National Technical University of Athens

School of Mechanical Engineering
Fluids Department
Lab. of Thermal Turbomachines
Parallel CFD
& Optimization Unit

Programming of a "Back-to-CAD" Tool for Use in the Analysis and Optimization of Turbomachinery Rows. Industrial Applications

Diploma Thesis

Markella Zormpa

Supervisor:
Kyriakos C. Giannakoglou, Professor NTUA

Athens, October 2018

Acknowledgements

First, i would like to express my deepest gratitude to my professor, K.Giannakoglou who has been guiding me for all the years of my studies and has provided me with hours of incredible teaching, the opportunity to learn more whenever I asked for it, the willingness to help and his extensive hours of advice and guidance throughout the integration of this diploma thesis. For all this generosity I consider myself very lucky to have been taught by himself.

I would also like to acknowledge Andritz Hydro, the company in Linz, Austria where this diploma thesis started. I would like to express my deep gratitude to my managers Peter Grafenberger and Simon Weissenberger for welcoming me and providing me with the opportunity to work in a real industry. I would also like to genuinely thank my supervisor there, Eugenia Kontoleontos who taught me valuable lessons about the industrial work environment. Our hours of discussing and problem solving taught me a way to think and react in such an environment. Also, our walks taught me how beautiful Linz is.

I would also like to deeply thank Kostas Tsiakas, who dedicated so many hours of his time, answering to my many questions. His software which I had the opportunity to work on in source form, is so carefully made that it made me understand what good programming looks like. I would also like to thank Dr. Xenofon Trompoukis and Dr. Varvara Asouti who both guided me and answered to my questions immediately although they are very busy. The whole research team of PCOpt/NTUA is always eager to help when needed.

Also, my friends and fellow students are always there when I need a break. Finally, I would like to thank my parents and my brother who love and support me in everything I do and they are indeed the greatest inspiration of all.

National Technical University of Athens
School of Mechanical Engineering
Fluids Department
Lab. of Thermal Turbomachines
Parallel CFD
& Optimization Unit

Programming of a "Back-to-CAD" Tool for Use in the Analysis and Optimization of Turbomachinery Rows. Industrial Applications

Diploma Thesis

Markella Zormpa

Supervisor: Kyriakos C. Giannakoglou, Professor NTUA

In this thesis, turbomachinery blade shape parameterization methods are further extended for use in an industrial design-optimization loop. Having the distinction between node-based blade representations (that use all CFD grid nodes as design variables) and CAD-based blade representations (that use geometrical quantities with a clear physical meaning as design variables, according to the method and tool developed by PCOpt/NTUA under the name GMTurbo) in mind, the software to bridge the two types of representation is programmed. The GMTurbo parameterization method is an indispensable part of an Evolutionary Algorithm optimization workflow, since it provides a complete representation of the blade using a limited amount of variables.

To be able to use the GMTurbo software in an optimization loop, even when only node-based representations are available, a Reverse Parameterization Tool (RPT), that performs the conversion from a node-parameterized to the equivalent GMTurbo-parameterized blade is programmed in this thesis, as an add-on to the GMTurbo software. In order to perform CFD simulations on the GMTurbo geometry, a surface Grid Adaptation Tool (GAT) is also programmed, its purpose being to adjust the initial surface CFD grid to the GMTurbo geometry, converting the GMTurbo representation back to the CFD grid representation.

To demonstrate the performance of the RPT and the GAT, comparisons on the geometry and CFD-results of the node-based blade shape and the equivalent reparameterized shape are performed in hydroturbine applications. Finally, Evolutionary Algorithm shape optimization of blades, the geometry of which is given in node-based form, is performed after converting them to the GMTurbo parameterization using the RPT. The GAT, is then integrated into the optimization workflow, its role being to generate a grid for each EA candidate blade geometry.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Εργαστήριο Θερμικών Στροβιλομηχανών
Μονάδα Παράλληλης ΥΡΔ & Βελτιστοποίησης

Προγραμματισμός ενός 'Πίσω-σε-CAD' Εργαλείου για την Ανάλυση και Βελτιστοποίηση Πτερυγώσεων Στροβιλομηχανών. Βιομηχανικές Εφαρμογές

Διπλωματική εργασία

Μαρκέλλα Ζορμπά

Επιβλέπων: Κ. Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Η διπλωματική αυτή εργασία αφορά μεθόδους παραμετροποίησης πτερυγίων στροβιλομηχανών. Η διάκριση μεταξύ των πλεγματικών αναπαραστάσεων και των CAD αναπαραστάσεων της μορφής γεωμετριών (εδώ πτερυγίων) γεφυρώνεται με την ανάπτυξη σχετικού λογισμικού. Η CAD αναπαράσταση που χρησιμοποιείται στη διπλωματική αυτή εργασία πραγματοποιείται με το λογισμικό GMTurbo το οποίο αναπτύχθηκε στο Εργαστήριο Θερμικών Στροβιλομηχανών του ΕΜΠ και στηρίζεται στις βασικές έννοιες των στροβιλομηχανών για την παραμετροποίηση πτερυγίων. Είναι ιδανικό για γέννεση της γεωμετρίας σε βελτιστοποίηση μορφής πτερυγίων με Εξελικτικούς Αλγορίθμους, αφού χρησιμοποιεί περιορισμένο αριθμό μεταβλητών σχεδιασμού για να περιγράψει το πτερύγιο.

Για να γίνει εφικτή η χρήση του GMTurbo, ακόμα και στις περιπτώσεις που μόνο το πλέγμα του πτερυγίου είναι διαθέσιμο, προγραμματίστηκε λογισμικό 'αντίστροφης παραμετροποίησης' (Reverse Parameterization Tool, RPT) που πραγματοποιεί τη μετατροπή του πλέγματος σε GMTurbo μορφή ως επέκταση στο λογισμικό GMTurbo. Για την πραγματοποίηση εφαρμογών Υπολογιστικής Ρευστοδυναμικής στην CAD γεωμετρία, αναπτύσσεται ένα λογισμικό 'προσαρμογής επιφανειακού πλέγματος' (Grid Adaptation Tool, GAT), το οποίο δημιουργεί πλέγμα γύρω από τη CAD γεωμετρία με κατάλληλη παραμόρφωση του αρχικού.

Για να αξιολογηθεί η πιστότητα τόσο του λογισμικού αντίστροφης παραμετροποίησης (RPT), όσο και του λογισμικού προσαρμογής επιφανειακού πλέγματος (GAT), συγκρίνονται οι γεωμετρίες και τα υπολογισθέντα ροϊκά μεγέθη μεταξύ της αρχικής και της παραμετροποιημένης γεωμετρίας. Τέλος, πραγματοποιείται η βελτιστοποίηση μορφής, με τη χρήση Εξελικτικών Αλγορίθμων, πτερυγίων των οποίων η γεωμετρία είναι διαθέσιμη σε μορφή πλέγματος, μετά την μετατροπή τους σε GMTurbo μορφή μέσω του RPT. Το λογισμικό προσαρμογής επιφανειακού πλέγματος (GAT) που προγραμματίστηκε χρησιμοποιείται για τη δημιουργία πλέγματος γύρω από κάθε γεωμετρία που δημιουργεί ο εξελικτικός αλγόριθμος.

Acronyms

CFD	Computational Fluid Dynamics
NTUA	National Technical University of Athens
PCOpt	Parallel CFD & Optimization unit
CAD	Computer-Aided Design
EA	Evolutionary Algorithm
LE	Leading Edge
TE	Trailing Edge
PS	Pressure Side
SS	Suction Side
IGV	Inlet Guide Vane

Contents

Contents **i**

1 Introduction **1**

1.1 Turbine Blades Representation 1

1.2 Optimization of Turbine Blades using EA 7

1.3 Structure of this Diploma Thesis 9

2 Basics of Parametric Geometry **11**

2.1 Introduction 11

2.2 Parameterization using Basis Functions 12

2.2.1 Power Basis Function 13

2.2.2 Bézier Curves 14

2.2.3 B-Splines 15

2.3 Non-Uniform Rational B-Splines(NURBS) 21

2.4 NURBS Surfaces 22

2.5 Fundamental Geometric Algorithms 24

3 Turbomachinery Blade Parameterization **25**

3.1 Meridional Plane 25

3.2 Mean Camber Line Parameterization 28

3.3 Blade Thickness Profiles 31

4 Reverse Parameterization Tool (RPT) **33**

4.1 Turbomachine Blade Grids 33

4.2 Reverse Parameterization of a Blade 34

4.2.1 Meridional Contour of the Grid 35
4.2.2 Mean Camber Line and Thickness Data Computation 36

5 Grid Adaptation Tool (GAT) 43
5.1 Description of the GAT Method 43
5.1.1 Step 1: Surface Grid Adaptation 44
5.1.2 Step 2: Volume Grid Adaptation 47
5.2 The GAT used in an EA Optimization 48

6 CFD Analysis 49
6.1 Hydraulic Turbomachines 49
6.2 The GPU-enabled CFD Solver PUMA 52
6.2.1 Flow Equations 52
6.2.2 The Spalart-Allmaras Turbulence Model 53
6.2.3 Boundary Conditions 54
6.3 Post-processing 56
6.3.1 Pressure Coefficient 56
6.3.2 Outlet Velocity Profile 56
6.3.3 Efficiency 58

7 Validation of the RPT and GAT and Optimization 59
7.1 Francis Runner 59
7.2 Propeller IGVs and Runner 62
7.3 Optimization 65
7.3.1 Guide Vane Optimization: 16 design variables 66
7.3.2 Guide Vane Optimization: 32 design variables 67
7.3.3 Guide Vane and Runner Optimization: 32 design variables . . 68

8 Overview and Conclusions 71
8.1 Overview 71
8.2 Proposals For Future Work 72

A Conformal Mapping **75**

B Surface Structured Grid-NURBS Surface Intersection **79**

C Constrained NURBS Approximation of Pressure and Suction Sides **83**

D Computation of the Mean Camber Line of an Existing Airfoil **87**

E Constrained Point Approximation with Cubic Bézier Curve **89**

Bibliography **93**

Chapter 1

Introduction

1.1 Turbine Blades Representation

Methods for the design and optimization of blades, based on aerodynamic or hydrodynamic criteria, is a key research topic in turbomachines. Turbomachinery blade's shapes vary, depending on the application, the operating conditions which they are designed for, manufacturing constraints etc. Thus, there are many types of turbomachines, to fit different operational requirements. They can be thermal or hydraulic, depending on the fluid that provides or absorbs energy. Thermal turbomachines are classified into two categories; compressors and turbines, while hydraulic turbomachines classify into pumps and turbines. These machines can be axial, radial or mixed flow, depending on the inlet and outlet flow direction. In the case of a multistage turbine or compressor, one has to design the shape of the blades for every stage. Turbomachines often involve Inlet Guide Vanes, that are stationary blades placed ahead of the first stage.

The above are just a few examples of the variety of turbomachinery blades used by the industry. Blade geometrical representation is essential for their design. Parameterization is the process of composing a geometry according to an algorithm, by firstly determining a set of design variables that correspond to the input to this algorithm. Different sets of parameters produce different shapes. There are various ways to parameterize a blade which, from a certain point of view, can be classified into **node-based** and **CAD-based** methods.

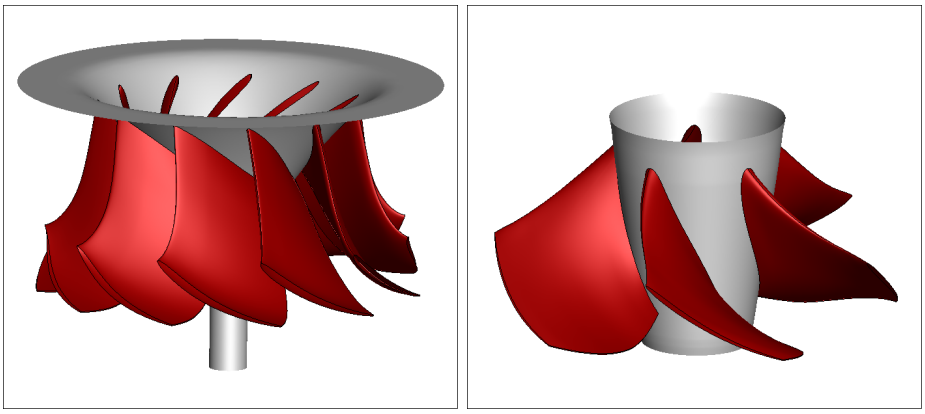


Figure 1.1: *Francis and Propeller Turbine runners.*

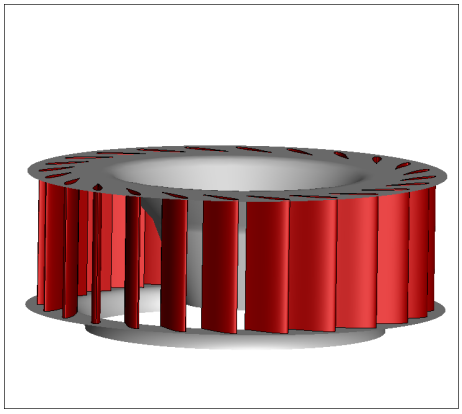


Figure 1.2: *IGV stationary blades.*

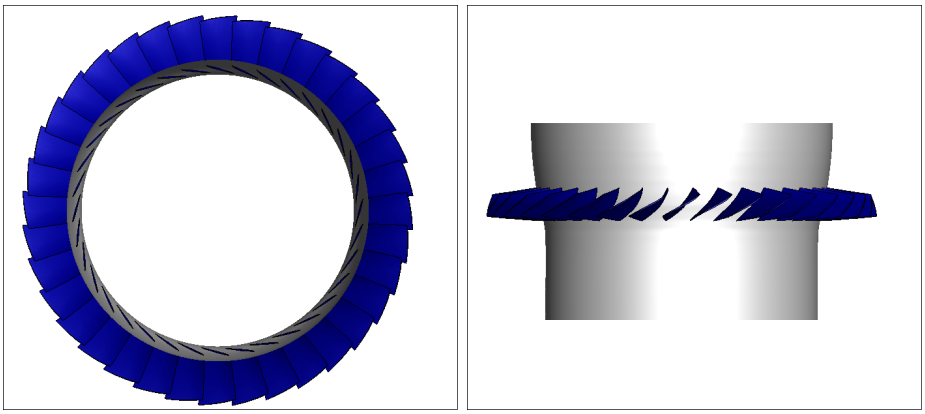


Figure 1.3: *A compressor stage with 36 blades.*

Node-based methods consider the blade as a surface grid that consists of distinct nodes given by their Cartesian coordinates and the connectivity between them. This

representation, although it offers the possibility to directly proceed to CFD simulations, is rather inconvenient in an optimization loop. Shape optimization using node-based representation may produce rough shapes and thus, invalid geometries. A smoothing technique is, thus, necessary to smooth these irregularities out.

