efficient for global optimization problems and the solutions are expected to consistently become better for increased simulation time, unlike single-solution methods.

Possible Optimization Configurations

As expressed above, both of the template solver chains are available within optiSLang, but both of the solver chains have distinctly different characteristics. The first solver chain is a direct optimization scheme, but the second solver chain includes multiple methods as a chain to perform optimization. In the following two sections, these solver chains will be discussed.

Option 1 - Direct Optimization:

A direct optimization is used to directly search for the best designs, in a multi-objective optimization problem. An evolutionary algorithm is used to search for optimal designs. This process involves the optimizer -optiSLang- generating many different input files, which are generated from an input file template. Each of these input files are sent as a command line argument to Foilmagic, so that with each input file, a new blade design is generated by GeometryGenerator, flow around a section of that blade is analyzed with MISES, and an output file is written. With Foilmagic, for each design, an output file is written that lists all chosen outputs. During this process, the optimizer is generating input files from an input file template.

For Option 1, each new design is generated as a result of the computations for the evolutionary algorithm. According to the selection and elimination process that is chosen for the Evolutionary Algorithm, with specific mutation and crossover settings, generation of successive blade designs replicates the evolution of a population in nature, a process that involves randomness but also fitness. Because of this, new blade designs are expected to improve in terms of optimizing objective functions.

The actual project workflow for this template chain can be seen in Appendix A.2.1

Option 2 - Optimization on Metamodel:

As the first step, a design-of-experiments study is conducted within optiSLang with Latin Hypercube Sampling. This sampling methodology involves generating many different designs with different input files that are generated from the same input file template. Each generated design is realized by each different input file and each generated file is processed by Foilmagic to evaluate the metrics to compare the designs. However, for this option, unlike Option 1, the generated designs are not generated as a result of optimization calculation parameters, but realized by pseudo-randomly sampling from the input parameter intervals assigned by the tool user. This is a pseudo-random process since Latin Hypercube Sampling deliberately generates a spread of design parameters which yield a set of different designs spread across the design-space. However, the actual combinations of parameters are random in nature which is the desired outcome. Since Foilmagic generates an output file for each design, after all output files are written for all of the designs optiSLang generates with a Latin Hypercube, optiSLang reads all of the output files to end the design-of-experiments study. As an outcome of it, response surfaces for each response with all input parameters are calculated. One can think of these response surfaces as a statistical model that can be used to predict the value of each output response from given input parameters. The only constraint for designs that are used for generating these response surfaces, is the flow convergence. So, if the flow study is not in converged state, the results from that particular design is not used to calculate any of the response surfaces. These response surfaces are collectively used to represent the results of the design-of-experiments study, and these response surfaces are collectively referred to as a meta-model or a surrogate model. This model, based on Latin Hypercube Sampling, is used to analyze the relations between the input parameters and output responses without wasting computation time to generate unnecessarily detailed information as in the case of a full-factorial design. This is also the reason why Latin Hypercube sampling was preferred within the default configuration of the tool.

As the second step, the metamodel (each response surface) is used for a separate optimization algorithm. Namely, an evolutionary algorithm is used to search for best blade designs. This time, possible designs are generated with evolutionary algorithm within optiSLang and their output responses are predicted using the design-of-experiments metamodel, instead of using Foilmagic to generate information. With this best-designs search on the response surfaces, a Pareto Front is formed and best designs are predicted to be on the Pareto front.

As the third step, the verification of the predictions of best-designs are

done. Validation runs, similar to the first step, are done by using Foilmagic script, on predicted best designs. These validation runs can optionally be used to update the metamodel, and the metamodel can then be used to predict the best designs again. As the last step, the optimization input parameters and the output responses for each best design predictions can be picked out by the user.

One should note that, the suggested optimum search framework is inspired by suggestions of optiSLang tutorials for optimization problems. However, depending on the number of input parameters and output responses, different algorithms can be chosen for the first and the second step.

The actual project workflow for this template chain can be seen in Appendix A.2.2

3.8 Constraint Programming and Handling Multiple Objectives

A special attention to constraint handling was not given, therefore if the problem is over-constrained, it is up to the user of the tool to distinguish such cases. In those cases, the tool will unnecessarily filter out solutions which are deemed unfeasible, and this will cause the sampled designs to converge to spurious solutions. Hence, the user should be aware that the tool does not govern satisfactory engineering accuracy for over-constrained problems.

The optimization is done on the Pareto Front and the cases where calculated Pareto-Front may have unconventional behavior, like a concave Pareto Frontier, are not tested with the tool. During prior tests, such unconventional behavior has not been observed, which is an assumption within the tool, also exemplified in Figure 3.7. However, especially if there are correlated responses, the sampling method and the optimum search method may fail to find best designs. These correlated responses may be difficult to detect within the scope of inverse design and it is up to the engineer to treat such cases before or after the study, within the established optimization framework. One example for such a case is the multi-objective optimization problem of attaining of a target skin-friction coefficient distribution and minimization of a loss coefficient. Since both objectives are kinetic responses and relate to the same physical phenomena, within the optimization problem they may appear to be correlated. This would cause these two responses to form an irregular Pareto front, and similar cases are left for the user to resolve.

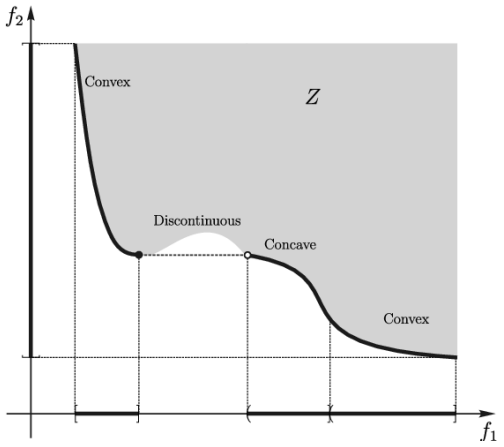


Figure 3.7: Unconventional behavior of a Pareto Front [31]

3.9 Assumptions and Negligences

Many assumptions were made before the tool was developed into fully-working order and there have been many aspects that were not considered for the development of the tool.

- Designed 2D sections are assumed to have very strong influence on closest analyzed 2D sections, and the flow study results are assumed to have very strong correlation with parameters of the designed section parametrization.
- It is assumed that the objective to minimize sum of squared errors between a target and resultant distribution will drive the optimizer to find designs that yield better matching distributions. So, a minimum sum of squared errors, is the best design that can possibly attain the target distribution.
- All geometric design parameters within the parametrization are assumed to be real-valued and not defined over a discrete range.
- All numerically represented meta-parameters are assumed to have discrete values and can be defined over an irregular range of values.
- For the optimum search framework, possible concave Pareto front or solution behavior that is not Pareto-optimal are not considered and not handled in any way.

- The cases where the optimum search is over-constrained are neither detected within runtime, nor they are handled in any way.

3.10 Test Runs Using the Tool

The tool is developed to working order, but the tool also has to be tested to prove that inverse design can be performed with the implementations within tool under certain assumptions and negligences.

In the current working version of the tool, the evaluation of any blade design is performed by Foilmagic (Python script explained in 3.7.1), and optimization of the reference blade design is achieved by Foilfinder (optiSLang project explained in 3.7.2 For the test runs, both of the two optimization schemes within Foilfinder is tested separately.

