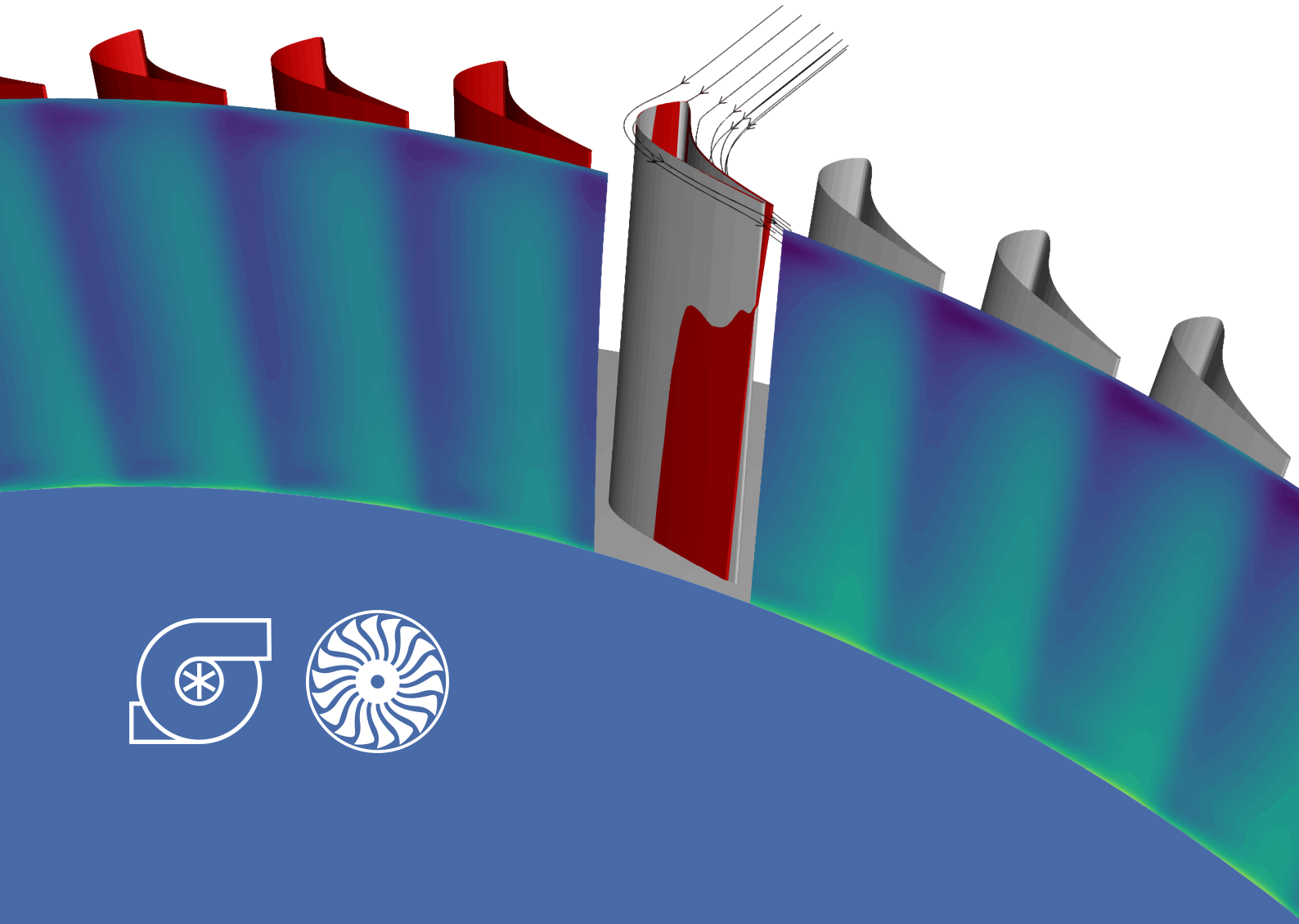


Adjoint-based 3D Shape Optimization for Turbomachinery Applications

P. Garrido de la Serna

TU Delft

Adjoint-based 3D Shape Optimization for Turbomachinery Applications

by

Pablo Garrido de la Serna

to obtain the degree of Master of Science in Aerospace Engineering
at the Delft University of Technology,
to be defended publicly on Thursday December 12, 2019 at 2:30 PM.

Student number:	4725131	
Project duration:	January 2019 – December 2019	
Supervised by:	Dr. Ir. M. Pini,	TU Delft
	Ir. N. Anand,	TU Delft
Thesis committee:	Dr. Ir. M. Pini,	TU Delft, supervisor
	Ir. N. Anand,	TU Delft, supervisor
	Prof. Dr. Ir. P. Colonna,	TU Delft
	Dr. Ir. A. H. van Zuijlen,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Delft University of Technology

Dedicada a mi abuelo Jaime, D.E.P.

Acknowledgements

This Master Thesis is the culmination of two and a half years of work that were accompanied by great and some tough moments that would have not been possible to achieve without the constant support from people that were there every time one needed them since I arrived to The Netherlands.

First of all, I want to thank my supervisor, Matteo, for purposing me this tough but at the same time challenging project that I have been working on for the past months. It is his motivation and enthusiasm what has enabled the execution of this project. Undoubtedly, all the achievements done in this Thesis would have not been possible without the help of my daily supervisor Nitish, who always had a solution to all the problems that arose during the course of this Thesis. His availability, motivation and determination were key to the accomplishment of this project. Thank you both.

También, el apoyo constante de mis padres, Salvador y Teresa, mis hermanos, Concha y Mario, mi abuela Monsi y toda mi familia durante estos dos años y medio ha sido vital. De ellos he aprendido a dar lo máximo de mí mismo, a continuar por muy difícil que parezca el camino y a no rendirme nunca. Gracias.

The journey that started in September 2017 in Delft would not have been the same without the people that I have met here. To Roberto, my housemate, for constantly supporting, motivating and teaching me to be a better chef than the one that I initially was. To Filippo, Johannes and Tomás for all those coffee breaks, dinners and funny moments shared together. To Steph, Tony, Jaime and Tino for all the good moments no matter where, Warsaw, Dublin or Delft. To Antonio, Íñigo, Ricardo, Marta and Hendrik for all the good times at the office. To Craig, Paula and Zoe for a great first year in Delft. To all the rest that I have met during these two years and have made my time at The Netherlands better. Thanks.

Por último, a Javi, Alberto, Manu, Nacho y todos mis amigos que desde España me han apoyado constantemente y cuya pasión y determinación por mejorar día tras día admiro y persigo. Gracias.

Pablo Garrido de la Serna
Delft, December 2019

Abstract

In order to reduce the carbon foot-print of turbo-generators to global warming, a step-change in the design process is needed. Extensive CFD simulations coupled with optimization algorithms are required, resorting to novel techniques to capture the complex flow phenomena occurring in the passage. However, the computation of the optimization gradients and the imposition of geometrical constrains is computationally expensive and non-trivial due to the large number of design variables. In the 1980s, the adjoint method arose as an alternative technique to compute the sensitivities at a cost independent of the number of design variables. This technique is extremely popular in external aerodynamic applications. However, its full potential is yet to be exploited to internal flows. The use of an adjoint solver, together with a surface parametrization technique based on traditional blade design parameters, presents a unique opportunity towards a fully-automated design methodology based on high-fidelity models for turbomachinery applications. Stemming from the above considerations, the aim of this thesis is to develop a common open-source numerical framework for the optimization of both axial and radial turbomachinery. In order to achieve so, the open-source CFD suite *SU2*, that includes an adjoint solver, is coupled with *ParaBlade*, an open-source blade parametrizer based on traditional turbomachinery design variables. The proposed methodology is successfully applied in an axial turbine stator, where a reduction of 7.59% in the entropy generation across the passage is achieved. The study performed in a mixed-flow turbine rotor highlights the need to parametrize the hub and shroud surfaces. However, its optimization revealed a reduction in the objective function value, showing the robustness of the proposed methodology in two types of turbomachinery.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

2D	Two-dimensional
3D	Three-dimensional
AD	Algorithmic/Automatic Differentiation
APU	Auxiliary Power Unit
BRep	Boundary Representation
CAD	Computer Aided Design
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
CPU	Central Processing Unit
FD	Finite Differences
FFD	Free-Form Deformation
JST	Jameson-Schmidt-Turkel
LE	Leading Edge
NURBS	Non-Uniform Rational B-Spline
RANS	Reynolds-Averaged Navier-Stokes
RBF	Radial Basis Functions
RMSE	Root-Mean-Square Error
SLSQP	Sequential Least SQuares Programming
SST	Shear Stress Transport
TE	Trailing Edge

Subscripts

0	Initial
$1-4$	Thickness distribution law points
AD	Relative to discrete adjoint
α	Relative to design variable
CAD	Relative to CAD
CFD	Relative to CFD
def	Deformed
FD	Relative to finite differences
gen	Generation
i, j	Numbers
in	Relative to inlet
is	Isentropic
new	New
opt	Optimum
out	Relative to outlet
ref	Reference
$S1 - S3$	Surface points
surf	Relative to surface coordinates
ax	Relative to axial/meridional direction
tan	Relative to tangential direction
t	Total

Greek letters

α	Design variable
β	Flow angle
Δ	Difference
δ	Step
ϵ	Blade wedge angle
η	Efficiency
ν	Poisson's ratio
ρ	Density
θ	Blade metal angle
ξ	Stagger angle
ζ_K	Kinetic energy loss coefficient
ζ_P	Total pressure loss coefficient

Symbols

$\mathbf{D}$	Distance
d	Thickness distribution law
$\mathbf{e}$	Unit vector
E	Young's modulus
ε	Tolerance
f	Relaxation factor
G	Constrain
$\mathbf{G}$	Flow solver iterations

h	Enthalpy	s	Entropy
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	Cartesian unit vectors	T	Temperature
J	Objective function	$\mathbf{U}$	Flow vector
k	Stiffness matrix	u, v	NURBS parametric coordinates
L	Lagrangian	$\bar{\mathbf{U}}$	Adjoint sensitivity
$\mathbf{lb}$	Lower bounds	$\mathbf{ub}$	Upper bounds
$\mathbf{M}$	Mesh deformation matrix	u_b	Blade speed
$\dot{m}$	Mass flow	$\mathbf{u}$	Displacement vector
N	Number of	$\mathbf{V}$	Volume
$\mathcal{N}$	Set of points	v	Velocity
$\mathbf{n}, \mathbf{t_1}, \mathbf{t_2}$	Local normal-tangential vectors	v_0	Spouting velocity
$\mathcal{O}$	Order of magnitude	$\mathbf{X}$	Grid points
p	Pressure	$\bar{\mathbf{X}}$	Mesh sensitivity
r	Radius	x_S, y_S, z_S	Surface points
$\mathbf{R}$	Flow residuals	x, y, z	Cartesian coordinates
$\mathbf{S}$	Surface parametrization method	y_p	Blade pitch

Chapter 1

Introduction

The increasing design requirements for future gas turbine engines, especially emphasizing on reducing pollutant emissions and noise, has irrevocably put the design of its underlying turbomachinery into the scope of research in the latest decades. In the past, the design of turbomachinery devices, in which compressors, fans, turbines and pumps can be included, was driven by the use of empirical correlations, charts and trial-and-error techniques where frequently the experience of the designer was key for the success of the device. However, due to the rapid development of computational resources and capabilities, this design process has nowadays resorted to an increased use of numerical methods to compute performance and releasing the process from the designer's experience, which is still needed for the assessment of the viability of the design, but in much less extent throughout the whole design process.

The introduction of Computational Fluid Dynamics in turbomachinery applications clearly differentiates a before and an after in shape design problems. New designs can now be rapidly assessed by discretizing the fluid domain and the use of numerical simulations, which in turn will give a better insight in the underlying flow phenomena and will save costs in the whole process. Early works on turbomachinery CFD are dated from the 1960s, and have evolved from simple blade-blade-throughflow models to two- and three-dimensional analyses [1]. Latest developments in the last decade have achieved multi-stage and unsteady simulations by the use of different methods such as mixing-planes and harmonic-balance, to mention some, allowing for a more extensive and accurate analysis of the flow through this type of machines, which are inherently unsteady and the interactions between its different components are of great importance [2].

However, one of the points of main interest of CFD is its coupling with numerical optimization techniques, as it can automatize the design process. Optimization problems are comprised by different parts. First of all, the main goal of the optimization needs to be established. This can be done by the definition of an analytical objective function. For instance, in wing design problems the objective could be to minimize the drag produced by a wing. In the turbomachinery field, one typical objective function is to reduce the entropy generation within a flow passage, which will in turn increase the thermodynamic efficiency of the stage. Afterwards, the constraints need to be defined. Another example of this could be that the mass flow rate needs to remain constant through the passage, as turbomachinery components are usually part of bigger systems for which some design specifications are given. Now, the key point is that the flow solution obtained by the aforementioned CFD tool can be used to evaluate the objective function value, instead of relying in extensive and time consuming experimental simulations.

In order to set up an optimization, there are several types of methods to minimize a given objective function, for which one can find different benefits and drawbacks. These are commonly classified into gradient-based and gradient-free or stochastic methods. For the former, the optimizer uses information on how the objective function changes with respect to its design variables in order to move towards a minimum. However, these techniques' success widely depends on the choice of the initial starting point and on how the problem is bounded and constrained. In gradient-free optimizations, on the other hand, the global minimum is searched. Nevertheless, stochastic models are computationally expensive, as they

require a large number of flow evaluations to converge [3] and are almost prohibitive for complex design problems with large number parameters, reason why they are usually deemed as inappropriate for turbomachinery-related problems.

For gradient-based techniques, the objective function's gradient with respect to the design variables has to be computed in each design step in order to provide the optimizer with the right information. Finite differences can be employed to compute this value; however, this comes at a cost of computing one flow solution per design variable, which again is prohibitive for turbomachinery problems as these are usually formed by design parameters in the order of tens and hundreds, depending on the parametrization technique. The adjoint method [4] arose in the late 1980s as a computationally-efficient way for calculating the objective function's gradient at a cost almost independent of the number of design variables chosen and its use is widely spread in external aerodynamic problems, but its application to internal flows and in particular to turbomachinery problems is yet to be explored.

Nevertheless, the gradients stemming from the adjoint solver cannot be directly used if the geometry is parametrized such that the surface coordinates depend on other variables. In this case, there will be two different set of gradients, one corresponding to the adjoint solver and the other to the underlying parametrization technique which need to be assembled before being passed to the optimizer. This operation is non-trivial and it is of great importance as it will determine how accurate the total gradient is.

In order to have an efficient and robust optimization routine, the geometry has to be correctly parametrized and represented so that a rich design space is available. Traditionally, the blade's surface points were directly, or indirectly via the use of perturbation functions, used as design variables. However, this resulted in a large number of design parameters which, for the increasingly demanding computational costs associated with the CFD simulations, made impossible to obtain results in a reasonable amount of time. In addition, the imposition of the constrains is non-trivial and has to be done by means of complex geometrical relationships. Furthermore, from a designer's point of view the resulting geometry consists of a cloud of surface points which without a further post-processing cannot be really evaluated.

One approach that started to gain popularity in the past years is the use of traditional turbomachinery parameters as design variables which define the blade's geometry, called CAD-based parametrization. This technique allows not only to have a reduced number of design variables which is translated into shortened design cycles, but also to have a more intuitive insight on how the geometry is being changed. However, this comes at a cost which is a non-trivial implementation and having a slightly reduced design space if compared to the other techniques mentioned in previous paragraphs.

Based on what was exposed in the preceding paragraphs, the use of the adjoint method together with an underlying CAD-based geometry parametrization is deemed to be the most appropriate towards the formulation of a robust design methodology based on high-fidelity models. Therefore, *the objective of this work is to couple an open-source in-house CAD-based parametrization tool with the open-source CFD suite SU2 relying on the adjoint method*. Special emphasis will be made on the robustness of the method, as it will be tested in both axial and radial turbomachinery, as well as how the different tools available in *SU2* are suited for this type of problem.

Research goal and motivation

In the last decade, new designs where fully- or hybrid-electric propulsion arose as an alternative to traditional aero engines has been a focus of research in both academia and industry. However, with the available technologies, its use is restricted to small-scale applications. On the other hand, in power generation, more than 80% of the electricity generated worldwide is obtained from gas and steam turbines, and it is still in the roadmap of many organizations to rely on turbomachinery in the development of low-carbon energy technologies [5]. Therefore, it is of great importance to have fast, efficient and robust design methods to meet with the increasingly more restrictive environmental regulations set for the upcoming decades.

Turbomachinery shape design problems are composed, commonly, of a large number of design variables that, together with the complex flow phenomena taking place in the internal passages, are time consuming and make necessary to move towards faster techniques. The adjoint method emerged as a computationally-efficient tool to be used in shape optimization, and its application to external aerodynamic problems is widely common. However, its full potential is yet to be exploited to internal flow applications.

Part of the lack of use of the adjoint method in the turbomachinery field can be attributed to the absence of robust design tools. Even though adjoint solvers are available in many flow solvers, such as OpenFOAM [6] or ANSYS Fluent [7], there is still a problem that needs to be tackled: how to parametrize the geometry.

As it was mentioned before, the complexity of turbomachinery design requires for an efficient parametrization tool able to describe a rich design space with the least amount of design variables as possible. It is here where CAD-based techniques stand out, with the added benefit of including traditional turbomachinery design parameters that allow for a better understanding of how the geometry is built and changing during the design process and, equally important, for an effective imposition of the geometrical constraints usually required in this type of problems.

Even though there are have been previous studies on the coupling of CAD-based techniques and adjoint solvers [8–19], it is observed that there is not a common methodology established for turbomachinery-related problems, nor there is a unified framework for different types of turbomachines. Most of the research done in the past assumes that the gradients provided by the adjoint solver are correct, without benchmarking those against other techniques to compute the aforementioned values. Furthermore, it is observed that none of the resulting frameworks built are released to the public, which is something considered as essential for the development and research on this type of techniques.

From the ideas presented above, the focus of this work will be on the coupling of the open-source CFD suite *SU2*, with an in-house CAD-based geometry parametrization toolbox, *ParaBlade*. *SU2* is chosen as the underlying suite as: *i)* it is an open-source project, *ii)* it includes several modules which are extremely beneficial for shape optimization, such as discrete adjoint solvers and mesh deformation algorithms and *iii)* its robustness has already been tested in axial and radial turbomachinery problems. The coupling of both *SU2* and *ParaBlade* will result in an optimization module which will be part of the CAD-based geometry tool. The methodology proposed is tested in two different types of turbines, axial and radial, and its different advantages and limitations will be analyzed in detail.

The research objective can formally be formulated as:

"The research objective is to develop a numerical framework for the shape optimization of axial and radial turbomachinery resorting to the adjoint method and a CAD-based geometry parametrization."

From the research objective, a the main research question can be posed

Is it possible to couple the in-house CAD-based geometry parametrization tool ParaBlade with the open-source suite SU2 to perform aerodynamic shape optimization for axial and radial turbomachinery?

which will in turn lead to different sets of sub-questions,

- *What is the improvement, in percentage, of the objective function value for an axial turbomachinery blade?*
  - *Is the spring-analogy-based algorithm in SU2 robust to perform volumetric mesh deformation on axial blades?*
  - *What is the RMSE between the gradients computed via the discrete adjoint to those from finite differences?*

  - *What is the improvement, in CPU-time, between computing the gradients via the discrete adjoint to that from finite differences?*
  - *Are the geometrical constraints respected throughout the optimization process?*

- *What is the improvement, in percentage, of the objective function value for a radial turbomachinery blade?*
  - *Is the spring-analogy-based algorithm in SU2 robust to perform volumetric mesh deformation on radial blades?*
  - *What is the RMSE between the gradients computed via the discrete adjoint to those from finite differences?*
  - *What is the improvement, in CPU-time, between computing the gradients via the discrete adjoint to that from finite differences?*
  - *Are the geometrical constraints respected throughout the optimization process?*

Originality of the work

The research objective and motivation described in the section above help understanding the lack of uniformity and especially accessibility, in terms of publicly-available tools for the design of turbomachinery components, relying on the adjoint method. Therefore, the originality of the present Thesis work can be summarized as follows:

- The shape optimization module that is created in this Thesis will be part of the open-source geometry parametrization tool *ParaBlade*. Therefore, its source code will be publicly available and open for further contributions.
- The computation of the total sensitivities is shown in a clear manner. Many of the papers analyzed do not show any validation of the adjoint gradients.
- To the knowledge of the author, all the observed previous work in adjoint-based shape optimization of turbomachinery has been performed uniquely considering one type of turbomachine. This analysis is extended to both axial and radial turbomachinery.
- Previous works have used *SU2* as the main tool for shape optimization, especially in external aerodynamics. However, for turbomachinery applications the test cases available are limited and mainly two-dimensional. Therefore, it is also a goal for the author to contribute to the development of this community by extending the turbomachinery test cases portfolio to the optimization of complex three-dimensional geometries.

Outline of the thesis

The present Thesis is divided into several chapters in order to ease the reading and understanding of the problem that is about to be tackled. **Chapter 2** summarizes all the previous work related to adjoint-based turbomachinery shape design, with a special emphasis on methods where CAD-based geometry tools have been used. Then, **Chapter 3** relates how the different modules that compose the aerodynamic shape optimization problem are coupled. **Chapter 4** describes the way this coupling is done in a numerical framework together with all the features that the resulting toolbox includes. In order to prove that the proposed methodology is functional, **Chapter 5** introduces two different test cases for which the results, in terms of computational and aerodynamic performance, are shown in **Chapter 6**. Finally, the main conclusions and points of future work will be summarized in **Chapter 7**.

Chapter 2

Background

In this chapter, a review of the literature on turbomachinery aerodynamic shape optimization problems is done. As argued in the previous Chapter, the focus of this work is on gradient-based techniques, in particular by resorting to the adjoint method. A schematic representation of the fluid-dynamic design chain in turbomachinery shape optimization problems is shown in Figure 2.1.

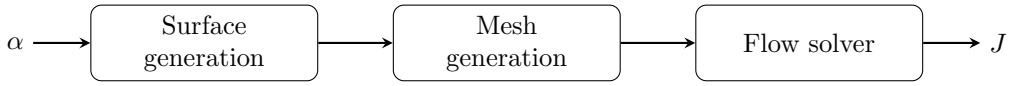


Figure 2.1: Schematic fluid-dynamic design chain in turbomachinery shape optimization problems.

The starting point of a shape optimization problem is to have a robust, efficient and fast geometry parametrizer. A review of the methods currently used will be done in Section 2.1. Then, the fluid domain needs to be spatially discretized, and currently two options are available: either meshing for each of the design steps within the optimization, or deforming an existing mesh. The benefits and drawbacks of such techniques will be critically analyzed in Section 2.2. Then, the objective function is evaluated, as argued in the previous Chapter, by the use of CFD.

Gradient-based methodologies require information about how the objective function changes with respect to the design variables. Therefore, the different terms needed to compute the sensitivity of the objective function will be explained in Section 2.3. The focus of this work will be in the adjoint method, and within this category one can find two approaches: the discrete or the continuous adjoint method. The adjoint solver is responsible for computing the gradient of the objective function, required to calculate the optimal search direction in a gradient-based optimization. However, this is not the only term necessary for the computation of the total gradient, therefore special emphasis will be put on how this term is assembled.

Section 2.4 will make a review of some outstanding works which have employed the open-source CFD tool *SU2*. Finally, Section 2.5 includes some relevant test cases where the coupling of CAD- and adjoint-based tools has been treated.

Figure 2.2 shows an overview of the previously mentioned modules. For each of them, the most commonly employed methods are schematized.

2.1 Geometry parametrization techniques

One of the most important parts in an optimization framework is the geometry modelling. Generally, the requirements are that the method should be *i)* fast, *ii)* robust, *iii)* efficient and *iv)* capable of representing a rich design space with the minimum number of design variables as possible.

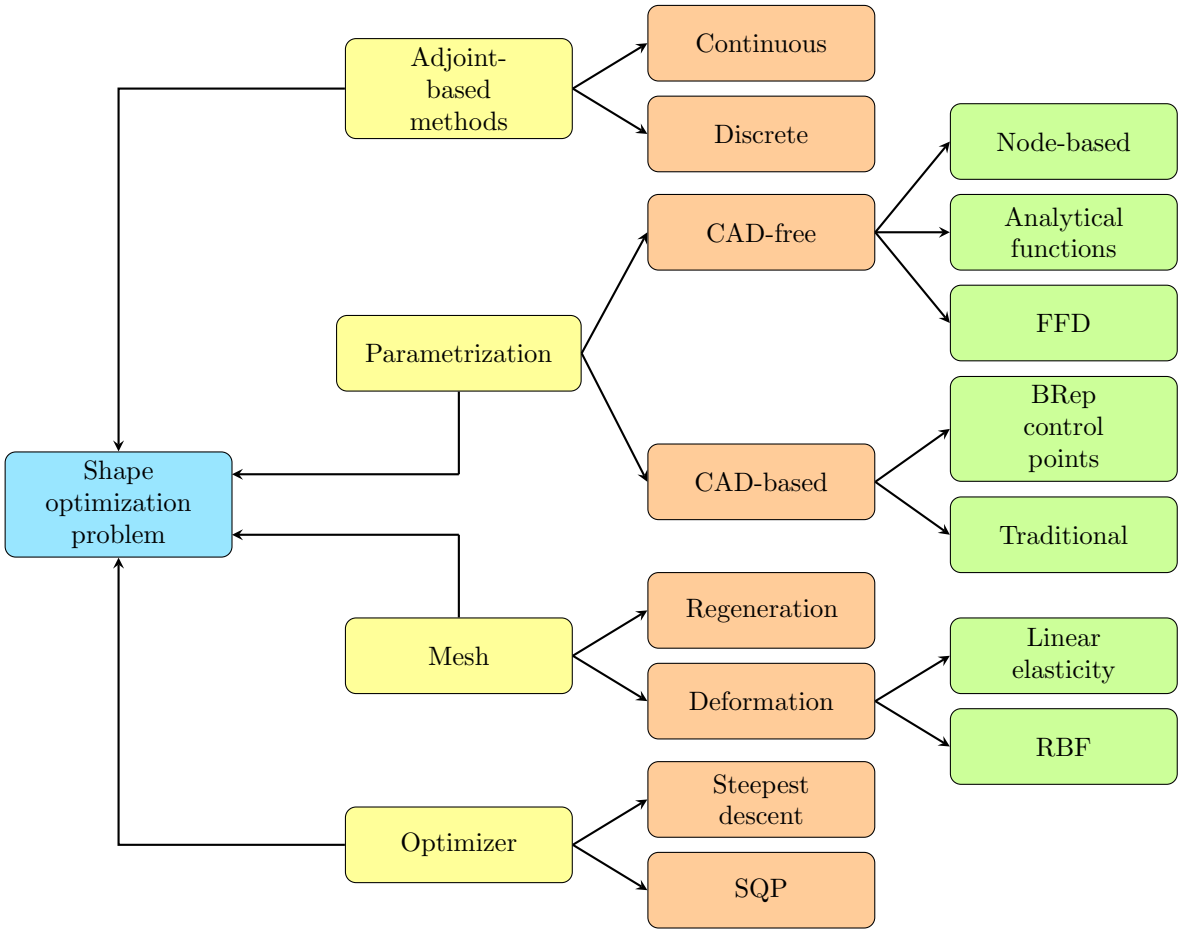


Figure 2.2: Common modules required to enable adjoint-based shape optimization. Yellow: blocks; orange and green: available methods.

As demonstrated by Anand et al. [20], the type of parametrization used when modelling the geometry has an impact on the optimal solution. In this work, two different parametrization techniques are employed which yield distinct optimal shapes. Therefore, it is not a trivial task to select the geometry modeller to use in an optimization problem.

Two different methods are distinguished under this block:

- **CAD-free:** use directly, or indirectly through perturbation functions, the mesh surface nodes as design variables.
- **CAD-based:** use of Computer Aided Design to parametrize the geometry: through the use of traditional design parameters or the control points of CAD models as design variables.

Both methods are analyzed in detail in the following subsections.

2.1.1 CAD-free

Within the CAD-free category, three main techniques are seen in the literature, for both 2D and 3D cases, that will be analyzed separately:

- Node-based approach.
- Analytical approach.
- Free-form deformation.

Node-based approach

In the node-based or discrete approach, the coordinates of the surface grid nodes are used as design variables. This represents one of the richest design spaces available, as the nodes are able to move individually in the curve (2D) or surface (3D) normal direction.

Amongst the publications where the mesh coordinates are directly used as design variables, one can find the work by Wu and Liu [21], which can be considered as one of the first researches where a RANS-based adjoint method was employed in turbomachinery shape design. In their work, a VKI turbine stator was successfully optimized where the flow constraints were directly included in the objective function, as penalties, while no mention to geometrical constrains is made.

However, there are several drawbacks that make this technique unattractive for aerodynamic shape optimization. Firstly, the implementation of the constraints is not straightforward and often will be based on complex distance calculations in the 2D or 3D domain. Secondly, due to the rich design space available it can lead to discontinuous shapes, which will require the application of smoothing shape functions difficult to implement [22]. Finally, the optimized shape is obtained in grid coordinate format, which will need a further transformation to other industry-standardized formats such as STEP, with the subsequent errors in the mapping of the coordinates. All the reasons given above make the node-based approach not suitable for complex problems like turbomachinery shape optimization.

Analytical approach

Similarly to the node-based method, the analytical approach adds shape functions linearly to the baseline shape. These shape functions are smooth functions that are based on a set of previous aerofoil designs [23]. Commonly found in the literature, one can find Hicks-Henne or Radial Basis Functions (RBFs).

Duta et al. [24] used perturbation functions to modify parameters defining the blade geometry, where the main goal was to prove the functionality of Automatic Differentiation on an existing CFD code, HYDRA [25], in order to obtain its discrete adjoint. The test case analysed is a NASA Rotor 37. Wang et al. [26, 27] develop a treatment of the adjoint mixing-plane for multirow designs based on steady RANS equations. In their work, three compressor test cases are studied, where for each of them the geometry is perturbed by the use of Hicks-Henne functions. Furthermore, Luo et al. [8] use Hicks-Henne functions to perturb the initial geometry of a turbine blade. The adjoint method is based on Euler flow equations. The authors conclude that, although the method proved to work, the Euler equations are very limited to provide useful information about the complex flow phenomena occurring in a turbine flow passage, pointing to the necessity to extend the method to RANS-based equations. Luo et al. [10] perform a multi-point shape optimization using the continuous adjoint method on a transonic compressor. The geometry is modified by using Hicks-Henne functions. Compared to their previous work exposed in the preceding paragraph, in this case the continuous adjoint equations are derived from viscous Reynolds-Averaged Navier-Stokes (RANS) equations. Walther and Nadarajah [11, 28] select a single stage axial compressor to be optimised using the discrete adjoint method. Hicks-Henne functions are selected to perturb the original geometry. The authors argue that, although no explicit geometrical constrains are applied, both leading- and trailing-edge radii are maintained throughout the optimization by controlling the placement of the perturbing bump functions. In a similar work, Walther and Nadarajah [29] highlight that instead of applying explicit geometrical constraints to the blade's thickness this can be ensured by prescribing aerodynamic constraints, such as keeping constant the blade loading, with the risk of violating mechanical requirements. Tang et al. [16] use the continuous adjoint method to optimize the performance of an transonic fan rotor, where both the geometry and the mesh are deformed employing a two-level mesh deformation method based on RBFs.

However, the drawbacks of this method are similar to those of the node-based approach: the imposition of the constraints is not straightforward and the optimized geometry needs to be converted into an industry-standardised format in order for further post-processing.

Free-form deformation

In this case, the shape is modified by moving the control points of a grid, or box, in which the blade surface is included. Ultimately, it treats the model as a rubber that can be twisted, bent, tapered, compressed or expanded while still maintaining its topology [23].

Free-form deformation is commonly regarded as a technique to modify the shape of wings in optimization problems. However, its application to turbomachinery, especially in the 3D domain, is still not widespread. Anand et al. [20] performed an optimization of a 2D turbine blade profile by using FFD. The authors address the complexity of applying geometrical constraints, especially mentioning the minimum trailing edge thickness, when using this type of technique. Similarly, Vitale [30] uses FFD boxes to validate the gradients from the discrete adjoint solver in the open-source software *SU2* [31]. Afterwards, an optimization of the entropy generation across the passage is achieved, overcoming the issue of maintaining the TE shape by leaving this region of the blade out of the FFD box.

Nevertheless, FFD boxes do not allow for a direct control of the geometry, as the design variables are their control points, and can produce unfeasible geometries, especially from a manufacturing point of view. They also require the previous step of creating the box, and the quality of the results will greatly reside on how the boxes are created and able to morph the baseline geometry.

From the above discussion it becomes evident that for CAD-free techniques the constraint imposition is not trivial, and usually authors tend to include aerodynamic constrains in form of penalty functions in the objective function in hope of maintaining certain geometrical blade characteristics. This is caused by the large number of design variables that the usage of these methods suppose. Furthermore, from a designer's point of view the geometry is in form of a cloud of points, either directly surface coordinates or control points, or coefficients in weight functions that are non-self explanatory and not intuitive to track throughout the optimization. Furthermore, an extra final step to convert the geometry to industry-standard formats, such as STEP, is needed, which can produce a loss of information and a departure from the optimal point.

2.1.2 CAD-based

All the above approaches are deemed as mesh-based methods as the grid coordinates were used directly (node-based) or indirectly (analytical and FFD) as design variables. The final optimized shape then needs to be converted to a standardised format for further post-processing or manufacturing of the design, and the control of how the design variables are changing through the optimization loop is not intuitive.

For several years, Computer Aided Design models have been widely used in industry. A typical file format resulting from CAD models is STEP (*.stp* or alternatively *.step*), and can be used in Finite Element Analysis tools to evaluate the stresses and low-life locations of the design. Furthermore, it is the common file format for the final manufacturing process.

Having the final optimized geometry in CAD format, as it can be seen, is really convenient as it is a step closer towards a fully-integrated design framework, where the different disciplines that set the design of turbomachinery blades can be considered. Therefore, the interest of keeping the CAD model throughout the optimization has grown in the recent years.

Nevertheless, within the geometry-based methods, two approaches can be followed: *i)* the use of traditional design parameters, such as blade metal angles, thickness distributions, amongst others, as design variables or *ii)* the use of control points of the Boundary Representation (BRep) as design variables. The following subsections will make an overview of both methods.

Traditional approach

Following this approach, traditional turbomachinery design parameters are used as design variables in order to define the aerofoil or blade shape. These parameters are in turn used to specify the control

points of polynomial and spline curves. For the latter, the most widely used are the non-uniform rational B-spline (NURBS) curves (2D) or surfaces (3D) [32], which can represent quadric primitives (cylinders, cones) as well as free-form geometry [23], and are well-established as standard in CAD software packages.

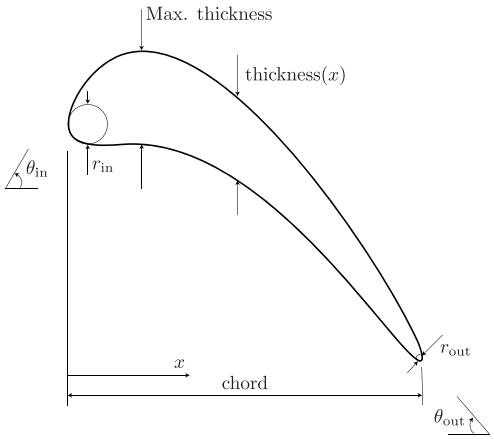


Figure 2.3: Parametrization of a blade section using traditional design parameters. Adapted from [33].

Figure 2.3 shows how a blade section is commonly built using this approach. Traditional parameters define the position of the leading and trailing edge. Then, the thickness distribution is built by constructing curves, usually NURBS-based, which reveal the variation with the chordline. In order to build a three-dimensional blade, it is a common practice to stack blade profiles in the span-wise direction.

The clear benefit of using such approach is that the geometry is generated in an intuitive way. Parameters that are commonly self-explanatory (such as outlet radius, stagger angle, etc.) generate the geometry, and allow for an efficient imposition of the geometrical constrains by just leaving them outside of the optimization loop. It is also useful to keep track of how the geometry is morphing, as the design vector will directly reveal how, for instance, the inlet radius is changing without the need of a further visualization.