Alternatively, CAD-based representation is employed. This type of representation describes the blade using parametric geometry. A blade can be described with one (or more) NURBS surfaces which are controlled by their control points. Such a representation can be modified or optimized by displacing the control points. The number of design variables is significantly reduced in comparison to node-based methods, while the use of parametric geometry guarantees a smooth result. However, such CAD-based methods have to be followed by a surface grid generator or morpher before generating the 3D CFD grid and running a CFD simulation.

To enforce the parameterization with physical concepts, intuitive algorithms are developed that use notions from the theory of turbomachines, such as the meridional contour, metal angles, thickness profiles, etc. In terms of optimization, this type of parameterization provides the design variables with a geometric meaning, therefore during a stochastic optimization, the designer can further reduce the number of the design variables, by choosing to modify the parameters that have the greatest impact on the objective function. For gradient-based optimization, the parameterization algorithm must provide the derivatives of grid nodal positions w.r.t the design variables, in order to compute sensitivity derivatives. This is done by the chain rule as follows

$$\frac{\delta F}{\delta b_n} = \underbrace{\frac{\delta F}{\delta x_k^i}}_{\substack{From \\ Adjoint}} \cdot \underbrace{\frac{\delta x_k^i}{\delta b_n}}_{\substack{From\ the \\ differenatiation \\ of\ the\ parameterization \\ model}} \tag{1.1}$$

Parameterization methods that construct the blade using parametric geometry and turbomachinery notions are used extensively in blade design, because they combine a strong geometric sense while introducing a small number of design variables that is ideal for the optimization.

For instance, [1] proposes the following parameterization method:
A blade can be constructed by first defining its meridional contour, then its mean camber lines in hub and shroud and finally, superposing thickness to the mean camber lines interpolated surface, resulting to the final blade. This method considers the meridional contour to be formed by 7 patches that are defined by 7 Bézier or B-spline curves along the hub and 7 along the shroud (fig. 1.4). Each Bézier curve has 3 control points. The blade lies on patch 4 of the meridional contour.

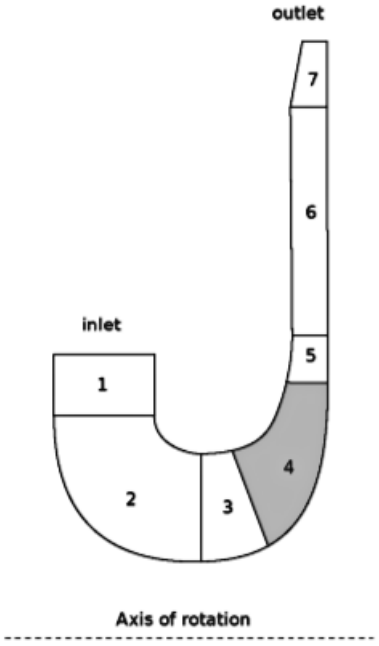


Figure 1.4: *The parameterization of the meridional contour according to [1].*

Next step is the definition of the two mean camber lines that lie on the meridional revolved surfaces of hub and shroud. These are defined by a cubic Bézier curve of the $\beta(u)$ angle distribution from leading edge (LE) (u=0) to trailing edge (TE) (u=1) that is the angle between the meridional plane and the tangent to the camber line at each streamwise position (fig. 1.5). To place this mean camber line in a circumferential position, angles $\theta = atan(\frac{y}{x})$ of the LE or TE have to be defined for hub and shroud.

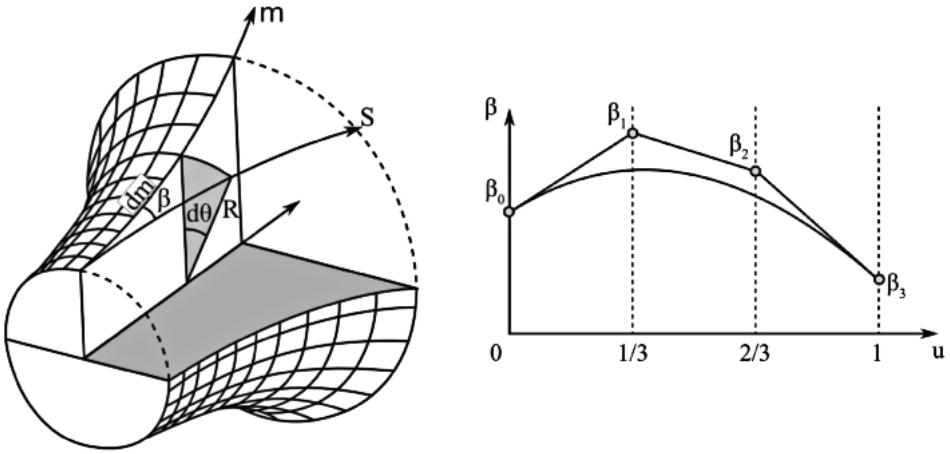


Figure 1.5: *The 3D surface of a meridional curve (left) and the definition of angle $\beta(u)$ distribution(right). From [1].*

Interpolation of the two mean camber lines leads to a thickless blade. Imposition of a thickness profile on this blade results the final blade. The thickness profile definition is seen in fig. 1.6 .

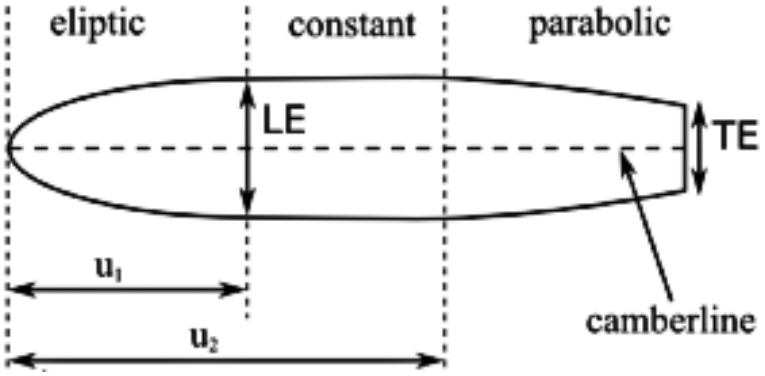


Figure 1.6: *Definition of the thickness profile. It consists of three parts, the eliptic the constant and the parabolic. u_1 and u_2 are measured along the camberline. From [1].*

The method presented for parameterizing blade shapes is the ancestor of the method this diploma thesis is based upon. This method, with some improvements made in the meantime, is implemented in the parameterization software GMTurbo, which is developed and used by the PCOpt/NTUA [2, 3]. Briefly, this software takes the necessary input values, that are the design variables of the blade, and returns the NURBS surface representations of the two sides of a 3D blade, pressure and suction, in a CAD compatible form (IGES format) (fig. 1.7).

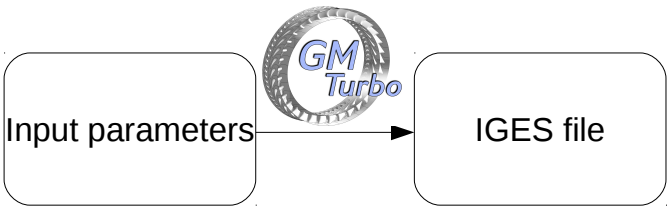


Figure 1.7: *The GMTurbo generates the CAD compatible geometry of the blade, using the necessary parameters.*

Having already emphasized on the value of intuitive parameterization methods in optimization, the need of a reverse parameterization tool compatible to GMTurbo rises. The Reverse Parameterization Tool (RPT) that is developed in this diploma

thesis as an add-on tool to the GMTurbo software, is a code that can translate a CFD grid of a turbomachine blade into a set of geometric parameters; compatible with the GMTurbo input parameters (figure 1.8). This conversion, from CFD grid to GMTurbo parameters, is needed if a blade CFD grid already exists and modifications (e.g. shape optimization) to the geometry have to be made without using any node-based parameterization or smoothers.

The case where the geometry is provided only as a CFD grid is very common. The design of a blade could have been carried out using parameterization tools that exist in the industry but are not compatible with each other, thus non exchangeable. Therefore, the standard data format when exchanging geometries, is the CFD grid. The information grids provide is very precise (connectivity, coordinates, boundary patches), so they are exchangeable, since the only difference is in the format and not the information each grid includes. Such format conversions are performed using grid transformation software, that exists at PCOpt/NTUA. Since grids are extensively used to represent the shape of blades in turbomachines, the implementation of the RPT within the parameterization algorithm, which is the subject of this diploma thesis, is a useful tool which may become an indispensable part of an optimization that uses GMTurbo as the geometry generation software.

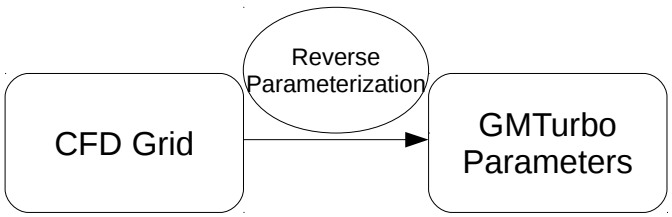


Figure 1.8: *The RPT uses the information provided by the CFD Grid in order to compute the parameters of GMTurbo.*

To be able to perform CFD simulations around the GMTurbo blade, a mesh has to be created around the geometry. In this thesis, a surface Grid Adaptation Tool (GAT) is developed that adapts the initial surface grid to the reparameterized geometry, using the 2D spring analogy technique developed by PCOpt/NTUA in [4] (fig. 1.9). The volume grid results from the deformation of the initial volume grid, in order to make it fit to the adapted surface grid provided by GAT. The 3D deformation is carried out using the 3D spring analogy technique also developed by PCOpt/NTUA [4]. That way the complete reparameterization is achieved and CFD runs can be performed to the geometry that is now available in both CAD and grid format.

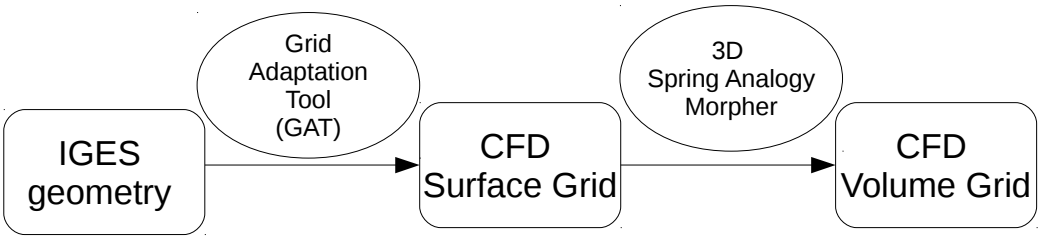


Figure 1.9: *To create a grid around the reparameterized surface, the GAT is developed that adapts the initial surface grid to the CAD geometry. Then the volume grid is displaced to fit to the adapted surface grid, using the 3D spring analogy technique.*

1.2 Optimization of Turbine Blades using EA

An evolutionary algorithm (E.A.) is a software, implementing a bio-inspired search method that optimizes a set of design variables w.r.t one or more target function(s), computed by an evaluation software. The software used in this diploma thesis, is the EASY (Evolutionary Algorithms SYstem) platform, developed at PCOpt/NTUA [5]. The evaluation code is a combination of the parameterization tool, the grid morpher, the CFD solver and the post processor code that computes the objective values from the results of the CFD software (fig. 1.10).

In this thesis, the optimization (using the GMTurbo parameterization) of a blade, the geometry of which is given as a CFD grid, w.r.t. some objective functions, is made possible. Before the optimization begins, the RPT programmed in this diploma thesis (in C++) generates the GMTurbo parameterization that describes the given node-based geometry. After specifying a set of the GMTurbo parameters as design variables, the optimization is set. The sequence of tasks of fig. 1.10, is called for every candidate geometry. In this diploma thesis, a surface grid displacement technique is also programmed (in C++) in order to be able to perform CFD simulations on the new, slightly different geometry. Also, post-processors of the CFD results are programmed (in Fortran 77) to compute the objective function values used by EASY.