In order to demonstrate that the tool actually works, an initial test run was performed just to prove the capability of inverse design for two target distributions, one for the suction side of the airfoil section and one for the pressure side of the airfoil section. This specific airfoil section is generated by the intersection of 50% streamline within the 3D flow and the generated blade designs that are only changed at blade mid-span. The reference blade designs are chosen from a sample turbine designed for research within the company.

After the initial test runs, certain modifications to the test cases are done. The tests for the tool include many scenarios:

- The tool is tested to see if it can find best designs, independently of the parametrization type within GeometryGenerator for the designed sections.
- The tool is tested for design of both rotor blades and stator blades.
- The tool is tested for constraints on outlet flow angle (S2deg), which has a lower and an upper bound to specify the allowed range.
- The tool is tested for potential unexpected behavior while optimization is performed, as a means of detecting bugs and execution errors.
- Final tests are done to detect scenarios when / if any of the two optimization schemes within Foilfinder, are not suitable for optimization. This will be discussed in more detail in subsection 4.3.3.

The important outcomes of the test runs, alongside the discussion about potential reasons and potential improvements, are presented in chapter 4.

Chapter 4

Discussion of Tool Performance

If the optimization problem that is formed by the user input is not over-constrained and if the design space contains the optimum solutions (which is hard to predict before executing the tool), the tool was expected to optimize the design so that it solves a multi-objective optimization problem, without violating the constraints. In order to prove this, test runs were performed for different conditions. The conditions that were tested, include:

- Optimization using optimization workflows, as proof-of-concept.
  - Option 1 - Direct Optimization
  - Option 2 - Surrogate Modelling
- Optimization with different optimization input representations in combination.
  - New value in design parameter
  - Percent change in design parameter
  - Delta change in design parameter
  - New value in meta-parameter
- Optimization for blades with different geometry parametrizations within GeometryGenerator
- Feasibility check of representing distribution tasks as sum of squared errors.

- Optimization of blades for both stators and rotors (stationary and moving).

Separate informal tests were done to time the execution of the tool and its computational performance. One should note that they are done to have an estimate about total execution time, uncover which operations can potentially be improved and give an idea to the user about how demanding the solution is.

4.1 Proof-of-concept

The two optimization schemes, that have been implemented and tested within the tool, were discussed in 3.7.2.

A base test with two objective functions and two constraints was performed with each optimization scheme, just to assess the two schemes and how suitable they are for design optimization. This test case has the inverse design purpose of finding blade designs that yield two target Mach number distributions, one for its suction side and another one for its pressure side, both of which are requested at the 50% streamline. The designs are constrained by an upper and lower bound for the outlet flow angle in addition to the default constraint on flow solver convergence.

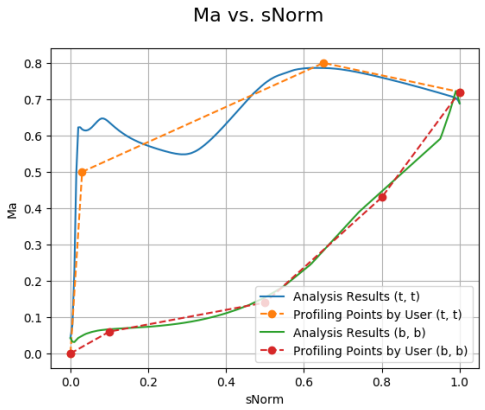


Figure 4.1: Flow Profile for the Reference Design Optimized for Proof-of-concept Test

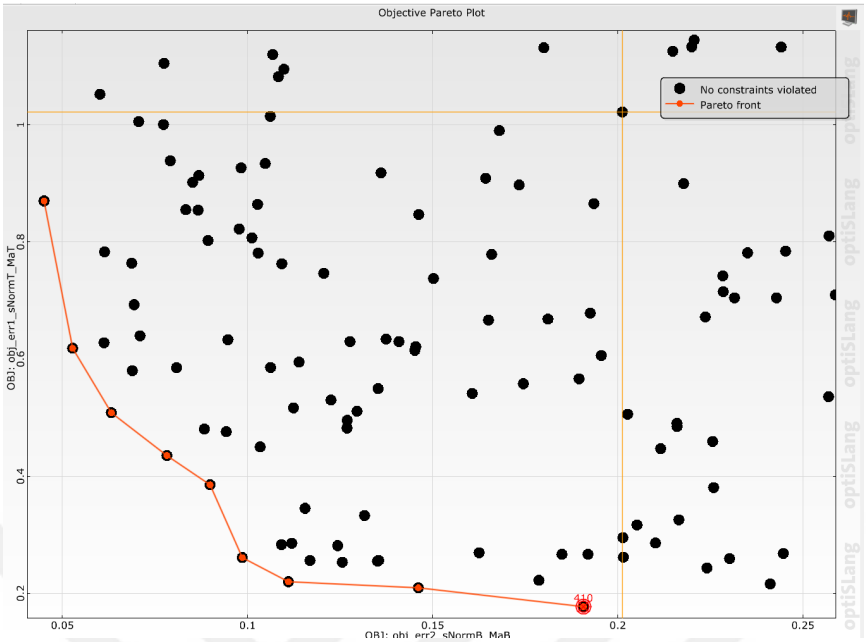
The flow profile around the reference design used for this test case can be seen in Figure 4.1. Both of the optimization schemes are initiated with this reference blade design (a turbine stator blade) with the same parametrization, and with the same target distributions. Under these conditions, the best designs are calculated with each scheme. The designs that will be discussed

are hand-picked from the set of best designs of optimization. The chosen designs match the suction side target Mach distribution most closely, but the mismatch for pressure side target Mach distribution is not considered while picking these designs. These specific designs are generated within the optimization schemes, and the optimizer did not compute other feasible designs that will further minimize the error between target and resultant suction side Mach distributions. In other words, each of the designs which will be discussed are the best generated designs in terms of purely minimizing the sum of squared errors between the target and calculated Mach distributions, for the blades suction side at 50% streamline.

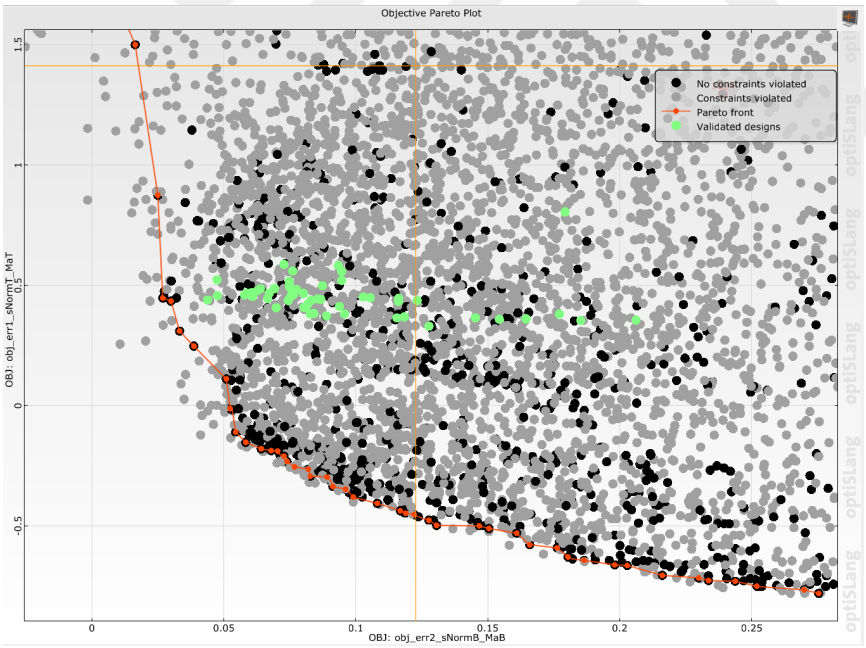
- With direct optimization scheme, 500 designs are generated, and all of them are validated with actual flow computations. Design 410 is hand-picked as the optimal case which satisfies all of the aforementioned conditions.
- For the scheme that involves optimization on metamodel, 2500 designs are generated and all of them are run with the flow solver. With all of these samples, the metamodel is calculated and 10000 designs are generated for the optimization on this metamodel, without any actual flow solutions. As the last step, 70 designs are automatically picked and validated with flow solutions. Design 9738 is hand-picked as the optimal case, which also satisfies all of the conditions discussed above. This is also one of the few designs that have been validated with flow solver, within this optimization scheme, after the optimizer predicted it to be on the Pareto Front. This Pareto Front, as discussed before, is formed by optimizing on the metamodel without doing any validation runs with flow solver.