The main drawbacks of this method is that the design space is limited when compared to mesh-based techniques, where the grid nodes are used directly as design variables therefore allowing for more freedom. Furthermore, its implementation in an optimization tool is not straightforward, and requires the formulation of complex functions especially to ensure geometrical continuity. Finally, the ability of representing an ample design space requires the designer's experience so as to chose the correct parameters.

Boundary representation control points

An alternative to CAD-based parametrization, where the parameters in the model's construction algorithm are changed, is working directly with the control points of the Boundary Representation (BRep). The BRep is the resulting geometry which, generally, consists of a set of NURBS curves (2D) or patches (3D).

Instead of needing to define a parametrization for the aerofoil or blade based on traditional design parameters, this approach uses the control points of the BRep as design variables, allowing their displacement in the normal direction. An example of a perturbed 3D NURBS patch can be seen in Figure 2.4, where the top left and bottom right control points have been perturbed upwards and downwards respectively. This, in turn, has some advantages and drawbacks.

The main advantage of this approach is that it is CAD-vendor independent as this method requires a generic CAD-file [19]. Therefore, the computation of the geometric gradients with respect to the design variables can be done without access to the geometry generation algorithm. Secondly, this approach

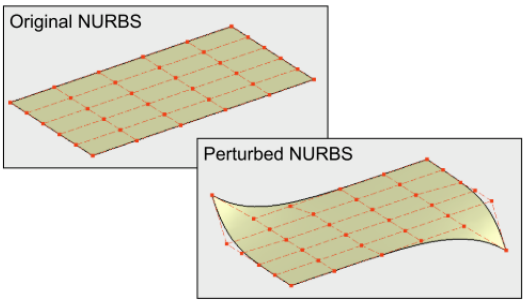


Figure 2.4: NURBS patch morphing [12].

also represents a rich design space, and the final product of the optimization is already in CAD format. Nevertheless, similarly to node-based and FFD approaches, by having control points as design variables the application of the constraints is not straightforward, because not only constraints need to be applied in the geometry itself, such as minimum thickness or limits in the twisting angles, but also in each of the NURBS curves and/or patches to ensure geometrical continuity.

In summary, the complex geometries found in turbomachinery components require for an efficient and intuitive geometry parametrization. While using the boundary representation might be convenient from an implementation point of view, it is not deemed as satisfactory for the present work, as it is again composed by a large number of design variables which can hinder the imposition of geometrical constraints. The traditional approach, on the other hand, allows for a much extensive control of the geometry which is key for the upcoming component of the shape optimization process.

2.2 Computational domain

In order to obtain a flow solution, the CFD solver needs to be provided with a computational volumetric mesh. The aforementioned grid needs to be accurate enough to be able to represent all the complex flow phenomena that can occur in turbomachinery passages, especially in regions close to the blade surface (and/or hub and shroud walls) where the boundary layer starts to develop.

In optimization problems where the design variables control the geometry, the shape will change in each optimization step. Therefore, in order to have a new evaluation of the flow solution, the CFD solver requires as input a volumetric mesh, which in this case differs to that of the previous iteration, as the geometry has been modified. This mesh, in turn, still needs to be robust and able to capture all the flow phenomena, maintaining the same characteristics as the initial one due to the fact that, if this does not occur, information can be lost in the optimization process.

Currently, two alternatives are available in order to represent these geometry changes in the volumetric grid: mesh regeneration or mesh deformation, which were already spotted in the previous section.

2.2.1 Mesh regeneration

Mesh regeneration, or re-meshing, consists of generating the mesh in each optimization step. This implies that the process has to be fully automated and integrated in the optimization chain, as minimum intervention from the designer is seeked.

In order to do so, a rapid and robust mesher is required, capable of generating good quality viscous meshes in three-dimensions as, in the end, the re-meshing process consists of solving partial differential equations which can be a greatly time consuming task, especially for unstructured meshes as argued

in [34].

In the works by Duta et al. [24] and Vasilopoulos et al. [33], an in-house geometry mesher named PADRAM [34] developed by Rolls-Royce is used and successfully integrated in the optimization loop. Furthermore, Mueller and Verstraete [15] employ an internal mesh generator to regenerate the grid in each step of an APU turbine optimization.

Amongst the benefits that grid regeneration offers, one can find an improved control of the boundary surfaces, such as the hub and shroud walls. Furthermore, it allows for large geometry changes. However, the drawbacks are that ensuring consistency between subsequent meshes is a hard task to be accomplished, especially with respect to achieving a similar grid density and boundary layer refinement next to the walls. This can even be worsened if the solution is not fully grid-independent, as it will lead to a loss of information between optimization steps.

2.2.2 Mesh deformation

Mesh deformation consists of morphing the initial grid in each of the optimization steps as determined by the new surface coordinates set by the optimizer. Therefore, meshing is only required at the first step of the optimization.

Common methods employed in mesh deformation are spring-analogy, linear-elasticity-based and RBFs. For the spring-analogy based algorithms, it is assumed that the mesh nodes are connected through springs [12], where a system of equations is built such that

$$k_{ij}\mathbf{u}_j = \mathbf{u}_i \quad \text{for } i = 1, ..., N_{\text{surf}}, \tag{2.1}$$

where k_{ij} is the stiffness matrix, and $\mathbf{u}_i$ and $\mathbf{u}_j$ are the prescribed and unknown displacements, respectively.

In case of linear-elasticity-based algorithms, the entire volumetric mesh is treated as a linear elastic material, and the unknown displacements are imposed as Dirichlet boundary conditions to the linear elasticity equations, similarly as done in Equation (2.1). However, these algorithms, as compared to spring-analogy, are able to perform better on meshes with very high-aspect ratio with boundary layer refinement, as highlighted in [12].

Similarly than in the analytical approach for CAD-free parametrizations (Section 2.1.1), RBFs are used to perturb the mesh node positions based on a prescribed deformation. Tsiakas et al. [13] use this type of analytical functions to deform the grid in an axial compressor shape optimization problem. Similarly, Walther and Nadarajah [28] employ RBFs to morph the mesh in an adjoint-based multi-point optimization of an axial compressor, where it is addressed the necessity of having an efficient mesh deformation tool for 3D shape optimization problems.

The main benefit of employing mesh deformation algorithms is that it saves the computationally expensive task of re-meshing. However, the mesh deformation process itself is not trivial and requires for a robust algorithm to ensure good quality results. It is observed in the literature that this step is commonly disregarded and only treated carefully in some works. In the same paper by Xu et al. [12], a robust mesh movement is incorporated in order to keep good quality cells in the tip gap of a high-pressure turbine rotor blade by restricting the movement in the tip surface normal direction. However, no special treatment of the hub and shroud surfaces is encountered during the literature review, and this is deemed to be a bottleneck in the mesh deformation process as, especially for mixed-flow geometries, these surfaces can be prone to be affected by the blade movement and this can not only alter the mass flow going through the passage but also the gradients predicted by the adjoint solver, topic treated in the next section.

In conclusion, it is observed that re-meshing is preferred when rapid, robust and efficient in-house grid generators are available, which allow having a more extensive control of the geometry. However, this is a

computationally expensive task, and it can also lead to a loss of information in the optimization process. On the other hand, the selection of mesh deformation algorithms will greatly depend on the extent to which these are able to maintain grid quality at critical regions such as the boundary and inflation layers.

2.3 Adjoint method

In gradient-based optimizations, the sensitivity of the objective function with respect to the design variables needs to be computed so that the optimizer has enough information, together with the value of the objective function itself, to determine the next design step towards the optimal solution search.

In adjoint-based turbomachinery shape optimization problems, where an underlying surface parametrization is included such that $\mathbf{X}_{\text{surf}} = \mathbf{X}_{\text{surf}}(\alpha)$, the objective function can be defined as

$$J = J(\mathbf{X}, \alpha), \tag{2.2}$$

where

$$\mathbf{X} = \mathbf{X}\big(\mathbf{X}_{\text{surf}}(\alpha)\big). \tag{2.3}$$

By applying the chain rule on Equation (2.2), the total gradient of the objective function J with respect to the design variables α can be written as

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = \frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}} \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\mathbf{X}_{\text{surf}}} \frac{\mathrm{d}\mathbf{X}_{\text{surf}}}{\mathrm{d}\alpha}. \tag{2.4}$$

The first term of equation (2.4), $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}}$, is obtained from the adjoint solver. These solvers are usually found to be integrated within existing CFD packages, such as HYDRA [25], OpenFOAM [6], ANSYS Fluent [7] or *SU2* [31].

The second term, $\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\mathbf{X}_{\text{surf}}}$, called mesh sensitivity, is commonly calculated by the adjoint solver as well and this can be done by finite differences (FD) or, alternatively, if there is access to the mesh deformation algorithm kernel, by differentiating the code. As mentioned in [12], this operation can be done with simpler mesh deformation algorithm than the used for the actual update for the update of the volumetric mesh for each optimization step. For instance, in this work, Xu et al. use a simple spring-based mesh algorithm for the computation of the sensitivities, while for each optimization step the volumetric mesh is updated using a more advanced linear-elasticity based mesh deformation algorithm, arguing a save in computational time. The dot product between the adjoint and mesh sensitivities can be regarded as the projection of the volumetric sensitivities onto the surface of interest, $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}_{\text{surf}}}$.

Finally, the last term $\frac{\mathrm{d}\mathbf{X}_{\text{surf}}}{\mathrm{d}\alpha}$, also known as surface sensitivities, can be computed in several ways: finite differences, using the complex-step method [35] or by differentiation of the underlying parametrization method. For the latter, this is commonly done by the use of algorithmic differentiation (AD), which requires access to the underlying parametrization source code. It has been observed that there is no tendency of using one method or the other; however, works that have access to the CAD kernel tend to use algorithmic differentiation. The advantage of using this technique is that it yields the exact gradient, but at a cost which is a non-trivial implementation. Finite differences is commonly employed when the CAD kernel is not accessible, treating the module as a black-box. It is also important to mention that if no underlying parametrization is included such that $\mathbf{X}_{\text{surf}} = \mathbf{X}_{\text{surf}}(\alpha)$ this term does not need to be computed. Table 2.1 summarizes all the parts needed to compute the total objective function gradient as well as where they are commonly obtained from in the literature.

The adjoint method has been of great success in the past decades, as it allows for a relatively fast computation of the gradient of the objective function with respect to its design variables. It was developed by Jameson based on control theory for its use in aerodynamic applications [4], and it permits the calculation of the objective function gradient independently of the number of design variables. Therefore, it is well suited for turbomachinery applications, where the large number of design variables requires

Table 2.1: Gradients needed to compute the total objective function gradient with respect to the design variables.

Name	Symbol	Obtained from
Objective sensitivity	$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}}$	Adjoint solver
Mesh sensitivity	$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}$	FD mesh deform. AD mesh deform.
Surface sensitivity	$\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}$	FD geom. parametrizer AD geom. parametrizer

short computational times for complex industrial applications.

Within the adjoint method, two approaches can be followed, namely the continuous and the discrete methods. For the former, the adjoint equations are derived from the linearized governing flow equations, and later discretized. In the case of the discrete approach, firstly the flow equations are discretized and then linearized. The continuous approach minimizes the memory requirements and CPU cost per iteration. Furthermore, as it was shown by Papadimitriou and Giannakoglou [36], the continuous adjoint equations can be derived in such a way that they are free of field terms, therefore the sensitivity of the interior grid nodes coordinates with respect to the design variables does not need to be computed. However, it may not give as good convergence as the original non-linear code. On the other hand, the discrete approach is more computationally expensive, but it yields the exact gradient of the objective function, ensuring good and fast convergence of the solver [37], well tailored for complex design problems.

2.4 *SU2* in aerodynamic shape optimization

Once all the different parts of the turbomachinery optimization problem have been presented, it is also important to evaluate the performance of the chosen CFD suite, *SU2*, in aerodynamic shape optimization problems.

Initially, *SU2* was conceived with an emphasis on external flow applications, highlighting its capabilities for an accurate sensitivity prediction using its continuous or discrete adjoint solver. Furthermore, it includes a mesh deformation algorithm which, together with the adjoint solver, makes *SU2* a complete framework for gradient-based optimization or uncertainty quantification [31].

However, in the recent years *SU2* has been extended for internal flow applications, where different types of boundary conditions and techniques for multi-row computations have been implemented, with even having the capability of performing unsteady fluid dynamic optimization of turbomachinery by resorting to the harmonic-balance technique [30, 38].

It is in these last two works where most of the studies on turbomachinery shape optimization using *SU2* reside. The discrete adjoint gradients have been validated for several axial and radial test cases, and in [30] the 3D Aachen turbine is successfully optimized by using FFD boxes. In the work by Rubino [38], a 2D organic Rankine cycle axial turbine stage is studied by performing an unsteady optimization of the geometry under non-ideal compressible conditions, again using FFD boxes.

However, the use of *SU2* together with a CAD-based geometry parametrizer is yet to be explored, and some questions remain unanswered yet:

- How accurately are the gradients, after performing the dot product between the adjoint and CAD sensitivities, being predicted.

- How robust is the mesh deformation algorithm linked with the CAD parametrizer for fine three-dimensional grids.

2.5 Test cases

As the objective of this thesis is to develop a common framework for 3D turbomachinery shape optimization, where a CAD-based parametrization approach is going to be followed, it is important to keep track of what has been done in the past. Several papers have been published using this specific framework; however, the objective of this section is to have those references in a more visual manner for a better understanding of the subject.

Tsiakas et al. [13] use the continuous adjoint method to optimize a compressor stator using NURBS coefficients as design variables. In their work, the row geometry is firstly created by parametrizing the shape of the blade's mean camber and then adding thickness, superimposing blade profiles in the spanwise direction to generate the 3D blade shape. The authors also elaborate on the computation of the total gradient of the objective function: in this case, while the adjoint solver provides the sensitivities with respect to the surface coordinates, finite differences are used to compute the geometrical sensitivities, therefore closing the computation of the final gradients. No validation of the adjoint sensitivities is shown in their work. Furthermore, the authors make use of a grid displacement algorithm based on RBFs to deform the mesh in each optimization step, where the hub and shroud surfaces are kept frozen.

Vasilopoulos et al. [33] perform an optimization of a compressor stator by using the discrete adjoint method incoporated in HYDRA [25]. The geometry is built by stacking 2D aerofoil profiles in the spanwise direction. A total of 192 design variables are considered, where five of them consist of discrete values such as inlet and outlet metal angles and maximum thickness, while the others are control points for thickness and camber distributions. The authors also elaborate on the computation of the total gradient: while HYDRA provides the gradient of the objective function with respect to the surface mesh coordinates, finite differences are applied to the CAD generation tool to compute the geometrical sensitivities. The inner product of both yields the total gradient of the objective function with respect to the design variables. However, no validation of the discrete adjoint gradients is shown in their work. A steepest descent algorithm will afterwards determine the search direction based on this information. For each optimization step, a new mesh is generated by the use of PADRAM [34].

Agarwal et al. [14] use traditional blade design parametes to define the geometry of a high-pressure turbine vane. The main goal of their work is to validate a novel technique based on finite differences to compute the geometrical sensitivities. The flow sensitivities are provided by the discrete adjoint solver within HYDRA. These gradients are compared to those computed by finite differences, showing large discrepancies in those design variables controlling the trailing-edge geometry. However, the author argues that, with the methodology implemented, one does not have control on the step size used for the finite differences gradient, therefore adding uncertainties. Furthermore, no optimization is performed in the aforementioned high-pressure turbine vane.

Mueller and Verstraete [15] optimize a radial turbine using the discrete adjoint method. The geometry is built by using 40 degrees of freedom, which include the definition of the meridional flow path, the blade angle distribution and the blade thickness distribution. The total gradient is computed by the inner product of the flow sensitivities, provided by the adjoint solver, and the geometrical sensitivities that are calculated by using the complex-step method in the CAD generation tool. Gradient validation is shown by comparing the adjoint gradients to those computed by the use of the complex-step method. For each optimization step, the mesh is regenerated and an SQP algorithm is used for the optimal direction search.

Xu et al. [12] use the control points of the BRep as design variables, which are allowed to move in the three space coordinates. This, together with the HYDRA discrete adjoint, is employed to optimize a one-stage turbine. Geometrical constrains for the NURBS patches (continuity) and blade thickness

and trailing-edge radius are imposed by implicitly incorporating these constraints to the parametrization. The authors argue that, although another possibility could be leaving the design variables that have more effect on the trailing-edge and blade thickness, this would drastically reduce the design space. Furthermore, in their work, while HYDRA provides the gradient of the objective function with respect to the volumetric grid coordinates, the gradient of the volumetric grid coordinates with respect to the surface grid coordinates is computed by automatic differentiation of a simple mesh deformation algorithm based on the spring analogy. The geometric sensitivity is computed analytically. No validation of the aforementioned gradients is shown in their work. In each optimization step, the mesh is deformed by using the linear elasticity method, and a steepest descent algorithm computes the search direction. The authors highlight the influence that the mesh deformation algorithm has on the robustness of the method, and argue that a SQP algorithm could potentially improve the results.

Mykhaskiv et al. [19] perform an optimization of a compressor stator by using a discrete adjoint solver. In their paper, the authors compare the outcomes of using the two CAD-based approaches mentioned in this work: traditional and BRep methods. For the latter, the control points of the NURBS patches that form the BRep of the model are used as design variables. Several geometrical constrains are added: NURBS patches continuity, radius of curvature and smoothnes. These are included by formulating complex geometrical links in the underlying parametrization. While the discrete adjoint solver provides the gradient of the objective function with respect to the surface mesh coordinates, automatic differentiation is used in both parametrization techniques to compute the remaining sensitivity. No gradient validation is shown in their work. In each optimization step, the mesh is re-generated. The authors conclude that parametric CAD models are useful when the underlying parametrization is well-established due to designer's experience, while the BRep-based approach can explore more unconventional designs as it has a richer design space.

Table 2.2: Relevant 3D cases in adjoint-based turbomachinery shape optimization using CAD-based parametrization techniques. Methodology.

Author	**Ref.**	**Type**[1]	**Adjoint**	**Geometry**	**Mesh**	**Optimizer**
Xu et al.	[12]	A-T	Discrete	BRep	Linear el.	Steepest
Tsiakas et al.	[13]	A-C	Continuous	Traditional	RBF	-
Agarwal et al.	[14]	A-T	Discrete	Traditional	-	-
Vasilopoulos et al.	[33]	A-C	Discrete	Traditional	Re-meshing	Steepest
Mueller and Verstraete	[15]	R-T	Discrete	Traditional	Re-meshing	SQP
Mykhaskiv et al.	[19]	A-C	Discrete	Both	Re-meshing	L-BFGS-B and Steepest

Table 2.2 gathers all the relevant previous works in the 3D domain for adjoint-based shape optimization using CAD-based parametrization techniques, as well as the type of adjoint solver used, the CAD-based approach, the mesh update technique per optimization step and the optimizer used.

Firstly, it can be noticed that none of the papers presented evaluate the same technique for different types of machines. In other words, even though there has been done research in different types of turbomachinery, the same method has not been evaluated yet in different kinds. Secondly, the preferred adjoint-method is the discrete approach. The reason behind this is that, even though being more computationally expensive, the gradients calculated are exact, therefore increasing the robustness of the total gradient computation. Regarding the modelling of the geometry, most of the studies employ the traditional CAD-based approach, arguing that the implementation of the constraints is more straightforward and intuitive. Moving to the update of the mesh in each optimization step, re-meshing is the preferred technique. As it was said before, having a robust mesh deformation algorithm is a key element in the

[1] A: Axial; R: Radial; T: Turbine; C: Compressor

robustness of the gradient computation. Finally, different optimizers are used in the literature reviewed, prominently steepest descend and SQP.

Another important aspect is the computation of the total gradient. Similarly than in the previous one, Table 2.3 gathers the different techniques to compute the terms in the total objective function gradient, as shown in Equation (2.4).

Table 2.3: Relevant 3D cases in adjoint-based turbomachinery shape optimization using CAD-based parametrization techniques. Gradient computation.

Author	**Ref.**	$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}}$	$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}$	$\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}$
Xu et al.	[12]	Adjoint solver	AD mesh def.	Analytically
Tsiakas et al.	[13]	Adjoint solver	Adjoint solver[2]	FD
Agarwal et al.	[14]	Adjoint solver	AD mesh def.	FD
Vasilopoulos et al.	[33]	Adjoint solver	AD mesh def.	FD
Mueller and Verstraete	[15]	Adjoint solver	Complex-step	Complex-step
Mykhaskiv et al.	[19]	Adjoint solver	AD mesh def.	AD

The adjoint solver is responsible of computing the gradient of the objective function with respect to the volume grid coordinates regardless of the number of design parameters considered. The next term necessary for the computation of the total gradient is the mesh sensitivity, $\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}$, which as it was mentioned before is usually computed also by common adjoint solvers included in CFD tools (e.g. *SU2*, HYDRA). However, in the work by Xu et al. [12], it is argued that a simpler mesh deformation algorithm can be used at this step, and they employ a spring-analogy algorithm in order to save computational costs. Finally, the geometric or CAD sensitivities $\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}$, are most commonly computed by using finite differences. However, those works that have access to the CAD generation algorithm use automatic differentiation or the complex-step method to obtain these sensitivities.

While there have been some works in the adjoint- and CAD-based turbomachinery field, it is observed that there is not a common approach to the problem, especially in the gradient validation process as it is not always shown. Furthermore, none of the works study the same methodology on different kinds of turbomachinery. Therefore, the focus of this work will be:

- The use of the discrete adjoint method to compute the objective function's sensitivities due to its save in computational time when compared to traditional finite difference methods. The discrete approach is preferred over the continuous as it yields the exact objective function gradient.
- The use of a CAD-based geometry parametrization tool, where the blade shape is modelled with traditional design parameters that ease the geometrical constrain implementation process for both axial and radial turbomachinery.
- The use of mesh deformation algorithms to morph the grid in each optimization step to maintain certain mesh properties and computational efficiency.
- The coupling of the open-source CFD suite *SU2* with the in-house geometry parametrizer *ParaBlade* will result in an open-source optimization framework for turbomachinery optimization.

[2] Use of the continuous adjoint method

Chapter 3

Methodology

In this chapter, the methodology which leads to the construction of the numerical framework for optimizing axial and radial turbomachinery is thoroughly explained. Special emphasis is put on how the different components of the total gradient of the objective function with respect to the design variables are obtained and assembled.

Section 3.1 introduces the design chain for CAD- and adjoint-based shape optimization problems. This leads to a problem decomposition for which the first module, shape parametrization, is explained in Section 3.2. Section 3.3 details the mesh deformation algorithm employed in this work. The flow and adjoint solvers, essential for the computation of the objective function value and gradient, are discussed in Section 3.4. The gradient assembly and validation are explained in Sections 3.5 and 3.6, to finally conclude in Section 3.7 with the optimization problem definition.

3.1 Design chain

Adjoint- and CAD-based optimization problems are composed of different modules which could be briefly seen in Chapter 2. These modules essentially correspond to the different operations that are needed in order to compute two main figures of merit which will be used by the optimizer to determine the optimal solution search direction:

- Objective function, J.
- Gradient of the objective function with respect to the design variables or total sensitivity, $\frac{\mathrm{d}J}{\mathrm{d}\alpha}$.

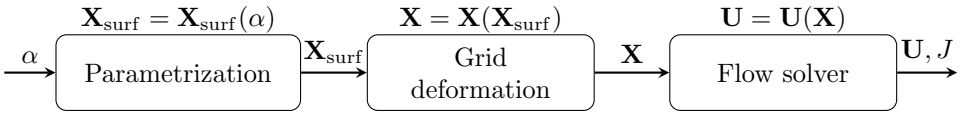


Figure 3.1: Objective function computation flow chart.

Figure 3.1 shows a schematic representation of how the objective function is computed. Firstly, the geometry is generated by a set of design variables, α, which will in turn generate a map of surface grid coordinates, $\mathbf{X}_{\text{surf}}$. A more detailed explanation on how the design parameters are obtained will be done in Section 3.2. The new surface coordinates will serve as an input to the mesh deformation algorithm, which will generate a deformed volumetric grid $\mathbf{X}$. The flow solution, $\mathbf{U}$, will then be obtained by the flow solver, where the objective function J can be evaluated and computed.

In a similar way, Figure 3.2 represents a schematic representation of how the total sensitivity is computed in this type of problems. In this case, three modules are called independently which will generate three different sensitivities: the parametrization tool will have as an input the design parameters

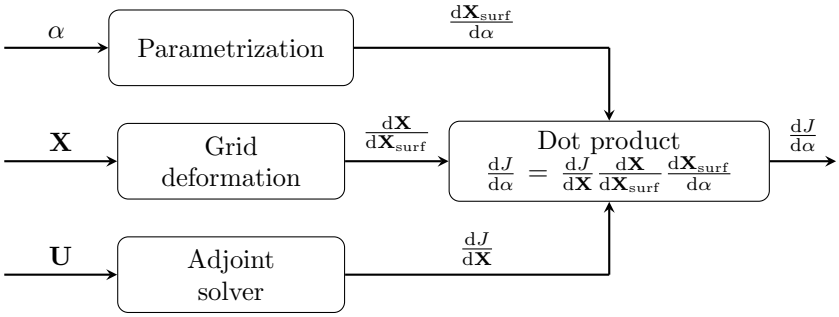


Figure 3.2: Total sensitivity computation flow chart.

and will compute the sensitivity of the surface coordinates with respect to the parametrization, $\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}$; the mesh deformation algorithm will calculate the mesh sensitivities, $\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}$; and the adjoint solver will read the flow solver solution $\mathbf{U}$ and compute the gradient of the objective function with respect to the volumetric grid coordinates, $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}}$. The dot product of these three gradients will yield the total sensitivity,

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = \frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}} \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}} \frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}. \tag{3.1}$$

3.2 Surface parametrization

In order to reduce the number of design variables of the optimization problem, which will in turn add simplicity and computational efficiency to the process, a shape parametrization is introduced. This way, the blade geometry, which is initially defined by a set of surface coordinates, is parametrized in such a way that

$$\mathbf{X}_{\mathrm{surf}} = \mathbf{X}_{\mathrm{surf}}(\alpha), \tag{3.2}$$

allowing for a substantial reduction in the number of design variables if compared to other techniques shown in Chapter 2.

In this case, the parametrization approach followed is CAD-based: the set of design parameters consists of typical turbomachinery design variables (i.e. blade metal angles, chord, stagger angle, amongst others) which will then construct a set of NURBS curves [32] defining the blade sections. The three-dimensional blade is then constructed by the stacking of blade sections in the span-wise direction, for which again NURBS curves are used.

3.2.1 Geometry generation

The steps followed in order to generate the geometry are [39]:

1. Parametrization of the meridional channel.
2. Parametrization of the blade sections.
3. Parametrization of the span variation of the blade sections.

The meridional channel is formed by four NURBS curves corresponding to the leading and trailing edge and the inner and outer surfaces, as depicted in Figure 3.3. Each of the curves contains an arbitrary number of control points, thus allowing for flexibility in case of simple (e.g. purely axial or radial) or complex (e.g. mixed-flow) turbomachinery.

The blade sections are parametrized by a set of seven engineering parameters, namely: stagger angle, ξ; leading edge metal angle, θ_{in}; leading edge wedge semi-angle, ϵ_{in}; leading edge radius, r_{in}; trailing edge metal angle, θ_{out}; trailing edge wedge semi-angle, ϵ_{out}; and trailing edge radius, r_{out}. The construction

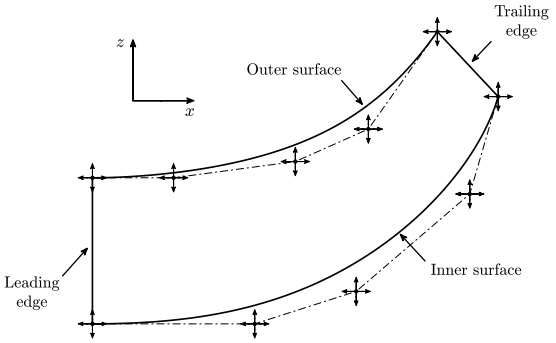
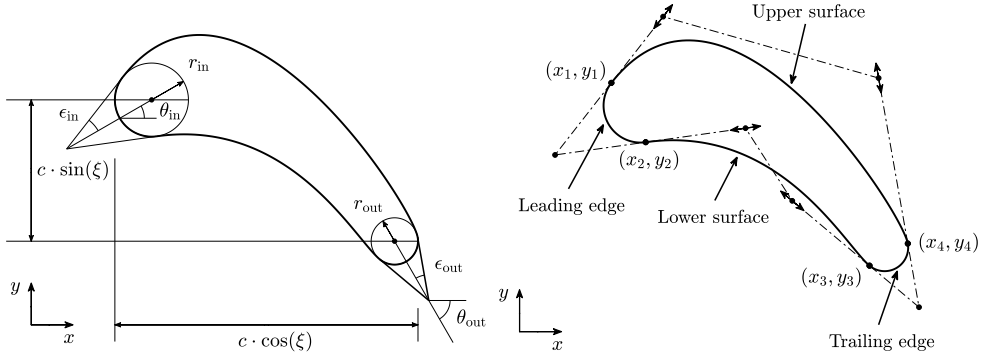


Figure 3.3: Meridional channel parametrization in *ParaBlade* [39].



(a) Parameters. **(b)** Segments.

Figure 3.4: Blade section parametrization in *ParaBlade* [39].

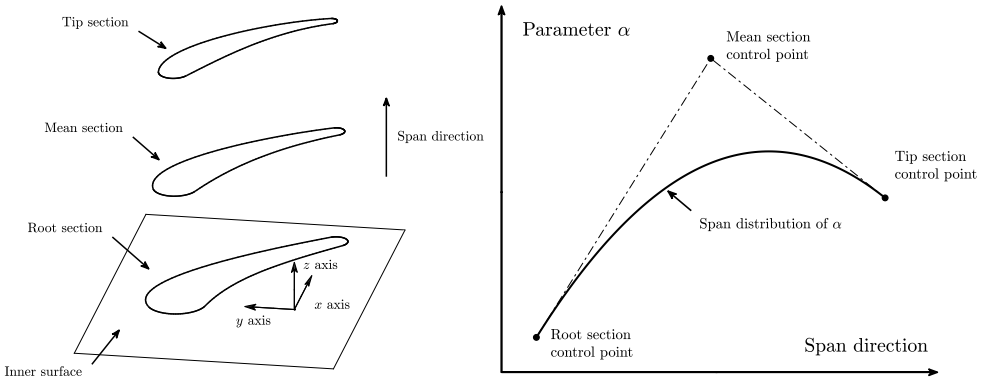
of the blade section based on this parameters can be seen in Figure 3.4a. Furthermore, the blade section is divided into four segments, which are depicted in Figure 3.4b, and correspond to the leading edge, from 1 to 2; lower surface, from 2 to 3; trailing edge, from 3 to 4; and upper surface, from 4 to 1. These segments are constructed by clamped second- (leading and trailing edge) and third-order (lower and upper surfaces) NURBS curves. The seven engineering parameters will define the location of the NURBS control points, and an additional flexibility is given by moving control points $1 - 4$ as previously explained.

Finally, once the different blade sections that form the geometry are defined, these are stacked in the span-wise direction, as shown in Figure 3.5a. In non-prismatic blades, there is a variation of the blade sections from the root to the tip section (either related purely to the section thickness, as the blade tip section is usually thinner; or due to a twist of the three-dimensional blade). This variation of the blade sections in the radial direction is achieved by providing a span-wise variation of the seven engineering parameters described earlier. This is done by means of NURBS curves, as shown in Figure 3.5b.

3.2.2 Geometrical surface sensitivities

In order to calculate the total objective function sensitivities, as shown in Equation (3.1), the so-called geometrical sensitivities or $\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}$ need to be computed beforehand. These will reveal how the surface coordinates change with a perturbation in the design variables, and are defined as the projection of the differential surface displacement into the normal direction, as depicted in Figure 3.6. These sensitivities can be calculated in different ways:

- Finite differences: as it was shown in Equation (3.2), the design variariables α univocally define the surface coordinates $\mathbf{X}_{\mathrm{surf}}$. Therefore, the design variables can be perturbed by a step δ which will in turn produce a change in the surface coordinates. Then, the geometrical gradient can be



(a) Sections. **(b)** Variation.

Figure 3.5: Span-wise section stacking in *ParaBlade* [39].

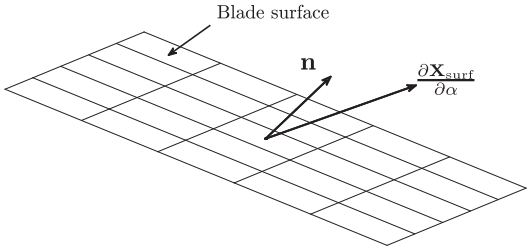


Figure 3.6: Surface sensitivity calculated in *ParaBlade* [39].

computed by using finite differences as

$$\frac{\mathrm{d}\mathbf{X}_{\text{surf}}}{\mathrm{d}\alpha} \approx \frac{\mathbf{X}_{\text{surf}}(\alpha+\delta) - \mathbf{X}_{\text{surf}}(\alpha)}{\delta}, \tag{3.3}$$

if a forward scheme is used, or as

$$\frac{\mathrm{d}\mathbf{X}_{\text{surf}}}{\mathrm{d}\alpha} \approx \frac{\mathbf{X}_{\text{surf}}(\alpha+\delta) - \mathbf{X}_{\text{surf}}(\alpha-\delta)}{2\delta}, \tag{3.4}$$

if a central scheme is selected instead.

- Complex-step: on the other hand, the complex-step method [35] inputs an imaginary step to the surface generation algorithm, such that $\mathbf{X}_{\text{surf}} = \mathbf{X}_{\text{surf}}(\alpha + i \cdot \delta)$. The first derivative of the objective function with respect to the design variables is then calculated as the imaginary part of the deformed surface coordinates,

$$\frac{\mathrm{d}\mathbf{X}_{\text{surf}}}{\mathrm{d}\alpha} \approx \frac{1}{\delta}\mathrm{Im}\Big[\mathbf{X}_{\text{surf}}(\alpha + i \cdot \delta)\Big]. \tag{3.5}$$

3.2.3 ParaBlade

For this study, the open-source blade parametrization tool *ParaBlade* is used. In particular, two modules are essential for the generation of the blade parametrization and computation of the geometrical gradient: `MatchBlade.py` and `MakeBlade.py`

MatchBlade.py

The main goal of the blade matching script is to parametrize an existing baseline geometry, given in the format of a cloud of surface points. Two parametrizations are performed:

- u, v parametrization.

- α parametrization.

In the $u-$ and $v-$ parametrization, for a set of surface cordinates $\mathbf{X}_{\text{surf}} = (x_S, y_S, z_S)$, a parametrization is established such that

$$\begin{aligned} (x_S, y_S, z_S) &\longleftarrow (u, v), \\ \mathbb{R}^3 &\longrightarrow \mathbb{R}^2, \end{aligned} \tag{3.6}$$

meaning that each of the surface design coordinates will be univocally defined by a pair of u and v parameters. This is of great importance as it will allow for a consistent generation of the new geometries in the optimization process. Figure 3.7 shows a generic three-dimensional blade in which the u and v parametrization has been schematized.

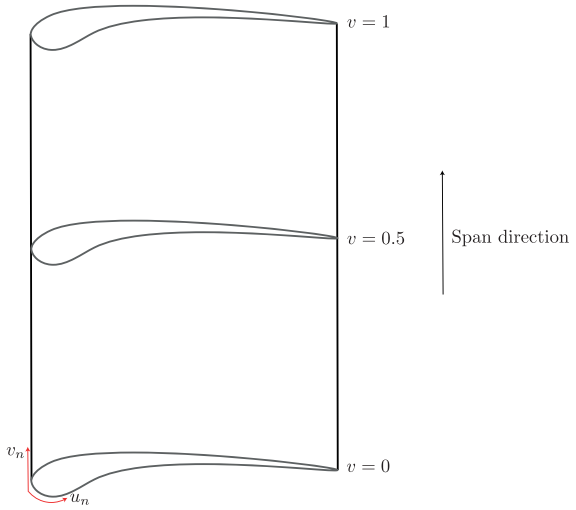


Figure 3.7: Three-dimensional blade u and v parametrization.