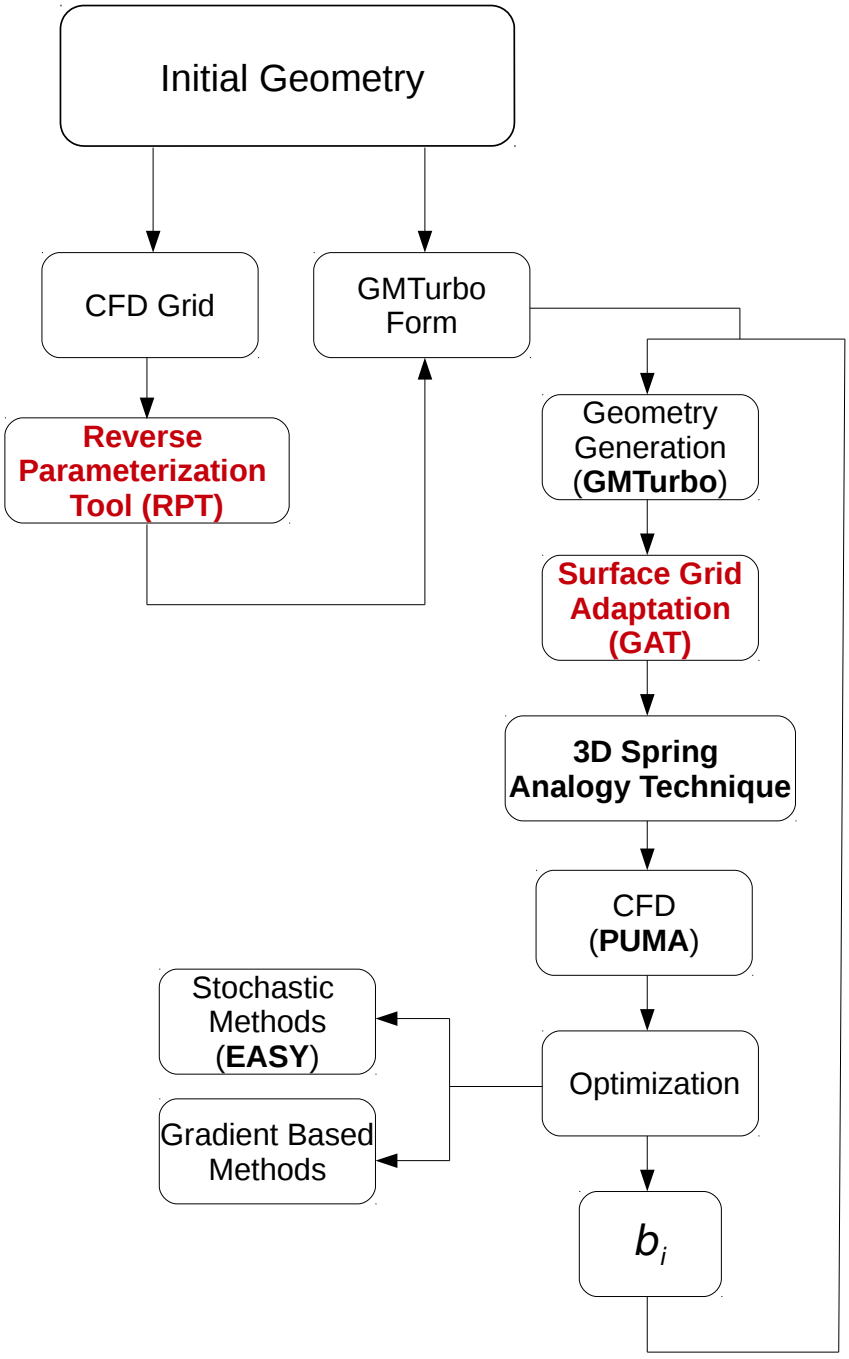


Figure 1.10: *Shape optimization of a turbomachinery cascade. Flowchart of the successive tasks that should be accomplished during one evaluation of a single individual (if stochastic optimization is performed) or an optimization cycle (if gradient methods are used). The initial geometry can be provided either in GMTurbo format or in CFD grid format (or in different formats that are not discussed in this thesis). In the rather common case where the CFD grid is available, the pre-processing of the optimization includes the conversion of the geometry from grid format to GMTurbo format. The software written in bold represents the integrated tools of PCOpt/NTUA. The software written in red corresponds to the steps that were programmed in this thesis to make the optimization possible.*

1.3 Structure of this Diploma Thesis

The chapters of this diploma thesis are the following:

- **Chapter 2** Introduction to parametric geometry, focusing on NURBS Curves and Surfaces, that is the main parametric representation used by GMTurbo.
- **Chapter 3** Presentation of the GMTurbo software. The parameterization method used by the PCOpt/NTUA is explained in detail.
- **Chapter 4** Presentation of the RPT that was programmed.
- **Chapter 5** Presentation of the GAT developed and programmed in this diploma thesis, as well as its incorporation into the optimization workflow.
- **Chapter 6** Presentation of the PUMA CFD solver used in this thesis and the post processing of the results.
- **Chapter 7** Applications of the RPT and GAT are presented, in order to demonstrate the method and software performance. Three shape optimizations are carried out, proving that the RPT and GAT are successfully integrated in the optimization procedure.
- **Chapter 8** Conclusions and ideas for future work.

Chapter 2

Basics of Parametric Geometry

2.1 Introduction

Geometry representation has always been a fundamental part of the designing procedure of machines. The different components of the machine can be seen as curves or surfaces in the $2D$ or $3D$ space. It is essential that a convenient method of representation is used to make geometry handling as flexible as possible.

The two main methods of representing a curve or a surface are *implicit equations* and *parametric functions*. The following analysis focuses on curves but the same definitions and properties, with just a few modifications, apply to surfaces as well.

The implicit or cartesian equation of a curve in the xy-plane is given by

$$f(x, y) = 0 \tag{2.1}$$

The same curve's parametric function is

$$\mathbf{C}(u) = (x(u), y(u)), a \leqslant u \leqslant b \tag{2.2}$$

where x and y are both functions of a single parameter u.

Parametric Geometry uses a single parameter to describe a curve and two parameters to describe a surface. It also has a great geometric sense as it uses polynomials that have certain useful properties. For the implementation of parametric curves and surfaces, algorithms that perform geometric operations are found in the literature. Some of the reasons why parametric representation is much more preferable than

implicit representation, during blade design and optimization, are listed below:

- Adding a third coordinate z to an arbitrary curve is a natural extention to eq. 2.2 of the parametric representation. On the other hand, in implicit representations, arbitrary $3D$ curves can only be mapped on the xy-plane or yz-plane. In aerodynamic or hydrodynamic simulations, the use of $3D$ geometries is fundamental, therefore the use of parametric geometry becomes indispensable.

- Cartesian representation of bounded curves or surface patches require the specification of an interval for every curve. On the other hand, the representation of bounded segments, is built in the definition of parametric representation, just by restricting the parameter u to a predefined $[a, b]$ interval.

- Parametric representation provides a natural direction for the curve and a rate of traversal, as u goes from a to b. This information is not produced by the cartesian representation but it is important to have it, because it provides an ordered sequence to the points of the curve which is really useful when programming software implementations of parametric geometry and gives the designer a sense of the position of each point along the curve.

- Parametric representation is associated with meaningful geometrical properties. The valuable properties, such as continuity along the curve, the fact that control points pull the curve towards them, the convex hull property and others that are presented in detail in this chapter, provide a better view of the geometry to the designer. Aerodynamic and hydrodynamic shape optimization using evolutionary algorithms would not be possible using implicit representation, because the selection of different geometries is stochastic, thus the design variables may take on values that expose the aerodynamic and hydrodynamic shape to unexpected deformations, that lead to discontinuity of the shape or violate the manufacturing constraints. Using parametric representations, the designer is able to apply deformations that always produce smooth results.

This chapter is an introduction to the basic definitions of parametric geometry that are going to be used extensively in the parameterization and reversed parameterization algorithms that will be presented in the following chapters.

2.2 Parameterization using Basis Functions

In eq. 2.2, it is obvious that letting the variables $x(u), y(u), z(u)$ be any function of u, a wide range of curves can be produced. Restricting the functions $x(u), y(u), z(u)$ to a specific form may reduce the flexibility but, on the other hand, provides a number of valuable attributes that come from the mathematical properties of these

functions. These functions are given by the equation

$$\mathbf{C}(u) = \sum_{i=0}^{n} F_i(u)\mathbf{a_i} \tag{2.3}$$

where $F_i(u)$ are basis functions, that vary according to the desired properties of the parametric representation, and $\mathbf{a_i}$ are the coefficients of the basis functions.

2.2.1 Power Basis Function

Polyonomials is a powerful tool in the parameterization method. The simplest polynomial basis function, that is hardly ever used because of its poor properties and geometric meaning, is the power basis function $F_i(u) = u^i$, that forms the N$^{\text{th}}$ degree parametric curve

$$\mathbf{C}(u) = \sum_{i=0}^{N} u^i\mathbf{a_i} = \mathbf{a_0} + \mathbf{a_1}u + \mathbf{a_2}u^2 + ... + \mathbf{a_N}u^N \tag{2.4}$$

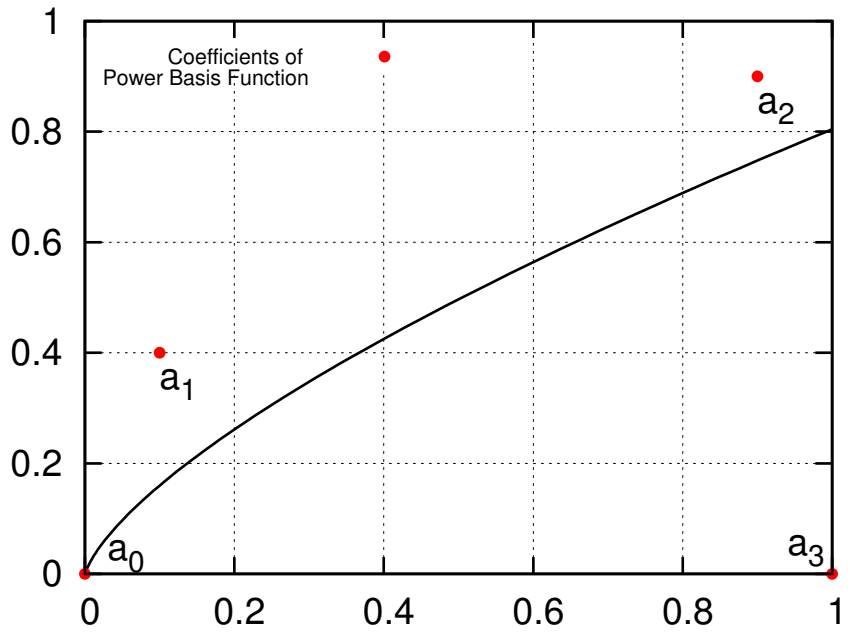


Figure 2.1: *Polynomial* $\mathbf{a_0} + \mathbf{a_1}u + \mathbf{a_2}u^2 + \mathbf{a_3}u^3$ *in black line. The coefficients, in the form of "control" points appear in red but they do not hold an obvious geometric meaning.*

2.2.2 Bézier Curves

Power basis functions have a purely algebraic sense and do not provide any geometric sense to the parameterization. This weakness is overcome by the use of other polynomials as basis functions namely the Bernstein polynomials, that are given by the formula:

$$B_i^N(u) = \frac{N!}{i!(N-i)!} u^i (1-u)^{N-i} \tag{2.5}$$

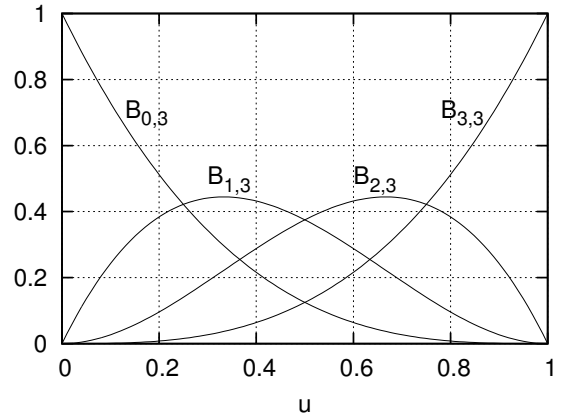


Figure 2.2: *3^d degree Bernstein Polynomial.*

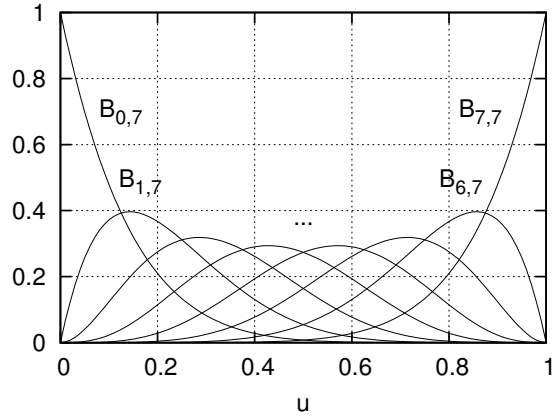


Figure 2.3: *7^{th} degree Bernstein Polynomial.*

Forming the N^{th} degree Bézier curve:

$$\mathbf{C}(u) = \sum_{i=0}^{N} B_i^N(u) \mathbf{P_i} \tag{2.6}$$

The $\mathbf{P_i}$ coefficients are referred to as Control Points or Bézier Points because, according to the basis function 2.5, they act like "magnets" to the curve, controlling its shape. The representation of a Bézier curve can also be explained, purely geometrically, based on the recursive De Casteljau's algorithm [6].