4.1.1 Effect Of Optimization Algorithm

The pareto front that is formed by these two optimization methods can be seen in Figure 4.2. In both figures, the pareto front is generated for two objective functions which is for a target Mach distribution of the airfoil section at the top and bottom surfaces. For a turbine blade, the suction side with higher flow speeds is the top surface and the pressure side with higher static pressure is the bottom side (which are abbreviated as t and b). As discussed before, the mismatch between target Mach number distributions and the distributions that flow solver yields, are quantified as the error. Each error for top and bottom distributions, are used across different designs to calculate the Pareto Front.



(a) Pareto Front for Test with Direct Optimization



(b) Pareto Front for Test with Optimization on Metamodel

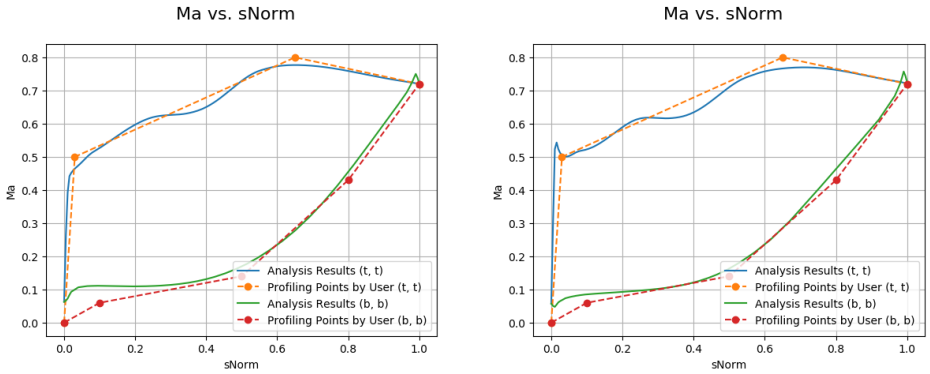
Figure 4.2: Pareto Fronts Computed by Different Optimization Methods for Attaining Top and Bottom Surface Target Distributions

The results in Figure 4.3 indicate that both optimization methods can actually generate a design that is improved for matching the suction side Mach distribution. More importantly, the two designs form flow-fields that are also close to each other's indicating that both methods can converge on similar designs. This is a cross-validation of the inverse design problem. This test demonstrates that the tool manages to successfully improve the reference design to achieve a Mach number distribution for the top surface that is closer to the target Mach number distribution on top surface. However, one should note that both designs cannot "exactly" match the specified target distribution. The exact matching with $0.\overline{0}$ error should not be expected from the tool for multiple reasons:

- In almost all cases, the pre-chosen parametrization does not have enough parameters to generate designs that yield the "exact" targets. In other words, it is generally not possible to control the curvature over the entire blade surface with any certain parametrization. This also means that even if the optimizer did a full-factorial analysis, resolving all possible design parameter combinations, the closest blade designs still cannot attain the specified distribution targets with absolute zero error.
- Also, since it is impossible to eliminate the linear approximation errors of target distributions and numerical truncation errors, the computational tolerance optimizer uses would not be enough to reduce the error to absolute zero.

However, if one inspects the actual airfoil sections which were optimized with the two options, there are important realizations about the nature of the problem. The flowfields of picked designs are plotted in the same cascade view in Figure 4.4. One can see that the designs are actually not very close to one another. Upon additional inspection of blade geometry files, it was seen that, with different optimization methods, different parameters are optimized by the optimizer to match the target Mach distributions. This peculiar outcome has two potential reasons that is worth mentioning:

- This phenomenon may have occurred during the tests because of bad selection of reference blade design, coupled with large parameter variations to search for the optimal designs. If one had started a separate optimization loop with the exact same objectives and constraints, but using one of the best designs as the reference design and setting small intervals for design parameters, the optimizer is expected to search small



(a) Ma. Distribution for Design #410 from Option 1 - Direct Optimization

(b) Ma. Distribution for Design #9738 from Option 2 - Optimization on Metamodel

Figure 4.3: Comparison of Mach Number Distributions for Best Designs Picked by Different Optimization Frameworks

variations and yield the same optimal designs that are much better, irrespective of the preferred optimization method. This expectation is reasonable but requires validation, which is left for future work.

- It can also be concluded that there may be non-unique solutions to the same optimization problem, that are approximated with different optimization methods. In other words, the current choice of optimization methods may not be robust for solving optimization problems which have non-unique solutions. Also, due to the available parameters for the optimizer, some other "approximately-optimal" designs may not be discovered. This is also an important realization from the test, but additional effort is required to automatically handle such cases.

4.1.2 Limitations of Surrogate Modelling

In this test case, Option 2 showed that it is a viable option for design optimization. However, the accuracy of the metamodel is a very limiting factor for automatic inverse design process. If the response surfaces which comprise the metamodel do not represent the true statistical nature of the design optimization problem, then there may be cases when the responses are incorrectly estimated from the metamodel. Such an occurrence will cause the design optimization process to yield spurious or erroneous best designs. Therefore, it is a

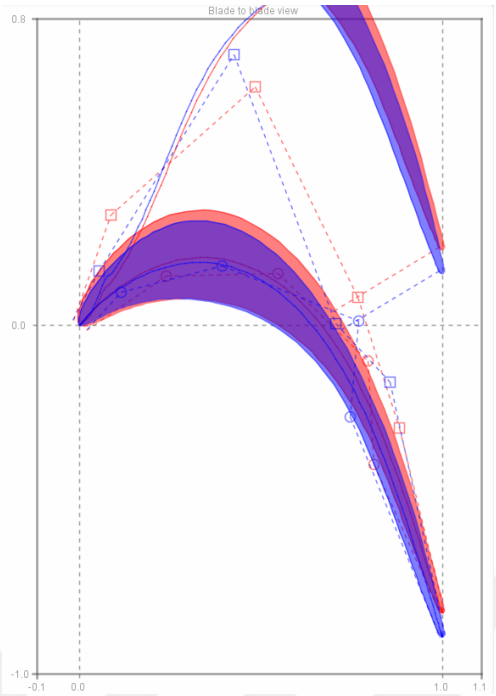


Figure 4.4: Comparison of Optimized Section at
Design 410 with Option 1 in red color
and
Design 9738 with Option 2 in blue color

general recommendation from the developer's perspective, to check the metamodel and its predicted accuracy by optiSLang, before relying on the design-of-experiments metamodel and continuing with the actual optimization process. It should be also noted that Option 1 does not have this limitation of user intervention.