These two parametrizations are obtained by solving an optimization problem such that

$$\begin{aligned} \underset{(u,\,v,\,\alpha)}{\text{minimize}} \quad & \text{dist}\big(\mathbf{X}_{\text{surf}}, \mathbf{X}_{\text{CAD}}(u, v, \alpha)\big), \;\; \alpha = \{\alpha_1, ..., \alpha_{N_\alpha}\}, \\ s.t. \quad & 0 \leq u_i \leq 1, \;\; i = 1, \ldots, N, \\ & 0 \leq v_i \leq 1, \;\; i = 1, \ldots, N, \end{aligned} \tag{3.7}$$

which implies that the design parameters u, v, α are found by minimizing the distance between the reference surface coordinates, $\mathbf{X}_{\text{surf}}$, and the coordinates calculated by the script with the given parametrization, $\mathbf{X}_{\text{CAD}}(u, v, \alpha)$.

The outcome of using this tool is a set of matched parameters and coordinates that will be later used by the optimizer to perturb the baseline geometry.

MakeBlade.py

Once the parametrization of the existing geometry is obtained, the gradient of the surface coordinates with respect to the design variables is calculated by `MakeBlade.py`. This tool can also be used to build two- and three-dimensional blades from scratch.

The three methods for computing the geometrical gradient explained in the previous section, namely finite differences (forward and central) and complex-step, are implemented in this script. Regardless of

the procedure used, this will result in a map of geometrical sensitivities such that

$$\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha} = \begin{pmatrix} \frac{\partial x_S}{\partial \alpha_1} & \frac{\partial x_S}{\partial \alpha_2} & \cdots & \frac{\partial x_S}{\partial \alpha_{N_\alpha}} \\ \frac{\partial y_S}{\partial \alpha_1} & \frac{\partial y_S}{\partial \alpha_2} & \cdots & \frac{\partial y_S}{\partial \alpha_{N_\alpha}} \\ \frac{\partial z_S}{\partial \alpha_1} & \frac{\partial z_S}{\partial \alpha_2} & \cdots & \frac{\partial z_S}{\partial \alpha_{N_\alpha}} \end{pmatrix}, \tag{3.8}$$

where N_α is the number of design variables.

3.3 Mesh deformation

As it was argued in Chapter 2, the introduction of mesh deformation in shape optimization problems allows for an efficient and robust method when generating the grid for each of the design steps.

Mesh motion is achieved by firstly moving the surface boundaries, displacement that is obtained from the optimizer in form of a new set of blade surface coordinates, and afterwards performing the volumetric deformation which is based on a classical spring-analogy method [40]. The system of equations that needs to be solved in order to compute the new volumetric grid displacements is

$$\left(\sum_{j \in \mathcal{N}(i)} k_{ij} \mathbf{e}_{ij} \mathbf{e}_{ij}^{\mathrm{T}} \right) \mathbf{u}_i = \sum_{j \in \mathcal{N}(i)} k_{ij} \mathbf{e}_{ij} \mathbf{e}_{ij}^{\mathrm{T}} \mathbf{u}_j, \tag{3.9}$$

where k_{ij} is the stiffness matrix, $\mathcal{N}(i)$ is the set of neighbouring points to node i and $\mathbf{e}_{ij}$ is the unit vector connecting both aforementioned points. The unknown displacements, $\mathbf{u}_i$, are then computed as a function of the prescribed surface displacement previously mentioned, $\mathbf{u}_j$. The system of equations is then solved iteratively using linear solvers.

The spring-analogy mesh deformation algorithm is implemented in the mesh deformation routine `SU2_DEF`. Furthermore, as the quality of the mesh deformation process will largely depend on the number of elements to be deformed (i.e. grid density, boundary layer refinement), certain options can be selected in order to ease the mesh deformation step:

- Type of element stiffness: in turbomachinery-like problems, which are comprised of relatively small domains (if compared to external flow simulations, for example) and are bounded by walls which usually include element refinement to capture the boundary layer effects, it is of great importance maintaining the y^+ value of the elements near the walls, as well as their expansion ratio, in order to have a consistent and robust method for mesh deformation. In other words, the mesh elements near the walls have to remain as stiff as possible in order to maintain consistency between the original and deformed mesh, and this can be achieved by selecting the correct element stiffness. Currently, there are three types implemented in `SU2_DEF`, namely wall distance, inverse volume and constant stiffness, which will affect the way the Young's modulus E is calculated and, in turn, the element's stiffness:
  - Wall distance: calculated as the inverse of the wall distance, i.e. stiffest elements are located next to the boundary walls:
  $$E \propto \frac{1}{\text{wall distance}}. \tag{3.10}$$
  - Inverse volume: computed as the inverse of the element volume, i.e. stiffest elements are those with the smallest volume (e.g. boundary layer cells):
  $$E \propto \frac{1}{\text{element volume}}. \tag{3.11}$$

– Constant stiffness: the Young's modulus is maintained constant as a prescribed value:

$$E = E_0. \tag{3.12}$$

Having a robust mesh deformation algorithm will not only depend on how the volumetric mesh is deformed, but also on how the boundary layer refinement is maintained, and this will largely depend on the aforementioned types of stiffness shown.

- Poisson's ratio: another important parameter that can be directly controlled in mesh deformation is the Poisson's ratio ν, which is the ratio of the transverse contraction strain to the longitudinal extension strain in the direction of the stretching force. This parameter can be used in order to reduce locking of the mesh elements.

3.3.1 Tangential boundary condition

Undoubtedly, one of the most critical points in turbomachinery shape optimization problems is the mesh deformation operation, not only because of possible runtime issues arising from fine meshes with large number of elements or high surface deformations, but also because the fluid domain needs to remain as unchanged as possible (besides the blade walls, of course, as they will be deforming according to the optimal direction search).

The boundaries that limit the flow domain in a turbomachinery CFD simulation are: inlet and outlet, periodic boundaries and adiabatic walls. For the former, they are usually few orders of magnitude of the blade's chord away from the moving geometry: therefore, they are deemed to remain unchanged during the mesh deformation and not being harmful for the robustness of the process; periodic boundary movement is also considered as harmless as both surfaces will move identically in the same direction and magnitude therefore not affecting the pitch; however, the movement of the hub and shroud walls can produce an alteration of the results as this can increase (decrease) the mass flow rate through the passage. Furthermore, the objective function value might be very sensitive to small movements of these two walls (especially if inflation layers are present in order to capture boundary layer effects), and if these two surfaces are not modelled by the CAD parametrization tool, a loss of information can occur as these walls will be moving but this information will not be available in the adjoint sensitivities. In conclusion, the movement of the hub and the shroud has to be avoided in the surface normal direction.

In order to restrict the movement of the hub and shroud in the normal surface direction its deformation components in the spring-analogy equation system previously described need to be suppressed and set to zero. Therefore, the so-called tangential boundary condition is implemented in the mesh deformation algorithm, which will be responsible for allowing deformation only in the tangential ($\mathbf{t}_1$) and bitangential ($\mathbf{t}_2$) directions, but not in the normal ($\mathbf{n}$) direction.

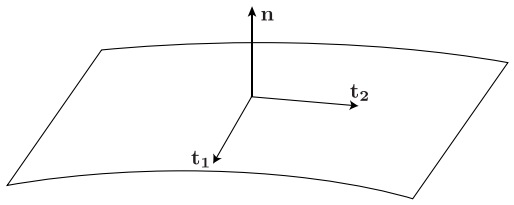


Figure 3.8: Local tangential coordinate system.

The implementation is as it follows: when building the linear mesh deformation system, the script loops through all the grid points and recognizes those in the hub and the shroud by the use of a tag in the configuration file, as specified by the user. In those boundary points, the local tangential coordinate system is computed, as shown in Figure 3.8. The normal vector is obtained internally by the geometry structure in `SU2_DEF`. Afterwards, the remaining two tangential directions are computed by comparing

the resulting normal vector's module from the cross product between $\mathbf{n}$ and the Cartesian unit vectors, $\mathbf{i}, \mathbf{j}$ and $\mathbf{k}$,

$$\begin{aligned} \mathbf{n}_i &= \mathbf{n} \times \mathbf{i} \rightarrow |\mathbf{n}_i| = \sqrt{\mathbf{n}_i \mathbf{n}_i}, \\ \mathbf{n}_j &= \mathbf{n} \times \mathbf{j} \rightarrow |\mathbf{n}_j| = \sqrt{\mathbf{n}_j \mathbf{n}_j}, \\ \mathbf{n}_k &= \mathbf{n} \times \mathbf{k} \rightarrow |\mathbf{n}_k| = \sqrt{\mathbf{n}_k \mathbf{n}_k}, \end{aligned} \tag{3.13}$$

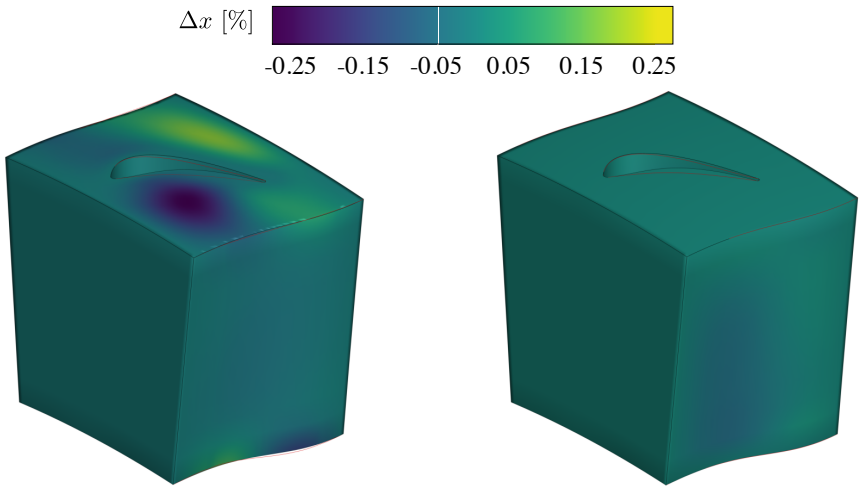
and taking the direction of the vector with maximum module, this is

$$\mathbf{t}_1 \iff \max(|\mathbf{n}_i|, |\mathbf{n}_j|, |\mathbf{n}_k|), \tag{3.14}$$

while the remaining tangential vector is calculated as

$$\mathbf{t}_2 = \mathbf{n} \times \mathbf{t}_1. \tag{3.15}$$

Based on these local tangential coordinate systems, a transformation matrix is built which is latter multiplied by the stiffness matrix, therefore converting it as well into the tangential coordinate system. Once this is done, the components of the stiffness matrix corresponding to the normal direction at the specified boundaries (commonly hub and shroud) are set to zero, restricting its movement in that direction.



(a) Without tangential boundary condition. **(b)** With tangential boundary condition.

Figure 3.9: Variation of span-wise coordinate (in percentage) for the same perturbation of the stagger angle in a prismatic axial blade.

Figure 3.9 shows the application of the tangential boundary condition on a prismatic axial turbine stator by plotting the difference in x-coordinates (approximately coincident with the radial direction) between the deformed and baseline mesh. Both figures have the same perturbation of the stagger angle. As it can be seen in Figure 3.9a, when the tangential boundary condition is not applied some patches can be encountered where there is displacement in the normal direction at both hub and shroud. However, when the tangential boundary condition is applied on both hub and shroud surfaces, Figure 3.9b, these patches disappear, but still allowing for a deformation within the plane as seen in the red edge lines.

Figure 3.10 displays the tangential boundary condition applied, in this case, in a mixed-flow turbine. The red surface corresponds to the deformed blade surface that serves as an input for the mesh deformation algorithm. The red arrows shown in the figure display the sliding movement of the mesh at the hub wall.

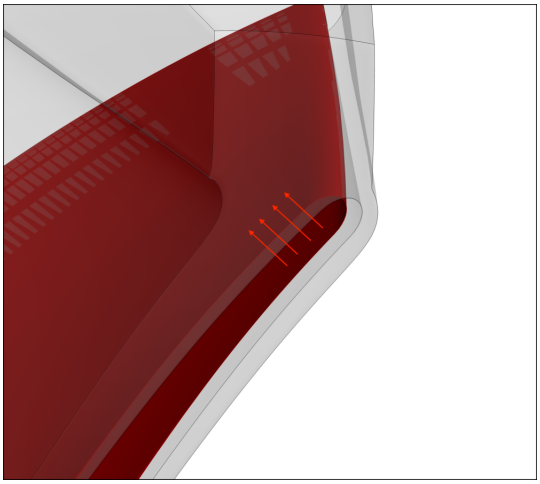


Figure 3.10: Tangential boundary condition applied on a radial turbomachinery test case. Red arrows show the sliding direction.

3.3.2 Mesh sensitivities

The computation of the gradient of the objective function with respect to the design variables requires the mesh sensitivities, $\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}$, as shown in Equation (3.1).

In order to compute this value, the underlying spring-analogy-based mesh deformation subroutine is differentiated using algorithmic differentiation. In the context of *SU2*, this results in a new module named `SU2_DOT_AD`, which yields a map of mesh sensitivities such that

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}} = \left(\frac{\partial \mathbf{x}}{\partial \mathbf{x}_{\mathrm{surf}}}, \frac{\partial \mathbf{y}}{\partial \mathbf{y}_{\mathrm{surf}}}, \frac{\partial \mathbf{z}}{\partial \mathbf{z}_{\mathrm{surf}}} \right). \tag{3.16}$$

3.4 Flow and adjoint solver

The open-source CFD suite *SU2* [31] is used as flow (`SU2_CFD`) and discrete adjoint (`SU2_CFD_AD`) solver. The following sections will describe each module thoroughly.

3.4.1 Flow solver

The CFD tool embedded in *SU2* includes both Euler and Reynolds-averaged Navier-Stokes (RANS) solvers. For this present study, focus will be put on the RANS solver, as turbulence plays an important role in internal flows. However, when utilizing RANS, the turbulence problem has to be closed. In this case, *SU2* includes both Spalart-Allmaras [41] and the two-equation Menter k-ω SST [42] turbulence models.

The partial differential equations are discretized using the finite volume method. Centered and upwind schemes are available for the discretization of the convective fluxes, such as JST [43] or Roe [44]. In case of using high-order upwind schemes, limiters are available, e.g. Venkatakrishnan [45] and Van Albada [46]. *SU2* does not include Law of the Wall functions, therefore special attention needs to be drawn to the value of y^+ in those regions, especially if second-order discretization schemes are to be employed.

In the present study, an Euler implicit time marching method is used, such that

$$\mathbf{U}^{n+1} = \mathbf{G}(\mathbf{U}^n, \mathbf{X}), \tag{3.17}$$

where $\mathbf{G}$ is the *fixed point* operator that represents the time integration of the RANS equations. The solution of the flow equations is found once

$$\mathbf{R}(\mathbf{U}^*, \mathbf{X}) = 0, \tag{3.18}$$

$\mathbf{R}$ being the residual vector and $\mathbf{U}^*$ the feasible point at which $\mathbf{G}$ is stationary.

Switching to the setting up of the problem, several types of boundary conditions can be applied, namely:

- No-slip adiabatic wall boundary condition.
- Symmetry walls.
- Periodic boundary condition.
- Characteristic-based inlet, outlet and mixing-plane boundary conditions:
  - Inlet: stagnation temperature and pressure.
  - Outlet: static back pressure.

The flow solver is responsible for evaluating the objective function and constrain, if any, specified. Currently, the following objective function and constrains are supported in *SU2*: entropy generation, total pressure loss, kinetic energy loss, outlet flow angle and dimensionless power output. These are explained in detail in Section 3.7.

The settings for each of the test cases studied will be reported in Chapter 5.

3.4.2 Adjoint solver

SU2 also incorporates both continuous and discrete adjoint solvers. For the benefits exposed in Chapter 2, the discrete adjoint solver will be the focus of this work.

The discrete adjoint solver included in *SU2* is obtained by using algorithmic differentiation on the primal flow solver [30] which was described in the previous section. This is done in the `SU2_CFD_AD` module.

3.4.3 Adjoint sensitivities

The objective function J, a scalar, in an aerodynamic optimization problem is a function of the flow vector $\mathbf{U}$ and the mesh coordinates $\mathbf{X}$, where both of them ultimately depend on the design variables α:

$$J = J\big(\mathbf{U}(\alpha), \mathbf{X}(\alpha)\big). \tag{3.19}$$

In an optimization problem, J is tried to be minimized while the problem is subject to:

$$\begin{aligned} \mathbf{U} &= \mathbf{G}(\mathbf{U}, \mathbf{X}), \\ \mathbf{X} &= \mathbf{M}(\alpha) = \mathbf{V}\big(\mathbf{S}(\alpha)\big). \end{aligned} \tag{3.20}$$

For the adjoint derivation using the Lagrangian approach, this can be written as

$$L(\alpha, \mathbf{U}, \mathbf{X}, \bar{\mathbf{U}}, \bar{\mathbf{X}}) = J(\mathbf{U}, \mathbf{X}) + [\mathbf{G}(\mathbf{U}, \mathbf{X}) - \mathbf{U}]^T \bar{\mathbf{U}} + [\mathbf{M}(\alpha) - \mathbf{X}]^T \bar{\mathbf{X}}. \tag{3.21}$$

If L is derived with respect to the design variables, α, and $\bar{\mathbf{U}}$ and $\bar{\mathbf{X}}$ are chosen so that the computationally expensive terms $\frac{\partial \mathbf{U}}{\partial \alpha}$ and $\frac{\partial \mathbf{X}}{\partial \alpha}$ are eliminated, this leads to

$$\bar{\mathbf{U}} = \frac{\partial}{\partial \mathbf{U}} J^T(\mathbf{U}, \mathbf{X}) + \frac{\partial}{\partial \mathbf{U}} \mathbf{G}^T(\mathbf{U}, \mathbf{X}) \bar{\mathbf{U}}, \tag{3.22}$$

$$\bar{\mathbf{X}} = \frac{\partial}{\partial \mathbf{X}} J^T(\mathbf{U}, \mathbf{X}) + \frac{\partial}{\partial \mathbf{X}} \mathbf{G}^T(\mathbf{U}, \mathbf{X}) \bar{\mathbf{U}}, \tag{3.23}$$

which are the adjoint and mesh sensitivity equations, respectively.

Therefore, once the adjoint solution has been found, $\bar{\mathbf{U}}$, the mesh sensitivities, $\bar{\mathbf{X}}$ can be computed using Equation (3.23). The total derivative of the objective function with respect to the design variables can be obtained from Equation (3.19) and yields

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha}^T = \frac{\mathrm{d}}{\mathrm{d}\alpha}\mathbf{M}^T(\alpha)\bar{\mathbf{X}}, \tag{3.24}$$

where

$$\frac{\mathrm{d}\mathbf{M}}{\mathrm{d}\alpha} = \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}, \tag{3.25}$$

which results in Equation (3.1). Therefore, for the computation of the total sensitivity, the gradient of the volumetric mesh with respect to the surface grid coordinates is needed.

It is important to note that, in the context of *SU2*, the consecutive call of the subroutines `SU2_CFD_AD` and `SU2_DOT_AD`, described in Section 3.3.2, yields the objective function sensitivities on the blade surface, this is

$$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}} = \left(\frac{\partial J}{\partial \mathbf{x}_{\mathrm{surf}}}, \frac{\partial J}{\partial \mathbf{y}_{\mathrm{surf}}}, \frac{\partial J}{\partial \mathbf{z}_{\mathrm{surf}}}\right). \tag{3.26}$$

3.5 Gradient assembly

Once the gradient information is available from all the different modules composing the design chain, the total sensitivity of the objective function with respect to the design variables needs to be assembled, as it will be later used by the optimizer to determine the next design step. The following procedure can be applied, as well, to the sensitivities of any given flow constrain, $\frac{\mathrm{d}G}{\mathrm{d}\alpha}$.

The process that is followed when assembling the total gradient can be illustrated with a simple example. Figure 3.11 shows the 2D cross section of a turbine blade, where a simple parametrization is introduced such that only one design variable, namely α_1, controls the position of three arbitrary points in the blade surface ($\mathbf{x}_{S1}, \mathbf{x}_{S2}$ and $\mathbf{x}_{S3}$) as shown in the figure.

The gradient of the objective function with respect to the surface coordinates will consist of a map of sensitivities such that

$$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}} = \begin{pmatrix} \frac{\partial J}{\partial x_{S1}} & \frac{\partial J}{\partial y_{S1}} \\ \frac{\partial J}{\partial x_{S2}} & \frac{\partial J}{\partial y_{S2}} \\ \frac{\partial J}{\partial x_{S3}} & \frac{\partial J}{\partial y_{S3}} \end{pmatrix}. \tag{3.27}$$

Similarly, the sensitivities corresponding to the parametrization will be

$$\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha_1} = \begin{pmatrix} \frac{\partial x_{S_1}}{\partial \alpha_1} & \frac{\partial y_{S1}}{\partial \alpha_1} \\ \frac{\partial x_{S_2}}{\partial \alpha_1} & \frac{\partial y_{S2}}{\partial \alpha_1} \\ \frac{\partial x_{S_3}}{\partial \alpha_1} & \frac{\partial y_{S3}}{\partial \alpha_1} \end{pmatrix}. \tag{3.28}$$

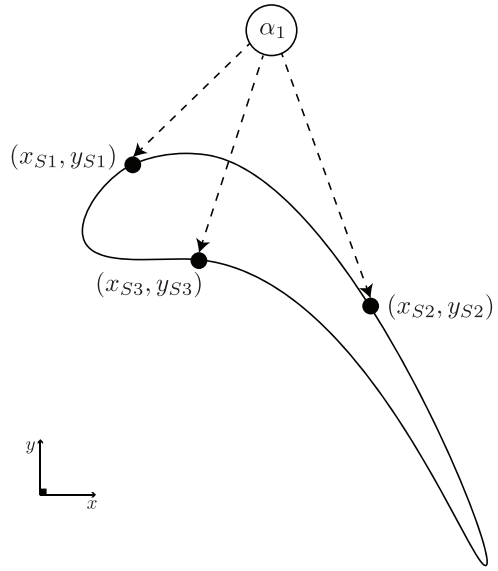


Figure 3.11: Representation of a simplified 2D turbine blade section parametrization.

Recalling Equation (3.1), the total sensitivities can be computed as the inner product of both sensitivity maps, such as

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha_1} = \left(\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}_{\text{surf}}}\right)\left(\frac{\mathrm{d}\mathbf{X}_{\text{surf}}}{\mathrm{d}\alpha_1}\right)^T = \begin{pmatrix} \frac{\partial J}{\partial x_{S1}} & \frac{\partial J}{\partial y_{S1}} \\ \frac{\partial J}{\partial x_{S2}} & \frac{\partial J}{\partial y_{S2}} \\ \frac{\partial J}{\partial x_{S3}} & \frac{\partial J}{\partial y_{S3}} \end{pmatrix} \begin{pmatrix} \frac{\partial x_{S_1}}{\partial \alpha_1} & \frac{\partial x_{S2}}{\partial \alpha_1} & \frac{\partial x_{S3}}{\partial \alpha_1} \\ \frac{\partial y_{S_1}}{\partial \alpha_1} & \frac{\partial y_{S2}}{\partial \alpha_1} & \frac{\partial y_{S3}}{\partial \alpha_1} \end{pmatrix} = \tag{3.29}$$

$$= \frac{\partial J}{\partial x_{S1}}\frac{\partial x_{S_1}}{\partial \alpha_1} + \frac{\partial J}{\partial y_{S1}}\frac{\partial y_{S_1}}{\partial \alpha_1} + \frac{\partial J}{\partial x_{S2}}\frac{\partial x_{S_2}}{\partial \alpha_1} + \frac{\partial J}{\partial y_{S2}}\frac{\partial y_{S_2}}{\partial \alpha_1} + \frac{\partial J}{\partial x_{S3}}\frac{\partial x_{S_3}}{\partial \alpha_1} + \frac{\partial J}{\partial y_{S3}}\frac{\partial y_{S_3}}{\partial \alpha_1}.$$

Equation (3.29) can be re-written in a generalized way, for 3D cases with more than one design variable as

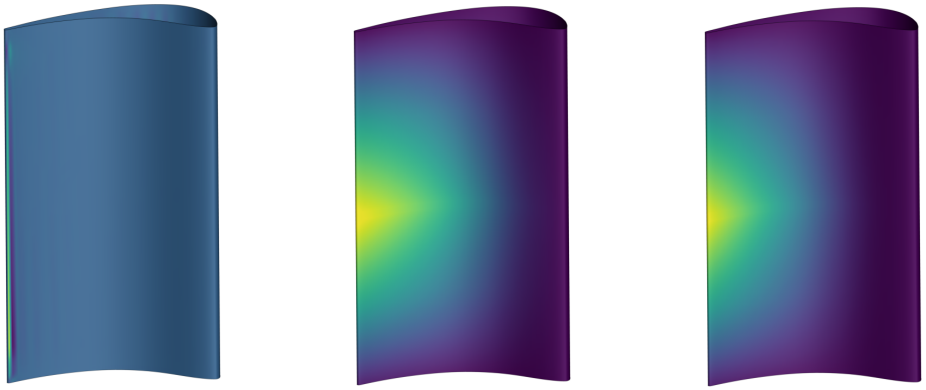
$$\frac{\mathrm{d}J}{\mathrm{d}\alpha_i} = \sum_{n=0}^{N_{\text{surf}}} \left(\frac{\partial J}{\partial x_{Sn}}\frac{\partial x_{Sn}}{\partial \alpha_i} + \frac{\partial J}{\partial y_{Sn}}\frac{\partial y_{Sn}}{\partial \alpha_i} + \frac{\partial J}{\partial z_{Sn}}\frac{\partial z_{Sn}}{\partial \alpha_i} \right) \quad \text{for } i = 1, ..., N_\alpha, \tag{3.30}$$

where N_α corresponds to the number of design variables and N_{surf} to the number of surface grid coordinates. The output will then be an array composed by the sensitivities of each of the design variables,

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = \left(\frac{\mathrm{d}J}{\mathrm{d}\alpha_1}, \frac{\mathrm{d}J}{\mathrm{d}\alpha_2}, ..., \frac{\mathrm{d}J}{\mathrm{d}\alpha_{N_\alpha}}\right). \tag{3.31}$$

As an example to illustrate this mathematical operation, Figure 3.12 shows the sensitivity maps for all the different gradients that are involved in the dot product operation: the adjoint (3.12a), CAD (3.12b) and the resulting (3.12c) dot product sensitivities.

The first figure reveals that, in order to minimize the objective function J, the most sensitive region is the trailing edge, as depicted by the yellow and dark purple bands along it. On the other hand, for the design parameter chosen, the surface coordinates are most affected by a variation in its value at the trailing edge of, approximately, a mid-span section. The last figure shows that, by performing the dot product between these two, the gradient will lead to changes at the mid-span's trailing edge in order to minimize the objective function.



(a) $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}$ (b) $\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}$ (c) $\frac{\mathrm{d}J}{\mathrm{d}\alpha}$

Figure 3.12: Sensitivity maps for a given parameter in an axial turbine blade. High sensitivity corresponds to yellow colors, while low sensitivities to dark purple colors.

3.6 Gradient validation

As a previous step before starting the shape optimization process, it is required to validate the gradients computed by the discrete adjoint solver. This will not only give a hint on how accurate their values are, but more importantly if the gradients are pointing towards the right direction (i.e. positive or negative).

A common practice seen in the literature review, Chapter 2, is to compare the adjoint sensitivities to those computed by finite differences, which are deemed to be correct. In an ideal situation, the gradients proceeding from the dot product will be equal to those computed by finite differences, this is

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\alpha}\right|_{\mathrm{AD}} = \left.\frac{\mathrm{d}J}{\mathrm{d}\alpha}\right|_{\mathrm{FD}}, \tag{3.32}$$

where

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\alpha}\right|_{\mathrm{FD}} = \frac{J(\alpha+\delta) - J(\alpha)}{\delta}, \tag{3.33}$$

in case of a forward finite differences scheme.

However, this will never happen due to the numerical errors that are inherent to the CFD simulations, therefore the gradients will be validated within a certain deviation from those of finite differences. A common measurement of the quality of the gradient validation is the root mean square error, or RMSE, between the difference of both gradients for all the design variables,

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_\alpha}\left[\sum_{n=0}^{N_\alpha}\left(\left.\frac{\mathrm{d}J}{\mathrm{d}\alpha}\right|_{\mathrm{FD}} - \left.\frac{\mathrm{d}J}{\mathrm{d}\alpha}\right|_{\mathrm{AD}}\right)_n^2\right]}. \tag{3.34}$$

Even though this is a computationally expensive step, as a direct solution will need to be computed for each of the design variables, it only needs to be performed once before the optimization is started. The quality of the method will mostly reside on how close the AD sensitivities are to those predicted by finite differences.

3.7 Optimization

The component which closes the entire process is the optimizer. For this study, as a numerical framework will be written in Python, the optimization package in NumPy [47] is be used, concretely the SLSQP algo-

rithm [48]. As implemented in this Thesis, the SLSQP algorithm is used to solve nonlinear optimization problems to minimize a scalar objective function,

$$J(\alpha),$$

subject to an inequality constrain,

$$G(\alpha) \geq 0.$$

As anticipated in Section 3.4, the different objective functions and constrains that are already defined in *SU2* are [30]:

1. Entropy generation, s_{gen}, defined as

$$s_{\text{gen}} = \frac{\langle s_{\text{out}} \rangle - \langle s_{\text{in}} \rangle}{v_0^2 / T_{\text{t,in}}}, \tag{3.35}$$

   where the values of the entropy at inlet and outlet of the domain are calculated using a mass-averaged/mixed-out average, $T_{\text{t,in}}$ is the total inlet temperature, and the spouting velocity, v_0, is defined as $v_0 = \sqrt{2(h_{\text{t,in}} - h_{\text{is,out}})}$ where $h_{\text{t,in}}$ and $h_{\text{is,out}}$ are the total enthalpy at the inlet and the isentropic outlet static enthalpy, respectively.

2. Kinetic energy loss, ζ_K, defined as

$$\zeta_K = \frac{\langle h_{\text{t,out}} \rangle - \langle h_{\text{is,out}} \rangle}{\langle v_{\text{is,out}} \rangle^2} \tag{3.36}$$

   being $\langle h_{\text{t,out}} \rangle$ the averaged total enthalpy, $\langle h_{\text{is,out}} \rangle$ the averaged isentropic enthalpy and $\langle v_{\text{is,out}} \rangle$ the averaged isentropic velocity all evaluated at the domain's outlet. All averages are computed using the mixed-out averaging procedure.

3. Total pressure loss, ζ_P, defined as

$$\zeta_P = \frac{\langle p_{\text{t}} \rangle_{\partial\Omega_1} - \langle p_{\text{t}} \rangle_{\partial\Omega_2}}{\langle p_{\text{t}} \rangle_{\partial\Omega_2} - \langle p_2 \rangle_{\partial\Omega_2}}, \tag{3.37}$$

   where $\langle p_{\text{t}} \rangle_{\partial\Omega_i}$ is the mixed-out average total pressure over boundary i, and $\langle p_2 \rangle_{\partial\Omega_2}$ the mixed-out averaged static pressure at the domain's outlet. All averages are computed using the mixed-out averaging procedure.

4. Outlet flow angle, β_{out}: certain values of the flow angle are required in order for the downstream components not to alter their operation and the overall turbine performance. In order to keep a certain flow deflection, the outlet flow angle is usually established as an inequality constrain, being calculated as

$$\beta_{\text{out}} = \text{arc tan}\left(\frac{v_{\text{tan}}}{v_{\text{ax}}}\right)_{\text{out}}. \tag{3.38}$$

5. Dimensionless power output, P^*: especially for radial turbomachinery, usually composed by a unique stage, it is of great important to keep the power output above a minimum threshold in order to deviate as least as possible from the initial operating point. The dimensionless power output is calculated as

$$P^* = \frac{w\ \dot{m}}{\rho_{\text{t,in}} y_{\text{p}} u_{\text{b}}^3}. \tag{3.39}$$

Once the design vector, α, the objective function and the constrains, if any, have been defined, the optimization problem can be solved. This process will have as a result a set of optimized design variables α_{opt}, that yield a minimum value of the objective function, J_{opt}.

The different modules that have been exposed in this Chapter are connected amongst each other in a numerical framework that is extensively described in the following Chapter.

Chapter 4

Numerical framework

In this chapter, the coupling of the in-house blade parametrizer *ParaBlade* with the open-source tool *SU2* is extensively explained and described. This implementation results in a new tool for the *ParaBlade* suite, which is written in Python and exploits the capabilities from `MatchBlade.py` and `MakeBlade.py` as explained in Chapter 3.

The way these three tools are interconnected between each other can be summarised in Figure 4.1. Firstly, from an existing geometry, the reference surface coordinates are matched with the existing parametrization used: u, v, α. Subsequently, the matched parameters, α as well as their $u-$ and $v-$ coordinates are fed into the optimizer, which will in turn also receive geometrical sensitivities information from `MakeBlade.py` and it will output the optimum surface coordinates.

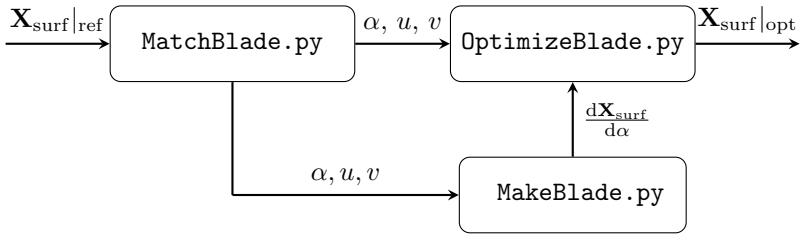


Figure 4.1: `ParaBlade` suite linkage.

Even though `MakeBlade.py` and `MatchBlade.py` are an essential part of this work, the focus of this Chapter will be on how `OptimizeBlade.py` was built and coupled with *SU2*.

4.1 OptimizeBlade.py overview

The main goal of `OptimizeBlade.py`, from now on `OptimizeBlade`, is to build a numerical framework capable of reading existing geometries, both their coordinates and parametrization, and couple these with the CFD suite *SU2*, which embeds a RANS fluid solver as well as a discrete adjoint and a mesh deformation algorithm based on the spring-analogy, needed for the efficient optimization of geometries as explained in Chapter 2.

The script is structured such that for each geometry a new class, named `OptiBlade`, is created. This allows for a modular-like code, where firstly an object is created with its specific attributes (such as surface coordinates, values of the parameters, amongst others) and then the different methods embedded can change these properties by performing operations or calling other modules on them (for example, running the discrete adjoint solver to generate the surface sensitivities).

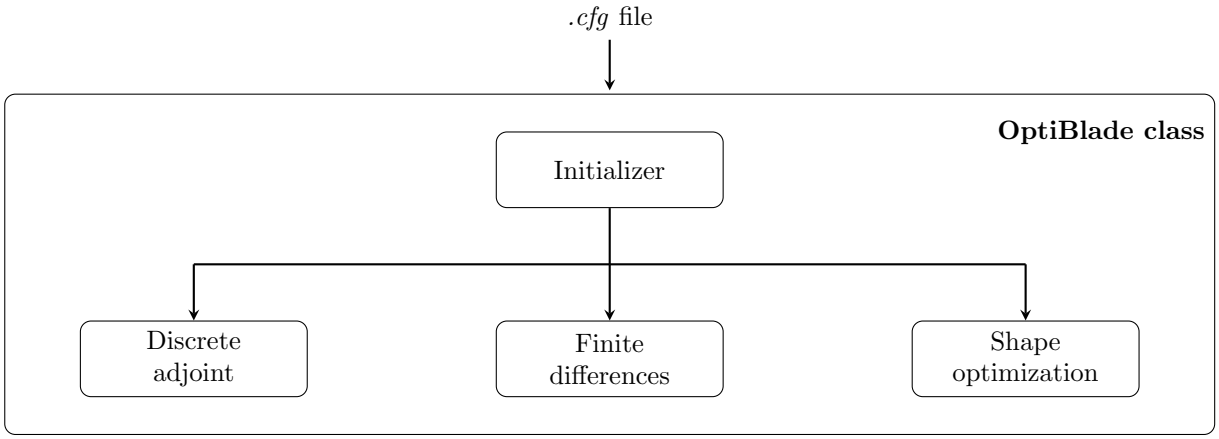


Figure 4.2: OptiBlade class creation and starting process.