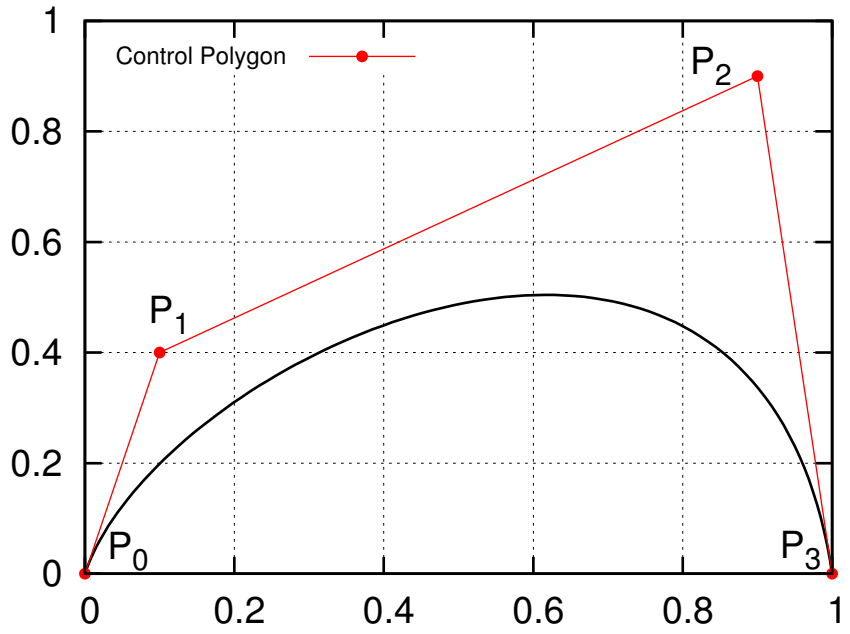


Figure 2.4: *Bézier Curve with 4 control points, the same points that are used as coefficients in the curve of figure 2.1. Herein the geometric meaning of the control points is obvious. The first and last points are interpolated by the curve, while the intermediate points pull the curve towards them.*

2.2.3 B-Splines

The above mentioned parametric representations use a single polynomial along the $[a, b]$ interval. Alternatively, partially continuous polynomials can be used, that is segments of parametric curves of different degree and basis function, defined in subsets of the interval $[a, b]$ (e.g. $[a, u_0]$,$[u_0, u_1]$, ... , $[u_m, b]$) that maintain continuity at the breakpoints $u_0, u_1, .., u_m$ (figure 2.5). This extension of the definition of the Polynomial Basis Function curves yields the definition of the B-Splines curves. Their distinguishing feature is that they provide local support, as the functions are defined in specific intervals. Therefore, any displacement of the control points affects only a certain interval, providing better handling and accuracy to the resulting curve. Furthermore, lower degree is required in order to interpolate or approximate N points (in contrast to Bézier curves where the degree of the polynomial has to be exactly $N - 1$).

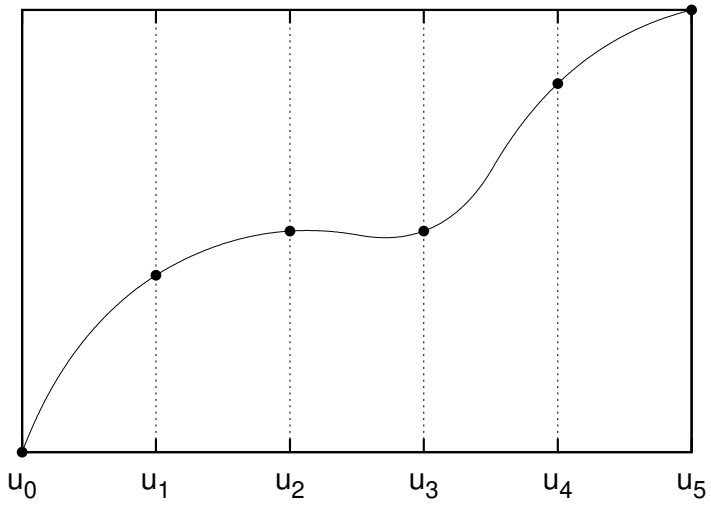


Figure 2.5: *Partially continuous curve. Parameter values where the function changes, are called breakpoints and are given in the form of vector* $\mathbf{U} = [u_0, u_1, u_2, u_3, u_4, u_5]$ *which is called knot vector.*

Knot Vectors:

A knot vector is an array that contains the breakpoints of a partially continuous polynomial in increasing order. In figure 2.5, the knot vector is $\mathbf{U} = [u_0, u_1, ..., u_m]$ and determines the shape of the curve. A knot vector can be *Uniform*, *Open-Uniform* or *Non-Uniform*. Uniform knot vectors are the vectors for which $u_{i+1} - u_i = const.$ (e.g. $\mathbf{U} = [0, 1, 2, 3, 4, 5]$). Open-Uniform knot vectors are Uniform knot vectors that have k-equal knot values at each end, as in

$$\begin{array}{rr} u_i = u_0, & i < k \\ u_{i+1} - u_i = const., & k - 1 \leqslant i < m + 1 \\ u_i = u_m, & i \geqslant m + 1 \end{array} \tag{2.7}$$

For $k = 3$ and $m = 5$ a valid knot vector is $\mathbf{U} = [0, 0, 0, 1, 2, 3, 4, 4, 4]$. Lastly, Non-Uniform knot vectors is the most general case. The only constraint is the typical one: $u_i \leqslant u_{i+1}$.

An important property of a knot vector is the effect of the multiplicity of a knot u_i in the vector $\mathbf{U}$. The multiplicity of u_i, is the number of times this knot is found in $\mathbf{U}$ and it is associated with the differentiability of the function at u_i.

B-Spline Basis Function:

The basis function of a p^{th} degree B-Spline Curve with a knot vector $\mathbf{U} = [u_0, u_1, ..., u_m]$

is given by the recursive formula:

$$N_{i,0}(u) = \begin{cases} 1, & u_i \leqslant u < u_{i+1} \\ 0, & \text{otherwise} \end{cases}$$
$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2.8}$$

Every basis function $N_{i,p}$ is a linear combination of functions $N_{i,p-1}$ and $N_{i-1,p-1}$ of a degree $p-1$, with weights depending on the knot vector $\mathbf{U}$.

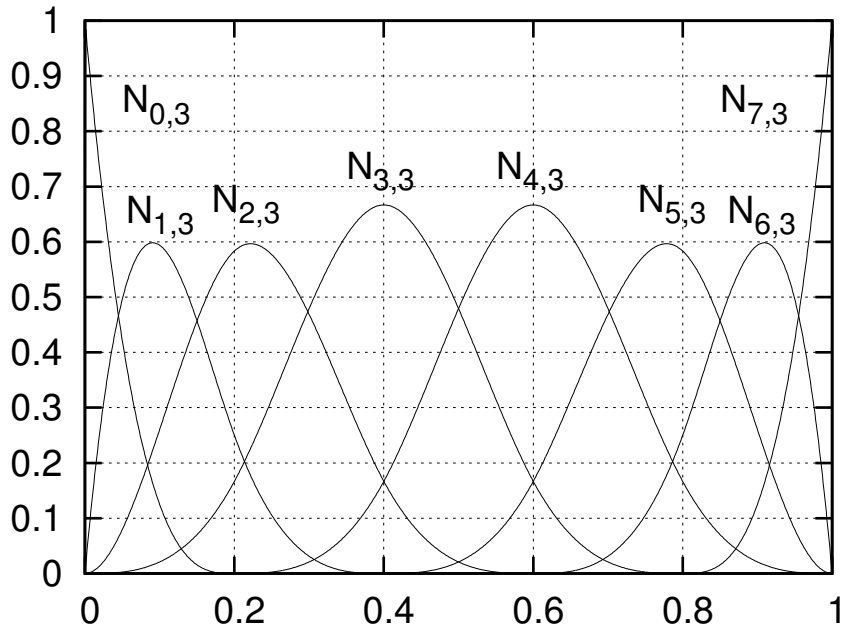


Figure 2.6: *Third-degree B-Spline basis functions with a knot vector* $\mathbf{U} = [0, 0, 0, 0, 0.2, 0.4, 0.6, 0.8, 1, 1, 1, 1]$.

The properties of the B-Spline Basis Functions that result from their definition, eq. 2.8, are the following:

- Function $N_{i,p}$ is non-zero for $u \in [u_i, u_{i+p+1})$. For example, in figure 2.6 $N_{3,3}$ is active only in the interval $[u_3, u_7]) = [0, 0.8)$
- At any knot span $[u_j, u_{j+1})$, the non-zero basis functions are at most p+1,namely $N_{j-p,p}, ..., N_{j,p}$. In figure 2.6, in the interval $[u_3, u_4]) = [0, 0.2)$ only functions $N_{0,3}, ..., N_{3,3}$ are active.
- $N_{i,p} \geqslant 0$ for every i,p and u.

- $\sum_{j=i-p}^{i} N_{j,p} = 1$ for any $u \in [u_i, u_{i+1})$.
- All the derivatives of $N_{i,p}$ exist in the interior of a knot span $[u_i, u_{i+1})$. At knot u_i, function $N_{i,p}$ is $p - k$ times continuously differentiable, where k is the multiplicity of the knot (figure 2.7).

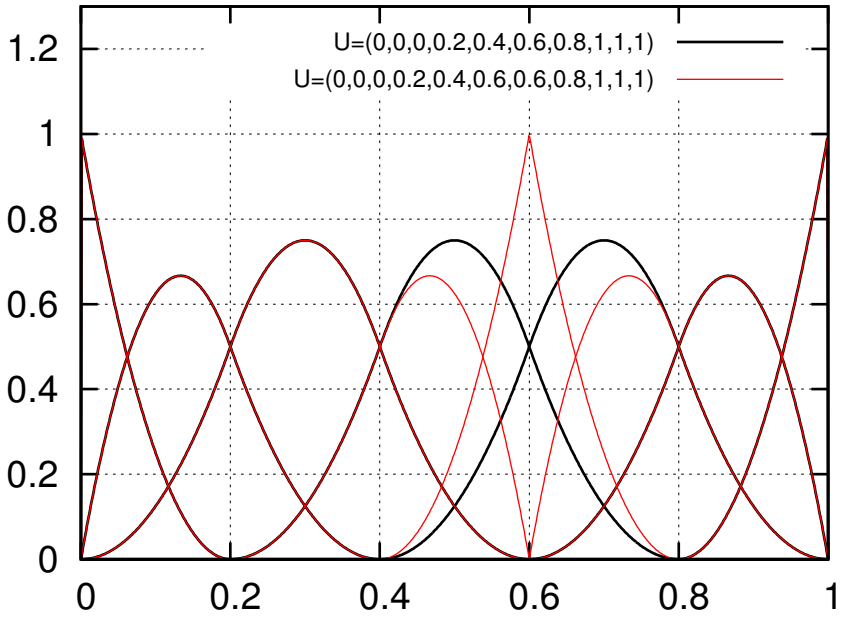


Figure 2.7: *Set of 2^{nd} B-Spline basis functions. One set of basis functions has multiplicity $k = 1$ for $u = 0.6$ and the other has $k = 2$ at the same knot. The effect of multiplicity is obvious as differentiability at this knot decreases while multiplicity increases.*

- $N_{i,p}$ has exactly one maximum value for $p \geqslant 0$.
- The B-Spline basis function with a knot vector

$$\mathbf{U} = [\underbrace{0, 0, 0}_{p+1}, \underbrace{1, 1, 1}_{p+1}] \tag{2.9}$$

  yields the Bernstein polynomials.
- If $m + 1$ is the number of the knots in the knot vector $\mathbf{U} = [u_0, ..., u_m]$, $n + 1$ the number of the $N_{i,p}$ functions and p the degree of the polynomials, then it is easily proved that

$$n = m - p - 1 \tag{2.10}$$

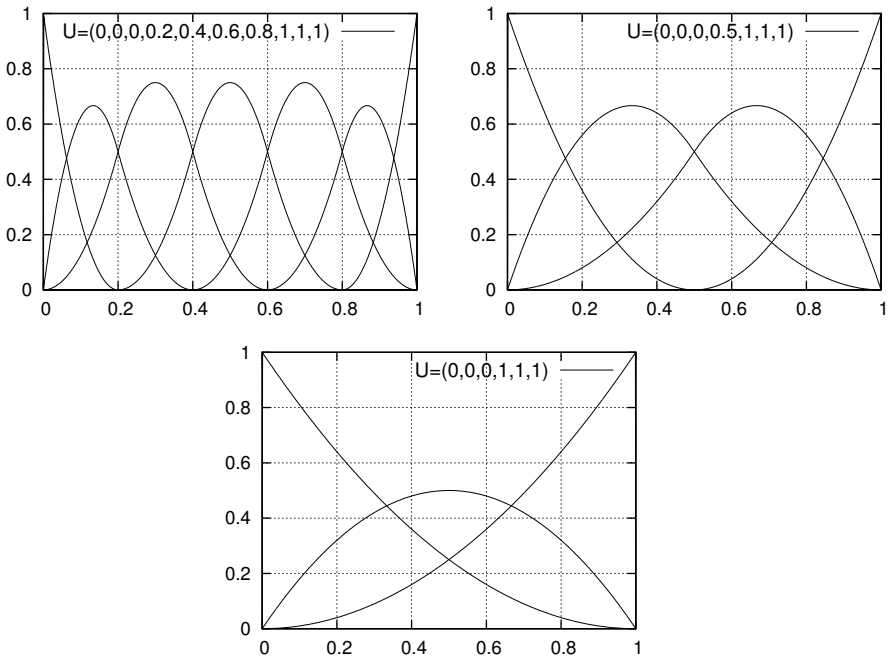


Figure 2.8: *The effect of the knot vectors to the B-Spline basis functions. Three sets of 2^{nd} degree basis functions are presented. At each breakpoint a basis function becomes zero and another one takes non-zero value. Therefore, the basis functions act like switches that activate and deactivate at the breakpoints.*