One should also recognize that the surrogate modelling method did not actually estimate the Pareto Front correctly for the predicted best designs. The predicted Pareto Front, shown as as a red line, and the validated values of the predicted best designs, shown as green dots, can be seen in mismatch at Figure 4.2b. Since this method involves calculation of response surfaces that model the relation between all input parameters and each output response, the accuracy of correct Pareto Front estimation is contingent to accurate statistical modeling. Accuracy of the statistical model (specifically the response surfaces) is represented with different coefficients within the optimizer and manual interpretation of the coefficients alongside the approval of the surrogate model is left to the engineer. However, as a suggestion to the tool's user,

these coefficients are known to depend very strongly on a few factors. Specifically:

- The higher number of samples used to generate the DoE study, the higher accuracy the statistical model is, and the more generalizable the DoE results are.
- The more optimization parameters there are for any given number of samples, the lower accuracy the surrogate model is.
- For any given number of samples, if the problem involves correlated inputs specified by the user, the accuracy decreases even though the optimizer detects the correlation, since there is less meaningful data generated per independent input.

4.2 Effect of Optimization Input Parameters

4.2.1 Influence of Number of Optimization Parameters

It was observed that within many tests that, the higher number of parameters are available for optimizer to change, the more likely it is to find the best designs by taking significantly longer time to do so. Depending on the optimization method, different correlations are available to express the relation between the number of optimization input parameters and reasonable number of design evaluations for accurate results. For DoE with Latin Hypercube Sampling, in order to build the meta-model with response surfaces, the minimum required design evaluations is the same as the number of multi-variable polynomial coefficients used to fit the response surfaces. This relation is expressed as in Equation 4.1, where m_c is the number of polynomial coefficients that is also equal to the number of minimum required design evaluations, n is the number of optimization input parameters and k is the degree of said polynomial. For a benchmark scenario of full factorial DoE study, the required number of design evaluations grow exponentially with chosen number of optimization input parameters, if one wants to confidently rely on the optimization search.

$$m_c = \frac{(n+k)!}{n! \cdot k!} \tag{4.1}$$

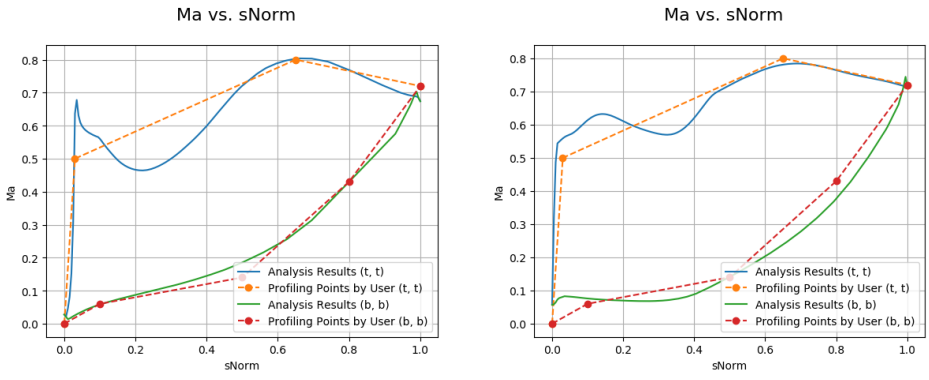
In any case, it is clear that, with increased number of parameters, it is necessary to expend a higher computational cost. This notion of trade-off is also

key to the selection of different blade parameterizations. Unfortunately, there is no direct functionality within the tool as a predictive step to suggest which parametrizations are more suitable for which optimization problems. In order to still function properly, the tool is programmed in a generic manner to leave that decision to the user before starting the optimization process. In a user-intervened scenario for the tool, this decision can be made by the engineer as a validation step by checking if flow-variable distributions and other aerodynamic variables, achieved by the best designs, are within tolerance for the design requirements.

It is also worth noting that it is not necessary to optimize all of the design parameters of the blade. It is possible to reduce the number of input parameters by different means, either by running the tool for different parameter combinations or by previous validated results. It is also possible for the engineer to reduce the parameters by hand, based on relevant understanding of the similar parametrizations and flow analysis results.

4.2.2 Effect of Parametrization Type

It was verified that the tool is compatible with different geometry parametrizations within GeometryGenerator. However, a thorough examination was not performed to understand how each parametrization affects the optimum design search. Two sample designs, that are designed while searching for the same target distributions, are generated with different parametrizations for the designed section, and is represented in Figure 4.5



(a) A sample design for blade with Parametrization I

(b) A sample design for blade with Parametrization II

Figure 4.5: Verification of Capability to Use Different Parametrization Types

It is important to note that different parametrizations have different number of parameters. Moreover, different parametrizations may require different number of optimized parameters to find the best designs with any given optimization method. In most circumstances, higher number of optimized parameters enables more fidelity over controlling the blade surfaces. This means that the optimizer has a higher probability of finding an arbitrary target shape (assigned by the target flow-field behavior) by varying more design parameters, with the trade-off for requirement of increased design evaluations, which was discussed in the previous subsection.

4.2.3 Influence of Geometric Design Parameters

Suitability of Different Representations of Geometric Parameters

It was observed that the tool can interpret different geometric parameter representations as optimization input parameters, as discussed in section 3.5, to design the same blade without any issues. This is a very simple and expected behavior. However, if the representation of geometric design parameters are expressed as the changes from their reference values, the information generated within optimization is more meaningful. Since the optimizer displays how much change in a parameter is relevant to reach one of the best designs, that variation in the parameter can be used in other similar design optimization problems.

4.2.4 Influence of Metaparameters

The metaparameters that can be included in the optimization process are the flow solver parameters, which were mentioned in previous chapter. The flow solver parameters and their effect on the search for the best design has not been thoroughly explored within this research.

However, during the test runs, it was spotted that there have been many cases in which flow solver cannot generate the grid around the airfoil section. As it was explained before, these cases are treated as failed designs, which may not be a very reasonable attitude to handle such cases. If one implements a way of estimating grid generation settings at runtime, and overwrites the grid generation settings to execute the tool successfully, it would significantly reduce the design cases that are wasted while searching for optimum designs. One should also note that the flow solver settings and grid generation settings within MISES does not necessarily vary over a continuous range, but they are discrete-valued. Therefore it may actually be better to formulate certain

mesh quality indicators over a continuous range, which can then be used for indicating how reliable the flow solutions are. Those mesh quality parameters may be assigned as output responses for the optimization problem, instead of having flow solver settings as inputs to the optimization process. This would, in theory, facilitate consistent selection of blade designs.

4.2.5 Determination of Ranges of Allowable Input Parameters

There is no built-in method within the tool to suggest suitable ranges for optimization input parameters. It comes down to prior experience of the user or previous data generated by similar design cases, for assignment of suitable ranges. If the ranges on any parameter is too small, then it may be impossible to find a satisfactory best design, even though the optimal designs found within the algorithm is still expected to be the solutions of the optimization problem. If the range for any optimization input is too large, more design evaluations will be required to resolve the effects of that input parameter over its specified range. In turn, this increases the computational cost of the inverse design problem.

4.3 Effect of Optimization Output Responses

The search for the optimal blade designs are directly influenced by how the objective functions are formulated and how the constraints are filtering the blade designs.