Figure 4.2 shows how the class is created and the possible modules that can be executed. The user provides a configuration *.cfg* file which contains information such as the design variables to be considered, the name of the required files and the type of operation to be performed, amongst others. Then, based on the type of operation selected by the user, the script will execute any of the modules displayed on Table 4.1. These will be described in the following sections.

Table 4.1: Modules embedded in `OptimizeBlade.py`, functionality and output.

Module	**Functionality**	**Output**
Discrete adjoint	Run discrete adjoint for validation	AD sensitivities
Finite differences	Run finite differences for validation	FD sensitivities
Shape optimization	Run shape optimization	Optimized geometry

It is also important to note the order of execution for the different modules. Figure 4.3 shows how the script is structured prior to running a shape optimization problem. It is of great importance to perform gradient validation before the shape optimization, as argued in Chapter 3, to ensure the accuracy of the discrete adjoint gradients. Furthermore, this order is chosen to save computational time, as finite differences (only forward-schemes, as it will be explained in Section 4.1.2) will use the flow solution from the discrete adjoint module if existent.

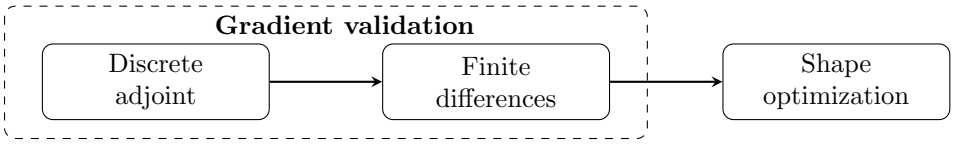


Figure 4.3: Order of execution of the different modules in `OptimizeBlade.py`.

4.1.1 Discrete adjoint module

In this module, the gradient of the objective function with respect to the design variables is obtained as a first step towards their validation against finite differences. A flow chart of its basic operations can be seen in Figure 4.4.

Firstly, the discrete adjoint embedded in *SU2* is executed (`SU2_CFD_AD`). This module outputs the adjoint solution in the volumetric grid, $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}}$ therefore the sensitivities need to be projected onto the blade's

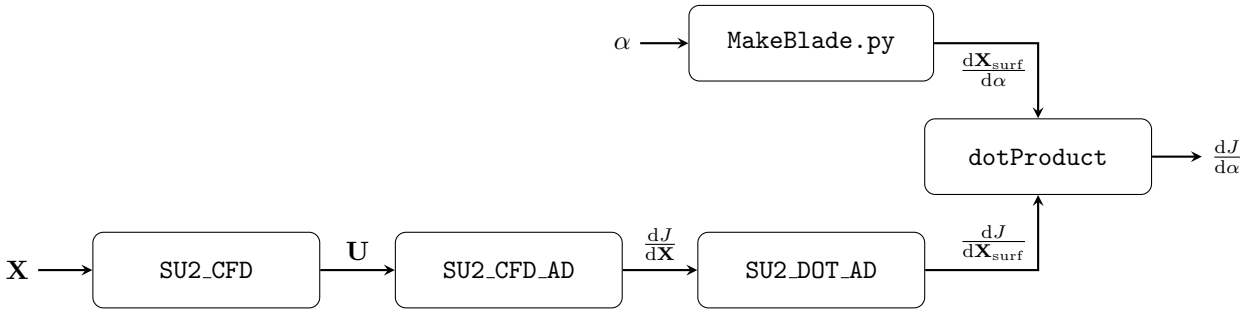


Figure 4.4: Discrete adjoint module flow chart.

surface. The latter is done in `SU2_DOT_AD`, where the spring-based elasticity equations are derived by automatic differentiation. As an outcome, `SU2_DOT_AD` generates a surface file containing a map of the gradient of the objective function with respect to the surface coordinates, $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}$.

Once the surface sensitivities are obtained, `MakeBlade.py` is used to obtain the gradient of the surface coordinates with respect to the design variables, $\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}$. This value, together with the surface sensitivities previously obtained, can be multiplied and it will output the gradient of the objective function with respect to the design variables, $\frac{\mathrm{d}J}{\mathrm{d}\alpha}$, also known as total sensitivities. This operation is performed in a method named `dotProduct`.

dotProduct method

This method is responsible for the coupling between the CFD adjoint solver and the blade parametrizer. It is not only used by the Discrete Adjoint module, but also at each design step it is called by Shape Optimization to assemble the corresponding gradients as explained in Chapter 3. Therefore, this module can be considered to be the core of the methodology implemented.

Firstly, as the solution written by the adjoint solver and by `ParaBlade` may be disordered, it is important to locate the points prior to the dot product operation. This is done by a closest neighbour algorithm, such that for each of the points written in the surface sensitivities file by the discrete adjoint, x_{CFD}, the distance, $\mathbf{D}$, to each of the points output by `ParaBlade`, $\mathbf{x}_{\mathrm{CAD}}$, is computed such that

$$\mathbf{D} = \sqrt{(\mathbf{x}_{\mathrm{CAD}} - x_{\mathrm{CFD}})^2 + (\mathbf{y}_{\mathrm{CAD}} - y_{\mathrm{CFD}})^2 + (\mathbf{z}_{\mathrm{CAD}} - z_{\mathrm{CFD}})^2}. \tag{4.1}$$

Then, its corresponding point in the `ParaBlade` output is found by computing the position of the minimum distance in the corresponding array,

$$(x, y, z)_{\mathrm{CAD}} \approx (x, y, z)_{\mathrm{CFD}} \iff \min(\mathbf{D}). \tag{4.2}$$

Subsequently, the dot product for a given design variable α between the adjoint and CAD sensitivities can be computed as

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = \sum_{n=0}^{N_{\mathrm{surf}}} \left[\left(\frac{\mathrm{d}J}{\mathrm{d}x_S} \frac{\mathrm{d}x_S}{\mathrm{d}\alpha} \right)_n + \left(\frac{\mathrm{d}J}{\mathrm{d}y_S} \frac{\mathrm{d}y_S}{\mathrm{d}\alpha} \right)_n + \left(\frac{\mathrm{d}J}{\mathrm{d}z_S} \frac{\mathrm{d}z_S}{\mathrm{d}\alpha} \right)_n \right], \tag{4.3}$$

where N_{surf} is the total number of surface coordinates, yielding the total sensitivity for the specified design variable. For 2D cases, the last addend will be equal to zero. Algorithm 1 shows how this is done in a numerical manner.

4.1.2 Finite differences module

As a following step towards gradient validation, the total sensitivities are computed by using finite differences and will be later compared to those proceeding from the discrete adjoint module. The flow chart

Algorithm 1 Dot product pseudo-code.

```
function dotProduct
    Initialize sensitivity array: dJ/dα;
    for each design variable α in α do
        Initialize sensitivity variable: dJ/dα_α;
        for each point x_CFD in X_CFD do
            Compute distance D between x_CAD and x_CFD;
            Find position in x_CAD such that min(D);
            Compute dot product between dX_surf/dα and dJ/dX_surf for the given point in the three spatial coordinates;
            Add the result of the dot product to the variable dJ/dα|_α;
        end for
        Include dJ/dα|_α in the sensitivity array dJ/dα;
    end for
    return Sensitivity array dJ/dα;
end function
```

for this module can be seen in Figure 4.5.

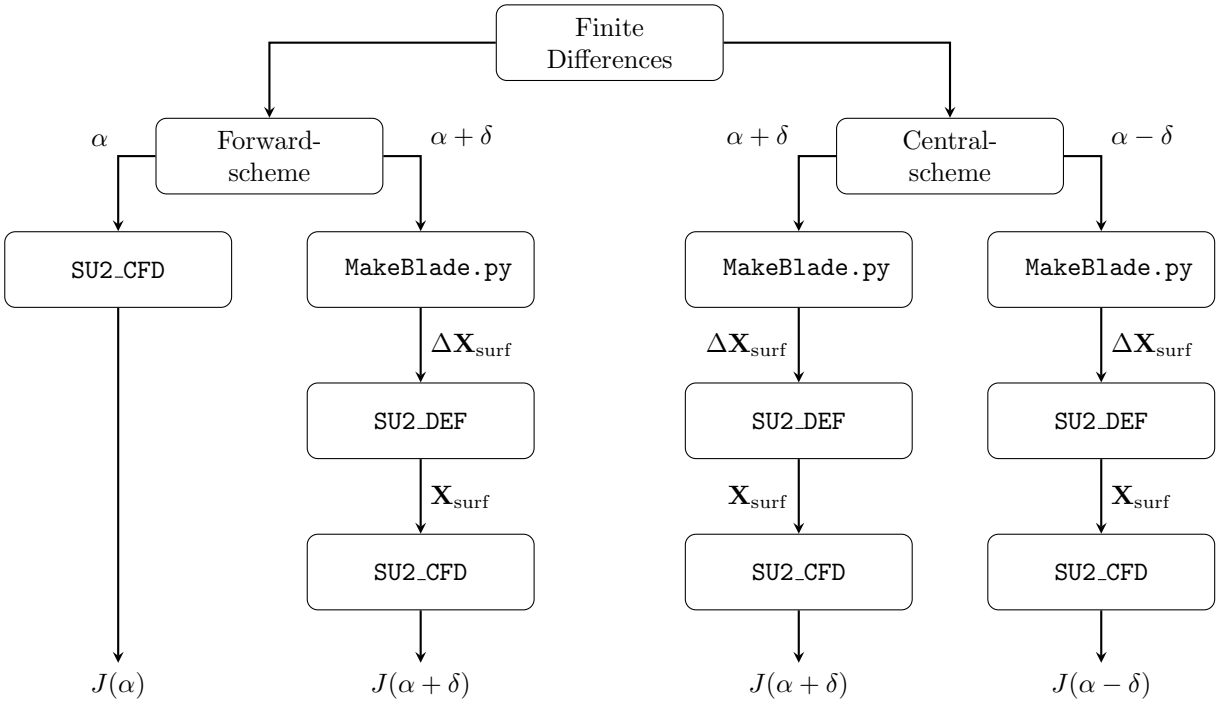


Figure 4.5: Finite differences module flow chart.

Two possible methods for computing the objective function gradient with respect to the design variables can be used: forward-scheme or central-scheme. For the former, two geometries are used: the reference (unperturbed) blade, and the perturbed blade with a step of plus δ. For the latter, on the other hand, the two geometries used are both perturbed by plus δ and minus δ.

In both of cases the process is identical: for those geometries that are perturbed, firstly the modified parameter α_{def} is computed,

$$\alpha_{\text{def}} = \left(1 + \frac{\delta}{100}\right)\alpha_{\text{ref}}. \tag{4.4}$$

Then, with the new perturbed parameter, the new surface coordinates $\mathbf{X}_{\text{surf}}$ are computed, which will

in turn be fed to SU2_DEF to perform a mesh deformation,

$$\alpha_{\text{def}} \rightarrow \Delta \mathbf{X}_{\text{surf}} \rightarrow \texttt{SU2\_DEF} \rightarrow \mathbf{X}_{\text{surf}}\big|_{\text{def}}.$$

Finally, with the new mesh, the objective function value is computed running SU2_CFD and the total gradient is calculated differently depending on which finite differences scheme is chosen:

Forward-scheme The total gradient is computed as

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} \approx \frac{J(\alpha+\delta) - J(\alpha)}{\delta}. \tag{4.5}$$

Central-scheme The total gradient is computed as

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} \approx \frac{J(\alpha+\delta) - J(\alpha-\delta)}{2\delta}. \tag{4.6}$$

From the computational point of view, it is evident that using central-scheme finite differences is more expensive, as the forward-scheme uses the flow solution obtained in the previous module shown in Section 4.1.1. However, this will come at a benefit which will be increased gradient accuracy.

4.1.3 Shape optimization module

The shape optimization module is the culmination of all the previous modules. Once gradient validation is obtained (i.e. the discrete adjoint gradients are comparable to those of the finite differences), shape optimization can be started.

Figure 4.6 shows the flow chart for this module. Firstly, the optimization instance is initialized by providing an array containing the initial design vector, such as

$$\alpha_0 = \{\alpha_1, \alpha_2, ..., \alpha_{N_\alpha}\}_0, \tag{4.7}$$

where N_α is the number of design variables. Together with the initial design vector, an array containing the bounds for each of the design variables can be specified, such that

$$\mathbf{lb} \leq \alpha \leq \mathbf{ub}, \tag{4.8}$$

subject to

$$\begin{aligned} \mathbf{lb} &= (1-\varepsilon)\{\alpha_1, \alpha_2, ..., \alpha_{N_\alpha}\}_0, \\ \mathbf{ub} &= (1+\varepsilon)\{\alpha_1, \alpha_2, ..., \alpha_{N_\alpha}\}_0, \end{aligned} \tag{4.9}$$

where ε can be defined as the accepted tolerance for the variation of a given design variable.

The shape optimization module is based on an optimizer which uses the SLSQP algorithm [48] integrated in NumPy [47]. Furthermore, two types of optimizations can be carried out:

Unconstrained optimization For this type of optimization, the objective function is minimized without being subjected to any constrain. Therefore, the optimization problem is formulated as

$$\min_{\alpha} \qquad J(\alpha),\ \alpha = \{\alpha_1, ..., \alpha_{N_\alpha}\}. \tag{4.10}$$

Constrained optimization On the other hand, the constrained problem is posed as

$$\min_{\alpha} \qquad J(\alpha),\ \alpha = \{\alpha_1, ..., \alpha_{N_\alpha}\}, \tag{4.11a}$$

$$s.t. \qquad G(\alpha) \geq 0, \tag{4.11b}$$

where only support to inequality constrains has been yet implemented.

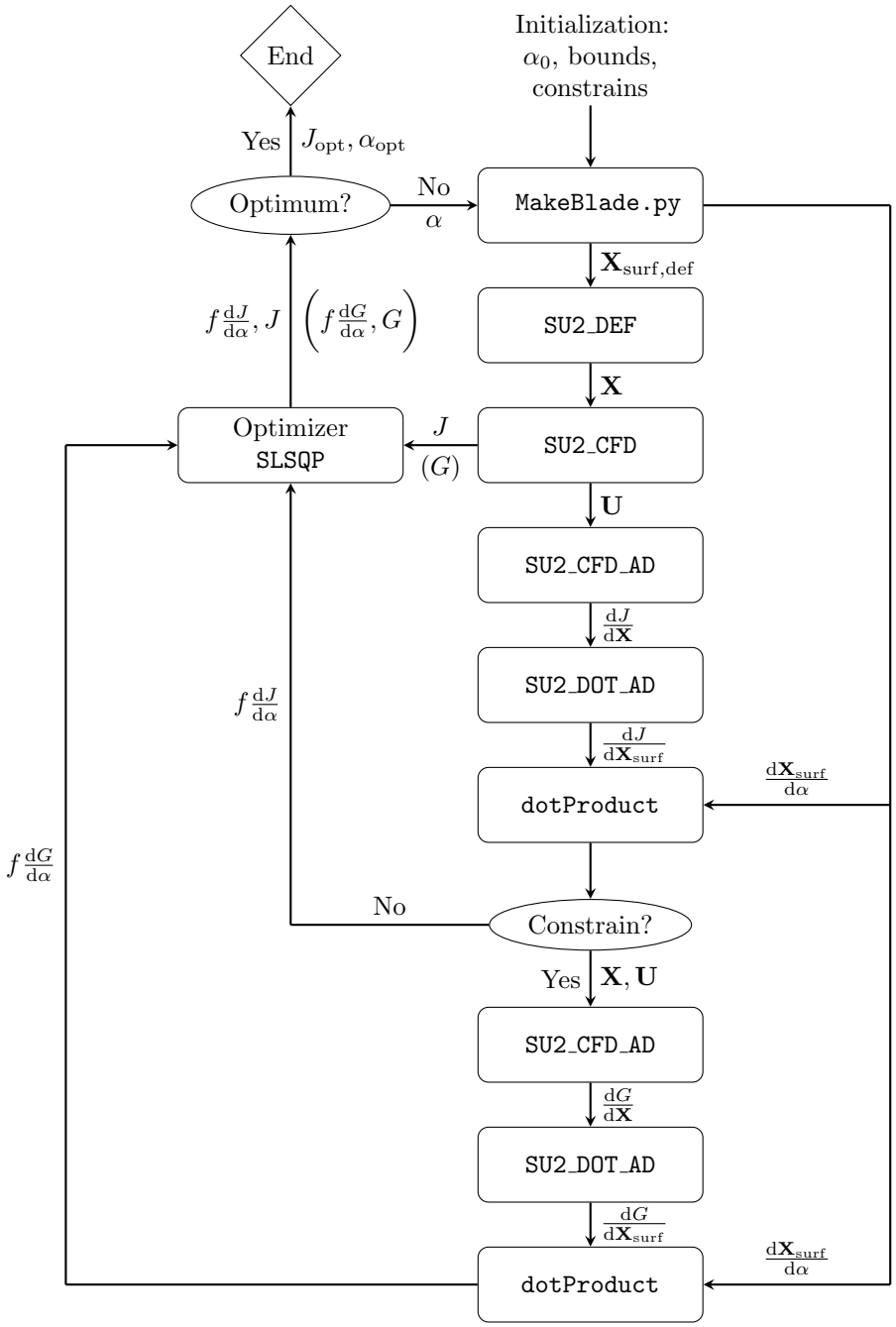


Figure 4.6: Shape optimization module flow chart.

The current support for objective functions and inequality constrains is, as explained in Chapter 3:

- Objective functions:
  1. Entropy generation.
  2. Kinetic energy loss.
  3. Total pressure loss.
- Inequality constrains:
  1. Flow angle out.

2. Dimensionless power output.

The optimization process is as it follows: firstly the design vector α, which consists of the new set of design parameters chosen by the optimizer, is fed to `MakeBlade.py`, having as an output the new set of surface coordinates, $\mathbf{X}_{\text{surf,new}}$ and the gradient of the surface coordinates with respect to the design variables, $\frac{\mathrm{d}\mathbf{X}_{\text{surf}}}{\mathrm{d}\alpha}$.

The new set of surface coordinates, $\mathbf{X}_{\text{surf,new}}$, is then input to the spring-analogy-based mesh deformation algorithm embedded in `SU2_DEF`, which will in turn apply the mesh deformation to an existing mesh, having as an output the new volumetric grid, $\mathbf{X}$. However, as it was mentioned before, mesh deformation can be a bottleneck if the new surface mesh coordinates, $\mathbf{X}_{\text{surf,new}}$, differ widely from surface coordinates of the grid to be deformed. Therefore, an algorithm is implemented for which, in each design step, a search of the most similar mesh is computed. This is done by calculating the root mean square of the difference between $\mathbf{X}_{\text{surf,new}}$, and the rest of the surface coordinates from all the existing design steps. Algorithm 2 shows a schematic representation of the mesh search function.

Algorithm 2 "Most similar mesh" pseudo-code.

```
function searchMesh
    for each design step k in N_steps do
        Initialize variable to store added up values of the squared distance D_sum;
        for each surface coordinate x_surf in X_surf do
            Compute distance D between the new point x_surf and the corresponding surface point in step k;
            Compute the square of the distance D^2 and add it to variable D_sum;
        end for
        Compute the RMSE.
    end for
    Find position such that min(RMSE).
    return Design step with closest mesh.
end function
```

Once the deformed mesh $\mathbf{X}$ is obtained, the new discretized domain is used to evaluate the objective function value J by running a CFD problem using the flow solver in `SU2_CFD`. The converged flow solution, $\mathbf{U}$, is later fed to the discrete adjoint solver in `SU2_CFD_AD`, which will in turn calculate the sensitivities of J with respect to the volumetric mesh coordinates, $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}}$. These sensitivities are then projected onto the blade surface by `SU2_DOT_AD`, obtaining $\frac{\mathrm{d}J}{\mathrm{d}\mathbf{X}_{\text{surf}}}$.

Then, the total gradient of the objective function with respect to the design variables, $\frac{\mathrm{d}J}{\mathrm{d}\alpha}$, is computed by using the `dotProduct` method, as explained in Section 4.1.1. This information, together with the value of the objective function J coming from `SU2_CFD`, is then fed to the optimizer which will in turn decide the step to be applied to the design variables towards the search of the minimum value of J.

Up to this point, it is important to mention that the gradient information that is passed to the optimizer is previously multiplied by the so-called relaxation factor, f, such that

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\alpha}\right|_{\text{opt}} = f\left.\frac{\mathrm{d}J}{\mathrm{d}\alpha}\right|_{0}, \tag{4.12}$$

which is of great utility in order to avoid having spurious gradients in highly-sensitive regions such as the trailing edge. Furthermore, it allows for an indirect control of the step size taken by the optimizer by giving less weight to the objective function sensitivities. In the current implementation, f is selected by the user.

If a constrained optimization is selected, after the computation of the objective function value and its gradient, a new discrete adjoint run is computed where, in this case, the constrain sensitivities are calculated, $\frac{\mathrm{d}G}{\mathrm{d}\alpha}$, in an identical process as before as the `dotProduct` function is again utilized.

Once the gradient information and the objective function's value, as well as the constraint value and its gradient in case of a constrained optimization, are fed to the optimizer, it is decided based on all the previous information whether the optimum has been reached or not. If the latter occurs, the process will repeat until J_{opt} is found.

4.2 Other features

Besides the aforementioned modules, `OptiBlade.py` incorporates other features which makes it attractive and eases the process of performing a shape optimization problem.

As it was already mentioned in the previous section, the main bottleneck of the shape optimization problem is the mesh deformation. Very fine meshes and/or large deformation steps can cause issues with the mesh deformation algorithm as the linear equation system becomes too stiff to resolve.

One of the main implementations that was introduced in `OptimizeBlade.py` was the "most similar mesh" search algorithm, for which the program will search amongst all the available meshes the closest one for the new set of surface coordinates and will perform the mesh deformation on it. This in turn reduces the deformation step as, for example, if the deformation was performed on the reference mesh.

However, this was not the only method for ensuring a good quality in the grid morphing module. The SLSQP optimization algorithm allowed for an indirect control of the optimization step by the use of the so-called relaxation factor, f.

Nevertheless, it is of extreme convenience to have the possibility of altering the relaxation factor during the shape optimization process itself. This, in turn, could allow for a better control of the design step once the solution is approaching the optimum or, for example, permitting for a more disruptive step at the first iterations of the optimization in order to jump rapidly to the optimal region.

Therefore, two possibilities could potentially be implemented in a turbomachinery-based shape optimization script to overcome this limitation:

1. Having a variable or dynamic relaxation factor which could automatically change in each optimization step based on mesh information.
2. Having the possibility of manually restarting the optimization from previous runs with a different relaxation factor.

While the first option may be attractive from an automated-design point of view, it might be firstly tedious to implement and secondly difficult to evaluate as most of the times mesh quality is assessed by simple visualization. Therefore, the second possibility is deemed to be the most convenient, as it as well helps in cases where the optimization is suddenly stopped (e.g. when using high performance computing clusters with limited walltime jobs).

The implementation of the restarting option is based on the use of the so-called *pickle* files [49], which enables to save variables in files and later on restoring those in other scripts. In this case the use of the *pickle* package is really convenient as it allows to store variables in any given format and later on, when restored, the same format is maintained.

Two *pickle* files are always created when running the shape optimization module (`ob_fun.pickle` and `sens_obj.pickle`) and a third one will be created in case of a constrained optimization (`sens_cons.pickle`). The content of each of these files is shown in Figure 4.7. Upon restarting, the script will loop over the existent design folders per optimization step and load these variables. As it will read the sensitivities per iteration, it is very important to check that the design vector for each of the steps coincides with the loaded *pickle* design vector, this is

$$\alpha_{\text{opt}} = \alpha_{\text{opt},0}. \tag{4.13}$$

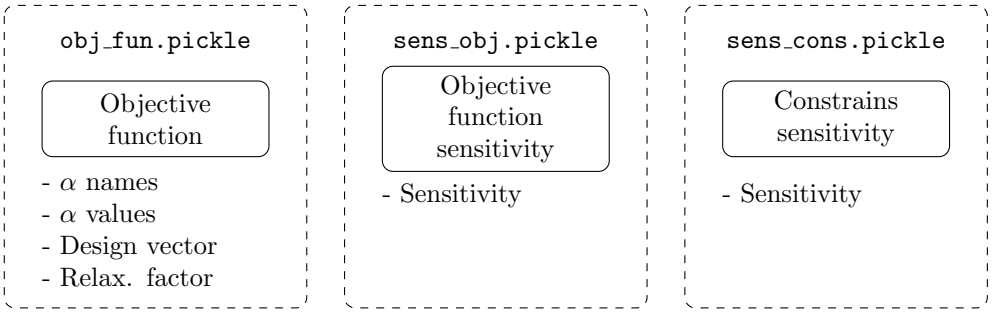


Figure 4.7: Contents per *pickle* file for the restarting option.

Furthermore, in order to change the relaxation factor, for the last of the design steps available the new sensitivity will be computed as

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\alpha}\right|_{\mathrm{new}} = \frac{f_{\mathrm{new}}}{f_{\mathrm{old}}} \left.\frac{\mathrm{d}J}{\mathrm{d}\alpha}\right|_{\mathrm{old}}, \tag{4.14}$$

where f is the relaxation factor and the subscripts old and new correspond to the loaded and computed values, respectively.

Chapter 5

Test cases

In this chapter, the test cases used to verify that the methodology exposed in Chapter 3 and 4 is functional and robust are documented. These consist of two different geometries, which cover the most-used types of turbomachines nowadays: axial and radial turbines. For the former, the well-known Aachen prismatic turbine stator will be exposed in Section 5.1, while for the latter an APU turbocharger turbine rotor will be introduced in Section 5.2.

5.1 Aachen turbine stator

In order to prove the functionality of the method proposed in this Master Thesis, the first test case consists of the transonic Aachen axial turbine stator. This is a common test case to be analyzed in validation of CFD codes, as its experimental results are publicly available.

The Aachen turbine is a 1.5-stage turbine which was widely studied and analyzed at the Institute of Jet Propulsion and Turbomachinery at RWTH Aachen University, Germany [50, 51]. The whole set-up consists of a first vane row, which is constructed by an untwisted Traupel profile, followed by a rotor row built with an untwisted VKI profile and lastly a vane row identical to the first one.

The selection of this test case is motivated by its ease of convergence, as the flow through the passage is transonic, and no strong shock waves nor huge separation can hinder the numerical stability of the simulations. Furthermore, the Aachen turbine has served as validation for *SU2*'s RANS solver [30], and has been used previously as a shape optimization test case using the same flow solver in [30] and [38].

From a design point of view, it will also be interesting to examine how a prismatic blade will change its form in an aerodynamic shape optimization if it is left free to morph independently in different sections defined along the blade span.

5.1.1 Test case definition

As a multi-row, multi-stage optimization is out of the scope of this Thesis, the test case to be analyzed will only consist of the Aachen turbine's stator, which is constructed by stacking Traupel profiles without twist from hub to shroud. No tip gap is included in the geometry.

5.1.2 Geometry parametrization

The blade geometry, which is shown in Figure 5.1, is discretized using the parametrization described in Chapter 3.

The parameters used to univocally define the blade surface geometry are:

- x, y and z position of the leading edge.
- x and z position of the trailing edge.
- Stagger angle, ξ.
- Inlet and outlet blade angles, θ_{in} and θ_{out}.
- Inlet and outlet wedge angles, ϵ_{in} and ϵ_{out}.
- Leading and trailing edge radii, r_{in} and r_{out}.
- Thickness distribution law, d_1, d_2, d_3 and d_4.

Following the methodology described in Chapter 3, the baseline geometry, which initially consists of a cloud of surface points, is first parametrized using the variables above. This is done using `MatchBlade.py`, from the geometry parametrization suite *ParaBlade*.

The maximum deviation found in the blade matching process was of 0.44 mm. The matched blade is deemed to be satisfactory as the deviation from the prescribed and matched coordinates is relatively low when compared to the order of magnitude of the surface coordinates. The reader can refer to Appendix A for more details about the resulting parametrization.

It is important to note that, even though the initial geometry is fully-prismatic (i.e. the blade sections do not change in the span-wise direction, where the blade angles remain constant), for the optimization five sections are created which initially coincide with the prismatic blade's section, but in this case they can independently vary based on the optimizer's information. This way, the blade is expected to switch from a fully-prismatic to a twisted design.

5.1.3 Domain discretization

Initially, the structured mesh is created using ANSYS TurboGrid [52]. Special care is taken of the mesh close to the wall regions, i.e. the hub, shroud and the blade, as a y^+ close to 1 is needed due to the fact that *SU2* does not include wall functions. This can be easily achieved with TurboGrid, where an expansion rate of the cells adjacent to the boundary layer between 1.1 and 1.5 is recommended for stability of the numerical solver. The grid consists of approximately 500,000 elements. An overview of the mesh used can be seen in Figure 5.1.

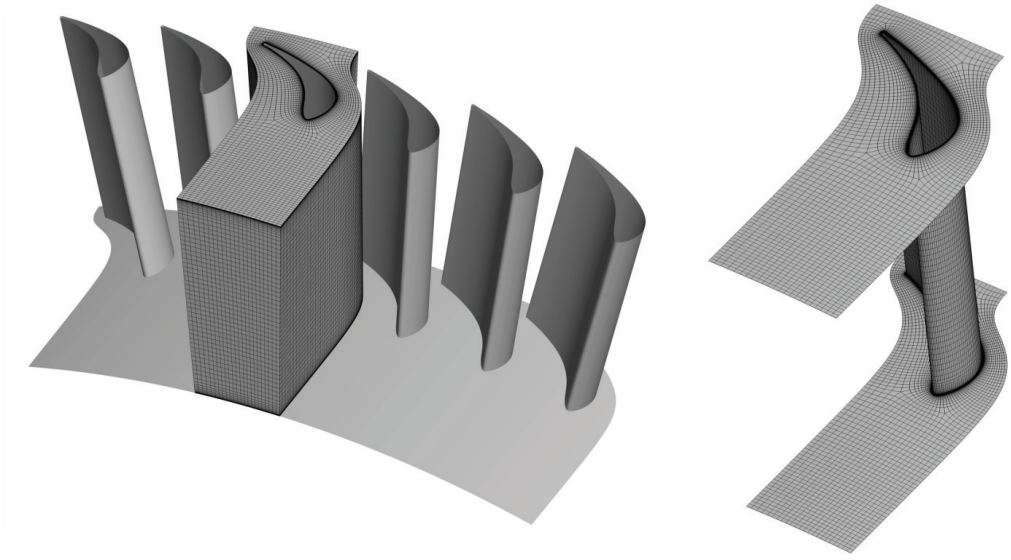


Figure 5.1: Grid used for the stator in the Aachen test case.

5.1.4 Flow and adjoint solver

The open-source code *SU2* is used as flow solver. The governing equations are the compressible Reynolds-averaged Navier-Stokes equations. The convective fluxes are discretized using a central second-order

Jameson-Schmidt-Turkel (JST) scheme [43]. The one-equaton Spalart-Allmaras turbulence model is employed as closure for the RANS equations. For the flow solver, the solution is obtained using an Euler implicit time-marching scheme, where a CFL of 1.0 is imposed. Non-reflective boundary conditions are used at both inlet and outlet, where the prescribed values can be checked on Table 5.1. A fully axial flow is imposed at the inlet.

Table 5.1: Non-reflective boundary conditions for the Aachen stator test case.

Inlet	Total pressure [Pa]	149623.33
	Total temperature [K]	305.76
Outlet	Static pressure [Pa]	99741.00

The hub, shroud and blade walls are deemed as adiabatic with the no-slip condition. In order to reduce the computational cost, only one blade row is simulated, therefore periodic boundary conditions in the circumferential direction are applied.

In order to obtain the adjoint and mesh sensitivities, the discrete adjoint solver embedded in *SU2* is used, which is obtained by automatic differentiation of the direct flow solver. Figure 5.2 shows the convergence history for both flow and adjoint solver. For the former, a residual reduction of, approximately, five orders of magnitude is achieved. In the case of the adjoint problem, residuals were reduced by three orders of magnitude, and the problem was deemed to be converged after 45,000 iterations.

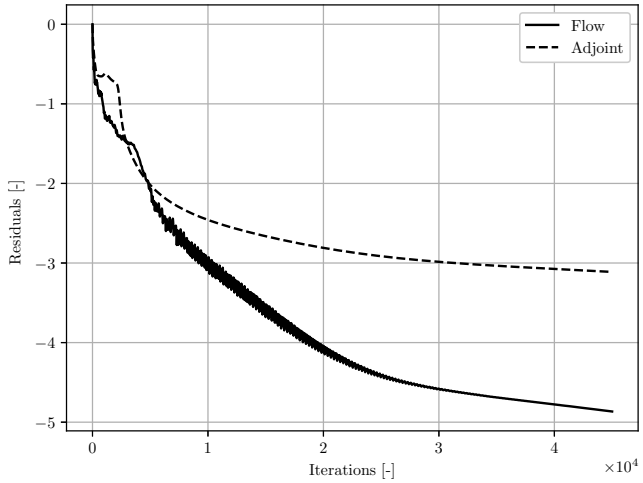


Figure 5.2: Residual evolution for both flow and adjoint problems for the Aachen turbine.

5.1.5 Computational performance

It is of great importance to keep track of the computational performance of both direct and adjoint runs, not only in terms of CPU-time but also on the used memory and resources. As it was mentioned before, a converged solution was obtained for the 500,000-element mesh in 45,000 iterations for both the flow and the adjoint solver. These solutions were obtained using 20 cores *Intel(R) Xeon(R) CPU E5-2670v2* CPU with 2.5 GHz clockspeed.

Figure 5.3 shows the computational performance, in CPU-time and memory usage, for all the required modules. On the right side, a pie chart with the percentage of CPU-time employed in each module is

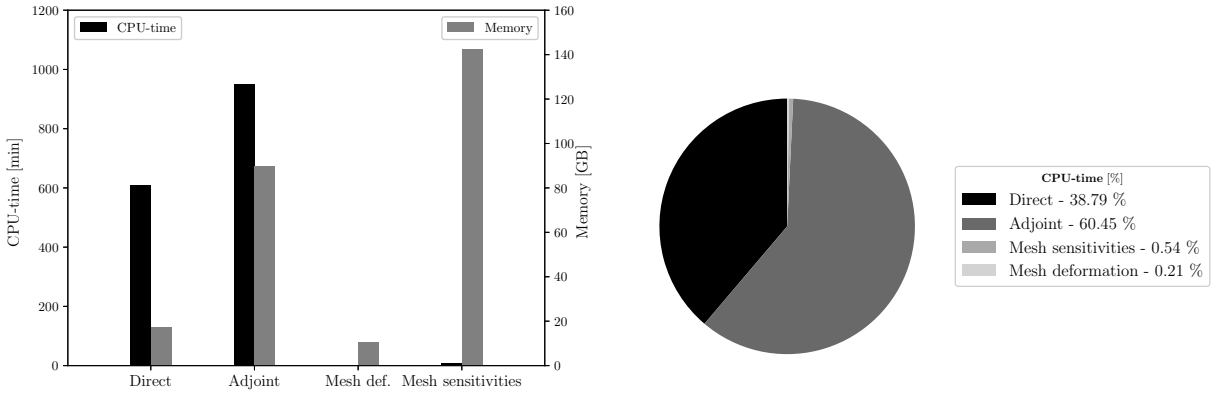


Figure 5.3: Computational performance, in CPU-time and memory usage, for the Aachen turbine stator test case.

shown. The solution was obtained in 609 minutes for the direct run, using a total of 17.3 GB, while 949 minutes and 89.9 GB were required for the discrete adjoint solver. However, the most memory exhausting process was the gradient projection, taking approximately 140 GB of memory. These values suppose a ratio of 1.6 between the CPU-time of the adjoint and direct CFD solver, and 5.2 between the memory usage. From these numbers, it becomes evident that the use of high-performance computing in order to run such test cases is compulsory.

5.2 APU turbine rotor

As the objective of the present Thesis is to prove the methodology for all types of turbomachinery, the next test case will correspond to the optimization of an auxiliary power unit (APU) turbocharger's turbine. The geometry will consist of a mixed-flow turbine, which is a 100 kW radial-inflow turbine extensively analyzed in [53], which has as well served for preliminary verification of the CFD solver *SU2* for radial-inflow configurations in [54].

This test case was selected for the following reasons:

- The flow across the rotor passage is transonic, therefore no strong shock-waves are expected to be formed that could hinder the primal and adjoint solver's convergence.
- The APU turbine has been widely tested [30, 54] in *SU2*, therefore it is expected to have a smooth convergence in the flow solver used that will ease the adjoint convergence.