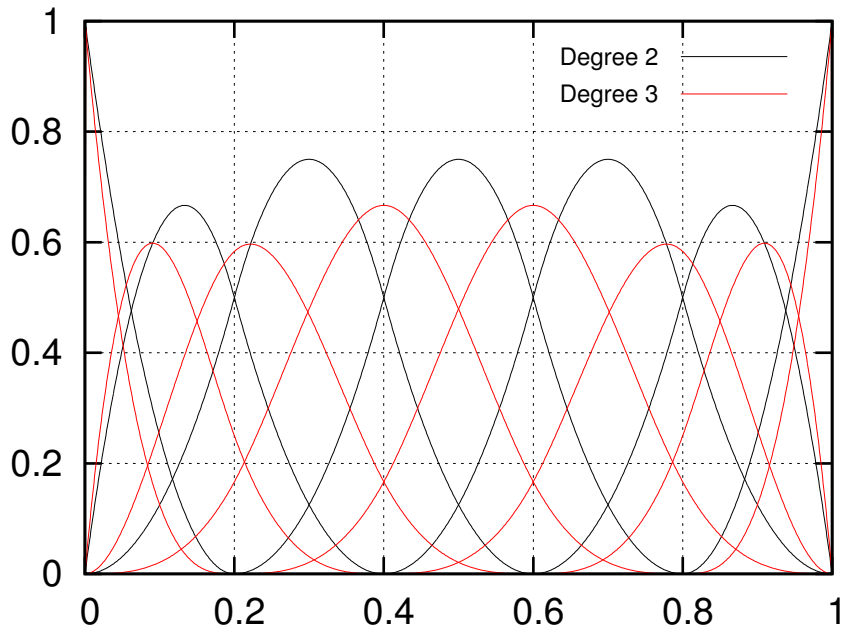


Figure 2.9: *The effect of the polynomial degree to the B-Splines basis functions. The shape of each function obviously changes as the degree changes, as well as the number of functions according to eq. 2.10.*

B-Spline Curve

A B-Spline curve of degree p is given by the formula:

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P_i} \tag{2.11}$$

that uses the polynomials of eq. 2.8 as basis functions.The knot vector is a Non-Uniform vector:

$$\mathbf{U} = [\underbrace{a, a, a}_{p+1}, u_{p+1}, ..., u_{m-p-1}, \underbrace{b, b, b}_{p+1}] \tag{2.12}$$

Usually $a = 0$ and $b = 1$. According to this definition, the properties of the B-Splines Curves are:

- Endpoint Interpolation: $\mathbf{C}(0) = \mathbf{P_0}$ and $\mathbf{C}(1) = \mathbf{P_n}$.
- Affine Invariance: An affine transformation is applied to the curve by applying it to its control points.
- Strong Convex Hull property: The curve lies into the convex hull of it's control points. Moreover for $u \in [u_i, u_{i+1})$, $\mathbf{C}(u)$ is contained in the convex hull of the points $\mathbf{P_{i-p}}, ..., \mathbf{P_i}$.
- Local Modification Schemes: Displacement of the control point $\mathbf{P_i}$ effects $\mathbf{C}(u)$ only in the interval $[u_i, u_{i+p+1})$.
- Moving along the curve from $u = 0$ to $u = 1$, the basis functions $N_{i,p}$ act like switches. As u passes u_i, the function $N_{i-p-1,p}$, which is the coefficient of $\mathbf{P_{i-p-1}}$ takes zero value, cancelling the effect of this control point, while $N_{i,p}$ takes a non-zero value, activating the effect of $\mathbf{P_i}$.
- Variation Diminishing property: There is no line that intersects the curve more times than it intersects its control polygon.
- Given that $\mathbf{C}(u)$ is a linear combination of the basis functions $N_{i,p}$ the continuity and differentiability follows that of the basis functions. Hence, in the interior of a knot span, $\mathbf{C}(u)$ is infinitely differentiable, while at a knot of multiplicity k, it is at least $p - k$ times continuously differentiable.

Rational B-Spline Curve:

Polynomials as basis functions are certainly well suited for parametric curves and surfaces. However, they lack the ability of exact representation of some geometric shapes, such as conics circle, ellipse, hyperbola. These curves are widely used during the design of an engine, so the need of a parameterization that can represent them is essential. This need leads to the definition of the Rational B-Spline curves that are presented in detail in section 2.3.

2.3 Non-Uniform Rational B-Splines(NURBS)

A NURBS Curve, as the acronym implies, is a Rational B-Spline curve, with a Non-Uniform knot vector. Therefore, it is given by

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P_i}}{\sum_{i=0}^{n} N_{i,p} w_i}, \quad a \leqslant u \leqslant b \tag{2.13}$$

where $N_{i,p}$ the B-Spline basis functions.

Control points $\mathbf{P_i}$, weights w_i and the non-uniform knot vector

$$\mathbf{U} = [\underbrace{a, a, a}_{p+1}, u_{p+1}, ..., u_{m-p-1}, \underbrace{b, b, b}_{p+1}] \tag{2.14}$$

can vary, providing a variety of curves. Alternatively, the curve is described by eq. 2.15 and the basis functions by eq. 2.16,

$$\mathbf{C}(u) = \sum_{i=0}^{n} R_{i,p} \mathbf{P_i} \tag{2.15}$$

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j} \tag{2.16}$$

The properties of a NURBS curve have already been partly mentioned as properties of B-Spline curves; however, these are repeated briefly for completeness:

- $R_{i,p}(u) \geqslant 0$ for every $i, p, u \in [0, 1]$
- $\sum_{i=0}^{n} R_{i,p}(u) = 1$ for every $u \in [0, 1]$
- $R_{0,p}(0) = R_{n,p}(0) = 1$
- For $p > 0$, $R_{i,p}(u)$ has exactly one maximum value for $u \in [0, 1]$
- Local support: $R_{i,p}(u) = 0$ for $u \notin [u_i, u_{i+p+1})$. This leads to the ability of local modifications. Displacing the control point $\mathbf{P_i}$ or changing the weight w_i effects only the part of the curve where $u \in [u_i, u_{i+p+1})$.
- All the derivatives of $R_{i,p}(u)$ exist in the interior of a knot span. For $u = u_i$ the function is $p-k$ times conituously differentiable, where k is the multiplicity of the knot. The same applies to $\mathbf{C}(u)$, being a linear combination of $R_{i,p}(u)$.
- If $w_i = 1$ for every i, $R_{i,p}(u) = N_{i,p}$, the NURBS curve yields a B-Spline curve.

- Endpoint Interpolation: $\mathbf{C}(0) = \mathbf{P_0}$ and $\mathbf{C}(1) = \mathbf{P_n}$.
- Affine Invariance
- Strong Convex Hull property
- Variation Diminishing property
- A NURBS curve without internal knots becomes a Bézier Curve. It is now obvious how the NURBS curve is the most general parametric representation since, by choosing appropriate parameters (weights and knot vectors), one can achieve a Bézier or B-Spline curve, using the basis function of the NURBS curve.

2.4 NURBS Surfaces

The ability of representing whole surfaces in space is crucial in a $3D$ design. Curve $\mathbf{C}(u)$ is a vector-valued function of one parameter. It is a mapping of a straight line segment $u \in [0, 1]$ into the $3D$ space. Surface $\mathbf{S}(u, v) = (x(u, v), y(u, v), z(u, v))$ is a two parameter vector-valued function that maps a (u, v) plane to the $3D$ space.

For every aforementioned parametric representation of a curve, the definition can expand to a $\mathbf{S}(u, v)$ Bézier, B-Splines and NURBS surface. Only NURBS surfaces are analyzed here since, as in curves, their basis function contains all the other parametric surfaces basis functions by definition.

A NURBS surface is given by the formula:

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P_{i,j}}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}}, 0 \leqslant u, v \leqslant 1 \tag{2.17}$$

where $\mathbf{P}_{i,j}$ are the grid-like control points, $w_{i,j}$ are the weights and $N_{i,p}(u)$ and $N_{j,p}(v)$ are the Non-Uniform Rational B-Spline Basis Functions with the knot vectors:

$$\mathbf{U} = [\underbrace{0, 0, 0}_{p+1}, u_{p+1}, ..., u_{m-p-1}, \underbrace{1, 1, 1}_{p+1}] \tag{2.18}$$

$$\mathbf{V} = [\underbrace{0, 0, 0}_{q+1}, v_{p+1}, ..., v_{m-p-1}, \underbrace{1, 1, 1}_{q+1}] \tag{2.19}$$

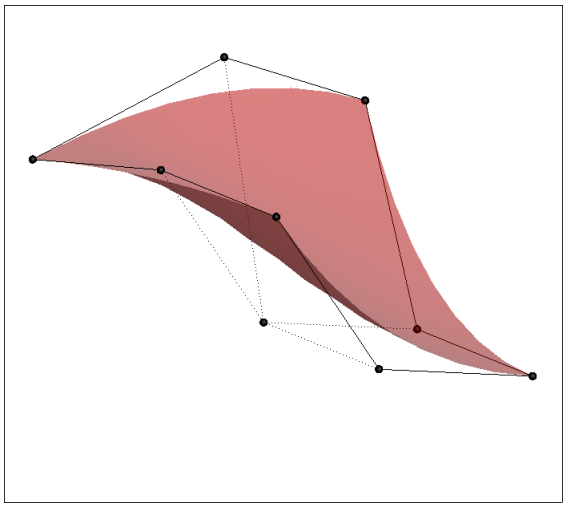


Figure 2.10: *A NURBS Surface with 9 control points and knot vectors* $\mathbf{U} = [0,0,0,1,1,1]$, $\mathbf{V} = [0,0,0,1,1,1]$

Alternatively, eq. 2.20 is used and the basis functions change to eq. 2.21

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}(u,v)\mathbf{P_{i,j}} \tag{2.20}$$

$$R_{i,j}(u,v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{k=0}^{n}\sum_{l=0}^{m} N_{k,p}(u)N_{l,q}(v)w_{k,l}} \tag{2.21}$$

The properties of the NURBS surface are analogous to the ones of the NURBS curve:

- $R_{i,j}(u,v) \geqslant 0$ for every i, j, u and v.
- $\sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}(u,v) = 1$ for $(u,v) \in [0,1] \times [0,1]$
- Local Support. $R_{i,j}(u,v) = 0$ if $(u,v) \notin [u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$
- In any region $[u_{i0}, u_{i0+1}) \times [v_{j0}, v_{j0+1})$ at most $(p+1)(q+1)$ basis functions are non-zero,moreover $R_{i,j}(u,v) \neq 0$ for $i_0 - p \leqslant i \leqslant i_0$ and $j_0 - p \leqslant j \leqslant j_0$
- If $p > 0$ and $q > 0$ then $R_{i,j}(u,v)$ attains exactly one maximum value.
- $R_{0,0}(0,0) = R_{n,0}(1,0) = R_{0,m}(0,1) = R_{n,m}(1,1) = 1$, which means that the control points on the four corners are interpolated by the surface.
- Affine Invariance
- Strong Convex Hull property
- Local Modification Scheme

- $\mathbf{S}(u, v)$ is $p - k$ times differentiable with respect to u at the knot u_i of multiplicity k and $q - k$ times differentiable with respect to v at the knot v_i of multiplicity k.

2.5 Fundamental Geometric Algorithms

Computational operations on parametric geometries are achieved through a series of algorithms that can be found in literature. These algorithms perform various tasks to the geometry and can be programmed and used to simplify the implementation of NURBS curves and surfaces in programming codes. They are used extensively in this thesis, during the programming of the RPT and GAT.

- **Interpolation of a set of points with NURBS curves.** Computation of the control points of a NURBS curve of a user defined degree, so that the NURBS curve passes through the points.
- **Approximation of a set of points with NURBS curves.** Computation of a user-defined number of control points of a NURBS curve of a user defined degree, so that the NURBS Curve is as close to these points as possible.
- **Approximation of a set of points with NURBS curves using Constraints.** Sometimes, constraints have to be applied during approximation, so that the resulting curve satisfies some requirements, such as point interpolation or derivative specification.
- **Local Nonrational Cubic Curve Approximation.** Approximation of a set of points with a cubic Bezier curve, that also interpolates one of the points (presented in appendix B).
- **Projection of a point onto a NURBS Curve.**
- **Projection of a point onto a NURBS Surface.**
- **Inversion of point of a NURBS curve.** Computation of the parameter u of a point that is known to lie on the curve.
- **Inversion of point of a NURBS Surface.** Computation of the parameters (u,v) of a point that is known to lie on the surface.
- **Intersection between two NURBS Surfaces.** Computation of the curve that belongs to both surfaces.
- **Intersection between two NURBS Curves.** Computation of the point that belongs to both curves.

Chapter 3

Turbomachinery Blade Parameterization

The blade parameterization this thesis is based upon is an intuitive method that exploits fundamental notions of turbomachinery to represent a blade. This method is programmed and used by the PCOpt/NTUA in the GMTurbo parameterization software [3, 7] and is presented in detail in this chapter. The implementation of the geometric shapes needed throughout the parameterization is carried out using parametric NURBS curves and surfaces, as discussed in the previous chapter.

3.1 Meridional Plane

The first step of the parameterization procedure is to create the meridional contour of the turbomachine. For a blading that revolves around the z axis, the meridional contour is an (r, z) projection of the axisymmetric parts of the blade, namely, the inlet and outlet planes, the hub and shroud, the LE and TE trace as the turbomachine revolves. These parts, being axisymmetric, exhibit symmetry around z axis, thus they can fully be represented on the (r, z) plane and, then, by revolution to a certain angle θ around the z axis , translate into (x, y, z) coordinates through the equation

$$(x, y, z) = (rcos\theta, rsin\theta, z) \tag{3.1}$$

Therefore, the first step is the definition of six meridional (r, z) NURBS curves. Namely, two meridional boundary curves (inlet and outlet), two meridional generatrices (hub and shroud) and two meridional edge curves (LE and TE) (fig.3.1).

To make it possible to superpose information about the metal angles and thickness profiles, at different spanwise positions of the blade, projected streamlines are computed as a linear interpolation between the hub and shroud generatrices, defined in the previous step. It is noted that the word *streamwise* refers to a distribution of data from inlet to outlet, while *spanwise* refers to a distribution from hub to shroud. Therefore, the blade can be seen as a combination of streamwise and spanwsie distributions of data. The blade meridional contour becomes

$$h(u, v) = (r(u, v), z(u, v)) \tag{3.2}$$

where shroud, hub, inlet, outlet, LE and TE are given by $h(u, 0)$, $h(u, 1)$, $h(0, v)$, $h(1, v)$, $h(u_{LE}, v)$ and $h(u_{TE}, v)$, respectively.