4.3.1 Aerodynamic Performance Indicators

The aerodynamic performance indicators, which were discussed in subsection 2.3.2 and subsection 3.6.2, are directly quantified as numbers and can be accessed within MISES. These indicative numbers can be directly used as objectives to be minimized / maximized, or constraints that filter out blades with intolerable performance. These indicative numbers do not have an unexpected effect on the optimization problem and can be directly used within any inverse design problem.

4.3.2 Quantification of Distribution Mismatch

Quantifying the mismatch between target and calculated distributions as the sum of squared errors is shown to be a good metric to minimize, to achieve designs that yield a flow variable distribution which is very close to the target distribution. However, it does not necessarily penalize flow behavior that is not wanted, like a pre-throat diffusion.

The chosen quantification also does not penalize the extrema points. More specifically, it does not particularly penalize the value of extrema points in the resultant distributions. So, there may be designs that yield distributions with extrema that do not necessarily match the extrema of the target distribution. The chosen formulation also does not penalize the position of the extrema points on the blade surfaces. As an example, in real-life cases, it may be important to achieve suction-side maximum Mach number at a specific location on the blade. Since it always occurs at the position of the throat (where the flow area is at its minimum), the match of the peak can be included in the formulation. These behaviors can be observed in Figure 4.6 visually for a sample design, where the suction side target distribution has one peak at a wrong position, and a single pre-throat diffusion region.

The values and positions of the extrema can be included in the design optimization problem by formulating a separate objective function that can be minimized. However, within the scope of this project, realization and effects of possible other formulations are not explored.

4.3.3 Effect of Constraints

The effects of constraints in optimization problems are studied as a whole research field of constraint programming. Such a technical discussion cannot be made within this project, as it is highly unpredictable in nature to detect if the design optimization problem is over-constrained. The problem is said to be over-constrained, when the predicted best designs are filtered out because of the constraints, so much that Pareto Front does not involve any best designs that is acceptable to the engineer. Cases where it is mathematically impossible to optimize an objective function without violating a specified constraint, is also an over-constrained problem. Both of these cases cause the optimization method to mis-represent the true behavior of the design optimization problem. Since the best design search requires some sort of sampling within the design space, the sampled designs should not be discarded whenever possible. This would enable the designs, in close-proximity of the to-be-discarded designs, to still be considered as a potential solution by the optimizer, and designs will still

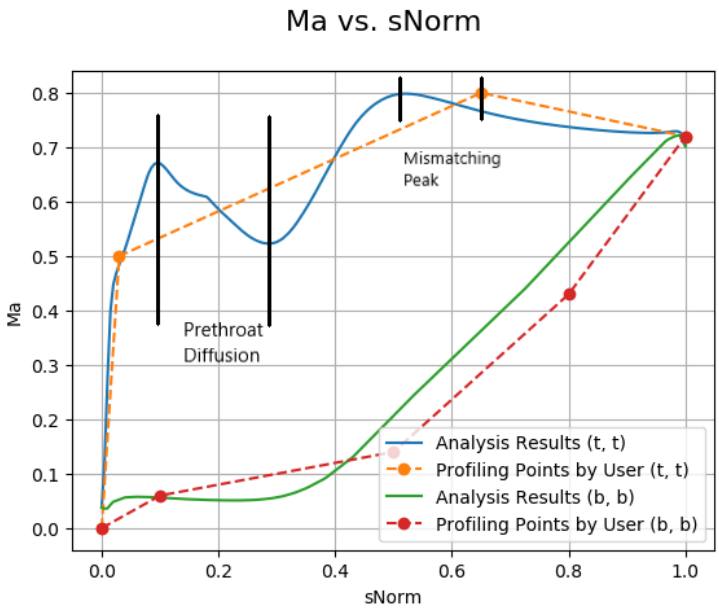


Figure 4.6: Sample Design With Multiple Undesired Behavior

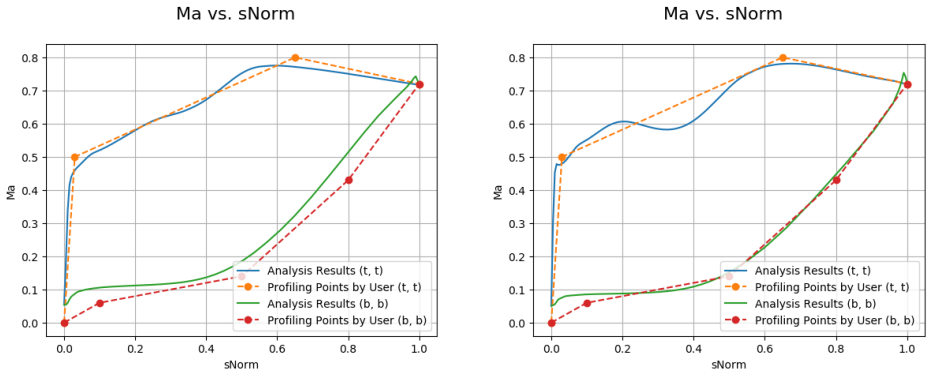
be searched around the supposedly-unfeasible designs. Since the optimization workflows include the design search with an Evolutionary Algorithm, unfeasible designs (violating a constraint) prevent the algorithm to not prefer that solution, and the solutions diverge from the unfeasible designs.

As a general suggestion, it may be logical to have soft constraints that are allowed to be violated. This might be relevant in cases where the optimal design problem has non-unique solutions. If the geometry parametrization yields a large design space, a subset of all geometric design parameters can be included in an initial design optimization. After the optimization process finishes, a separate study can be done with previous optimal designs as a starting point and a different subset of geometric design parameters can be changed to search for similar designs which strictly should not violate the constraints.

As an example, in some problems, if there is a real-life exit flow angle constraint, this can be set free for the first step of the optimization problem. After the best designs are determined by the optimizer, the desired best design can be hand-picked and a separate optimization problem can be set using this hand-picked design as the reference design to the new optimization problem. However, this time, in order to enforce the real-life limitation, it can be set as a constraint to the optimizer and additional design parameters within the parametrization can be involved. This would enable the optimizer to search for best-designs for the initial constrained optimization problem by splitting

the problem into two separate design searches.

To further demonstrate possible implications of over-constraining the problem, consider the cases in Figure 4.7. Within an evolutionary algorithm, it is expected to have better designs as iteration count increases. However, since an outlet angle constraint for the test run forces the Evolutionary Algorithm to reject Design 77, designs which are generated at following iterations are deliberately chosen by the optimization algorithm to diverge from the rejected design. Design 204 is the only design that is detected as one of the best designs at the end (on the pareto front), which has been generated at a later iteration than Design 77. Even then, one can see that this solution is generated by the evolutionary algorithm to match the pressure-side distribution and not generated to further optimize the suction-side distribution. Since no other designs generated after Design 77 are chosen as one of the optimal designs, the tool user can incorrectly reason that the problem is over-constrained. However, in reality it is inconclusive if the problem is actually over-constrained and impossible to solve, or if the constraint prevented the algorithm to generate designs that are close to Design 77. For the latter, there is a chance that longer simulation times would enable the optimization framework to generate similar designs to Design 77. For the former possibility, the engineer would be wasting time to resolve the issue by hand, when in fact it is impossible to resolve without adding new parameters or changing optimizer settings which erases all the data generated for that optimization task.