5.2.1 Test case definition

As a multi-row optimization is out of the scope of this Thesis, the subject of investigation of this test case will only be the rotor (from now on named APU turbine rotor). No tip gap is included in the geometry.

5.2.2 Geometry parametrization

The geometry will be parametrized using the open-source suite *ParaBlade*. An overview of the turbine rotor can be seen in Figure 5.4.

The variables that are used to parametrize the rotor's geometry are:

- x, y and z position of the leading edge.
- x and z position of the trailing edge.
- x and z position of the hub and shroud.
- Stagger angle, ξ.
- Inlet and outlet wedge angles, ϵ_{in} and ϵ_{out}.
- Leading and trailing edge radii, r_{in} and r_{out}.
- Inlet and outlet blade angles, θ_{in} and θ_{out}.
- Thickness distribution law, d_1, d_2, d_3 and d_4.

Unlike the test case presented in Section 5.1, which was prismatic, the APU blade's geometry is extremely twisted. Therefore, more than one section needs to be defined to parametrize the entire geometry.

The maximum deviation found in the blade matching process is of 0.18 mm, which is deemed to be satisfactory as it is several orders of magnitude below the length scale of the blade. The reader can refer to Appendix B for more details about the resulting parametrization.

5.2.3 Domain discretization

The domain is discretized using a structured mesh created in ANSYS TurboGrid. It is important to mention that the stator is as well simulated, as the flow at the rotor's inlet will not be uniform and can be a bottleneck to obtain a good convergence based on a user-specified inlet velocity profile. Figure 5.4 shows both grids used for stator and rotor.

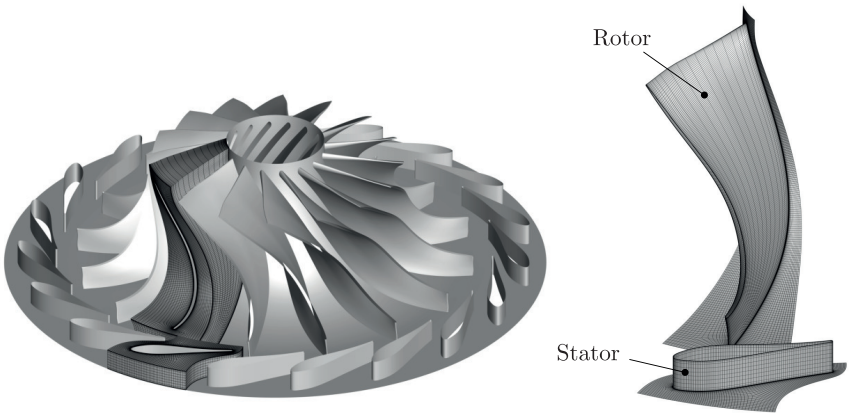


Figure 5.4: Grid used for both stator and rotor in the APU test case.

Table 5.2 gathers the number of elements for both stator and rotor. As it can be seen, the total passage contains almost 900,000 elements. Although this might result in longer runtimes for simulations (especially for the discrete adjoint solver), it is necessary for second order simulations, where a very low y^+ value is a compulsory condition in order to achieve a smooth residual convergence.

Table 5.2: Number of mesh elements for the APU turbine test case.

Zone	Number of elements [-]
Stator	230190
Rotor	633906

5.2.4 Flow and adjoint solver

The open-source flow solver embedded in *SU2* will be used. RANS equations govern the physical problem, where the one-equation Spalart-Allmaras turbulence model is used as closure. Convective fluxes are discretized using a second-order Roe scheme with MUSCL reconstruction, where a Venkatakrishnan slope limiter is used. An implicit Euler scheme is used for the time integration, where the CFL number

is set to 1.0.

Non-reflective boundary conditions are set at both inlet and outlet of the passage, which are summarized in Table 5.3. No-slip adiabatic wall conditions are defined for all the surfaces: blades, hub and shroud. Furthermore, the shroud is modelled as a fixed wall (i.e. the rotational speed is zero), whereas the rotational speed of the rotor is 71,700 rpm. Periodic boundaries are introduced in order to model the entire annulus. Due to the fact that a simulation for both stator and rotor is seeked, a mixing-plane interface with a linear interpolation algorithm is introduced between both zones. The discrete adjoint solver based in the RANS equations from *SU2* is used to compute the objective function sensitivities.

Table 5.3: Non-reflective boundary conditions for the APU turbine test case.

Inlet	Total pressure [Pa]	413600
	Total temperature [K]	477.6
Outlet	Static pressure [Pa]	67460

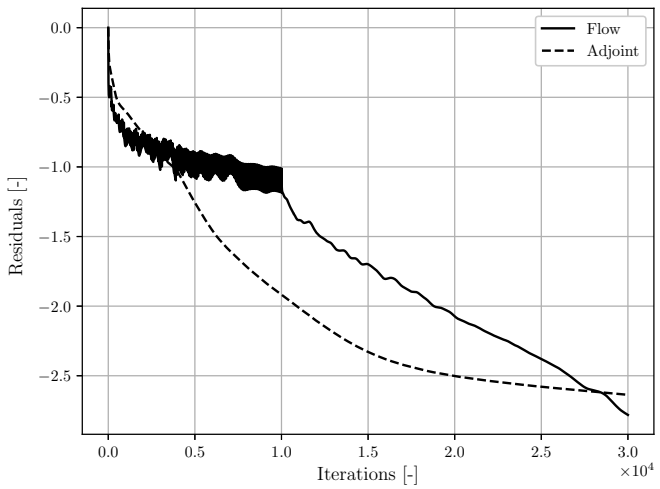


Figure 5.5: Residual evolution for both flow and adjoint problems for the APU turbine.

Figure 5.5 shows the convergence history for both the direct and adjoint problem. After 30,000 iterations, the flow residuals were reduced approximately three orders of magnitude, while almost three orders of magnitude reduction was achieved for the adjoint residuals.

5.2.5 Computational performance

Similarly than in the previous test case, the computational performance of the APU test case is evaluated. The entire mesh is composed by approximately 900,000 elements and a converged solution is obtained in 30,000 iterations. These solutions were obtained using 20 cores *Intel(R) Xeon(R) CPU E5-2670v2* CPU with 2.5 GHz clockspeed.

Figure 5.6 gathers all the values for the CPU-time and memory usage for this test case. On the right side, a pie chart with the percentage of CPU-time employed in each module is shown. The direct solution is obtained in 911 minutes, using approximately 25 GB of memory, while the discrete adjoint run took 1971 minutes and used almost 175 GB of memory. The gradient projection process, were the mesh sensitivities are calculated, was computed in 21 minutes using approximately 150 GB. These values suppose an adjoint-to-direct time ratio of 2.16 and memory-wise of 7. This test case, due to the increased

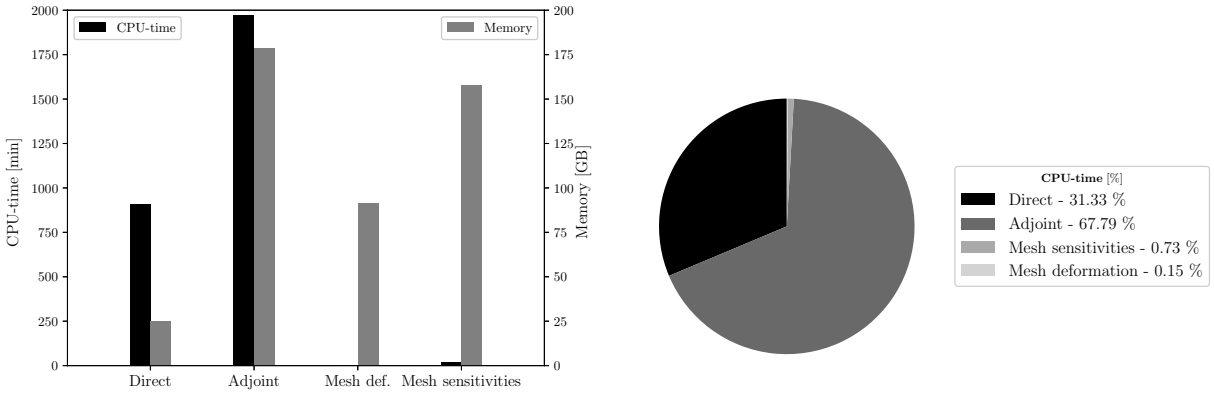


Figure 5.6: Computational performance, in CPU-time and memory usage, for the APU turbocharger test case.

number of elements when compared to the Aachen one, is more memory-exhausting and, again, it becomes evident that the use of high-performance computing is necessary.

Chapter 6

Results and discussion

In this chapter, the results for the optimization of the previously described test cases are shown. Section 6.1 covers the Aachen turbine stator case, while Section 6.2 analyzes the APU turbine rotor. This is followed by a discussion, emphasizing in how accurate the discrete adjoint gradients are and how the changes in fluid-dynamic performance of the new design are related to those of the blade surface.

6.1 Aachen turbine stator

The Aachen turbine stator optimization problem presented in Chapter 5, Section 5.1, is described in this section, where the results of the gradient validation, as well as the optimization progress is thoroughly analyzed.

6.1.1 Gradient validation

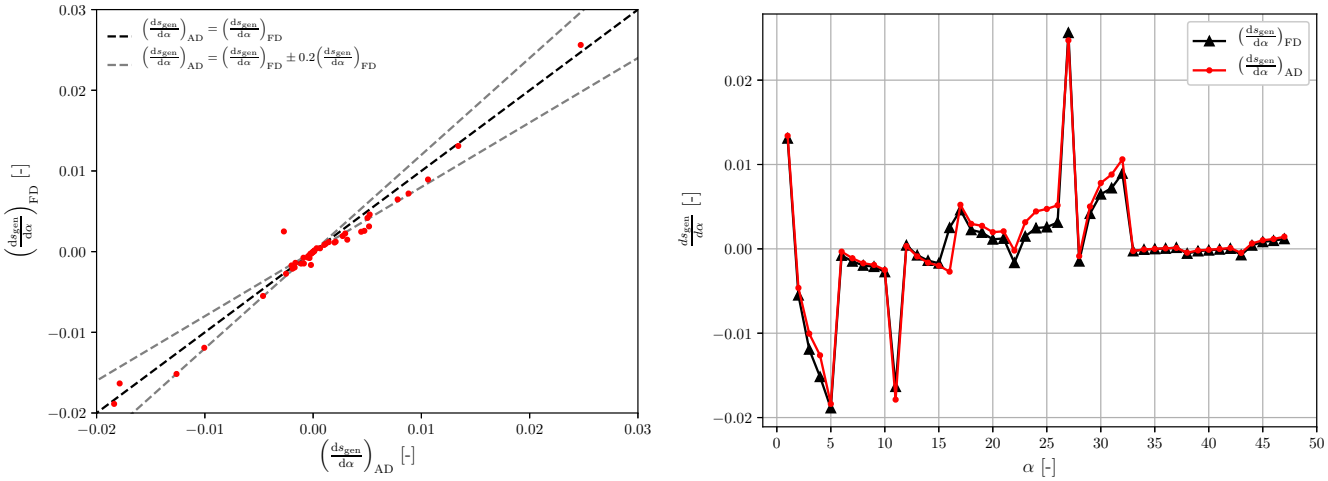
Prior to the shape optimization problem, the gradients provided by the discrete adjoint solver in *SU2* are validated. As it was already mentioned in Chapter 3, these sensitivities are compared against finite difference results. In this process, the geometry is slightly perturbed and the gradient is computed as the ratio between the change in the figure of merit and the perturbation of the design variable.

The gradients are computed using the non-dimensional entropy generation, s_{gen}, as objective function. The design variables that are going to be validated are the following: ξ, θ_{out}, θ_{in} ϵ_{in}, ϵ_{out}, r_{in}, r_{out}, d_1, d_2, d_3, d_4, all of which are explained in detail in Chapter 3. Five sections are created for all but θ_{out} and r_{out}, as they are out of the scope of the optimization as it will be argued in the following section, yielding a total of 47 design variables. The finite difference gradients are computed using a first-order scheme (forward-differences) with a step size of 1E-3 and 1E-1 for the variable d_2.

Figure 6.1 shows the results from the gradient validation study. It is important to note that the gradients for r_{out} have been scaled by a factor of 20 for visualization. All design variables show a good accordance between the gradients provided by finite differences and by the discrete adjoint solver, pointing for all design variables towards the same direction. This is of great importance for the latter shape optimization problem as the discrete adjoint gradient will be capable of pointing towards the direction of reduction of the objective function. A RMSE value of 2.89E-3 is found between the adjoint and finite difference gradients.

In addition, Table 6.1 shows the root-mean-square error per design variable. For most of the design variables, the RMSE between the adjoint sensitivities and finite differences is in the order of $\mathcal{O}(10^{-3})$ or $\mathcal{O}(10^{-4})$, which is deemed as satisfactory. However, for some others, the mismatch can be explained by some factors:

- Discretization errors: possibly caused by the fact that the solution is not fully grid-independent. A mismatch of 3% in the entropy generation of the passage was observed between the grid-independent



(a) Adjoint against finite differences.

(b) Gradient per design variable.

Figure 6.1: Gradient validation for the Aachen turbine stator test case.

Table 6.1: Root-mean-square error per design variable for the Aachen turbine stator test case validation.

Design variable	**Symbol**	**RMSE**
Stagger angle	ξ	1.66E-3
LE metal angle	θ_{in}	3.56E-4
TE metal angle	θ_{out}	1.53E-3
LE wedge semi-angle	ϵ_{in}	2.61E-3
TE wedge semi-angle	ϵ_{out}	8.60E-4
LE radius	r_{in}	2.10E-3
TE radius	r_{out}	1.80E-2
Distance 1	d_1	1.44E-3
Distance 2	d_2	1.92E-5
Distance 3	d_3	8.11E-5
Distance 4	d_4	2.54E-4

solution and the used mesh. Truncation errors may also play an important role in this mismatch. For more details about the grid convergence study, the reader is referred to Appendix A.

- Adjoint convergence: as it could be seen in the previous chapter, the adjoint problem was converged by approximately three orders of magnitude. Having a reduction of, approximately, four orders of magnitude would imply having a better-converged adjoint solution. Nevertheless this would come at a large computational cost.

- For the design variables that show greater mismatch, the values for both adjoint and finite differences sensitivities are extremely low, between $\mathcal{O}(10^{-3})$ and $\mathcal{O}(10^{-4})$. For such low values, it is expected a validation with a lower error to be achieved by investigating further finite difference steps, or even a second-order finite differences, which due to the time limitations of this project could not be achieved. Nevertheless, these gradients are also deemed as valid as they point towards the same direction.

It is also important to note that the region with higher spurious sensitivities with respect to a change in the surface coordinates position is the trailing edge, as displayed in Figure 6.2. Therefore, for those design variables which have a higher impact on the trailing edge coordinates, it is difficult to capture

correctly the gradients by the discrete adjoint solver, thus showing a greater mismatch between AD and FD sensitivities (e.g. trailing edge radius, trailing edge metal angle).

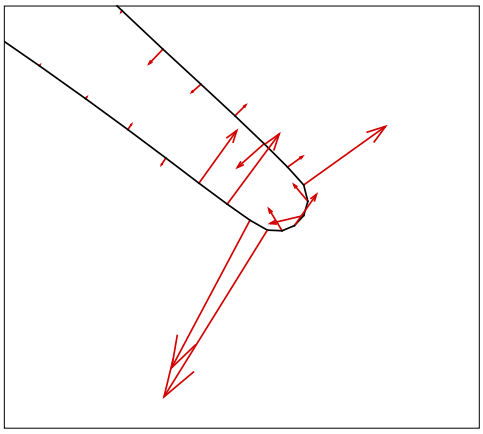


Figure 6.2: Adjoint sensitivity $\frac{\mathrm{d}s_{\mathrm{gen}}}{\mathrm{d}\alpha}$ vectors at the mid-span section's trailing edge for the Aachen turbine stator.

Even though the gradient validation is deemed as satisfactory, with a RMSE of 2.89E-3, it is important to note that due to geometrical and flow constraints, which will be explained in detail in the following section, the design variables r_{out} and θ_{out} are left out of the optimization loop. This, in turn, causes the RSME of the gradient validation to reduce to 1.29E-3.

Finally, it is important to highlight the computational time saving by using the discrete adjoint as compared to finite differences when computing the design variable gradients. Recalling Chapter 5, Section 5.1, one direct solution is obtained in 609 minutes CPU-time, while the adjoint solution is computed in 949 minutes CPU-time. The cost for obtaining the gradients through the discrete adjoint solver is a total of 1,558 minutes (26 hours), accounting for both the flow and adjoint runs. In contrast, in order to obtain the gradients through finite differences, as 48 direct solutions are required, this sums up to a total of 29,232 minutes (487 hours), which are approximately 20 days. From here it is evident the benefit of using the adjoint solver to compute the objective function gradient.

6.1.2 Optimization setup

The optimization problem that will be solved can be posed as

$$\min_{\alpha} \quad s_{\mathrm{gen}}, \tag{6.1a}$$

$$s.t. \quad \beta_{\mathrm{out}} \geq 69°. \tag{6.1b}$$

Both entropy generation, s_{gen} and flow outlet angle, β_{out}, are computed using the mixedout average [55], being

$$s_{\mathrm{gen}} = \frac{\langle s_{\mathrm{out}} \rangle - \langle s_{\mathrm{in}} \rangle}{v_0^2 / T_{\mathrm{t,in}}}, \tag{6.2}$$

as defined in Chapter 3. The flow outlet angle is computed as

$$\beta_{\mathrm{out}} = \mathrm{arc\ tan}\left(\frac{v_{\mathrm{tan}}}{v_{\mathrm{ax}}}\right)_{\mathrm{out}}, \tag{6.3}$$

where v corresponds to the absolute velocity, and the subscripts "tan" and "ax" correspond to the tangential and axial components, respectively. The objective function and constrain value are computed using *SU2*'s RANS solver with the Spalart-Allmaras turbulence model.

The design vector will be

$$\alpha = \{\xi, \theta_{\text{in}}, \epsilon_{\text{in}}, \epsilon_{\text{out}}, r_{\text{in}}, d_1, d_2, d_3, d_4\}, \tag{6.4}$$

based on the parametrization described in Chapter 5, Section 5.1. In an effort to maintain certain flow and geometrical constrains, the variables r_{out} and θ_{out} are left out of the optimization for the following reasons:

- Trailing edge's radius, r_{out}: it should be a natural tendency of the optimized blade to reduce this value as it will lead to a thinner trailing edge and, therefore, reduce the wake thickness and thus the entropy generation related to the trailing edge mixing losses. However, due to manufacturing constrains, extremely thin trailing edges are not achiveable. Furthermore, they could potentially be a region of very high-concentrated stresses. Additionally, turbine vanes and blades commonly include cooling passages which need to be accomodated through the blade's cross-section, which would be very challenging for very thin trailing edges.
- Blade metal angle out, θ_{out}: it is a common practice in constrained optimizations to maintain a minimum threshold for the outlet flow angle as it affects the performance of the downstream components. If the outflow angle is reduced the stage expansion decreases, affecting the overall turbine performance. As the outlet metal angle is deemed to be one of the main contributors to the outlet flow angle, it is left out of the optimization.

A total of 45 design variables will be used, corresponding to five sections created for each of the parameters in the design vector α. The optimization problem will be ran without upper and lower bounds: instead, the step size will be indirectly controlled by the use of the so-called relaxation factor f which was introduced in Chapter 4. The constrained optimization is set for an accuracy of 1E-14 and for a maximum number of iterations of 20.

6.1.3 Optimization progress

The optimization history for both the objective function and the inequality constraint imposed is displayed in Figure 6.3, along with the final blade shape. It can be clearly seen that the optimization is mainly divided into two parts, separated by a big jump in both the s_{gen} and β_{out} value. These correspond to a re-meshing step, together with a contentious relaxation factor, that was introduced in the eighth design step.

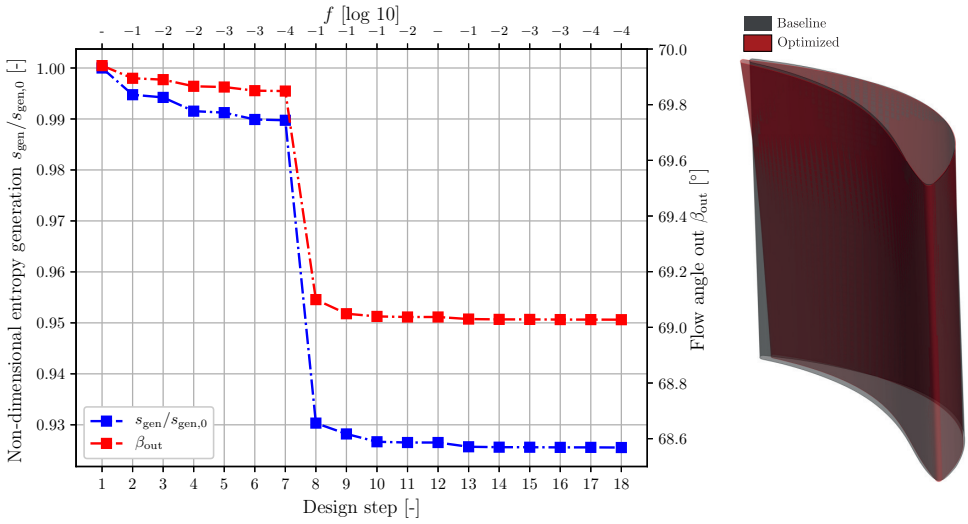
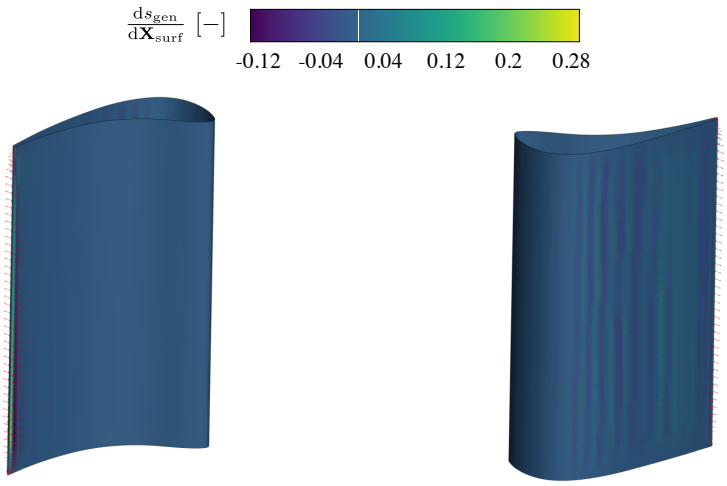


Figure 6.3: Optimization history and final optimized blade shape for the Aachen turbine stator test case.

The strategy for the optimization stage was the following: for the first design steps, the optimizer was left free to morph the mesh accordingly with the optimal search direction. However, as mesh deformation was a known issue and can be a bottleneck if the change in surface grid coordinates is very large, it was expected beforehand that re-meshing at some point of the optimization was going to be required. Therefore, the structure of this section is divided into two parts: the first one corresponding to the first mesh deformation stage and the second to the runs where remeshing was introduced. The issues of using both techniques are going to be detailed and analyzed.

First mesh deformation stage

For the first design step, corresponding to the baseline design, the flow and adjoint solutions are already available from the gradient validation. Before computing the dot product between the adjoint and the CAD sensitivities, as explained in Chapter 3, by plotting the adjoint sensitivity map one can get a better insight on how the surface coordinates, $\mathbf{X}_{\text{surf}}$, will morph in order to reduce the entropy generation of the passage.



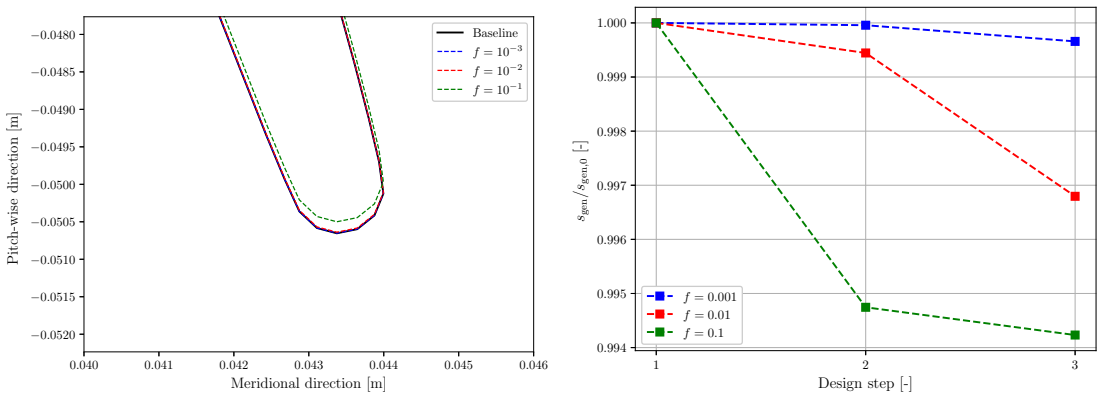
(a) Pressure side. **(b)** Suction side.

Figure 6.4: Adjoint sensitivity $\frac{ds_{gen}}{d\mathbf{X}_{surf}}$ map for the Aachen turbine stator after the first design step. Red arrows display the surface movement direction.

Figure 6.4 shows the projected adjoint sensitivities on the Aachen turbine stator surface. The largest sensitivities are located next to the trailing edge, therefore it is expected that small changes in that region can possibly cause larger reductions in the objective function value if compared to other locations of the blade. As the trailing edge radius is left out of the optimization, presumably this changes will arise from variations in the stagger angle.

As it was mentioned in Chapters 3 and 4, the user has control on the optimization step length indirectly through the relaxation factor, f. Therefore, it is important to select the right value for this parameter as too aggressive ones can cause issues with mesh deformation and too conservative values can ease the deformation process but at a cost which is a slower rate of reduction in the objective function.

Figure 6.5 shows the influence of the relaxation factor on the trailing edge for a cross-section at the hub. Higher relaxation factors imply a larger deformation as the gradient has more weight on the selection of the step. This can clearly be noticed at the trailing edge: while the deformation cases with $f = 0.01$ and $f = 0.001$ are almost identical to the baseline shape, the shape movement for $f = 0.1$ is much noticeable and contentious. This, in turn, causes a steeper reduction in the objective function's value right after the first design step, as it can be seen in Figure 6.5b. It is noticeable that, even after two



(a) Hub section – TE detail.

(b) Objective function.

Figure 6.5: Relaxation factor f influence on the blade shape **(a)** and the objective function value **(b)** for the Aachen turbine stator test case.

design steps, the entropy generation for the $f = 0.01$ and $f = 0.001$ cases has not even reduced further than for one design step with $f = 0.1$.

Table 6.2: Relaxation factor f per design step for the first deformation stage in the Aachen turbine stator test case.

Design step	f	**Mesh quality**
01	-	Good
02	0.1	Good
03	0.01	Good
04	0.01	Good
05	0.001	Good
06	0.001	Good
07	0.0001	Good
08	0.0001	Negative volumes

In conclusion, the relaxation factor has to be carefully selected in order to have a robust, but at the same time computationally efficient, optimization. Table 6.2 summarizes the relaxation factors employed for this pure mesh deformation stage. As it can be seen, initially a relaxation factor of 0.1 was selected as a rapid jump to the optimal region was seeked. It is important to note that all the changes in f were motivated by mesh deformation issues.

Figure 6.6 shows a common issue found when performing mesh deformation with large relaxation factors. This part of the mesh corresponds to the trailing edge at the shroud. As it could also be seen in Figure 6.5, the region which has the largest variation in surface coordinates is the trailing edge. Because of the implementation explained in Chapter 3, the movement of the mesh at the hub and shroud regions is restricted in the normal surface direction. Therefore, all the deformations occurring in these surfaces need to be accommodated in the two tangential directions. The consequence is that, if the mesh is very fine, the linear system of equations to be solved can be very stiff and negative volumes after the mesh deformation can be found. These issues were commonly encountered in regions with boundary layer refinement, such as the blade, hub and shroud walls. After design step number seven, the mesh was deemed to be in a bad state especially at the hub and shroud surfaces, therefore a re-mesh for iteration number eight was introduced, which will be explained in the next section.

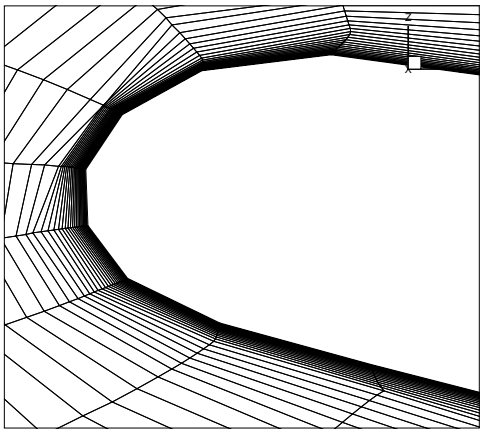


Figure 6.6: Mesh defects at the trailing edge of the shroud surface after the seventh design step for the Aachen turbine stator test case.

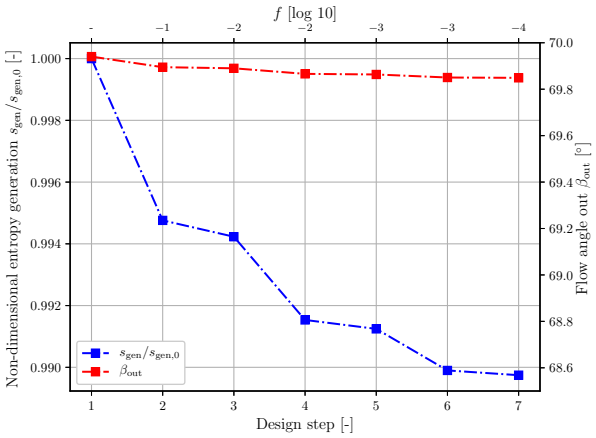


Figure 6.7: Optimization history for the first optimization stage in the Aachen turbine test case.

The optimization history for the first seven design iterations is shown in Figure 6.7, which is extracted from Figure 6.3. The relatively-high relaxation factor for the first design step causes a steep reduction of the objective function's value, entropy generation. After this design step, applying a second iteration with the same relaxation would have caused similar mesh issues as the one seen in Figure 6.6. Therefore, a lower relaxation factor of $f = 0.01$ was applied instead. As it was explained in the previous paragraphs, lowering f caused a less-aggressive (mesh deformation-wise) design step, and, expectedly, less reduction in the objective function's value. However, once the optimizer has read the information from several design steps, the user has less control of the step size. This can be seen from iteration 3 to 4, where a lower value of the relaxation factor is chosen but the jump in the entropy generation decrease is higher than from 2 to 3, where f was an order of magnitude higher. From this same figure it can be seen that a reduction of, approximately, 1.1% in the passage's entropy generation is achieved after the seventh design step. The flow angle constraint is satisfied for all the iterations as $\beta_{out} > 69°$.

It is interesting to highlight that, in this first optimization stage, most of the blade shape changes were observed in the stagger angle. Figure 6.8 shows the blade shape change for the last design step, namely 7, with respect to the baseline design. The figure depicts some valuable facts:

- The blade turns from fully-prismatic to a twisted shape: the stagger angle tends to reduce near the hub, while it increases progressively in sections close to the shroud.
- The blade is slightly thickening towards the shroud and becoming thinner at the hub.

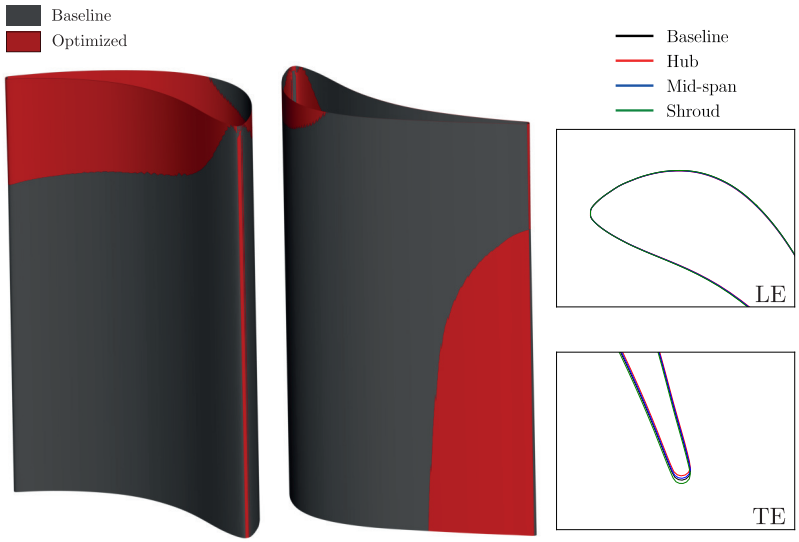


Figure 6.8: Geometrical comparison for the first optimization stage between the optimal (red) and baseline (dark gray) designs for the Aachen turbine stator.

- The mid-span cross-section shows small deviations in shape when compared to the baseline design.
- The trailing edge radius is successfully maintained through the optimization.

However, the negative volumes shown earlier in this section caused the original mesh to be considered as inadequate for continuing with the optimization process: therefore, the geometry after design step number seven was re-meshed in order to carry on with the optimization. This will be explained in the following section.

Second mesh deformation stage

The mesh issues observed in the previous optimization stage caused the original mesh to be in a bad state towards the last design steps. As it is expected that the optimizer might take more aggressive steps as it moves towards the minimum, it was deemed appropriate to re-mesh the geometry after the last available design step, and continue from there on with the mesh deformation. However, as it was argued in Chapter 2, a regeneration of the mesh can introduce a loss of information in the optimization, especially if the solution is not completely mesh-independent. Nevertheless, it will be assumed that these mismatches will be present in the results obtained and, in order to evaluate the impact of the re-mesh, once the optimized geometry is obtained the baseline design will be evaluated in a mesh with a similar element density.

Similarly than in the previous section, Figure 6.9 shows the optimization history for the second stage, corresponding to the design steps after re-meshing the geometry, which is extracted from Figure 6.3. Again, the relaxation factor f had to be carefully chosen in order to avoid issues with mesh deformation, especially towards the last design steps of the optimization shown. The optimization history shows a steep reduction of the objective function value at the first steps after the re-meshing, shifting to a plateau from design step number 13 onwards.

As a matter of fact, in an initial run, shown in Figure 6.10 as the black curve, the SLSQP algorithm stopped using the information stemming from the discrete adjoint and the CAD sensitivities and started taking linear steps (i.e. same optimization step for the subsequent iterations) from design step number 11 onwards. In the figure, this can be seen as a plateau where the objective function is barely improved. In order to restore the gradient information handled by the SLSQP optimizer, at design step number 12 the optimization was restarted using the previous step, number 11, as the initial design. Therefore, the algorithm will not have any previous information from the gradient and, potentially, can take a different step than before. In fact, the blue curve, corresponding to these new set of restarted design steps, shows

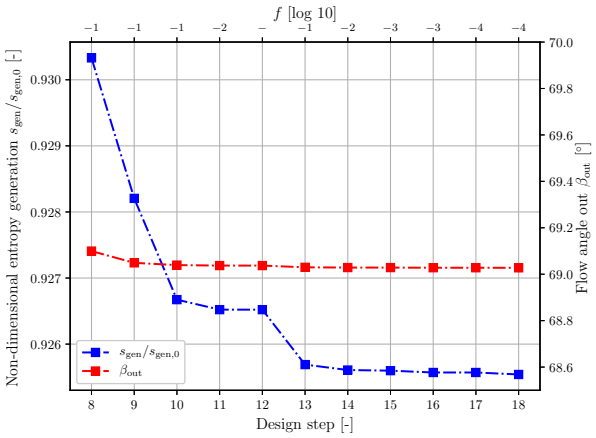


Figure 6.9: Optimization history for the second optimization stage in the Aachen turbine test case.