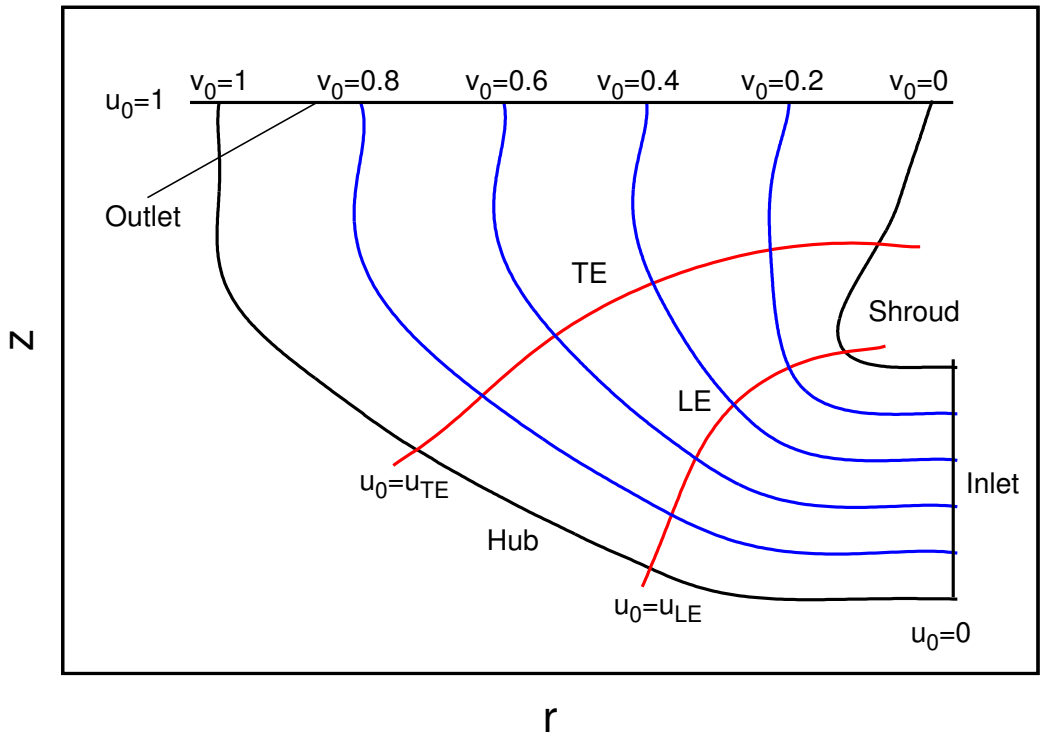


Figure 3.1: *Meridional contour of a Francis hydroturbine blade. The streamline projections (blue lines) lie between hub and shroud.*

Each projected streamline $h(u, v_0) = (r(u, v_0), z(u, v_0))$ corresponds to a revolved surface (fig. 3.3) by adding the angle θ using eq. 3.1.

Conformal Mapping:

Every surface can be described by two parameters, using a transformation from $3D$ coordinates to a $2D$ plane. Hereupon, the conformal mapping is introduced. It is known that a $3D$ surface of revolution can be mapped onto the $2D$ (m, θ) plane through the transformation

$$\Phi(v_0) : (r(u, v_0)cos\theta, r(u, v_0)sin\theta, z(u, v_0)) \mapsto (m(u, v_0), \theta) \tag{3.3}$$

where $m(u, v_0)$ is given by

$$m(u, v_0) = \int_0^u \frac{\sqrt{r_u(t, v_0)^2 + z_u(t, v_0)^2}}{r(t, v_0)} dt \tag{3.4}$$

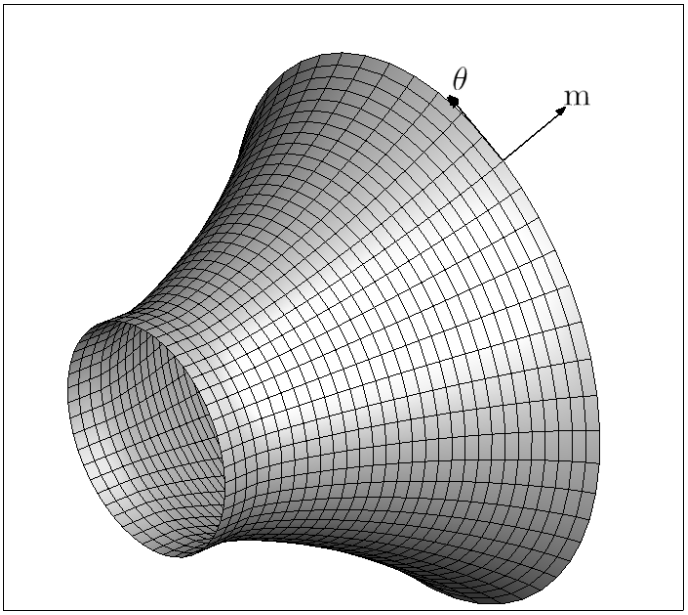


Figure 3.2: *Conformal mapping of a revolved surface to* (m, θ) *coordinates.*

The mapping 3.3 is conformal (see appendix A). The most important property of any conformal mapping is the angle preservation property. *Conformal mappings preserve the magnitude and direction of the angle between two curves [8].* This property contributes to a better understanding of the parameterization, considering that the angles defined on the (m, θ)-plane to be presented in the next section, are preserved on the (x, y, z) surface as well.

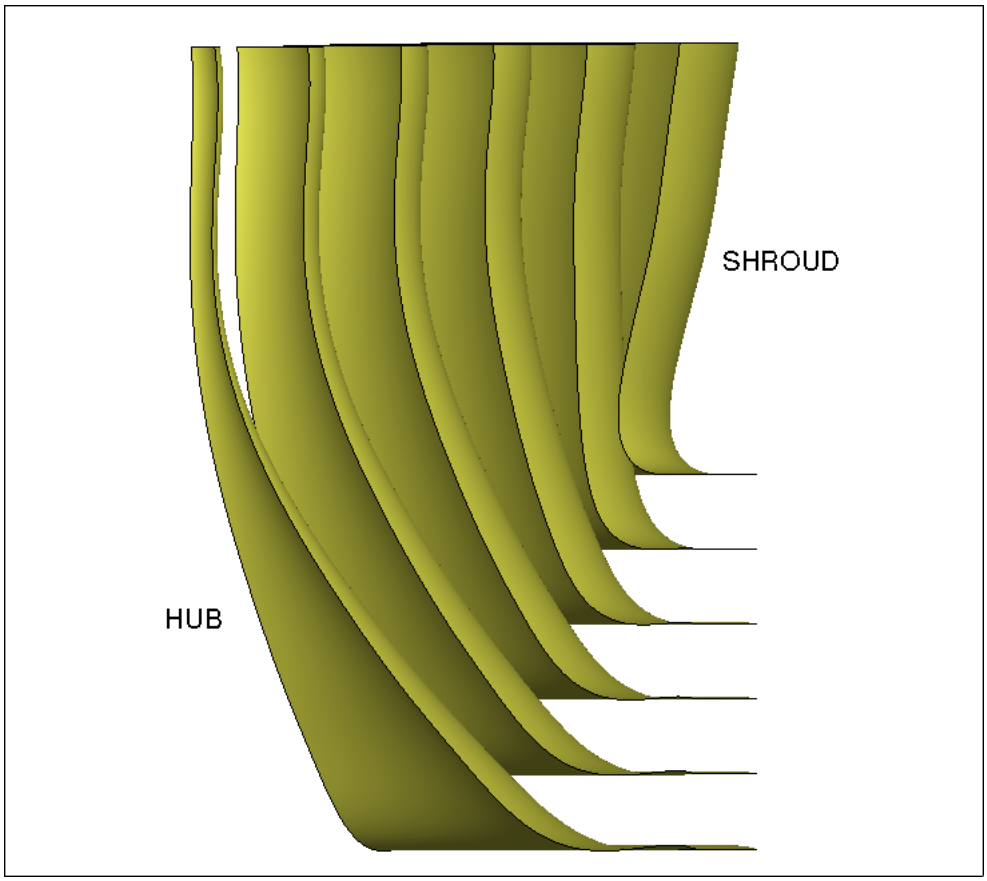


Figure 3.3: *Surfaces of revolution that result from the (r, z) streamlines of figure 3.1. It is noted that for a $v_0 = const.$, that is, for a streamline between hub ($v_0 = 1$)and shroud ($v_0 = 0$), the (r, z) projected streamline, the (r, θ, z) surface of revolution and the $\Phi(v_0)$ transformation to (m, θ) plane have equivalent geometric meaning.*

3.2 Mean Camber Line Parameterization

At each spanwise position $v_0 \in [0, 1]$, a mean camber line is defined, to add information about the blade's metal angles. The mean camber line is chosen to be represented by a cubic Bézier curve on the (m, θ) plane given by the transformation $\Phi(v_0)$. The four control points of the cubic Bézier curve are defined by six parameters that correspond to different angles at the LE and TE of the mean camber line, providing the designer with a better understanding of the mean camber line's slope along the arc length. These are:

- θ_{LE} and θ_{TE} are the peripheral positions of the LE and TE respectively.
- β_{LE} and β_{TE} are the metal angles. These are the angles between the tangent to the mean camber line at the LE or TE and the tangent to the circle that

revolves around z-axis and passes through the LE or TE respectively (an $m = const.$ circle in fig. 3.2 that passes through the LE and TE points).

- δ_{LE} and δ_{TE} are the angles that determine magnitudes ζ_{LE} and ζ_{TE} respectively. These magnitudes specify for how long the directions β_{LE} and β_{TE} are kept.

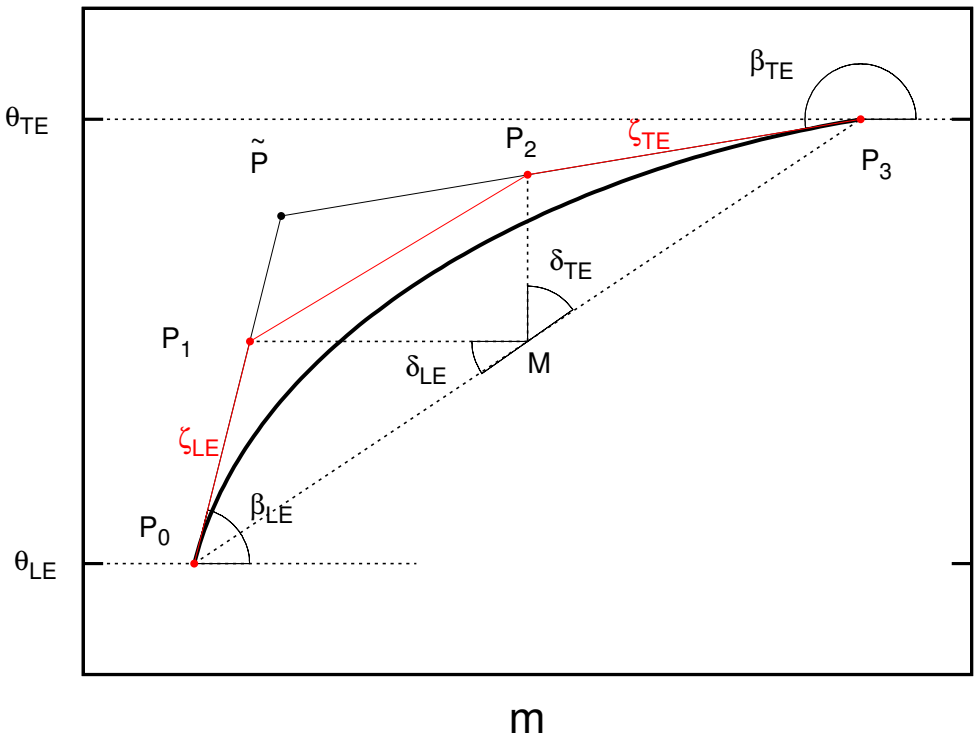


Figure 3.4: *Definition of control points of a cubic Bezier mean camber line, from angles θ_{LE} , θ_{TE} , β_{LE} , β_{TE} , δ_{LE} and δ_{TE}.*

According to fig. 3.4, the control points $\mathbf{P_0}, \mathbf{P_1}, \mathbf{P_2}, \mathbf{P_3}$ are functions of θ_{LE} , θ_{TE} , β_{LE} , β_{TE} , ζ_{LE}, ζ_{TE} and δ_{LE}, δ_{TE} given by:

$$\mathbf{P_0} = (m_{LE}, \theta_{LE}) \tag{3.5}$$

$$\mathbf{P_3} = (m_{TE}, \theta_{TE}) \tag{3.6}$$

where $m_{LE} = m(u_{LE}, v_0)$ and $m_{TE} = m(u_{TE}, v_0)$ from equation 3.4.

Point $\tilde{\mathbf{P}}$ is the intersection point between the tangent lines at LE and TE, defined by the angles β_{LE} and β_{TE}. Points $\mathbf{P_1}$ and $\mathbf{P_2}$ are functions of ζ_{LE} and ζ_{TE}:

$$\mathbf{P_1} = \zeta_{LE}\mathbf{P_0} + (1 - \zeta_{LE})\tilde{\mathbf{P}} \tag{3.7}$$

$$\mathbf{P_2} = \zeta_{TE}\mathbf{P_3} + (1 - \zeta_{TE})\tilde{\mathbf{P}} \tag{3.8}$$

Alternatively, using δ_{LE} and δ_{TE} angles, control points $\mathbf{P_1}$ and $\mathbf{P_2}$ are given as solution to the systems:

$$(\mathbf{P_1} - \mathbf{M}) \cdot (\mathbf{P_0} - \mathbf{M}) = cos\delta_{LE}||\mathbf{P_1} - \mathbf{M}|| \cdot ||\mathbf{P_0} - \mathbf{M}|| \tag{3.9}$$

$$(\mathbf{P_2} - \mathbf{M}) \cdot (\mathbf{P_3} - \mathbf{M}) = cos\delta_{TE}||\mathbf{P_2} - \mathbf{M}|| \cdot ||\mathbf{P_3} - \mathbf{M}|| \tag{3.10}$$

where $\mathbf{M}$ is the midpoint of the $\mathbf{P_0P_3}$ chord.