(a) Generated Design #77
Violating Exit Angle Constraint

(b) Generated Design #204
One of Many Best Designs

Figure 4.7: Effect of constraints on optimum design search

It is possible that the constraints are violated by a very small discrepancy and any small change in an input parameter can generate a design that does

not violate any constraints. For the test problem, this was actually the case and doing a small perturbation to stagger angle (one of the parameters used in the parametrization) of the airfoil section, results in a design that doesn't violate any constraints while yielding a flow distribution which is nearly identical to the design in Figure 4.7a, which is a very good design. However, in a general design scenario there is no way to predict that the globally optimal designs are not violating any constraints. In other words, it would have been impossible to conclude that unfeasible Design 77 can be perturbed to yield a feasible and possibly an optimal design. If there were additional constraints that Design 77 did not violate, but the perturbed design violates while avoiding the constraint which made Design 77 unfeasible, then searching for perturbed candidate solutions is also a part of the design optimization problem. Therefore, employing soft constraints at the initial step would enable the dataset to include possible candidate solutions which may have violated the hard constraints. Then, the optimal solutions of the first analysis can be perturbed to find similar designs, which are not violating any constraints. However the design scenario is handled, it is very clear that constraints can significantly alter the optimization process and yield unexpected predictions of optimal designs.

4.4 Tool Diagnostics

It is worthwhile to mention how computationally efficient the tool is and how costly the computations are. The discussion is based on the computational complexity of the tool, specifically the time and memory complexity of the algorithms and methods used within the optimization schemes.

4.4.1 Computational Cost for Design and Evaluation of Individual Designs

The python script can be executed in a standalone fashion, which also enables independent timing of the script's execution time. Since the script is creating subprocesses, piping input and output files and waiting for some of them to return to continue execution, the subprocesses' execution times have non-negligible importance to overall execution time. The script can operate within both Windows and UNIX shells, and the subprocesses significantly differ in time across the operating systems.

An extensive timing study is not done, but 15 runs in succession were performed with the same files under different operating systems and the execution times are averaged. According to the average execution times:

- Execution time averages at 20 seconds in Windows.
- Execution time averages at 8 seconds in Linux.

This major difference is caused by the 2 specific subprocesses that affect the performance:

- Launching Pollygraph / GeometryGenerator, changing the blade geometry and exporting the file.
- Launching MISES / flow-solver, executing the flow-solver modules in the required order and generating result files.

The launch speed of the two programs and the flow-solution computations are significantly faster in UNIX. Therefore it is suggested to use UNIX for batch execution of the script if it is necessary to cut down execution time. However, the total memory used for storing all of the blade generation and flow analysis data is the same for both operating systems, which is around 4 Megabytes worth of files stored in a single directory. Therefore, in cases of single or validation uses of standalone code, it may be unnecessary to revert to UNIX platform, just to save time.

4.4.2 Computational Cost for Optimization Calculations

The optimization calculations are timed once in Windows, for the two optimization methods discussed previously. It is for a general comparison but the execution times of optiSLang is still noteworthy.

- Surrogate Modelling and Optimization on Metamodel, with 2570 verified designs and 10000 designs picked on metamodel, took 10 hours of computation time.
- Direct optimization with 610 verified designs, took 4 hours.

If one considers the computational accuracy of the two optimization methods, as discussed in subsection 4.1.1, it should be remembered that Direct Optimization finds the best designs much faster by searching significantly less number of different candidate designs. This comes down to Evolutionary Strategy that is used within Direct Optimization, which greatly reduces redundant design searches. This reduced number of design searches is also

advantageous in terms of total memory expenditure to store all of the information, generated for each of the candidate designs. Therefore, it is highly recommended to use Direct Optimization in cases where the sole purpose of the study is to calculate the Pareto Front and find the optimal designs. On the contrary, if the inverse design study is done to generate a dataset to capture as much detail as possible for future reference and performed for many different objective functions, Surrogate Modelling as a design-of-experiments approach is worth-while even when the computational cost is considered.

One should also remember that both of the aforementioned methods can be used to run design evaluations in parallel. This will greatly increase the Random Access Memory (RAM) requirement for the computer, significantly reducing the total execution time for the design optimization.

4.4.3 Built-in Limitations

Within the tool, there are some built-in limitations to have an indirect control over time and memory expenditure.

The most prominent limitation is the maximum allowed flow solver iterations per design. The flow solver convergence is checked at most 10 times, and it is allowed to go with 25 solver iterations at most before checking for convergence. This puts a hard limit of 250 maximum allowed iterations; discussed in detail in section 3.3. This flow solver convergence check and the maximum allowed iterations, jointly enables the tool to quickly reject designs that yield unrealiable flow results.

The other hard-coded limitation is with the generated files. In order not to generate files that are not related to the set inverse design problem, only figures related to the inverse design objectives are generated in the designated folder. So, if the user does not set a target distribution to be achieved, no distribution plots are generated and saved. If the user sets a target distribution for Mach number at top surface only, there will be only one figure that only includes the resultant Mach distribution on the suction side. If target distributions for both airfoil surfaces are set and if both of them are the distribution of the same variables, then they are plotted in the same figure to conserve memory space.

4.5 Possible Capabilities

The tool is primarily designed for aerodynamic design optimization problems regarding turbomachinery airfoils, with concentration on turbine airfoils specifically. It is developed for the user to change the 3D blade shape at any of

the 2D designed sections in the GeometryGenerator and analyzed on a single streamline with the flow solver. However, the tool, within the aforementioned capabilities, is developed to be a viable option for other automated applications.

The tool can be used to run a separate mesh-study to determine a standard set of mesh parameters, by only changing the grid parameters and not the design parameters or any other parameters to the problem. This would enable to find the set of grid parameters that does not cause large variation in built-in optimization outputs, as discussed in subsection 3.6.2, for a reference design. In this sense, this capability is useful for parameter meta-optimization, where a subset of all optimization parameters (specifically grid generation parameters) are fine-tuned before the actual design optimization.

The tool can also be used to understand the effect of other designed sections on the flow on the chosen streamline. This capability is not very meaningful for the current version of the tool, since the full 3D blade cannot be optimized with all streamlines simultaneously, but it is actually the foundation for the tool to be extended to handle 3D aerodynamic optimization.

Chapter 5

Conclusions

The aerodynamic design optimization requires many different considerations. When the task at hand is to automate this optimization process, these different considerations must be addressed and potential errors must be handled in a consistent manner, so as not to affect the search for the best designs.

Geometry Considerations

The geometry parametrization is an integral consideration for the design optimization problem. If the geometry parametrization involves too few parameters, then the design cannot be changed to the extent that the best designs are of sufficient quality according to their aerodynamic performance. On the contrary, if the geometry parametrization involves too many parameters, the computational expense of running the tool cannot be compensated by its convenience and it is advisable to reduce the number of parameters involved in the optimal design search before progressing.

It is also important to assess the suitability of the chosen parametrization of the blade. Having physically interpretable parameters enables the surface curvature of the blades to be intuitively modified before starting with the optimization process. Since the choice of the reference design is shown to be influential on the accuracy of prediction of best designs, utilizing a parametrization that includes parameters that are in close relation to blade surface shape can have unforeseen advantages.