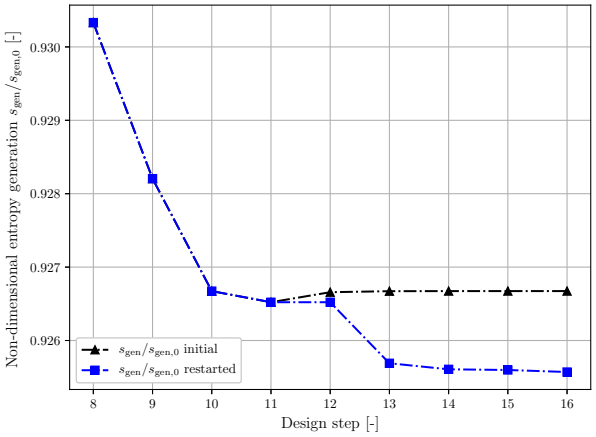


Figure 6.10: Optimization history comparison: initial (black) and restarted from design step number 12 onwards (blue) for the second optimization stage in the Aachen turbine test case.

a better improvement of the objective function reduction than in the previous case.

For this last optimization stage, the non-dimensional entropy generation across the blade passage showed a reduction of 0.51% from the eigth to the eighteenth, while the flow angle constraint is satisfied for all the design shapes explored. Finally, the optimization is stopped after design step number 18, as the objective function value showed no further reduction. The final blade shape, as well as the fluid-dynamic comparison between the baseline and the optimized geometries, are analyzed in the following section.

6.1.4 Optimized geometry

The optimized Aachen turbine stator geometry consists of the best achieved reduction in entropy generation, s_{gen}, while respecting the flow angle constraint, $\beta_{\text{out}} \geq 69°$, as explained in Section 6.1.2. In this section, the blade shape changes will be compared, from a fluid-dynamic performance point of view, with the baseline initial geometry.

Nevertheless, as it was discussed in the previous section, the re-meshing process can introduce some uncertainties in the result as the solution is not fully grid independent. Therefore, in order to assess the optimized geometry against the baseline design, a new CFD run is prepared where, for the reference design, the grid has a similar density than the re-meshed one. The grid statistics, as well as some key

performance parameters, are shown in Table 6.3.

Table 6.3: Final optimization results for the Aachen turbine stator test case, compared to the baseline design.

Geometry	**Elements** [−]	s_{gen} [−]	Δs_{gen} [%]	β_{out} [°]	ζ_K [−]	$\Delta\zeta_K$ [%]	η_{is} [%]	$\Delta\eta_{\text{is}}$ [%]
Baseline	505,164	0.02344	-	69.94	0.044	-	96.56	-
Optimized	506,766	0.02166	-7.59	69.03	0.039	-9.26	96.83	+0.28

After performing a constrained optimization where the non-dimensional entropy generation across the blade passage was aimed to be minimized, the final optimized blade shows a reduction of 7.59% in s_{gen}. The outlet flow angle constraint was satisfied as the optimized design features a value of $\beta_{\text{out}} = 69.03°$, which supposes a reduction of 1.30% of its initial value. It is important to point out that even though the outlet blade metal angle is not included explicitly as a design variable, the optimizer still tends to decrease the value of β_{out}, slightly shifting towards less-curved airfoil geometries. This confirms the right selection of design variables as including θ_{out} could have hampered the optimization chain by moving towards designs with lower outlet flow angle. Furthermore, the total pressure loss coefficient, ζ_K, decreases 9.26%, and the isentropic efficiency of the stator is increased by 0.28%.

In the following sections, a link between the blade shape changes and its fluid-dynamic performance is constructed.

Blade shape assessment

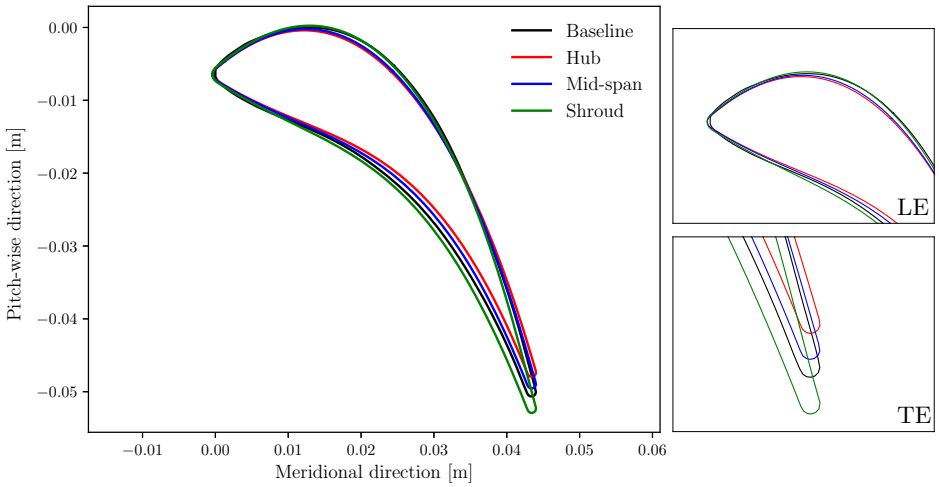


Figure 6.11: Detailed comparison between the optimized and baseline (prismatic) Aachen turbine stator shape for three different sections.

Figure 6.11 shows the changes in the blade shape geometry for three sections: one located near the hub, another at approximately mid-span and the final one near the shroud. A detailed view of both leading and trailing edge is displayed in the same figure. As it can be seen, most of the geometrical changes are related to the stagger angle ξ and opposite trends are featured between the hub and shroud sections: while this value decreases near the hub, causing the trailing edge to shift upwards, it increases towards the shroud, having the opposite effect. Therefore, as it was also seen in the previous section, the blade changes from a prismatic design (i.e. equal sections and blade angles along the blade span) to a

twisted shape.

Another important change in the blade shape can be related to the airfoil's thickness distribution. This can be observed in the same figure by comparing the thickness changes along the chordline: while the blade tends to narrow down towards the hub section (red), by a movement upwards of the pressure side, the opposite effect can be seen progressively the section near the shroud (green).

The geometrical constrain imposed by leaving out of the optimization loop the design variable r_{out} is successfully achieved as it can be seen in the trailing edge detail of Figure 6.11. This demonstrates the convenience of having CAD-based parametrization techniques: while other methods imply creating complex geometrical relationships to maintain certain geometrical characteristics, the traditional approach allows for a more intuitive and efficient imposition of the constrains, especially in 3D cases.

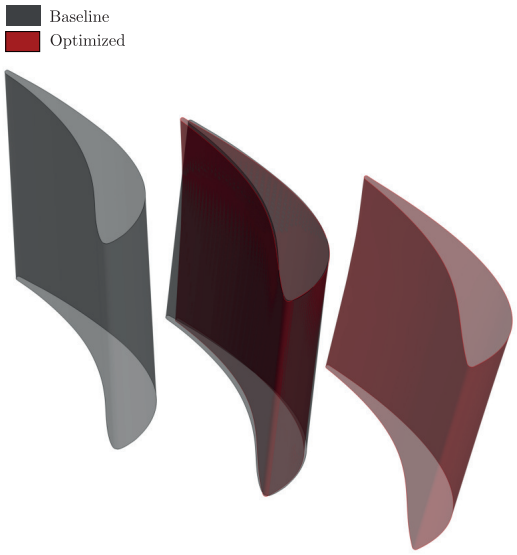


Figure 6.12: Geometrical comparison between the optimal blade shape (red) and the baseline geometry (dark gray) for the Aachen stator.

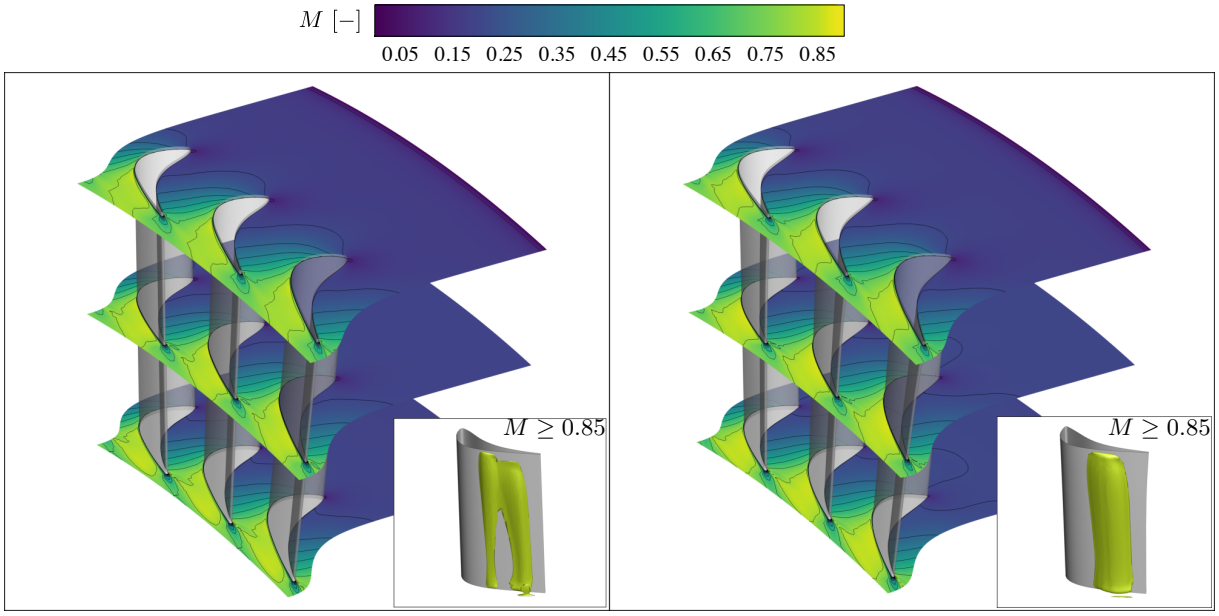
Similarly, Figure 6.12 shows a 3D view of the optimized and baseline geometries, where it can be better appreciated the change in stagger angle across the blade span. It may also be noticed that the optimized geometries feature a smaller inlet radius, causing a more pronounced curvature. Ultimately, these geometrical changes will have an impact on the fluid-dynamic performance of the flow across the passage, which is analyzed in detail in the following section.

Fluid-dynamic assessment

The objective of this section is to link those blade shape changes shown in earlier paragraphs to the fluid dynamic performance of the passage, comparing both baseline and optimized designs. Summarizing, the main changes in the optimal blade geometry, when compared to the baseline design, are:

- Stagger angle increases from hub to shroud, causing a twist of the initial prismatic blade.
- The blade thickness increases from the hub to the shroud, producing thicker blade sections in the span-wise direction.
- More pronounced leading edge curvature in all the blade sections.

Figure 6.13 shows Mach contours for three different blade cross-sections, corresponding to hub, mid-span and shroud for the baseline and the optimized design. Before entering into comparisons between the two designs, it is interesting to analyze the baseline flow field. First of all, a stagnation point is



(a) Baseline. **(b)** Optimized.

Figure 6.13: Comparison between the Mach number for different cross-sections for the baseline **(a)** and optimized **(b)** designs.

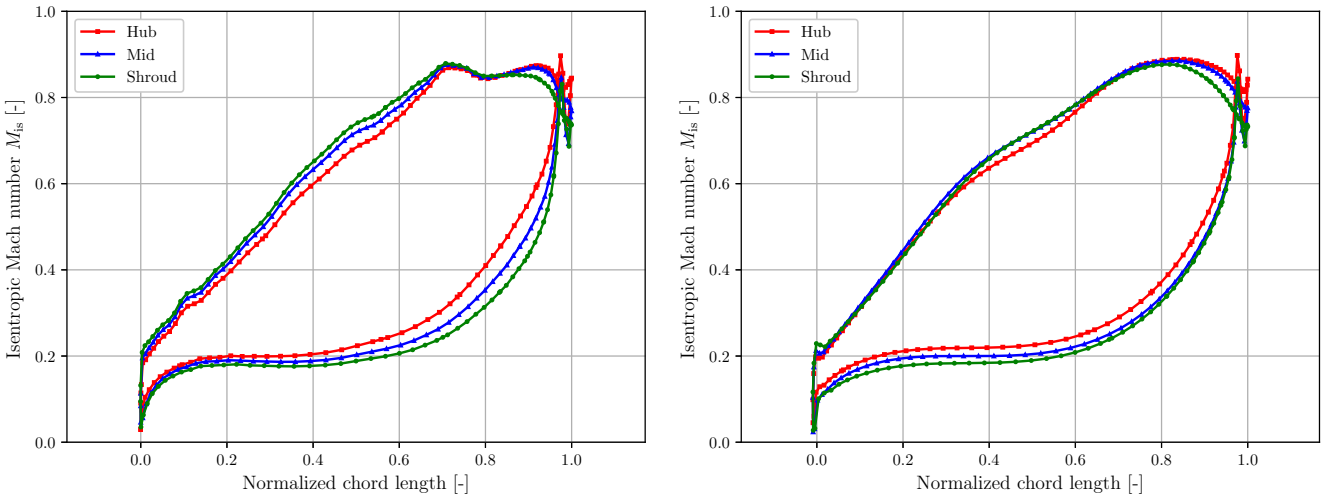
located at the leading edge as seen in all three slices. The fluid gets accelerated as it advances through the geometrical throat, and no shockwaves are found as this is a transonic test case. Furthermore, it is evident that by performing a three-dimensional simulation there is a span-wise variation of the fluid properties across the passage. In this case, the flow acceleration taking place in the geometrical throat is more prominent in the vicinity of the shroud, as shown in the bottom right iso-surface plot for $M \geq 0.85$. The slower flow at the bottom causes a slightly increased wake region at the hub.

Now, if both designs are compared these plots show two interesting flow features caused by the changes in the blade geometry explained in the previous section. Firstly, at the region near the leading edge, it can be seen that for the optimized designs the stagnation zone is decreased due to the reduced inlet radius. Secondly, due to the combined effect of the stagger and thickness variation from the hub to the shroud, the flow accelerates more in the cascade geometrical throat. Furthermore, at the bottom right of each figure an iso-surface for passage Mach numbers greater than 0.85 is displayed, which provides valuable information about what the optimized geometry features with respect to the baseline design: not only the flow is increasing its acceleration at the hub, but also this increase in velocity is more uniform across the whole blade span.

This phenomenon becomes evident in Figure 6.14, where the isentropic Mach number against the normalized chord length is plotted for the same sections as displayed before. It is important to notice that a greater flow acceleration between the optimized and baseline design is achieved at the hub and progressively this value decreases towards the shroud, again confirming the tendency of the optimizer of accelerating the flow at the hub. Furthermore, the comparison between Figures 6.14a and 6.14b also shows some interesting facts:

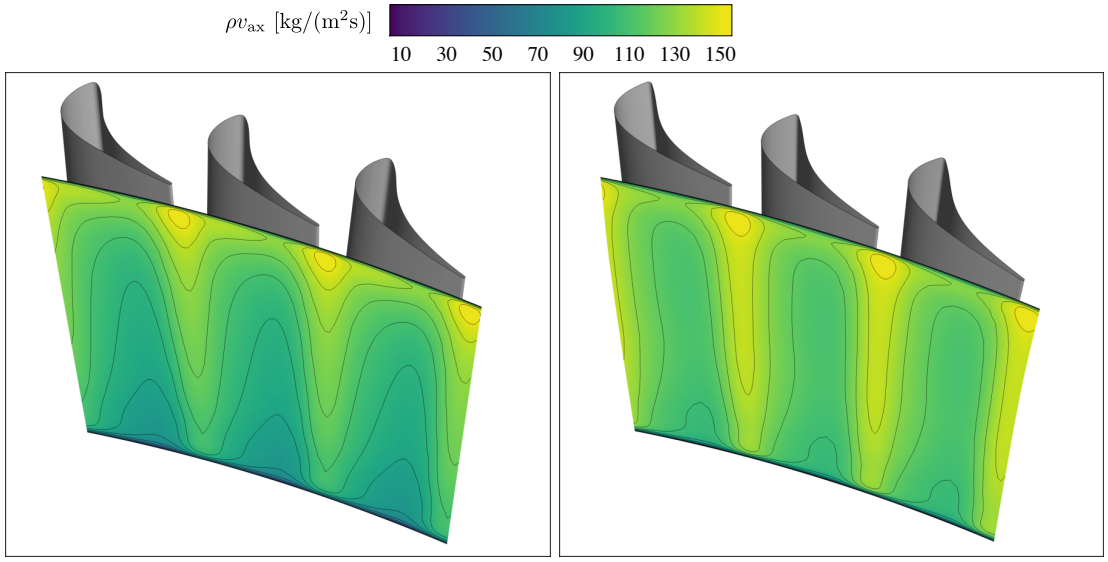
- The M_{is} profiles for the sections from the optimized blade tend to collapse (i.e. become uniform with the span-wise coordinate), especially at the suction side (upper curve).
- Flow gets more accelerated in the vicinity of the trailing edge (i.e. from 0.7 to 1.0 normalized chord length) which is in agreement with what was seen in Figure 6.13.
- The flow acceleration near the geometrical throat becomes more uniform and smooth in the opti-

mized design.



(a) Baseline. **(b)** Optimized.

Figure 6.14: Comparison between the isentopic Mach number across three different blade span-wise sections for the Aachen turbine stator.



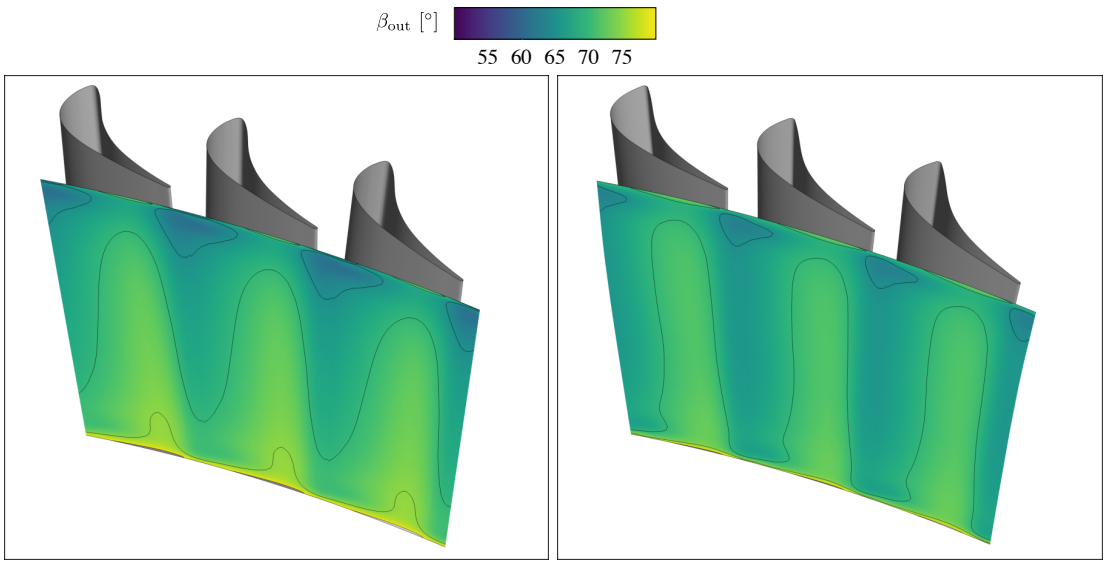
(a) Baseline. **(b)** Optimized.

Figure 6.15: Comparison between the outflow meridional momentum for the baseline **(a)** and optimized **(b)** designs in the Aachen turbine stator test case.

In order to understand why the flow is being accelerated more at the hub than at the shroud, when compared to the baseline design, Figure 6.15 can be analysed. In this case, the momentum in the meridional/axial direction is contoured at the outflow surface. This figure shows how for the baseline design initially there is a low-momentum (dark purple) region at the hub, which progressively gets accelerated when moving upwards in the span-wise direction, showing high-momentum (yellow) flow at the shroud region. This difference causes mixing losses due to the co-existence of flows with different momentum, a loss mechanism widely known for high-aspect ratio blades. However, for the optimized geometry this

gradient is less prominent, because the hub gets more accelerated as it was argued in the previous paragraph. Furthermore, the flow is more uniform across the whole span which will in turn reduce the mixing losses. This will also be beneficial for the downstream component, the turbine's rotor.

The aforementioned flow uniformity can also be seen in Figure 6.16, where the outlet flow angle is plotted for the same outflow slice as in the previous analysis. Initially, the flow angle has a span-wise profile which decreases from hub to shroud. In the optimized geometry, as there is a distribution of the stagger angle for the different blade sections defined, together with an increase of the thickness from the hub to the shroud, the flow is more restricted and is able to follow the blade's geometry in an improved manner.

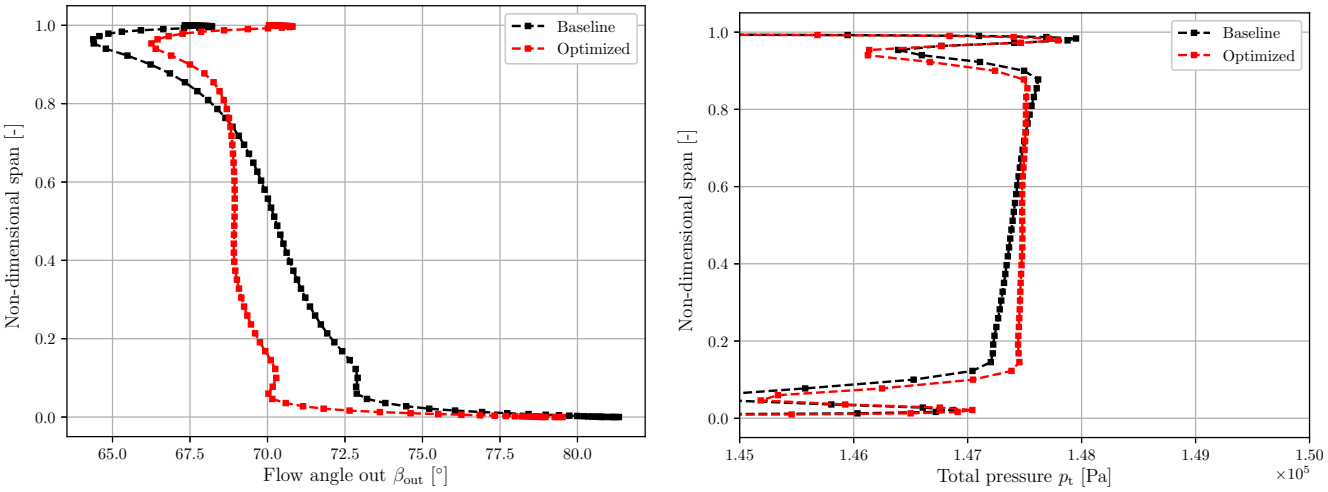


(a) Baseline. **(b)** Optimized.

Figure 6.16: Comparison between the outlet flow angle β_{out} for the baseline **(a)** and optimized **(b)** designs in the Aachen turbine stator test case.

Figure 6.17 shows the averaged span-wise distributions of two key values which give a final insight on how the geometry is changing in order to reduce the entropy generation across the blade passage while satisfying the imposed constrain $\beta_{out} \geq 69°$. Figure 6.17a displays a comparison between change in outlet flow angle in the span-wise direction for both baseline and optimized geometries. As it can be seen, initially this constrain is not satisfied in the vicinity of the shroud, as relatively low flow angles can be observed. On the other hand, the opposite phenomenon takes place in the proximity of the hub where relatively large outlet flow angles are encountered.

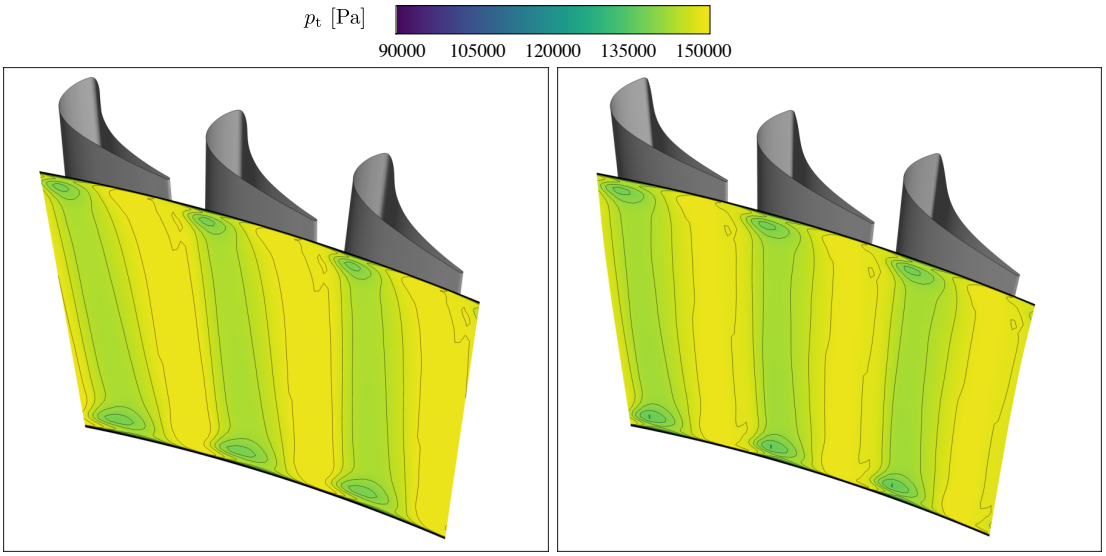
However, from the optimizer's point of view, two different criteria have to be met in order to satisfy both objective function and constrain: the entropy generation across the passage has to be reduced and the outlet flow angle needs to be maintained above a given threshold, which supposes a trade-off and produces different changes of geometry across the blade span. Firstly, it is important to note that the constrain's role is to maintain a certain flow turning, as it would be intuitive that an unconstrained optimization would lead to flattened geometries where the flow turning is reduced (i.e. decreasing the outlet flow angle) as this is a phenomenon that intrinsically generates entropy. Therefore, as the constrain will limit how much the flow angle can be reduced, flow uniformity will be the main mechanism for the entropy reduction across the passage. As it was mentioned in earlier paragraphs, this uniformity is beneficial because it reduces mixing of low- and high-momentum fluid losses. This trend can be clearly seen in regions in the vicinity of the hub: even though initially the constrain was greatly satisfied, as large outlet flow angles are encountered in this zone, the geometry is changing so that β_{out} is uniformed



(a) Outlet flow angle. **(b)** Total pressure.

Figure 6.17: Span-wise distribution comparison for outlet flow angle **(a)** and outlet total pressure **(b)** for the baseline and optimized Aachen turbine stator.

with respect to the rest of the span-wise flow angles. Simultaneously at the shroud, as the constrain is not satisfied, the opposite effect takes place: the outlet flow angle increases for the optimized geometry, and this is done by means of increasing the blade thickness, which will ultimately affect the geometrical throat length. Furthermore, as greater differences between the baseline and optimized values for β_{out} are observed in the proximity of the hub, it can be concluded that this uniformity is set by the values encountered at the shroud. For example, if the constrain would have instead been $\beta_{out} \geq 65°$, the uniform profile would shift towards the left showing again greater differences at the hub.



(a) Baseline. **(b)** Optimized.

Figure 6.18: Comparison between the outlet total pressure p_t for the baseline **(a)** and optimized **(b)** designs in the Aachen turbine stator test case.

Figure 6.17b compares the total pressure profile for both baseline and optimized design. Besides

regions near the hub and shroud where end-wall vortices are observed, not only the total pressure distribution is almost uniform for the rest of the blade span, but also for most of the blade the total pressure loss is reduced, which ultimately leads to a reduced passage entropy generation.

Finally, Figure 6.18 shows the outlet total pressure contour comparison. It is interesting to point out that not only the total pressure profile is more uniform in the optimized geometry, but also that there is a change in the intensity of both hub and shroud endwall vortices intensity. While the shroud wall vortex gets slightly bigger, which is accompanied by a larger total pressure loss as shown in Figure 6.17b; the hub wall vortex gets slightly smaller, showing an improved recovery of the total pressure loss in the vicinity of this endwall.

The reader can refer to Appendix A for further plots of other fluid-dynamic properties.

6.1.5 Conclusions

In conclusion, the outcome of this turbine stator aerodynamic shape optimization can be summarized as follows:

- The gradients predicted by the adjoint solver accurately match those from finite differences.
- The non-dimensional entropy generation across the blade passage is mainly reduced in the optimized geometry due to changes in the blade thickness and stagger angle which in consequence cause a uniform flow acceleration across the entire blade span.
- The aforementioned passage acceleration causes in turn lower mixing losses, which are translated in a uniform outlet flow angle and total pressure profiles.
- The outlet flow angle constraint prevents the optimized geometry from shifting towards a flatter airfoil. The value of this constraint determines at which extent the flow is getting uniformly accelerated through the blade's geometrical throat.
- The constant trailing-edge radius geometrical constrain is effectively maintained through the whole optimization.

6.2 APU turbine rotor

Following the test case introduced in Chapter 5, Section 5.2, the APU turbine rotor will be as well subject to an aerodynamic shape optimization to prove that the methodology developed in this Thesis can be applied also to radial turbomachinery.

6.2.1 Gradient validation

Prior to the optimization, the gradients provided by the dot product between the adjoint and CAD sensitivities have to be compared against finite difference results and validated.

The gradients are computed using the non-dimensional entropy generation, s_{gen}, as objective function. The design variables that are going to be validated are: ξ, θ_{in}, θ_{out} ϵ_{in}, ϵ_{out}, r_{in}, d_1, d_2, d_3, d_4, all of which are explained in detail in Chapter 3. Four sections will be created for each design variable, therefore the total number of parameters validated is 44. As opposed to the previous test case, where several sections were created as well in order to allow for a full three-dimensional optimization, in this case it is compulsory as the geometry is highly twisted. The adjoint solution was obtained in a total of 2,882 minutes (two days), corresponding to one direct and adjoint run. The finite difference gradients, instead, took a total of 40,084 minutes (28 days). From these numbers, it becomes evident the benefits of using the adjoint solver to compute the objective function's gradient instead of the traditional finite differences method. The finite differences results are computed using a forward-scheme with a step equal to 0.1%.

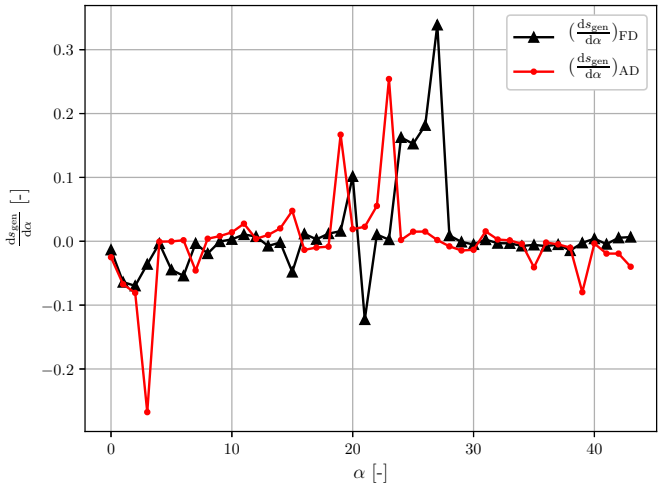
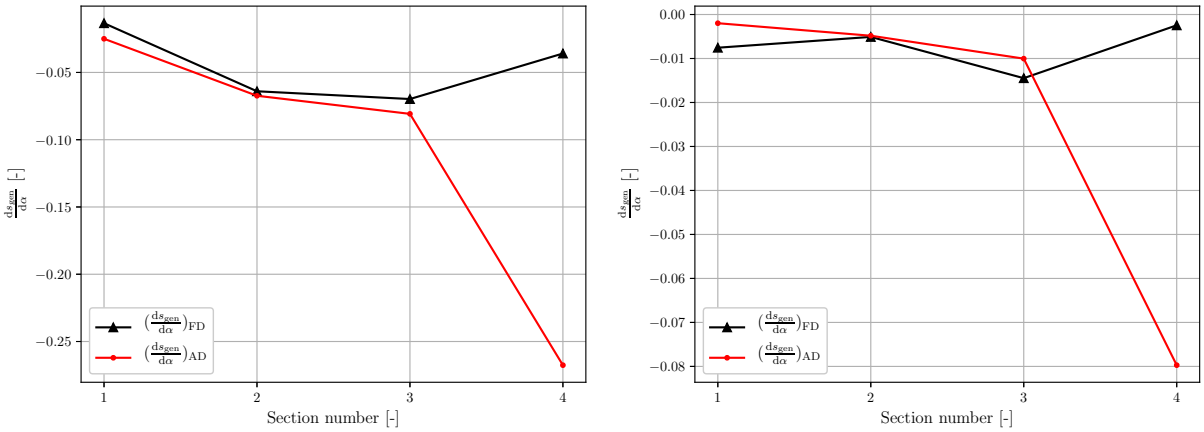


Figure 6.19: Gradient validation for the APU turbine rotor.

Figure 6.19 shows the gradient validation for the APU turbine rotor. The calculated RMSE between the AD and FD gradients is equal to 9.37E-2. As it can be seen, both the plot and the value of the RMSE reveal a poor validation of the gradients, which requires a further investigation before performing shape optimization.



(a) Stagger angle.

(b) Thickness distribution.

Figure 6.20: Gradient comparison per design variable section for the APU turbine rotor.

In order to get a better insight on why some of the gradients are not getting properly validated, Figure 6.20 shows the gradient validation plot for two different design variables: the stagger angle ξ and the thickness distribution d_3, showing the gradient values per each section, where section 1 corresponds to the hub and section 4 to the shroud. It is noticed that the trend of the gradient is correctly captured for the first three sections; however, the shroud section shows a large disagreement between both values. The reader can refer to Appendix B for other plots for the rest of the design variables.

Figure 6.21 displays the adjoint sensitivity map for the APU turbine rotor test case. First of all, it is noticeable the relatively high values of the sensitivities found in the blade surface: ranging from -9 to 5 approximately, values much larger than those seen in the previous test case. Very high sensitivities, with respect to the rest of the blade surface, are observed precisely at the shroud, shown by the red

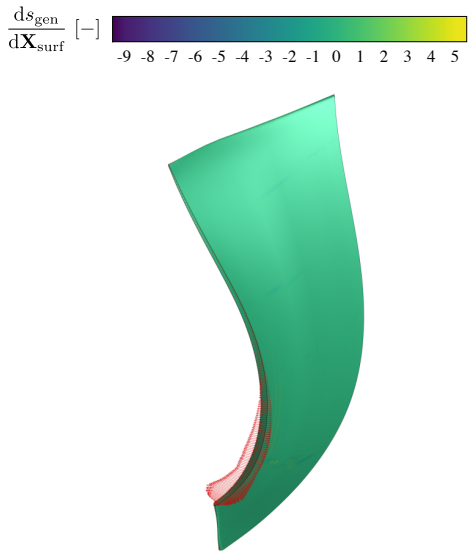


Figure 6.21: Adjoint sensitivity $\frac{ds_{gen}}{d\mathbf{X}_{surf}}$ map for the APU turbine rotor. Red vectors correspond to the adjoint sensitivities per surface points.

arrows. This means that any movement of the shroud surface causes a large change in the objective's function value. Possibly, the mismatch seen in the gradient validation plots can be attributed to these high sensitivities, as the greater disagreement in Figure 6.20 was observed at the shroud section. However, a more extensive study is documented in the next section.

6.2.2 Euler test case

In order to investigate the poor validation of the adjoint gradients shown in the previous section, a simpler test case is prepared which will allow to obtain results in a reasonable amount of time. The aforementioned test case consists of the same geometry but ran on a coarser mesh with Euler as governing equations. The same boundary conditions at inlet and outlet as described in Chapter 5, Section 5.2 are utilized. Even though by using Euler governing equations the viscosity is not taken into account, it is expected that the same behaviour and trends are shared between these test cases and the previously shown RANS case.

Two different test cases for the same geometry are prepared:

(a) The first one representing the blade and hub as Euler walls (i.e. solid walls where flow tangency is imposed) and the shroud as a fixed wall (i.e. the velocity equal to 0.0).

(b) The second one representing the blade as an Euler wall, and the hub and the shroud as symmetry walls.

The second test case is prepared in hope of obtaining a better understanding of how the hub and shroud surfaces are affecting the solution. As opposed to the Aachen test case, this geometry is extremely different, as it consists of a mixed-flow turbine. Furthermore, the blade's height is considerably smaller than the Aachen turbine stator. This implies that the perturbation of certain design variables can have an influence on the hub and shroud surfaces to a greater extent.

The convergence history for these test cases is shown in Figure 6.22. As it can be appreciated, due to the easiness of convergence of Euler problems, a six-order of magnitude reduction in the density residuals for the direct problem is achieved in 2,000 iterations. Similarly, the adjoint residuals show a reduction of five orders of magnitude, which is deemed to be more than satisfactory for results accuracy.