Through this cubic Bezier parameterization, spanwise distributions of θ_{LE} , θ_{TE} , β_{LE} , β_{TE} , δ_{LE} and δ_{TE} are defined, producing the mean camber line for each spanwise position as seen in fig. 3.5 .

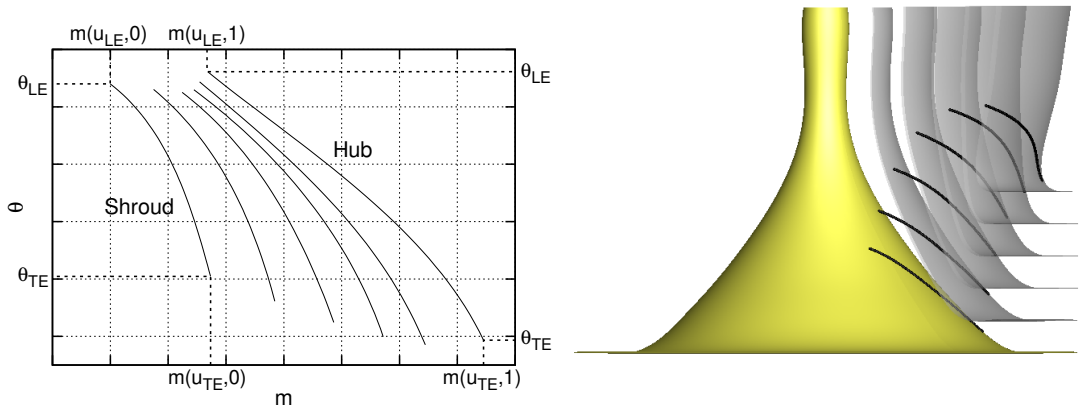


Figure 3.5: *Spanwise mean camber lines, defined on the (m, θ) plane(left figure), and transformed to the 3D space through equation 3.3 (right figure). Each mean camber line lies on the corresponding surface of revolution.*

3.3 Blade Thickness Profiles

By determining the mean camber line on each surface of revolution, a skeleton of the blade has been defined. The superposition of streamwise thickness profiles, along each spanwise position of the blade, creates the final blade. The thickness profile is imposed in two steps, to increase flexibility. First, the normalized thickness profiles $(\hat{t})$ with respect to the normalized arc-length (s) of the mean camber line is defined separately for the pressure and suction sides. Then, a thickness factor (t_f) that scales the thickness profiles is specified for each profile, resulting to a thickness distribution at each spanwise position $v_0 \in [0, 1]$:

$$\begin{aligned} t^{PS}(s, v_0) &= \hat{t}^{PS}(s, v_0) t_f^{PS}(v_0) \\ t^{SS}(s, v_0) &= \hat{t}^{SS}(s, v_0) t_f^{SS}(v_0) \end{aligned} \tag{3.11}$$

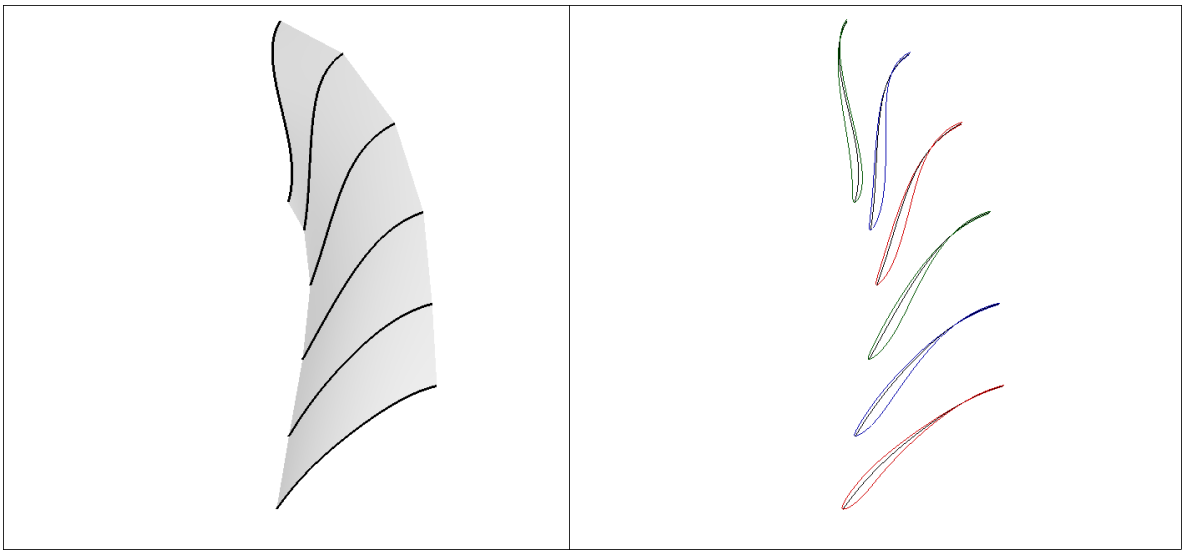


Figure 3.6: *The thickless blade (mean camber surface) on the left, composed by interpolation of the mean camber lines in the 3D space. On the right, the thickness profiles are superposed.*

Having determined a mean camber line $\mu_{m\theta}$ for the spanwise section on the (m, θ) plane, the imposition of the thickness profiles requires the computation of the normal vector $\hat{n_{m\theta}}(s, v_0)$ at each normalized arc length s point of the mean camber line and the application of the equation (to both pressure and suction side with the

appropriate sign):

$$\mathbf{c}^{PS}_{m\theta}(s, v_0) = \boldsymbol{\mu}_{m\theta}(s, v_0) \pm \mathbf{n}_{m\theta}(s, v_0)\frac{t^{PS}(s, v_0)}{r^2(m(s, v_0))}$$
$$\mathbf{c}^{SS}_{m\theta}(s, v_0) = \boldsymbol{\mu}_{m\theta}(s, v_0) \mp \mathbf{n}_{m\theta}(s, v_0)\frac{t^{SS}(s, v_0)}{r^2(m(s, v_0))} \tag{3.12}$$

where $r(m(s, v_0))$ is the corresponding radius of the (m, θ) point $\mu_{\mathbf{m}\theta}(\mathbf{s}, \mathbf{v_0})$ of the mean camber line, through $\Phi^{-1}(v_0)$, and is used to transform the legnth $t(s, v_0)$ of the $3D$ space to a length on the (m, θ) plane (see appendix A). These airfoil curves (eq. 3.12) are mapped back onto the (x, y, z) coordinates, through $\Phi^{-1}(v_0)$, to create the $3D$ skeleton of the blade(figure 3.6).

The final step is the skinning of the two sides, to create two NURBS surfaces, using an algorithm that passes a smooth surface through a set of curves, giving rise to the final $3D$ blade (fig. 3.7).

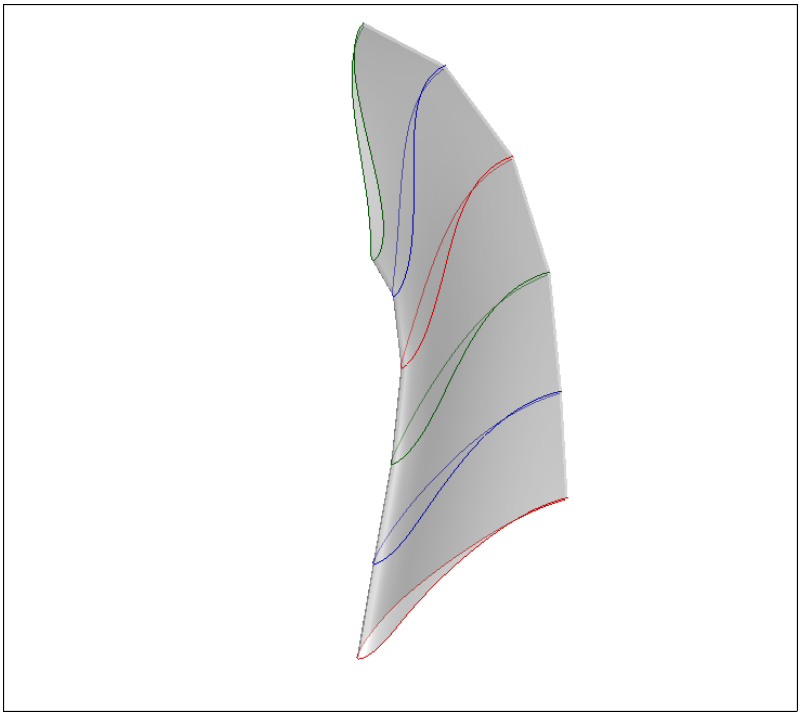


Figure 3.7: *The final skinned blade surface.*

Chapter 4

Reverse Parameterization Tool (RPT)

A CFD grid is a common but unhandy form of representation of a blade's shape, since, in terms of design, it is not easy to modify the geometry given in grid form. It is thus essential that it is converted to a more useful CAD form, so that changes in the geometry can be made. In this thesis, the CAD representation is the blade design parameterization GMTurbo presented in chapter 3. A software to transform a CFD grid into a GMTurbo compatible form is the main purpose of this thesis and this is presented in this chapter.

4.1 Turbomachine Blade Grids

A CFD 3D grid summarizes the following information:

- A set of nodes, with their (x, y, z) coordinates.
- The connectivity of these nodes.
- The boundary patches of the volume grid, namely surface patches formed by the nodes that have already been defined. These are the patches where the various boundary conditions must be applied by the CFD solver.

The RPT is programmed to reparameterize structured surface grids, composed by quads. The boundary patches of these grids must be in a standardized format and contain the following:

- An Inlet and an Outlet Patch where the CFD solver applies the inlet and outlet boundary conditions.
- Wall Patches i.e. solid boundaries where the wall boundary conditions are applied. These usually are hub and shroud patches, pressure and suction side patches and a trailing edge patch. In turbomachinery CFD grids, wall patches might be rotating (with the exception of the shroud that is usually stationary or the stationary blades), thus the computation of the velocity flow variables depends on the speed of the rotating parts.

  The trailing edge type can vary. In the most common cases, it is rounded, sharp or blunt type. In the cases analyzed in this thesis the trailing edge type is blunt, so the blade in the trailing edge is cut off, and the thickness at that point is represented by a trailing edge patch (fig. 4.1).
- Periodic Patches. When the CFD domain has a periodic repeating nature (repeating geometry and flow field), periodic boundary conditions are applied on the periodic patches. In the grids used in this thesis, the periodic patches surround each blade as seen in fig. 4.1.

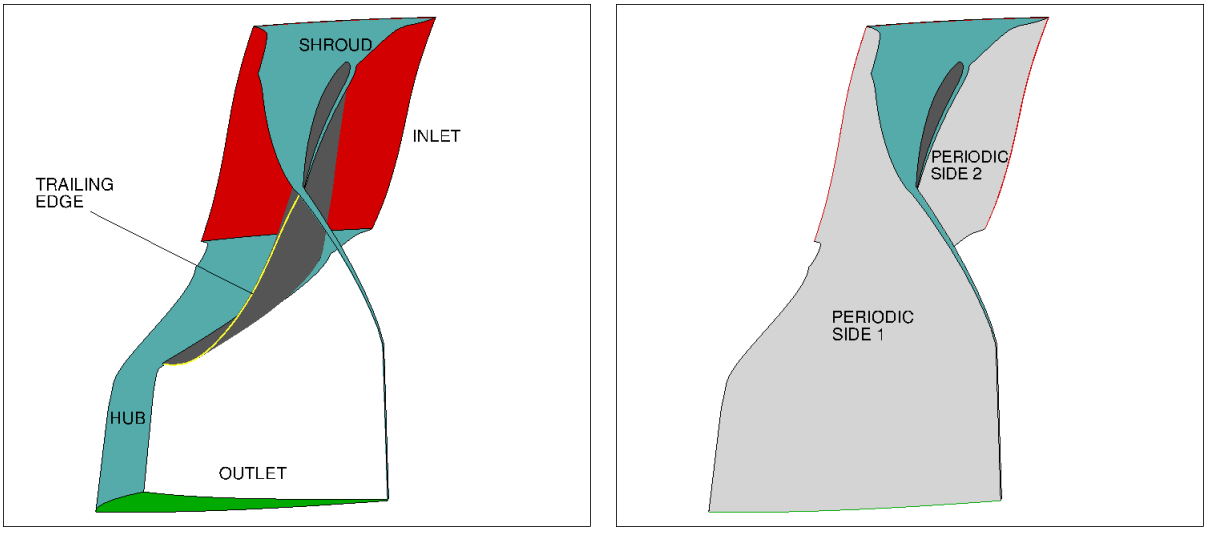


Figure 4.1: *Standard patches of the CFD grid used for the analysis of a turbomachinery blading.*

4.2 Reverse Parameterization of a Blade

Starting with a turbomachinery surface grid like the one presented in section 4.1, a back-to-CAD method, converting the CFD grid to the parameterization presented in chapter 3 is described below. The meridional contour, mean camber lines and thickness profiles of the existing grid are to be computed.