Optimization

The behavior of the optimization framework has a significant influence on finding the actual best designs of the problem. If a sampling-based method like design-of-experiments is included in the framework, one should inspect how representative the metamodel and the response surfaces are, before making further decisions on parameter reduction and searching for optimum designs. If population-based algorithms are included in this framework, the settings for elimination of individuals and how new individuals are generated has to be thoroughly assessed, so that optimal solutions are approximated to sufficient precision of input parameters, after the algorithm finishes execution.

The formulation of objective functions also plays a huge role within optimization. Objective functions have to be formulated in such a way that they truly reflect what the user wants to attain. As an example for distribution matching, if it is very important to have the peaks of target and resultant distributions to match exactly, the mismatch should be included in the objective function formulation in some way. Within this project, countless other possbile considerations for quantifying distribution matches were ignored and just a relative mismatch of the whole resultant distribution is formulated with sum of squared errors.

Spotting if the problem is over-constrained is very difficult to do beforehand, but it is another fundamental consideration for the optimal design search, and unfortunately dependent on the optimization method that is used. So, it is up to the user of the tool to change the optimizer settings and adjusting the behavior to successfully solve inverse design problems within an optimization framework.

Future Work

Improvements for the tool

There can be changes in the current version that were not implemented due to time limitation of the project. From the developer's perspective, they are easy to implement into the code, without doing major modifications in the overall architecture.

- The blade geometries which are considered as failed designs can be handled differently according to the reason why they are considered as failure. For different reasons like grid generation failure or constraint violation, different modifications can be done. An "intelligence" can be programmed to systematically suggest changes in the blade design.
- Different representations of distribution objectives can be implemented. In subsection 3.6.3, the discussion was about how mismatches in target and resultant distributions can be quantified to be an objective function, which is to be minimized within optimization to find the best designs. The implementation of the discussed methods, may be relevant to find more relevant designs for the inverse design problem.
- Additional flow variables can be incorporated to the tool to define target distributions. The available flow variables are defined in subsection 3.6.3. Other relevant flow quantities like coefficient of pressure or static pressure can be implemented, so that the user can have more options to choose when specifying target distributions.
- The tool was tested specifically for turbine airfoils, but in principle, it should accomodate design of other engine components. Specific flow variables relevant for other components, can be formulated like diffusion rate at leading edge of airfoil.

- Geometric constraints like maximum allowed thickness or thickness distribution limitations can be implemented to the tool.
- The tool would benefit a lot, if any significant effort is spent to reduce the computational cost of the tool. More specifically, if the optimization method is improved, it will significantly cut time and memory requirements.
- The tool can be fully integrated to optiSLang as an add-on for blade design.

Extending the capabilities of the tool

The tool can be further improved, by extending the current capability to handle different scenarios. From the developer's perspective, to implement these changes, one must overhaul the overall structure of the code.

- The tool currently can design the blade at any designed section that the GeometryGenerator allows. However, one specific streamline at a time can be used to evaluate the output responses for the blade, which are used for optimizing the blade. If multiple streamlines are to be used to assess the blade design, then the tool would be able to handle 3D constraints and objectives for the blade. Right now, the responses can only be defined for individual 2D sections at the intersection of chosen streamline and 3D blade shape.
- The inverse design framework can be modified all-together, so that at specific points during the optimization, the blade designs can be verified with 3D flow solutions. Those solutions can then be used to modify the blade shape.

Bibliography

[1] M. L. Henderson. “Inverse Boundary Layer Technique For Airfoil Design”. In: *Advanced Technology Airfoil Research, volume 1, part 1. [conference on development of computational codes and test facilities]*. 1979.

[2] Raja Ramamurthy and Wahid Ghaly. “Dual Point Redesign Of An Axial Compressor Airfoil Using A Viscous Inverse Design Method”. In: *ASME Turbo Expo 2010: Power for Land, Sea and Air*. 2010.

[3] Siva Kumaran Nadarajah. “The Discrete Adjoint Approach To Aerodynamic Shape Optimization”. PhD thesis. Stanford University, 2003.

[4] K. Aainsqatsi. *Schematic diagram illustrating the operation of a 2-spool, high-bypass turbofan engine, with LP spool in green and HP spool in purple*. 2020. URL: https://en.wikipedia.org/wiki/File:Turbofan_operation.svg.

[5] George Dulikravich, Brian Dennis, Daniel Baker, Stephen Kennon, Helcio Orlande, and Marcelo Colaco. “Inverse Problems in Aerodynamics, Heat Transfer, Elasticity and Materials Design”. In: *International Journal of Aeronautical and Space Sciences* 13 (Dec. 2012), pp. 405–420. DOI: 10.5139/IJASS.2012.13.4.405.

[6] Kasra Daneshkhah and Wahid Ghaly. “Aerodynamic Inverse Design for Viscous Flow in Turbomachinery Blading”. In: *Journal of Propulsion and Power* 23.4 (2007), pp. 814–820. DOI: 10.2514/1.27740. eprint: https://doi.org/10.2514/1.27740. URL: https://doi.org/10.2514/1.27740.

[7] Helmut Sobieczky. “Parametric Airfoils and Wings”. In: *Notes on Numerical Fluid Mechanics,* Vol. 68 (1998), pp. 71 –88.

[8] Brenda M. Kulfan. “A Universal Parametric Geometry Representation Method – “CST””. In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. 2007.

[9] Dev Rajnarayan, Andrew Ning, and Judd Mehr. “Universal Airfoil Parametrization Using B-Splines”. In: *2018 Applied Aerodynamics Conference*. 2018.

[10] D. A. Masters, N. J. Taylor, T. C. S. Rendall, C. B. Allen, and D. J. Poole. “Review of Aerofoil Parameterisation Methods for Aerodynamic Shape Optimisation”. In: *53rd AIAA Aerospace Sciences Meeting*. 2015.

[11] Patrice Castonguay and Siva K. Nadarajah. “Effect Of Shape Parameterization On Aerodynamic Shape Optimization”. In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. 2007.

[12] *Airfoil Optimization Using Area-Preserving Free-Form Deformation*. Vol. Volume 1: Advances in Aerospace Technology. ASME International Mechanical Engineering Congress and Exposition. V001T01A013. Nov. 2015. DOI: 10.1115/IMECE2015-50904. eprint: https://asmedigitalcollection.asme.org/IMECE/proceedings-pdf/IMECE2015/57342/V001T01A013/2491830/v001t01a013-imece2015-50904.pdf. URL: https://doi.org/10.1115/IMECE2015-50904.

[13] Thomas W Sederberg and Scott R Parry. “Free-form deformation of solid geometric models”. In: *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 1986, pp. 151–160.

[14] David Ives and Robert Zacharias. “Conformal mapping and orthogonal grid generation”. In: *23rd Joint Propulsion Conference*. DOI: 10.2514/6.1987-2057. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.1987-2057. URL: https://arc.aiaa.org/doi/abs/10.2514/6.1987-2057.

[15] *Turbine Airfoil Design Optimization*. Vol. Volume 1: Turbomachinery. Turbo Expo: Power for Land, Sea, and Air. V001T01A055. June 1996. DOI: 10.1115/96-GT-158. eprint: https://asmedigitalcollection.asme.org/GT/proceedings-pdf/GT1996/78729/V001T01A055/4217213/v001t01a055-96-gt-158.pdf. URL: https://doi.org/10.1115/96-GT-158.