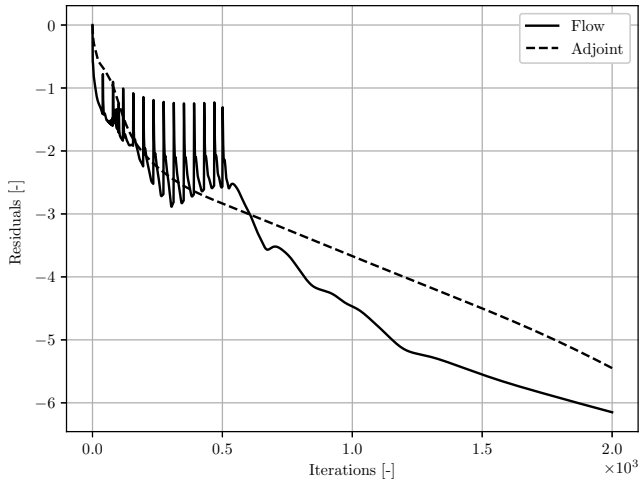
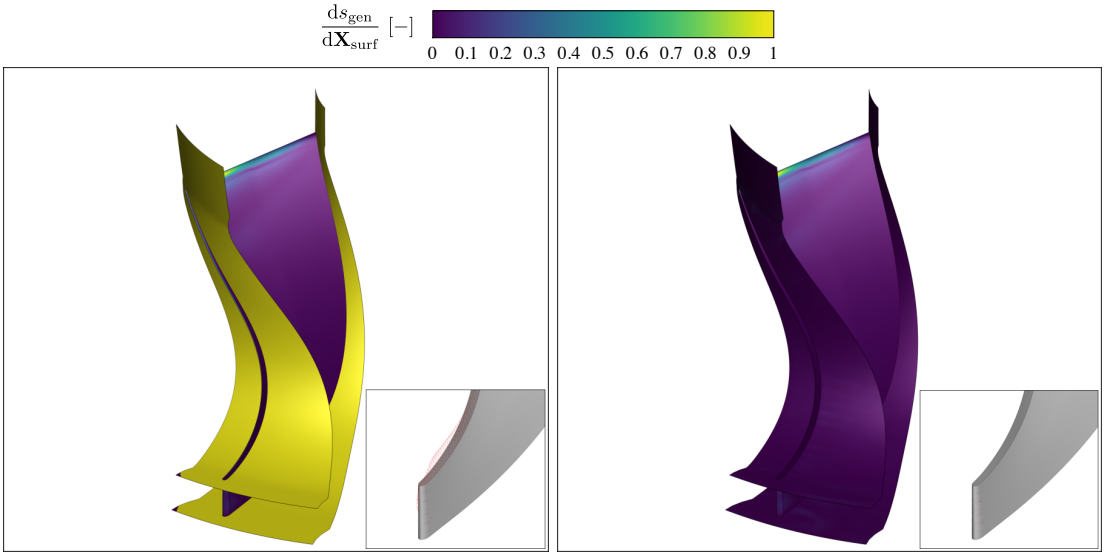


Figure 6.22: Convergence history for both flow and adjoint solution for the Euler APU turbine rotor test case.

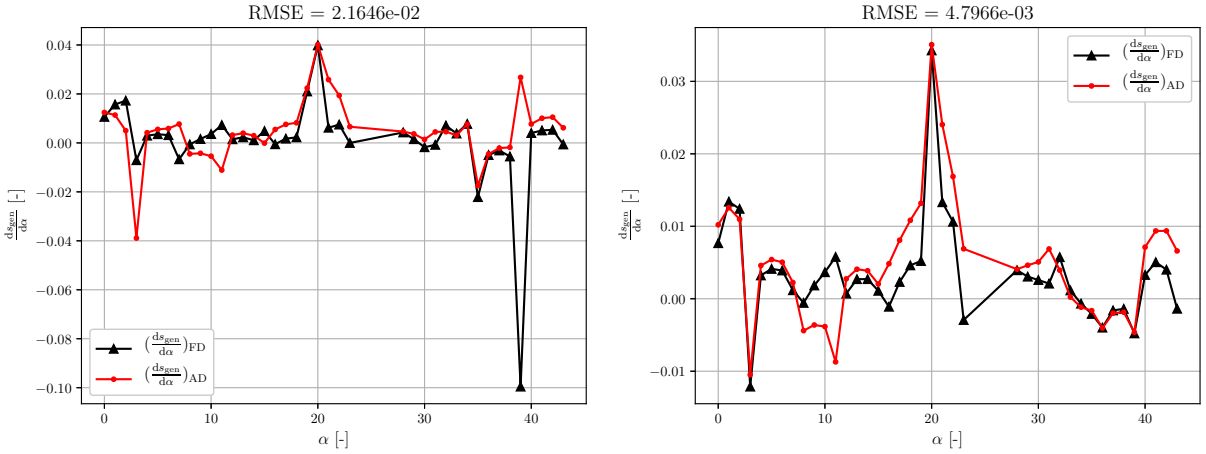


(a) Solid (Euler) walls. **(b)** Symmetry walls.

Figure 6.23: Non-dimensional adjoint sensitivity $\frac{ds_{gen}}{d\mathbf{X}_{surf}}$ map for the Euler APU turbine rotor test case. Red arrows show the surface sensitivity vectors.

Figure 6.23 shows the dimensionless adjoint sensitivity for both cases prepared. The adjoint sensitivities have been normalized so their value range from 0 to 1 for visualization purposes. Besides the high sensitivity region encountered at the trailing edge of most of the blade's height, which is expected similarly than in the previous Aachen turbine stator test case, it is important to note that for the solid (Euler) walls case, Figure 6.23a, both hub and shroud show large sensitivities. Furthermore, the sensitivity vectors are shown at the bottom right of each figure, and it can be seen that the addition of the fixed shroud in space produces a high sensitivity region, as well, in the blade surface.

The gradient validation performed for both of the test cases, which is shown in Figure 6.24, reveals that, by comparing the RMSE, a lower error is found when the hub and shroud walls are considered as symmetric. This implies that, when both surfaces are deemed as solid walls, there are very strong sensitivities at the hub and shroud surfaces which might need to be considered due to the movement



(a) Solid (Euler) walls. **(b)** Symmetry walls.

Figure 6.24: Gradient validation for the Euler APU turbine rotor test case.

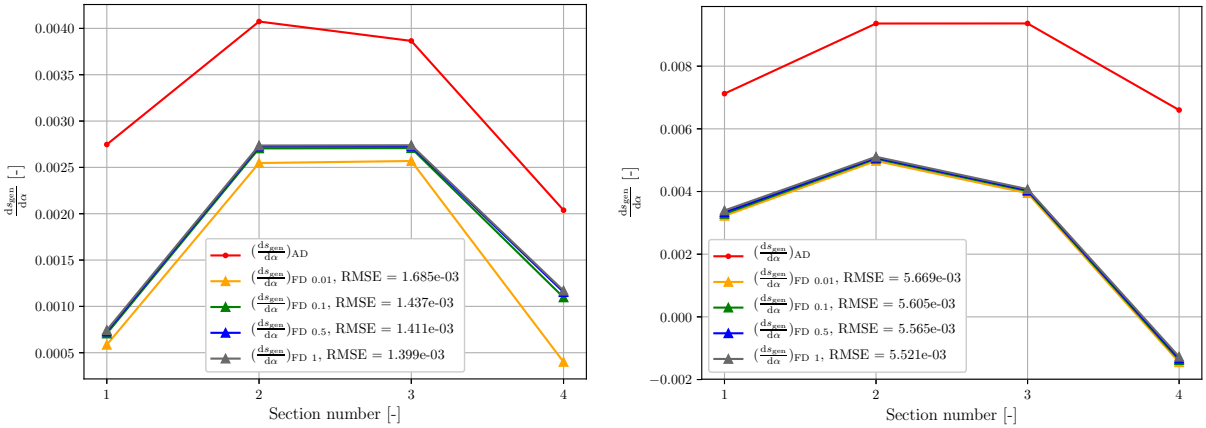
of these surfaces. In other words: when the total objective function gradient is calculated, which only considers contributions from the movement of the blade surface, two additional terms might be added, one corresponding to the hub and a different one for the shroud. Even though the deforming tangential boundary condition is applied in both cases, as explained in Chapter 3, this movement is captured by the finite differences, as the mesh employed to compute the gradient value is indeed the resulting deformed mesh, but not by the adjoint solver, as all the sensitivities are obtained at the blade surface but not at the hub and shroud walls.

Table 6.4: Root-mean-square error per design variable for the Euler APU turbine rotor.

Design variable	**Symbol**	**RMSE solid**	**RMSE sym.**
Stagger angle	ξ	1.72E-2	1.61E-3
LE metal angle	θ_{in}	7.41E-3	1.14E-3
TE metal angle	θ_{out}	1.08E-2	8.75E-3
LE wedge semi-angle	ϵ_{in}	2.90E-3	1.40E-3
TE wedge semi-angle	ϵ_{out}	5.18E-3	6.47E-4
LE radius	r_{in}	1.19E-2	5.97E-3
Distance 1	d_1	3.27E-3	2.73E-3
Distance 2	d_2	2.69E-3	1.04E-3
Distance 3	d_3	6.32E-2	2.80E-4
Distance 4	d_4	5.28E-3	5.52E-3

Table 6.4 gathers the RMSE for both of the cases studied. Overall, validation is poorer for the first case where hub and shroud are considered as solid walls. In the symmetric walls case, however, some design variables show some significant difference, such as the trailing edge metal angle, which is something expected as this is a very sensitive region at which the adjoint gradients are difficult to capture. Nevertheless, given the convergence history of both flow and adjoint simulations, shown in earlier paragraphs and featuring a six-order of magnitude residual reduction, this validation is not deemed as satisfactory.

In order to have a better understanding of the discrepancies shown in the validation with symmetry walls, Figure 6.25 displays the results for two different design variables per section. The finite differences results, for both of them, have converged towards a value. However, a jump between FD and AD sensitiv-



(a) Inlet wedge semi-angle. **(b)** Thickness distribution.

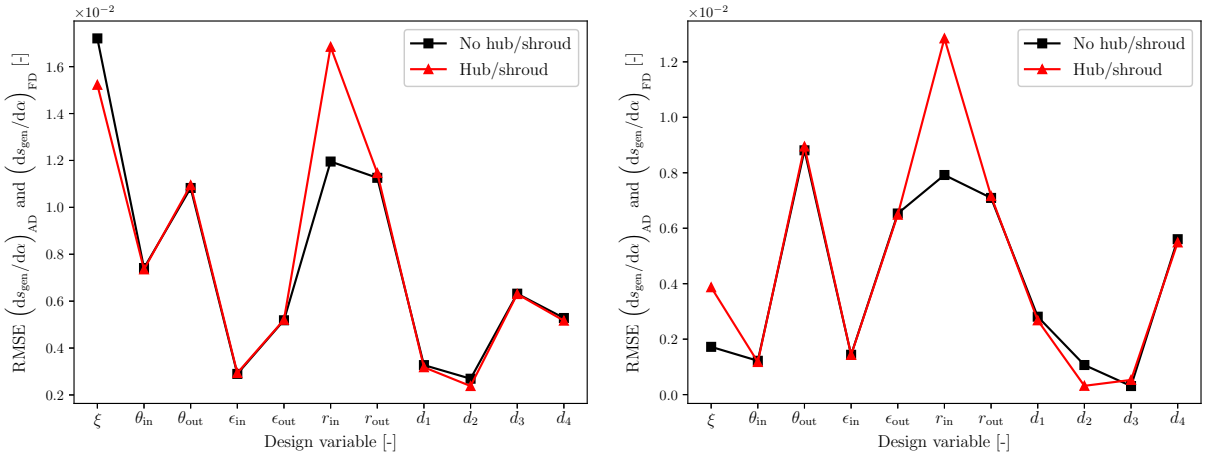
Figure 6.25: Gradient validation per section for the Euler APU turbine rotor test case with symmetry walls.

ities is observed for all the sections, which could possibly be caused by the fact that there is information missing stemming from the hub and shroud sensitivities not being included.

For the sake of getting a further insight on the poor gradient validation for both Euler test cases, a new test case where the hub and shroud sensitivities are included will be shown. In this case, the total objective function sensitivity can be computed as

$$\frac{\mathrm{d}s_{\mathrm{gen}}}{\mathrm{d}\alpha} = \left(\frac{\mathrm{d}s_{\mathrm{gen}}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}\right)_{\mathrm{blade}} + \left(\frac{\mathrm{d}s_{\mathrm{gen}}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}\right)_{\mathrm{hub}} + \left(\frac{\mathrm{d}s_{\mathrm{gen}}}{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}\frac{\mathrm{d}\mathbf{X}_{\mathrm{surf}}}{\mathrm{d}\alpha}\right)_{\mathrm{shroud}}. \tag{6.5}$$

As the hub and shroud surfaces are not directly modelled in the geometry parametrizer, the term $\mathrm{d}\mathbf{X}_{\mathrm{surf}}/\mathrm{d}\alpha$ is computed using finite differences from the resulting surface coordinates after performing mesh deformation for a given perturbation of a design variable α. Then, the dot product is computed using the same closest neighbour algorithm as for the blade surface.



(a) Solid (Euler) walls. **(b)** Symmetry walls.

Figure 6.26: Root-mean-square error between the adjoint and finite differences gradients per design variable, including hub and shroud sensitivities for the two APU turbine Euler cases studied.

Figure 6.26 shows the root-mean-square error (RMSE) between the total adjoint and finite difference sensitivities per design variable. Overall, two main points can be extracted from these figures:

- In agreement with what was seen before, case **(a)** shows a higher error than case **(b)** even when including the hub and shroud sensitivities.
- Including the hub and shroud sensitivities does not necessarily reduce the RMSE between both sensitivities. However, there is a substantial contribution to some design variables that needs to be further analyzed.

In conclusion, the perturbation of some design variables can produce movements in the hub and shroud, even if the mesh deforms sliding at these surfaces due to the tangential boundary condition, therefore the quality of the validation can potentially be improved if the hub and shroud surfaces are modelled as well in the CAD parametrization tool.

6.2.3 Optimization setup

Even though the gradient validation was not as satisfactory as the previous test case, a preliminary aerodynamic shape optimization of the APU turbine rotor is still performed. This way, it can be proved that the proposed methodology works as well with radial turbomachinery.

The optimization problem that is solved can be described as

$$\min_{\alpha} \qquad s_{\text{gen}}, \tag{6.6}$$

where s_{gen} is defined equally as in the test case described in Section 6.1. The objective function value is computed using *SU2*'s RANS solver with the Spalart-Allmaras turbulence model. Due to time limitations, no constraints are directly introduced; however, special attention will be considered for the stage work output.

The design vector will consist of the following parameters,

$$\alpha = \left\{\xi, \theta_{\text{in}}, \epsilon_{\text{in}}, \epsilon_{\text{out}}, r_{\text{in}}, d_1, d_2, d_3, d_4\right\}, \tag{6.7}$$

where no upper nor lower bounds are set for any of the design variables. Similarly as in the previous test case, there will not be any geometrical constrain directly imposed over the blade shape: instead, the step of the optimizer will be indirectly controlled by the use of the relaxation factor f. The outlet blade metal angle and trailing edge radius are left out of the design loop for the same motivation as the Aachen turbine stator.

6.2.4 Optimization progress

Due to time constraints, as this test case is very computationally expensive (shown in Chapter 5, Section 5.2) only four design steps were run for the APU turbine shape optimization. This, in turn, can prove that even though the gradient quality is not as good as expected, the methodology still can be applied to radial turbomachinery.

Figure 6.27 displays the evolution of the non-dimensional entropy generation per design iteration. A relaxation factor of 1E-4 was used in all the steps due to issues observed with mesh deformation: the mesh used is relatively fine, if compared to the turbine stator test case, more compact and contains wall refinement at both blade, hub and shroud surfaces. The spring analogy mesh deformation algorithm showed a bad performance for all optimization steps with relaxation factors ranging from $f = 0.1$ to $f = 0.001$, especially due to the strong gradients that are present at the vicinity of the shroud as observed in Figure 6.21.

The entropy generation across the stage is being reduced in each design step, which reveals that even though the gradient validation was not satisfactory still the sensitivities point towards the optimal

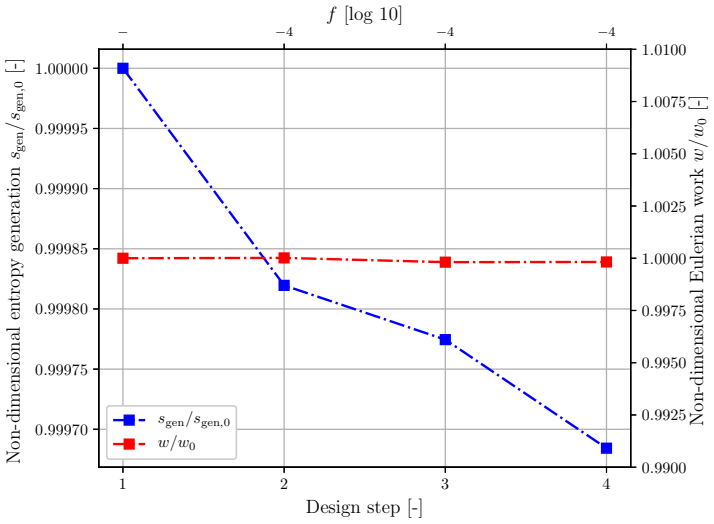


Figure 6.27: Optimization history for the APU turbine rotor.

direction. It is believed that this slow reduction in the objective function's value is produced by the conservative relaxation factor value used throughout the whole optimization. Furthermore, the Eulerian work, which was not directly included in the optimization loop but still is necessary to track, shows small, but not drastic, deviations from its initial value.

6.2.5 Conclusions

The following conclusions can be extracted from the APU turbine rotor test case study:

- The gradient validation is poor, even for a six-order-converged Euler test case. The research conducted points out that hub and shroud gradients may be included as well in the total sensitivity computation in order to improve accuracy. However, a further investigation of this topic is required.
- Even though the sensitivities were not as accurate as desired, the optimization showed (small) improvements in the entropy generation reduction across the blade passage.
- The spring analogy grid deformation algorithm showed poor performance for this test case due to the compactness of the mesh used.

Chapter 7

Conclusions and recommendations

Finally, this chapter serves to summarize and highlight the key contributions of the present Thesis in Section 7.1, and to make further recommendations for future work in the adjoint- and CAD-based turbomachinery shape optimization field in Section 7.2.

The objective of this work is to build an open-source framework for the aerodynamic shape optimization of both axial and radial turbomachinery. The novelty of the methodology implemented is that it relies on the discrete adjoint solver incorporated in *SU2*, which has been widely used in external flow applications but its capabilities for internal flows have yet not being exploited. Furthermore, the underlying CAD-based parametrization provided by *ParaBlade* allows to model axial and radial turbomachinery, as well as enabling a robust and intuitive control of the blade shape during the optimization process.

7.1 Conclusions

The main conclusions that are be drawn from this Master Thesis can be evaluated by answering to the main research question introduced in the first chapter.

Is it possible to couple the in-house CAD-based geometry parametrization tool ParaBlade with the open-source suite SU2 to perform aerodynamic shape optimization for axial and radial turbomachinery?

The developed methodology was successfully applied to perform an aerodynamic shape optimization of an axial turbine stator blade using the CAD- and adjoint-based sensitivities. These gradients were validated against those calculated by finite differences, where a good agreement was found with a RMSE of 1.29E-3. Computing the sensitivities using the discrete adjoint solver when compared to finite differences reported a reduction, in CPU-time, of the 95% (from 29,232 to 1,558 minutes), clearly showing the benefits of resorting to the adjoint method. Using a CAD-based parametrization tool allowed for an intuitive and effective treatment of the geometrical constraints during the optimization process, where the entropy generation across the blade passage was reduced by 7.59% while maintaining a constant trailing edge thickness and the outlet flow angle above a minimum threshold.

The initial mesh, which contained boundary layer refinement for the blade surface and inflation layer for both hub and shroud walls was effectively deformed during the optimization process but exclusively for small deformation steps. A re-mesh step was necessary to be included as the initial grid contained negative volumes after several optimization steps. The mesh morphing process accounted only the 0.2% of the total CPU-time in each optimization step, being deemed as computationally inexpensive. However, it highlighted the inability of the spring-analogy-based mesh deformation algorithm to deal with fine grids with boundary layer refinement, showing a slow and not sufficient reduction of the residuals when solving the linear system of equations.

The same methodology employed in the axial turbine blade was tested in a mixed-flow turbine rotor.

Validation was performed by comparing the discrete adjoint gradients to those stemming from finite differences, showing a RMSE of 9.37E-2, one order of magnitude higher than the axial test case. As the sensitivities did not show a good agreement, different test cases based on Euler governing equations were prepared. From those studies, it was concluded that the hub and shroud walls may have an effect on the discrete adjoint sensitivities which is currently not captured as these surfaces are not modelled in the geometry parametrization tool. Nevertheless, the gradient computation via the discrete adjoint method supposed a saving, in CPU-time, of the 93% when compared to finite differences, again showing the benefits of this method. A preliminary shape optimization of the rotor was performed, due to time constrains, where the entropy generation across the blade passage was successfully reduced.

The spring-analogy-based mesh deformation method was challenged in this test case compared to the previous axial case. The APU turbine's mesh is more compact, when compared to the Aachen stator's, and has an increased refinement to improve the flow and adjoint convergence. This hindered the mesh deformation especially in the vicinity of the hub and shroud walls. Nevertheless, during the optimization this process accounted only for the 0.2% of the total CPU-time, being again deemed as computationally inexpensive.

In conclusion, a framework for optimizing axial and radial turbomachinery has successfully been created. The underlying CAD-parametrization allows for an intuitive and efficient imposition of the geometrical constrains. The gradients calculated via the adjoint method accurately predict the optimal search direction. However, the radial turbine analyzed might lack the information stemming from the movement of the hub and shroud surfaces. The spring-analogy-based mesh deformation algorithm embedded in *SU2* can morph axial and radial grids. Nevertheless, it lacks of robustness when operating on fine meshes, especially containing boundary layer refinement at both blade, hub and shroud walls.

7.2 Recommendations for future work

Some further developments and improvements for the methodology implemented in this Thesis can be summarized as:

- Modelling of the hub and shroud surfaces: it was observed that for radial turbomachines the hub and shroud surfaces could have an influence when performing mesh deformation. These terms need to be included in the gradient computation and this can only be achieved if these walls are modelled in the geometry parametrization tool. This will also allow to have control of the channel shape, enabling endwall contouring and the inclusion of blade fillets.
- Implementation of a more robust and efficient mesh deformation algorithm: especially for fine meshes with boundary layer refinement at the hub and shroud walls, the spring-analogy method has been observed to be non-robust. Some volumetric meshes were unable to reproduce the change in surface shape correctly, as negative volume elements originated. Furthermore, it is deemed as non-efficient, where the residual reduction was not sufficient and the large memory required made the process impossible to be run in common workstations, especially for the radial case. The implementation of a linear-elasticity-based algorithm, capable of handling wall refinement and high-aspect ratio cells, can be beneficial for turbomachinery applications, as highlighted in [12]. Other technique that might be evaluated is the use of RBFs in several deformation steps to handle the restrictive requirements of the volumetric mesh, as seen in [28] and [16].
- Evaluate the impact of a different optimizer: even though the SLSQP algorithm was deemed as satisfactory, especially because the gradients were relaxed using a relaxation factor, and therefore the extent to which the grid morphing was performed, further optimizers could be tested once a more efficient mesh deformation algorithm is implemented. Steepest descent algorithms might be used to have a better control of the mesh deformation as argued in [12]; however, it has a slow convergence rate not suited for this type of problems. On the other hand, the optimization package pyOpt [56] provides a numerical framework for non-linear constrained problems with several SQP algorithms such as SNOPT, which is well-suited for large scale problems such as the ones analyzed.

- Increase the test case database to axial and radial compressors: due to the inability of efficiently computing direct CFD and adjoint solutions for compressors, these test cases were considered out of the scope of this work.

Even though the present work is a first step towards the use of an open-source integrated optimization framework for turbomachinery applications, relying on the fast computation of gradients by using the adjoint method and an efficient and robust geometry parametrization tool, some further investigations and extensions need to be made to keep in pace with the latest developments in turbomachinery CFD simulations, for instance:

- Multi-row and multi-stage shape optimization: *SU2* allows for multi-row calcualtions by the use of mixing-planes. Due to the modularity of the developed framework, its implementation can be straightforward.
- Reduced order models to solve for unsteady effects, such as the harmonic-balance method already implemented in *SU2* [38].
- Extension for multi-disciplinary shape optimization, taking into account structural and/or heat transfer phenomena, for instance.

Bibliography

[1] W. Dawes, "Turbomachinery computational fluid dynamics: asymptotes and paradigm shifts," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1859, pp. 2553–2585, 2007.

[2] L. He, "Multi-component Multi-disciplinary Multi-scale Interactions. Challenges and Opportunities in Turbomachinery CFD." 2019.

[3] Z. Li and X. Zheng, "Review of design optimization methods for turbomachinery aerodynamics," *Progress in Aerospace Sciences*, vol. 93, pp. 1–23, May 2017.

[4] A. Jameson, "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, vol. 3, no. 3, pp. 233–260, 1988.

[5] EUTurbines, "Enabling New Energy Technologies under Horizon 2020. A Roadmap on Turbomachinery Research 2014-2020.," 2014.

[6] "OpenFOAM. The open source CFD toolbox." `https://www.openfoam.com`. Accessed: 02-11-2019.

[7] "ANSYS Fluent Adjoint Solver." `https://www.ansys.com/products/fluids/shape-optimization`. Accessed: 02-11-2019.

[8] J. Luo, J. Xiong, F. Liu, and I. McBean, "Three-Dimensional Aerodynamic Design Optimization of a Turbine Blade by Using an Adjoint Method," *Journal of Turbomachinery*, vol. 133, p. 011026, January 2011.

[9] G. Yu, J.-D. Mueller, D. Jones, and F. Christakopoulos, "CAD-based shape optimisation using adjoint sensitivities," *Computers and Fluids*, vol. 46, no. 1, pp. 512–516, 2011.

[10] J. Luo, C. Zhou, and F. Liu, "Multipoint Design Optimization of a Transonic Compressor Blade by Using an Adjoint Method," *Journal of Turbomachinery*, vol. 136, p. 051005, May 2014.

[11] B. Walther and S. Nadarajah, "Optimum Shape Design for Multirow Turbomachinery Configurations Using a Discrete Adjoint Approach and an Efficient Radial Basis Function Deformation Scheme for Complex Multiblock Grids," *Journal of Turbomachinery*, vol. 137, p. 081006, August 2015.

[12] S. Xu, D. Radford, M. Meyer, and J.-D. Mueller, "CAD-based adjoint shape optimisation of a one-stage turbine with geometric constraints," *Proceedings of ASME Turbo Expo 2015: Turbine Technical Conference and Exposition. GT2015*, pp. 1–14, 2015.

[13] K. T. Tsiakas, F. Gagliardi, X. S. Trompoukis, and K. C. Giannakoglou, "Shape optimization of turbomachinery rows using a parametric blade modeller and the continuous adjoint method running on GPUs," *ECCOMAS Congress 2016. VII European Congress on Computational Methods in Applied Sciences and Engineering*, pp. 3972–3984, 2016.

[14] D. Agarwal, T. T. Robinson, C. G. Armstrong, S. Marques, I. Vasilopoulos, and M. Meyer, "Parametric design velocity computation for CAD-based design optimization using adjoint methods," *Engineering with Computers*, 2017.

[15] L. Mueller and T. Verstraete, "CAD Integrated Multipoint Adjoint-Based Optimization of a Turbocharger Radial Turbine," *International Journal of Turbomachinery Propulsion and Power*, vol. 2, no. 14, pp. 1–22, 2017.

[16] X. Tang, J. Luo, and F. Liu, "Adjoint aerodynamic optimization of a transonic fan rotor blade with a localized two-level mesh deformation method," *Proceedings of ASME Turbo Expo 2017: Turbomachinery Technical Conference and Exposition. GT2017*, pp. 1–11, 2017.

[17] G. Ntanakas, M. Meyer, and K. C. Giannakoglou, "Employing the Time-Domain Unsteady Discrete Adjoint Method for Shape Optimization of Three-Dimensional Multirow Turbomachinery Configurations," *Journal of Turbomachinery*, vol. 140, p. 081006, August 2018.

[18] M. Luers, M. Sagebaum, S. Mann, J. Backhaus, D. Grossman, and N. R. Gauger, "Adjoint-based Volumetric Shape Optimization of Turbine Blades," *AIAA Aviation Forum. 2018 Multidisciplinary Analysis and Optimization Conference*, pp. 1–14, 2018.

[19] O. Mykhaskiv, M. Banovic, S. Auriemma, P. Mohanamuraly, A. Walther, H. Legrand, and J.-D. Mueller, "NURBS-based and parametric-based shape optimization with differentiated CAD kernel," *Computer-Aided Design and Applications*, vol. 15, no. 6, pp. 916–926, 2018.

[20] N. Anand, S. Vitale, M. Pini, and P. Colonna, "Assessment of FFD and CAD-based shape parametrization methods for adjoint-based turbomachinery shape optimization.," *Proceedings of Montreal 2018 Global Power and Propulsion Forum*, pp. 1–8, May 2018.

[21] H.-Y. Wu and F. Liu, "Aerodynamic Design of Turbine Blades Using an Adjoint Equation Method," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, vol. 1006, pp. 1–13, 2005.

[22] A. Jaworski and J.-D. Mueller, "Toward Modular Multigrid Design Optimisation," *Lecture Notes in Computational Science and Engineering. Advances in Automatic Differentiation.*, pp. 281–292, 2008.

[23] J. A. Samareh, "A Survey of Shape Parameterization Techniques," pp. 333–343, 1999.

[24] M. C. Duta, S. Shahpar, and M. B. Giles, "Turbomachinery design optimization using automatic differentiated adjoint code," *Proceedings of GT2007. ASME Turbo Expo 2007: Power for Land, Sea and Air*, pp. 1–10, 2007.

[25] L. Lapworth, "HYDRA-CFD: A Framework for Collaborative CFD Development," *International Conference on Scientific and Engineering Computation (IC-SEC), Singapore*, vol. 30, no. 1987, pp. 1–6, 2004.

[26] D. X. Wang and L. He, "Adjoint Aerodynamic Design Optimization for Blades in Multistage Turbomachines - Part I: Methodology and Verification," *Journal of Turbomachinery*, vol. 132, p. 021011, April 2010.

[27] D. X. Wang, L. He, Y. S. Li, and R. G. Wells, "Adjoint Aerodynamic Design Optimization for Blades in Multistage Turbomachines - Part II: Validation and Application," *Journal of Turbomachinery*, vol. 132, pp. 2156–2169, April 2010.

[28] B. Walther and S. Nadarajah, "An adjoint-based multi-point optimization method for robust turbomachinery design," *Proceedings of ASME Turbo Expo 2015: Turbine Technical Conference and Exposition. GT2015*, pp. 1–12, 2015.

[29] B. Walther and S. Nadarajah, *Constrained Adjoint-Based Aerodynamic Shape Optimization in a Multistage Turbomachinery Environment.*

[30] S. Vitale, *Advancements in Automated Design Methods for NICFD Turbomachinery.* PhD thesis, Delft University of Technology, November 2018.

[31] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso, "SU2: An Open-Source Suite for Multiphysics Simulation and Design," *AIAA Journal*, vol. 54, pp. 828–846, March 2016.

[32] L. Piegl and W. Tiller, *The NURBS Book*. Springer, 2 ed., 1996.

[33] I. Vasilopoulos, P. Flassig, and M. Meyer, "CAD-based aerodynamic optimization of a compressor stator using conventional and adjoint-driven approaches," *Proceedings of ASME Turbo Expo 2017: Turbomachinery Technical Conference and Exposition. GT2017*, pp. 1–12, 2017.

[34] S. Shahpar and L. Lapworth, "PADRAM: Parametric Design and Rapid Meshing system for Turbomachinery Optimisation," *Proceedings of ASME Turbo Expo 2003. Power for Land, Sea and Air*, 2003.

[35] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso, "The Complex-Step Derivative Approximation," *ACM Transactions on Mathematical Software*, vol. 29, pp. 245–262, September 2003.

[36] D. Papadimitriou and K. C. Giannakoglou, "Compressor Blade Optimization Using a Continuous Adjoint Formulation," *Proceedings of GT2006. 2006 ASME TURBO EXPO*, January 2006.

[37] M. B. Giles, M. C. Duta, J.-D. Mueller, and N. A. Pierce, "Algorithm Developments for Discrete Adjoint Methods," *AIAA Journal*, vol. 41, pp. 198–205, February 2003.

[38] A. Rubino, *Reduced Order Models for Unsteady Fluid Dynamic Optimization of Turbomachinery*. PhD thesis, Delft University of Technology, July 2019.

[39] R. Agromayor and N. Anand, *ParaBlade documentation*, 2019.

[40] F. Palacios, M. Colonno, A. Aranake, A. Campos, S. Copeland, T. Economon, A. Lonkar, T. Lukaczyk, T. Taylor, and J. Alonso, "Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design," 01 2013.

[41] P. Spalart and S. Allmaras, "A one-equation turbulence model for aerodynamic flows," *AIAA*, vol. 439, 01 1992.

[42] F. Menter, *Zonal Two Equation k-w Turbulence Models For Aerodynamic Flows*.

[43] A. Jameson, "Origins and further development of the jameson–schmidt–turkel scheme," *AIAA Journal*, vol. 55, pp. 1–23, March 2017.

[44] P. L. Roe, "Approximate riemann solvers, parameter vectors, and difference schemes," 1997.

[45] V. Venkatakrishnan, "On the accuracy of limiters and convergence to steady state solutions," in *31st Aerospace Sciences Meeting*, 1993.

[46] G. van Albada, B. van Leer, and Roberts, "A comparative study of computational methods in cosmic gas dynamics," *Astronomy and Astrophysics*, vol. 108, pp. 76–84, 04 1982.

[47] T. Oliphant, *Guide to NumPy*. January 2006.

[48] D. Kraft, *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht, Wiss. Berichtswesen d. DFVLR, 1988.

[49] "The Python Standard Library - Python 3.7.4 documentation." `https://docs.python.org/3.7/library/`. Accessed: 30-09-2019.

[50] H. Gallus, C. Poensgen, and J. Zeschky, "Three-dimensional unsteady flow in a single stage axial-flow turbine and compressor," 1995.

[51] R. Walraevens and H. Gallus, "Testcase 6 – 1-1/2 stage axial flow turbine," 1997.

[52] "ANSYS TurboGrid: Blade Meshing for Turbomachinery." `https://www.ansys.com/products/fluids/ansys-turbogrid`. Accessed: 30-09-2019.

[53] A. C. Jones, "Design and Test of a Small, High Pressure Ratio Radial Turbine," *Journal of Turbomachinery*, vol. 118, pp. 362–370, 04 1996.

[54] J. Keep, S. Vitale, M. Pini, and M. Burigana, "Preliminary verification of the open-source CFD solver SU2 for radial-inflow turbine applications," *Energy Procedia*, vol. 129, pp. 1071–1077, 09 2017.

[55] A. P. Saxer, *A numerical analysis of 3-D inviscid stator/rotor interactions using non-reflecting boundary conditions*. PhD thesis, Massachusetts Institute of Technology, March 1992.

[56] R. E. Perez, P. W. Jansen, and J. R. R. A. Martins, "pyOpt: A Python-based object-oriented framework for nonlinear constrained optimization," *Structures and Multidisciplinary Optimization*, vol. 45, no. 1, pp. 101–118, 2012.

Appendix A

Aachen turbine stator

In this appendix, all the relevant configuration files and others for the Aachen turbine stator test case will be included.

A.1 Geometry parametrization

Blade matching in `MatchBlade.py`.