[16] Brian Dennis, Igor Egorov, Zhen-Xue Han, George Dulikravich, and Carlo Poloni. “Multi-objective optimization of turbomachinery cascades for minimum loss, maximum loading, and maximum gap-to-chord ratio”. In: *8th Symposium on Multidisciplinary Analysis and Op-*

timization. DOI: 10.2514/6.2000-4876. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2000-4876. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2000-4876.

[17] Pasquale Cardamone. "Aerodynamic Optimisation of Highly Loaded Turbine Cascade Blades for Heavy Duty Gas Turbine Applications". PhD thesis. University of the German Armed Forces Munich, 2006.

[18] *Two-Dimensional Airfoil Shape Optimization Using Highly Differentiable Splines and Evolution Strategies*. Vol. Volume 2C: Turbomachinery. Turbo Expo: Power for Land, Sea, and Air. V02CT45A001. June 2016. DOI: 10.1115/GT2016-56040. eprint: https://asmedigitalcollection.asme.org/GT/proceedings-pdf/GT2016/49712/V02CT45A001/2429476/v02ct45a001-gt2016-56040.pdf. URL: https://doi.org/10.1115/GT2016-56040.

[19] Dirk Büche. "Multi-Objective Evolutionary Optimization of Gas Turbine Components". PhD thesis. SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZÜRICH, 2003.

[20] Ignacio Santín, Carles Pedret, and Ramón Vilanova. *Control and Decision Strategies in Wastewater Treatment Plants for Operation Improvement*. Modfied from the figure on Page 116. https://doi.org/10.1007/978-3-319-46367-4: Springer International Publishing, 2017.

[21] Rainer Storn and Kenneth Price. "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces". In: *Journal of Global Optimization* 11.4 (1997), pp. 341–359. ISSN: 1573-2916. DOI: 10.1023/A:1008202821328. URL: https://doi.org/10.1023/A:1008202821328.

[22] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4.

[23] M. Dorigo, M. Birattari, and T. Stutzle. "Ant colony optimization". In: *IEEE Computational Intelligence Magazine* 1.4 (2006), pp. 28–39.

[24] Dervis Karaboga. *An idea based on honey bee swarm for numerical optimization*. Tech. rep. 2005.

[25] A. Safari and H. G. Lemu. “Optimum Nurbs Curve Fitting For Geometry Parameterization Of Gas Turbine Blades’ Sections; Part I: Evolutionary Optimization Techniques”. In: *ASME 2012 International Mechanical Engineering Congress & Exposition*. 2012.

[26] A. Safari and H. G. Lemu. “Optimum Nurbs Curve Fitting For Geometry Parameterization Of Gas Turbine Blades’ Sections; Part Ii: Swarm Intelligence Techniques”. In: *ASME 2012 International Mechanical Engineering Congress & Exposition*. 2012.

[27] Neil W. Bressloff Thomas R. Barrett and Andy J. Keane. “Airfoil Design and Optimization Using Multi-Fidelity Analysis and Embedded Inverse Design”. In: *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2006.

[28] George R. Anderson and Michael J. Aftosmis. “Adaptive Shape Control for Aerodynamic Design”. In: *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. DOI: 10.2514/6.2015-0398. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2015-0398. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2015-0398.

[29] Mark Drela. *MISES - Software for Design and Analysis of Turbomachinery Blading*. 2020. URL: https://tlo.mit.edu/technologies/mises-software-design-and-analysis-turbomachinery-blading.

[30] Eleftherios I. Amoiralis and Ioannis K. Nikolos. “Freeform Deformation Vs B-spline Representation In Inverse Airfoil Design”. In: *8th Biennial ASME Conference on Engineering Systems Design and Analysis*. 2006.

[31] Ioannis Giagkiozis. “Nonconvex Many-Objective Optimisation”. PhD thesis. University of Sheffield, 2012.

Appendix A

Tool Flowcharts

A.1 Simplified Representation of optiSLang Project

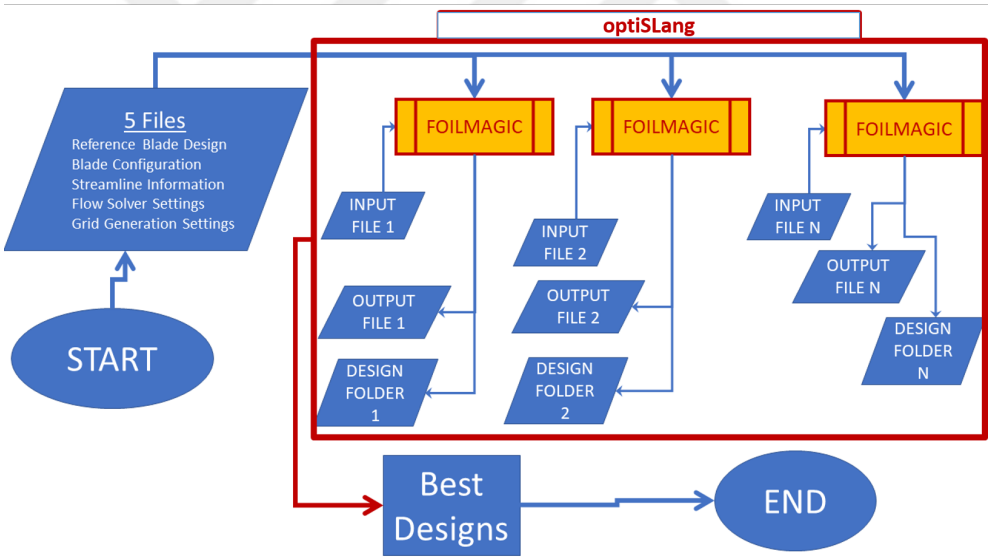


Figure A.1: Relation of Optimization Scheme and Single Design Evaluation

A.2 Optimization Schemes within optiSLang

A.2.1 Direct Optimization

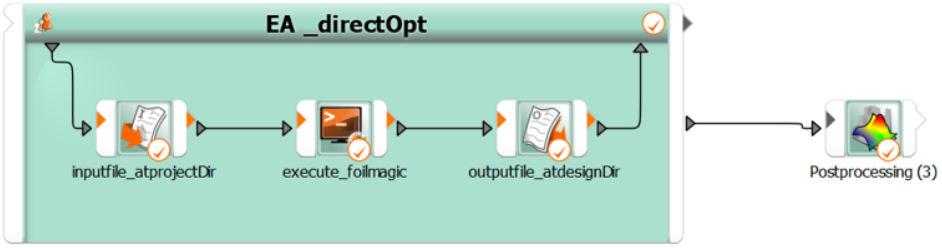


Figure A.2: Direct Optimization in optiSLang to Find Best Designs

A.2.2 Optimization on Metamodel

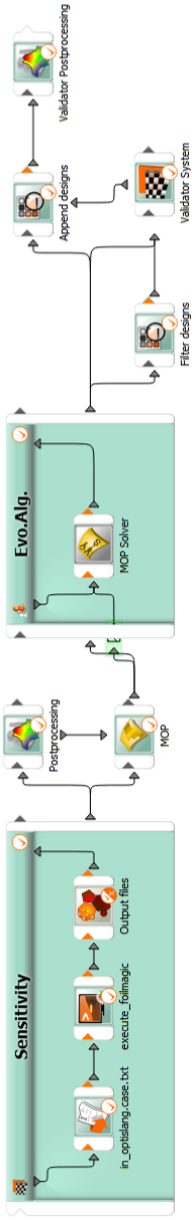


Figure A.3: Optimization on Metamodel in optiSLang to Find Best Designs