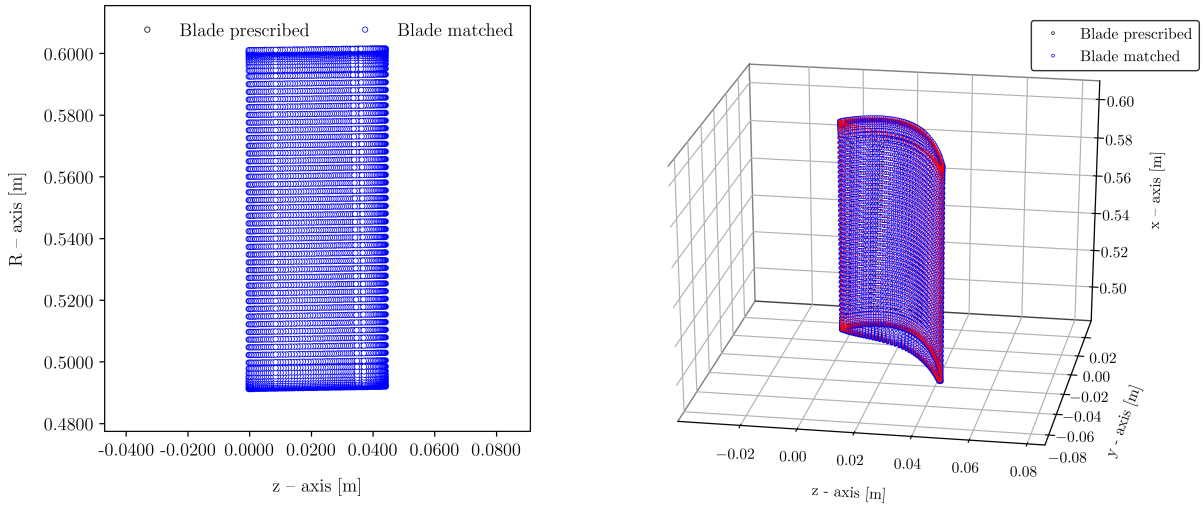


Figure A.1: Blade matching surface plots for the Aachen turbine stator baseline design.

ParaBlade configuration file for the baseline and optimized Aachen turbine stator.

baseline_aachen.cfg

```
NDIM=3.0
REP_TYPE=AXISYM
NBLADES=50.0
x_leading=-0.0004055388565050492
y_leading=-0.006441361249941034
z_leading=0.6010480843273525, 0.543357665136443, 0.49113621235877963
x_trailing=0.044019040998267554
z_trailing=0.6015831969951295, 0.5375028999298332, 0.489277416819547
x_hub=
z_hub=
x_shroud=
z_shroud=
stagger=-0.7768012787765164, -0.7768012787765164, -0.7768012787765164, -0.7768012787765164,
    -0.7768012787765164
theta_in=0.065712340134762, 0.065712340134762, 0.065712340134762, 0.065712340134762, 0.065712340134762
theta_out=-1.2366839860979455
wedge_in=0.6365499505472934, 0.6365499505472934, 0.6365499505472934, 0.6365499505472934,
    0.6365499505472934
wedge_out=0.0503002146893723, 0.0503002146893723, 0.0503002146893723, 0.0503002146893723,
    0.0503002146893723
radius_in=0.028081467868199164, 0.028081467868199164, 0.028081467868199164, 0.028081467868199164,
    0.028081467868199164
radius_out=0.013405435245379524
dist_1=0.5924530430090768, 0.5924530430090768, 0.5924530430090768, 0.5924530430090768,
    0.5924530430090768
dist_2=0.3076453282422903, 0.3076453282422903, 0.3076453282422903, 0.3076453282422903,
    0.3076453282422903
dist_3=0.6295207167282275, 0.6295207167282275, 0.6295207167282275, 0.6295207167282275,
    0.6295207167282275
dist_4=0.16184026731581858, 0.16184026731581858, 0.16184026731581858, 0.16184026731581858,
    0.16184026731581858
PLOT_FORMAT=NONE
PRESCRIBED_BLADE_FILENAME=MoveSurface.txt
OPERATION_TYPE=SENSITIVITY
```

optimized_aachen.cfg

```
NDIM=3.0
REP_TYPE=AXISYM
NBLADES=50.0
x_leading=-0.0004055388565050492
y_leading=-0.006441361249941034
z_leading=0.601047740303605, 0.5433514427801326, 0.49113705186229323
x_trailing=0.044019040998267554
z_trailing=0.6016524573034141, 0.535446470034067, 0.4892833082352871
x_hub=
z_hub=
x_shroud=
z_shroud=
stagger=-0.80243521604241219, -0.76891415770729887, -0.75385100222281909, -0.7516438510872413,
    -0.74301370512602238
theta_in=0.066733282274462055, 0.06872181475426492, 0.068606836591233358, 0.06981409613667626,
    0.070711914321581498
theta_out=-1.2366839860979455
wedge_in=0.63635792546065895, 0.63919437288850545, 0.63710326182585797, 0.6404404177282631,
    0.6419407021874437
wedge_out=0.039225440928813873, 0.04489243689026548, 0.0438650991787084, 0.047107286162834584,
    0.045913523499894957
radius_in=0.028455514458357405, 0.022034150121905152, 0.019373677796396344, 0.019139889910082794,
    0.018591921442930506
radius_out=0.013405435245379524
dist_1=0.59355562800830042, 0.58337147141807988, 0.57638337814578577, 0.57604034825437644,
    0.57213015675487955
dist_2=0.30828034730397175, 0.30800532878681758, 0.30754039556379231, 0.30776384246964478,
    0.30742304893086414
dist_3=0.6305325237474575, 0.63117938073636126, 0.62720357246251257, 0.62979013026914221,
    0.62930663833521128
dist_4=0.1630266022384122, 0.16162645359556108, 0.15810183166948266, 0.16002886509922312,
    0.15897232598170699
PLOT_FORMAT=NONE
PRESCRIBED_BLADE_FILENAME=MoveSurface.txt
OPERATION_TYPE=SENSITIVITY
```

A.2 Preliminary grid convergence study

Prior to the gradient validation and further optimization, it is advised to perform a preliminary grid convergence study in order to work with a mesh for which its solution is grid-independent. This is done by preparing four different meshes, for which the number of elements on each one is shown in Table A.1.

Table A.1: Grid convergence study meshes for the Aachen stator.

Grid	Elements
Grid 1	307050
Grid 2	416431
Grid 3	514242
Grid 4	707194
Grid 5	909847

The figure of merit that is going to be analysed is the dimensionless entropy generation in the passage, calculated as [38]

$$s_{\text{gen}} = \frac{\langle s_{\text{out}} \rangle - \langle s_{\text{in}} \rangle}{v_0^2 / T_{0,\text{in}}} \tag{A.1}$$

where the values of the entropy at inlet and outlet of the domain are calculated using a mixed-out average, $T_{0,\text{in}}$ is the total inlet temperature, and v_0 is defined as $v_0 = \sqrt{2(h_{0,\text{in}} - h_{is,\text{out}})}$ where $h_{0,\text{in}}$ and $h_{\text{is,out}}$ are the total enthalpy at the inlet and the isentropic outlet static enthalpy respectively.

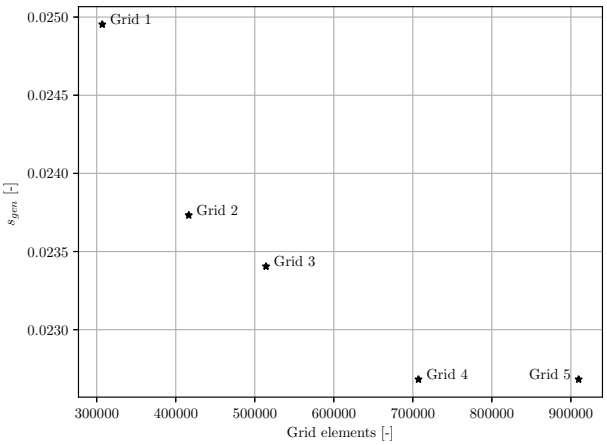


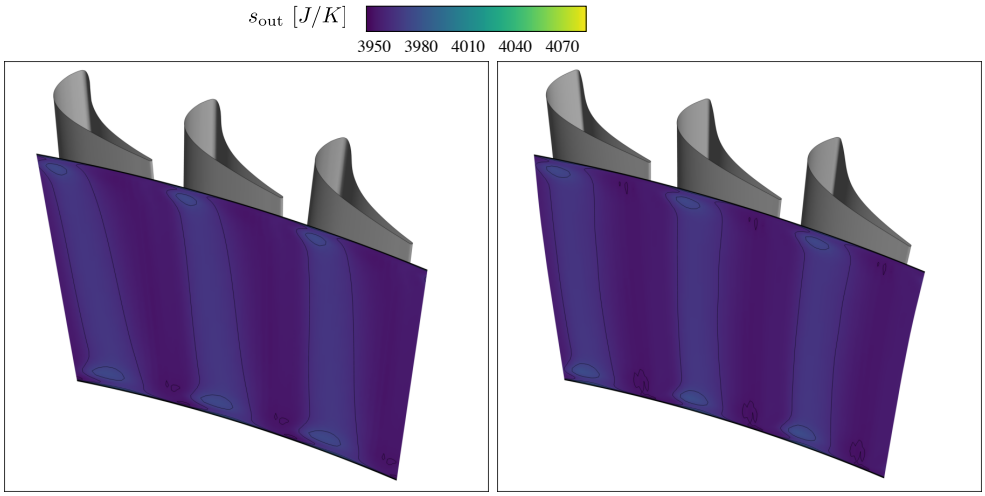
Figure A.2: Dimensionless entropy generation per grid for the Aachen stator.

The results from the preliminary grid convergence study can be seen in Figure A.2, where the dimensionless entropy generation in the passage is plotted against the number of elements per grid. As it can be seen, the solution converges at, approximately, a number of elements of 700,000.

However, the final mesh used for the gradient validation and shape optimization for the Aachen test case is grid number 3. This decision was exclusively motivated by runtime issues: while grid 4 was already deemed as being converged, the computational requirements, not only in terms of CPU-time, but also on memory usage were too large. Furthermore, the mesh deformation algorithm employed (spring-analogy-based) showed a slow reduction of the residuals, which can be translated into poor performance even for

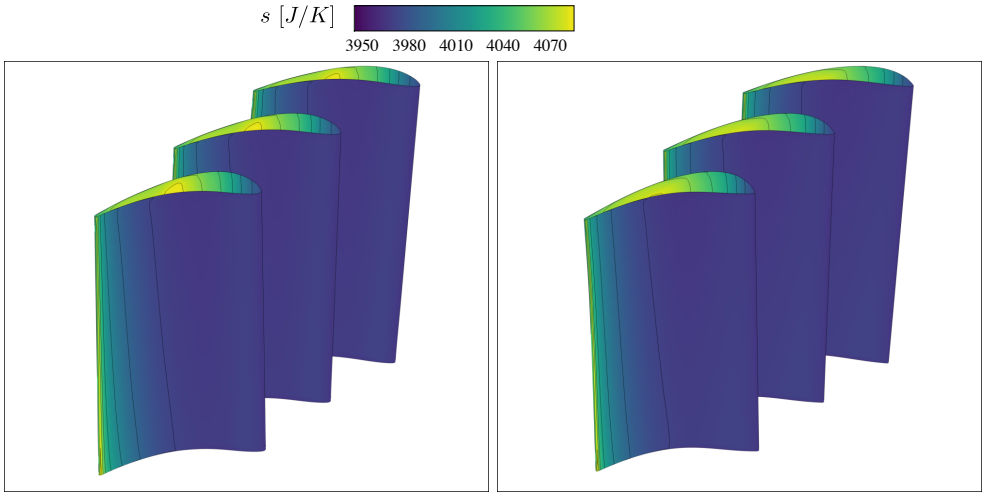
small deformations. Therefore, the study was conducted with grid number 3, assuming that the solution is not fully-grid independent. Nevertheless, the use of mesh deformation can ensure, as the number of elements remains constant, a continuity of the gradient information being passed from one iteration to another.

A.3 Optimized design plots



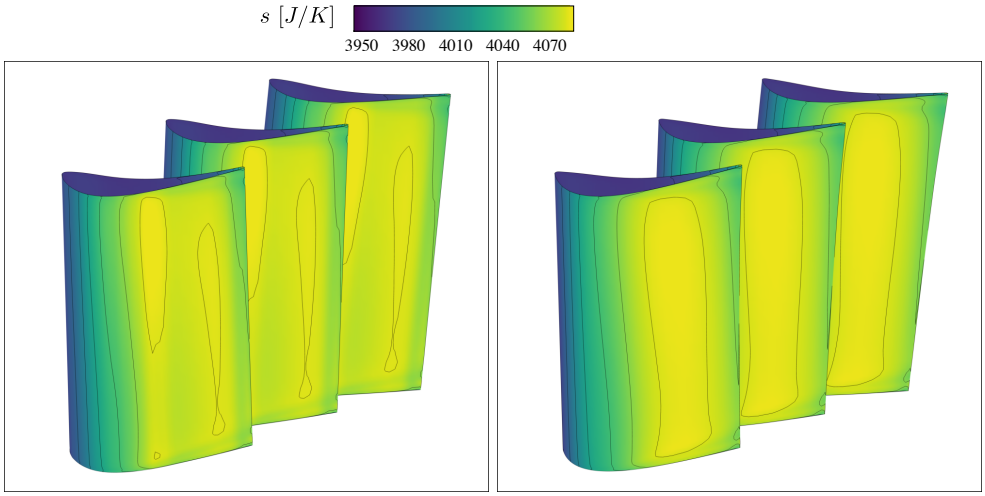
(a) Baseline. **(b)** Optimized.

Figure A.3: Comparison between the outflow entropy for the baseline **(a)** and optimized **(b)** designs in the Aachen turbine stator test case.



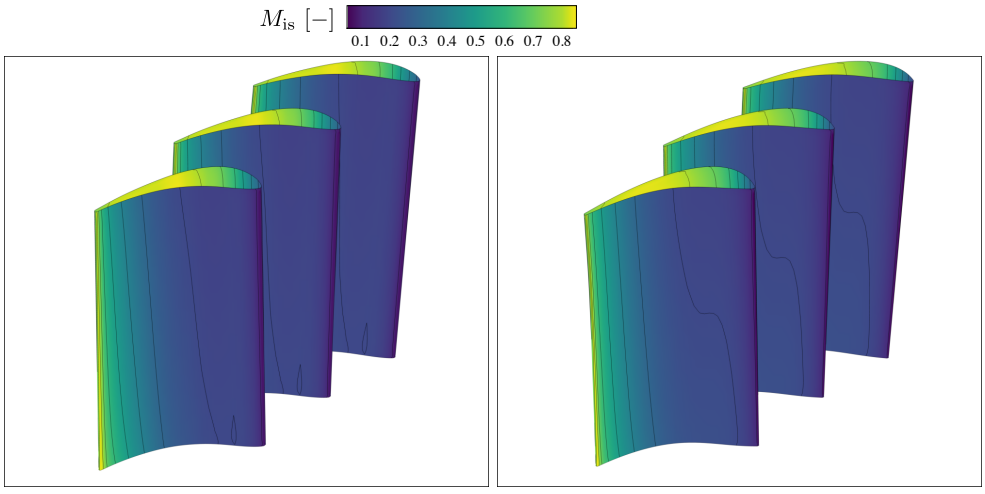
(a) Baseline. **(b)** Optimized.

Figure A.4: Comparison between the pressure-side surface entropy for the baseline **(a)** and optimized **(b)** designs in the Aachen turbine stator test case.



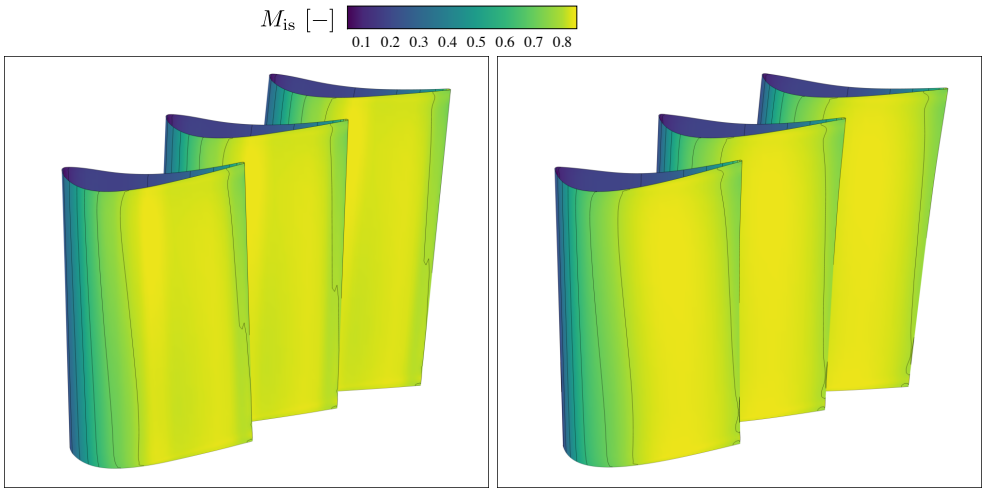
(a) Baseline. **(b)** Optimized.

Figure A.5: Comparison between the suction-side surface entropy for the baseline **(a)** and optimized **(b)** designs in the Aachen turbine stator test case.



(a) Baseline. **(b)** Optimized.

Figure A.6: Comparison between the pressure-side surface isentropic Mach number for the baseline **(a)** and optimized **(b)** designs in the Aachen turbine stator test case.



(a) Baseline. **(b)** Optimized.

Figure A.7: Comparison between the suction-side surface isentropic Mach number for the baseline **(a)** and optimized **(b)** designs in the Aachen turbine stator test case.

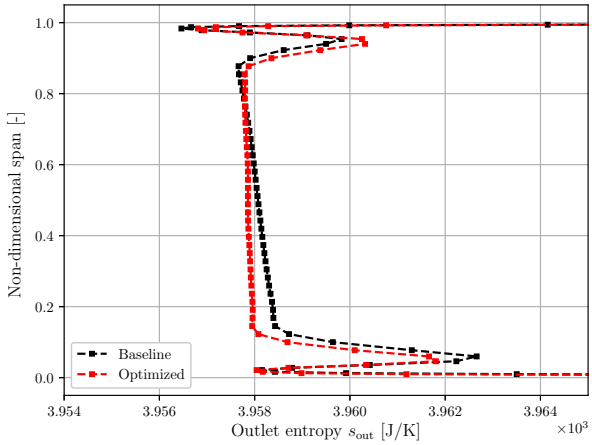
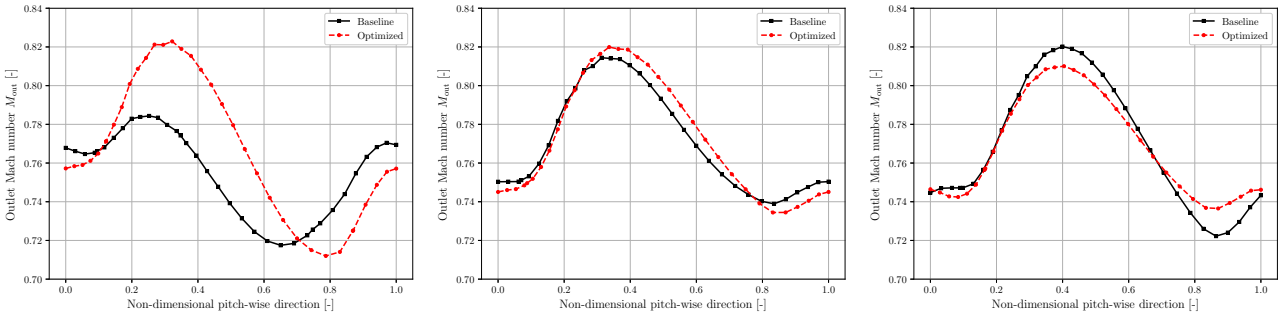
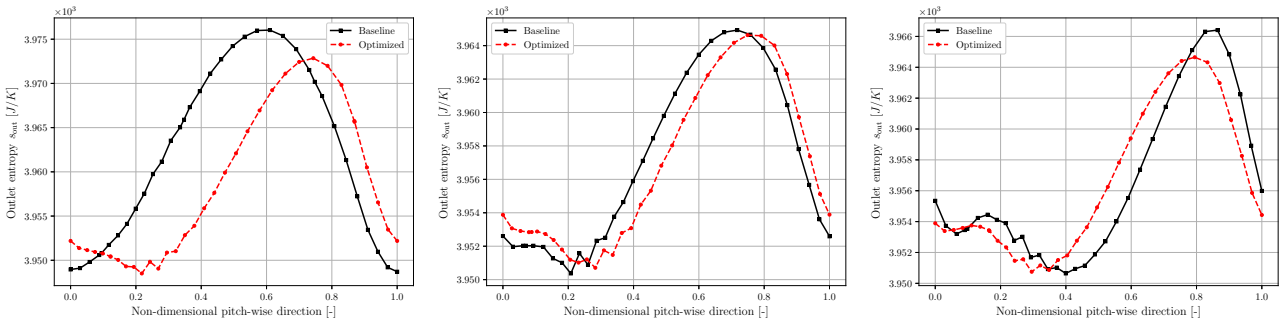


Figure A.8: Comparison between the outlet entropy for the baseline (black) and optimized (red) designs for the Aachen turbine stator.



(a) Hub. **(b)** Mid-span. **(c)** Shroud.

Figure A.9: Outlet Mach number M_{out} comparison between the baseline (black) and optimized (red) designs for three different span-wise sections for the Aachen turbine stator.



(a) Hub. **(b)** Mid-span. **(c)** Shroud.

Figure A.10: Outlet entropy s_{out} comparison between the baseline (black) and optimized (red) designs for three different span-wise sections for the Aachen turbine stator.

Appendix B

APU turbine rotor

In this appendix, all the relevant configuration files and others for the APU turbine rotor test case will be included.

B.1 Geometry parametrization

Blade matching in `MatchBlade.py`.

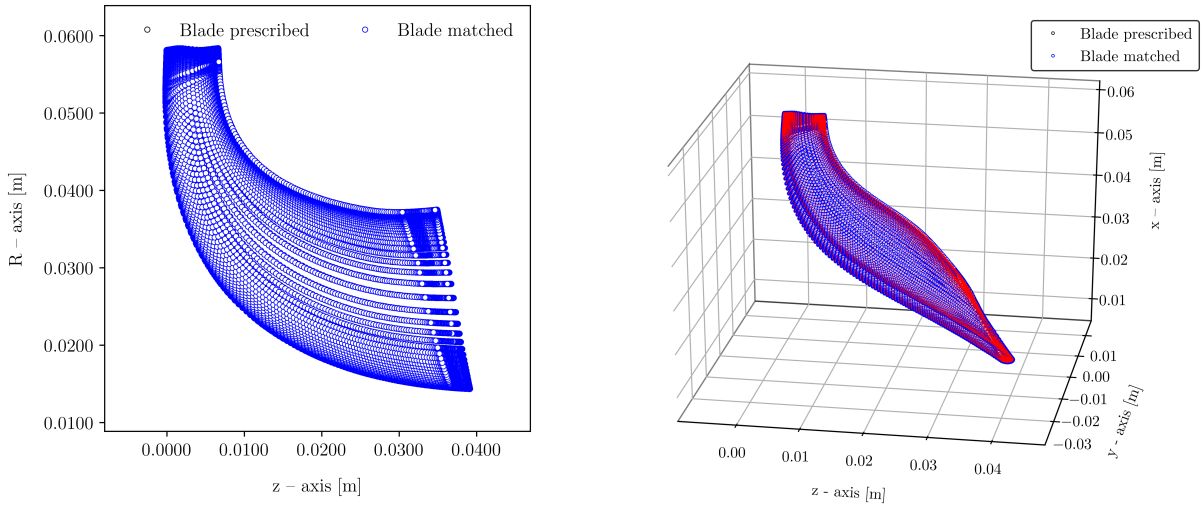


Figure B.1: Blade matching surface plots for the APU turbine rotor baseline design.

ParaBlade configuration file for the baseline and optimized APU turbine rotor.

baseline_apu.cfg

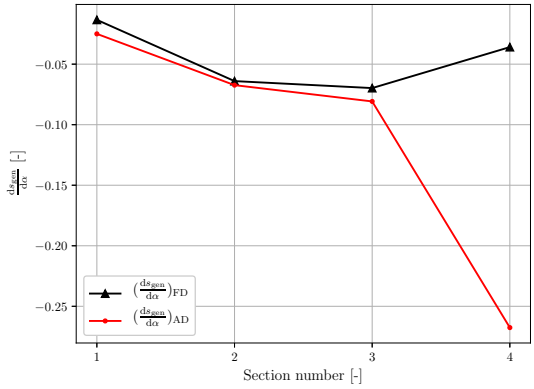
```
NDIM=3.0
REP_TYPE=AXISYM
NBLADES=50.0
x_leading=-3.646194798474673e-05, 0.002412672420951997, 0.003174714712943419, 0.0066934076728881915
y_leading=0.00018441780829557116, 1.2335082593509619e-05, 8.439395000114186e-06,
     -0.0003908035053091713
z_leading=0.05814444575395078, 0.05897865299088579, 0.05758999611884934, 0.05838433191320845
x_trailing=0.03914631099715976, 0.03730615940542024, 0.03691134272242666, 0.03488488094535926
z_trailing=0.014251395755852195, 0.021775511343813704, 0.017501807067298307, 0.037582470335704624
x_hub=-0.0016764095587437527, 0.004447985657424302, 0.005083326473182464, 0.02018377966101807
z_hub=0.037668524174860545, 0.032844457636426745, 0.02210075690574945, 0.015265441165930538
x_shroud=0.006776732406679054, 0.006727163630613125, 0.02631330749306577, 0.021493563846726942
z_shroud=0.05083661766747923, 0.03624821053919685, 0.040469112907640964, 0.036144983535726316
stagger=-0.22424884116177887, -0.3383177924943896, -0.3862605978997724, -0.6482982677287391
theta_in=-0.09543152957049936, -0.016947746420696345, -0.022185587259216263, 0.1588072094294682
theta_out=-0.845776191809606, -1.022982247618011, -1.2033704160573362, -1.3375620385880842
wedge_in=0.18485367138561715, 0.15084785287883917, 0.19459562922475904, 0.037189710547092844
wedge_out=0.03237222991628105, 0.06321104106294874, 0.10657633503212763, 0.050151855319982355
radius_in=0.005447148417316654, 0.0012652507977993573, 0.0028366678421749654, 0.008932502990141158
radius_out=0.011955794350015474, 0.007633437795294236, 0.0012772516734290494, 0.0019626594389498 69
dist_1=0.4266156881384261, 0.4050044486682227, 0.3588919855248973, 0.34288149593405426
dist_2=0.4310565168084504, 0.40746758852349024, 0.3652831967034764, 0.3429178521248926
dist_3=0.37894577252962125, 0.4086864865054647, 0.440750365485985, 0.386559033278867
dist_4=0.38518336454819047, 0.3753731733195784, 0.3394635819027098, 0.3650769149069278
PLOT_FORMAT=NONE
PRESCRIBED_BLADE_FILENAME=MoveSurface.txt
OPERATION_TYPE=SENSITIVITY
```

optimized_apu.cfg

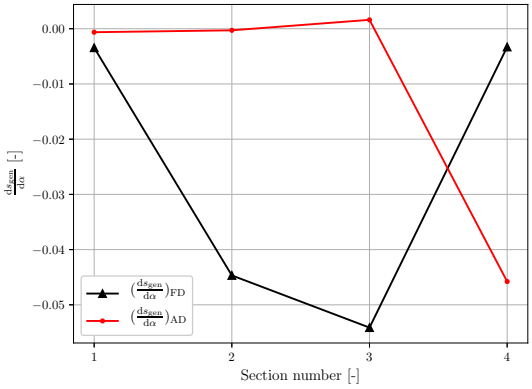
```
NDIM=3.0
REP_TYPE=AXISYM
NBLADES=50.0
x_leading=-3.646194798474673e-05, 0.002412672420951997, 0.003174714712943419, 0.0066934076728881915
y_leading=0.00018441780829557116, 1.2335082593509619e-05, 8.439395000114186e-06,
     -0.0003908035053091713
z_leading=0.05814444575395078, 0.05897865299088579, 0.05758999611884934, 0.05838433191320845
x_trailing=0.03914631099715976, 0.03730615940542024, 0.03691134272242666, 0.03488488094535926
z_trailing=0.014251395755852195, 0.021775511343813704, 0.017501807067298307, 0.037582470335704624
x_hub=-0.0016764095587437527, 0.004447985657424302, 0.005083326473182464, 0.02018377966101807
z_hub=0.037668524174860545, 0.032844457636426745, 0.02210075690574945, 0.015265441165930538
x_shroud=0.006776732406679054, 0.006727163630613125, 0.02631330749306577, 0.021493563846726942
z_shroud=0.05083661766747923, 0.03624821053919685, 0.040469112907640964, 0.036144983535726316
stagger=-0.22359002846276751, -0.3372497677197383, -0.38547687672325387, -0.64358191116235486
theta_in=-0.095458729210573273, -0.016918374255071433, -0.02215151108234328, 0.16016978265756437
theta_out=-0.845776191809606, -1.022982247618011, -1.2033704160573362, -1.3375620385880842
wedge_in=0.18473570988930352, 0.15072753294393992, 0.1944873741875284, 0.038242720792436562
wedge_out=0.032275205010784458, 0.062988640764485487, 0.10641925572167671, 0.049123293118898244
radius_in=0.0045034120115755418, 0.00086292139970960385, 0.0026446434868280188, 0.0033262202486732621
radius_out=0.011955794350015474, 0.007633437795294236, 0.0012772516734290494, 0.0019626594389498 69
dist_1=0.42658333122150216, 0.40499194795668186, 0.35890181675317229, 0.33675678749514609
dist_2=0.43106901689950455, 0.40749985247261933, 0.36528181646841462, 0.34426756505007505
dist_3=0.37892018130260097, 0.40866174387538401, 0.4407608184624201, 0.38613676318075318
dist_4=0.38510022696417007, 0.37543891700582038, 0.33960040902905392, 0.36767230649066962
PLOT_FORMAT=NONE
PRESCRIBED_BLADE_FILENAME=MoveSurface.txt
OPERATION_TYPE=SENSITIVITY
```

B.2 Gradient validation

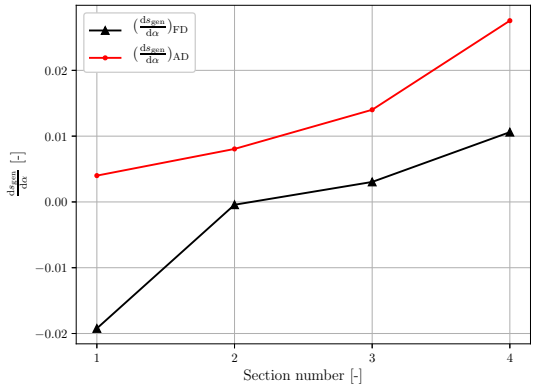
Gradient validation per design variable for the RANS second-order test case.



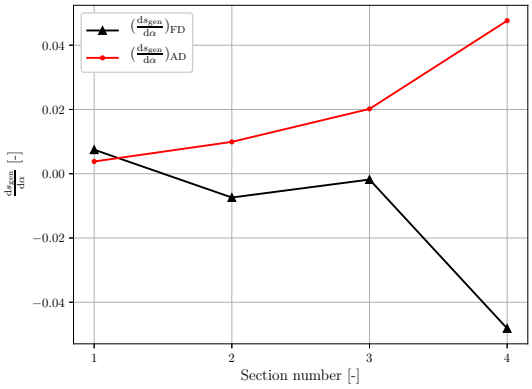
(a) Stagger angle.



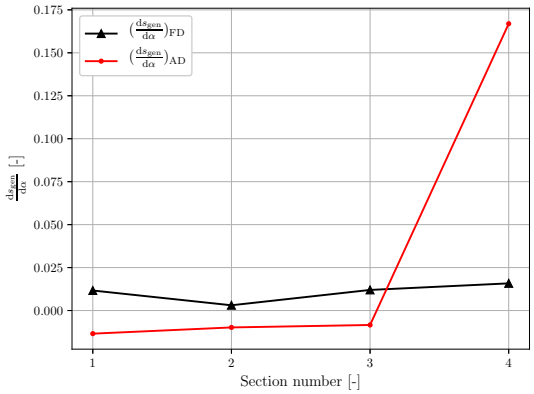
(b) Inlet blade metal angle.



(c) Outlet blade metal angle.

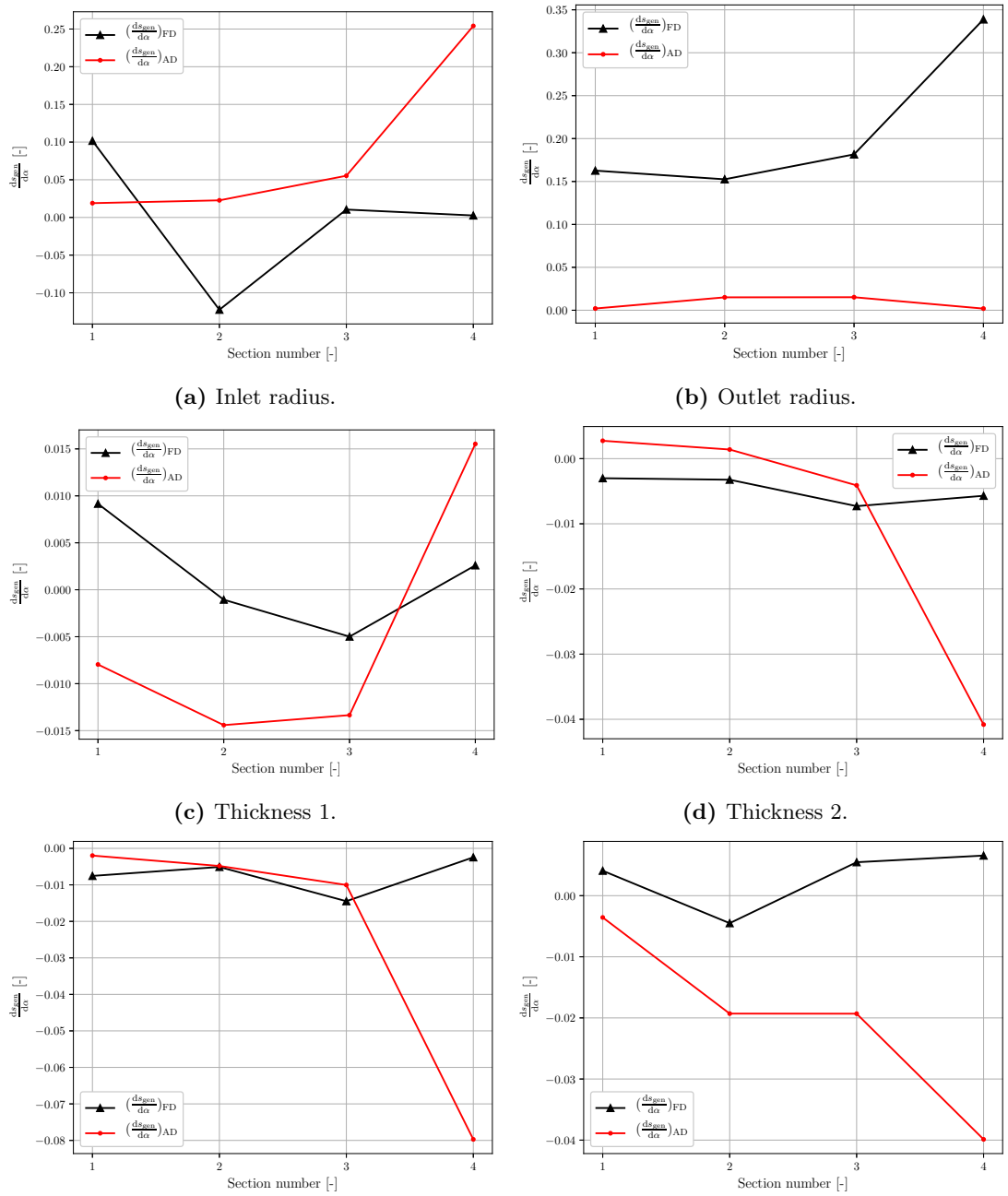


(d) Inlet wedge semi-angle.



(e) Outlet wedge semi-angle.

Figure B.2: Gradient validation per design variable for the APU turbine rotor RANS second-order test case.



(a) Inlet radius.

(b) Outlet radius.

(c) Thickness 1.

(d) Thickness 2.

(e) Thickness 3.

(f) Thickness 4.

Figure B.3: Gradient validation per design variable for the APU turbine rotor RANS second-order test case.