4.2.1 Meridional Contour of the Grid

The first step is the computation of the meridional contour. Having the surface grid of the meridional patches, namely hub, shroud, inlet and outlet, the (r, z) generatrices of each meridional surface are computed. The edge between the surface grid of a meridional patch with one of the periodic patches (whichever) (fig. 4.2) is a node representation of one generatrix of the meridional surface in (x, y, z) coordinates. Projecting this generatrix onto the (r, z) plane using equations

$$r = \sqrt{x^2 + y^2}$$
$$z = z \tag{4.1}$$

produces the meridional projection of hub, shroud, inlet and outlet patches. Also, projecting the grid edges that correspond to the LE and TE onto the (r, z) plane, produces the meridional curves of the two edges. The hub and shroud generatrices are, then, approximated by a NURBS curve of a user-defined degree and number of control points.

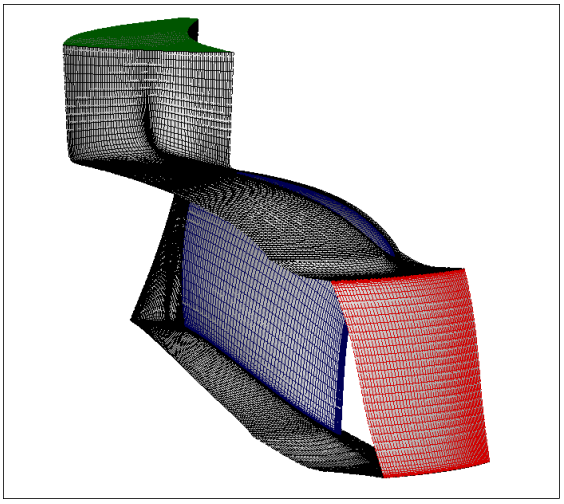


Figure 4.2: *The CFD surface grid. The axisymmetric patches namely hub (black), shroud (black), inlet (red) and outlet (green) are shown.*

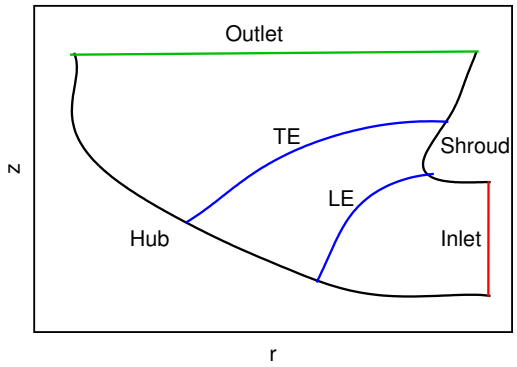


Figure 4.3: *The meridional contour, as it results from the projection of the axisymmetric patch nodes and the edge nodes onto the (r, z)-plane.*

Next, a user-defined number of N spanwise generatrices is generated. Having the NURBS curves of hub and shroud at the meridional plane (in (r, z) coordinates), a linear interpolation of the control points, produces intermediate streamlines (in NURBS representation)(fig. 4.4). After defining the N generatrices, the operations take place for each spanwise generatrix, thus the following are applied to each and

every spanwise generatrix, in order to attain spanwise distributions of data.

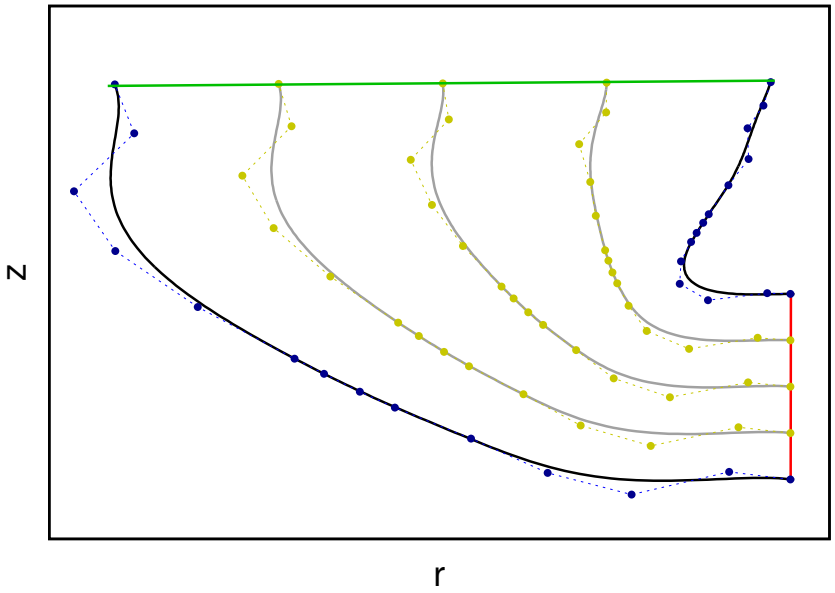


Figure 4.4: *A linear interpolation between the control points of the hub and shroud results to the control points of the intermediate spanwise generatrices.*

4.2.2 Mean Camber Line and Thickness Data Computation

Based on the (r, z) NURBS curve of the N^{th} generatrix, a revolved surface in the (x, y, z) space and the Φ transformation function to the (m, θ) plane are generated.

A NURBS revolved surface that rotates around the z-axis to a certain angle θ, can be computed using a single (r, z) NURBS generatrix. Rotating this generatrix control points at discrete angles from 0 to θ, leads to sets of control points of type (r, θ_i, z). These control points generate a NURBS revolved surface with angle of revolution θ, the generatrix of which is the initial (r, z) generatrix. The cartesian representation of the (r, θ, z) surface points is given by

$$(x, y, z) = (rcos\theta, rsin\theta, z) \tag{4.2}$$

Using the N^{th} (r, z)-generatrix and eq. 3.3, a transformation function Φ that maps every (r, θ, z) point of the revolved surface to the (m, θ) plane can be computed.

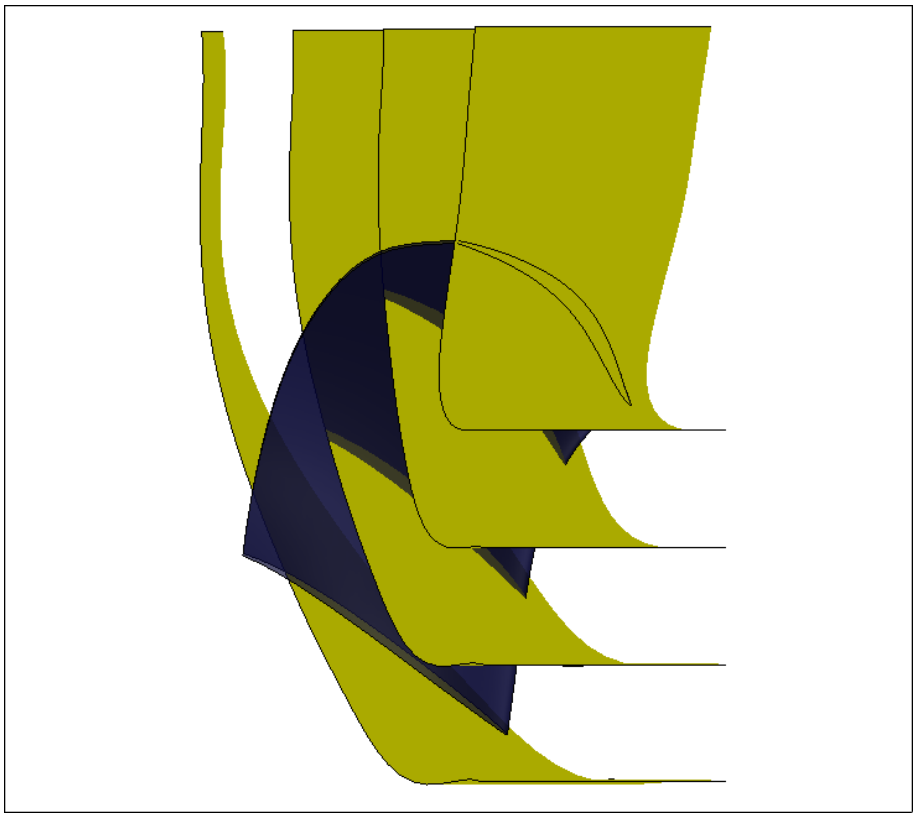


Figure 4.5: *Up to this point, the revolved surfaces of the spanwise generatrices of figure 4.4, between hub and shroud have been produced by the RPT. The grid of the blade lies within hub and shroud and intersects the spanwise revoved surfaces. In the RPT, instead of superposing data to the spanwise revolved surfaces, as in the GMTurbo, data is extracted from the spanwise revolved surfaces.*

Pressure and Suction sides:

The definition of the pressure and suction side curves that lie on the revolved surface is the result of the intersection between the blade grid and the revolved surface produced. An algorithm that finds the intersection points between a structured surface grid consisted of quads and the NURBS surface that intersects it, was developed in order to find the points that lie both on the revolved surface and the blade (appendix B), resulting to a set of points such as in fig. 4.6.

Having identified the airfoil (x, y, z) points in the 3D space and using the transformation function Φ that has also been defined (knowing the (r, z) generatrix), the blade (x, y, z) points can transform to (m, θ) coordinates and the analysis turns from 3D cartesian space to the (m, θ)-plane (fig. 4.7). It is helpful to approximate the (m, θ) points of the two sides of the airfoil with two NURBS curves in order to achieve a continuous representation of the two sides. To do so, a NURBS constrained approximation algorithm was programmed, so that the resulting curves respect the continuity of first derivative at the LE (appendix C).

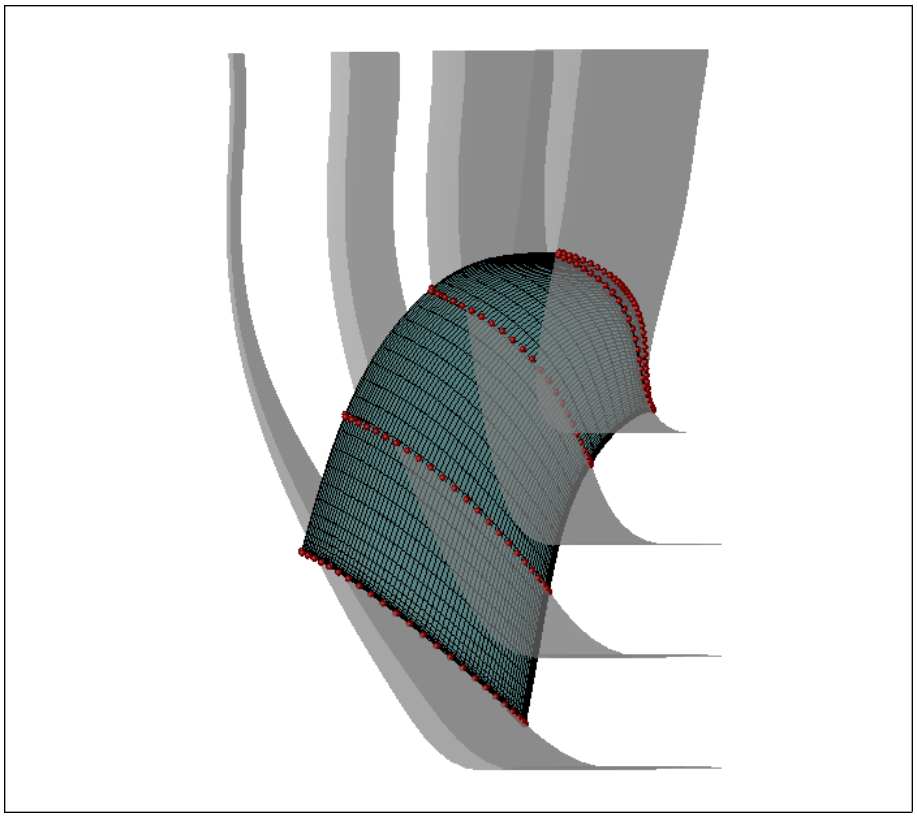


Figure 4.6: *Discrete points (in red) of the blade grid that lie on the surfaces of revolution.*

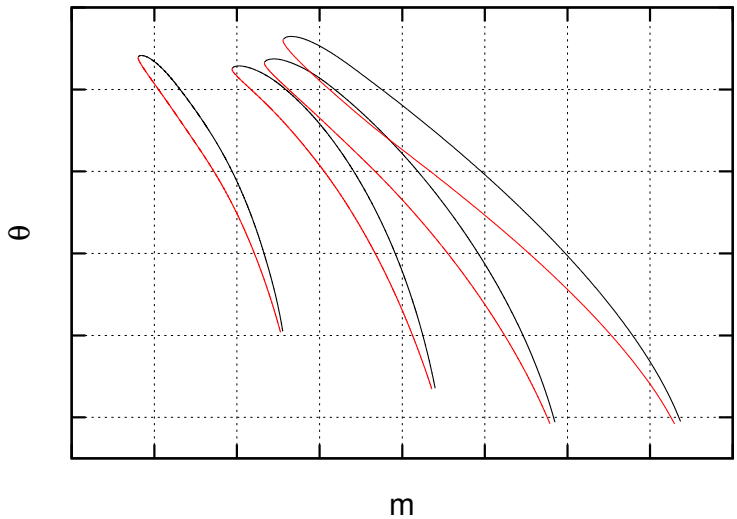


Figure 4.7: *The points of fig. 4.6 transformed into the (m, θ)-plane and approximated by NURBS curves, creating spanwise airfoils on the (m, θ)-plane.*

Mean Camber Line:

To compute the mean camber line of an existing blade airfoil, on a 2D plane, an algorithm was developed that computes the line referred to as the Exact Mean Camber Line because it is computed according to the following definition; *a line joining the leading and trailing edges of an airfoil equidistant from the upper and lower surfaces*(appendix D).

To produce a mean camber line compatible with the parameterization method of chapter 3 (4 point Bézier representation), the exact mean camber line points are approximated with a Cubic Bézier curve that is referred to as the 4 Point Mean Camber Line. A cubic Bézier approximation technique enforced with constraints was developed to approximate the exact mean camber line points, while preserving (through constraints), the endpoints (LE and TE preservation) and endpoint tangents (metal angles preservation) of the airfoil (appendix E).

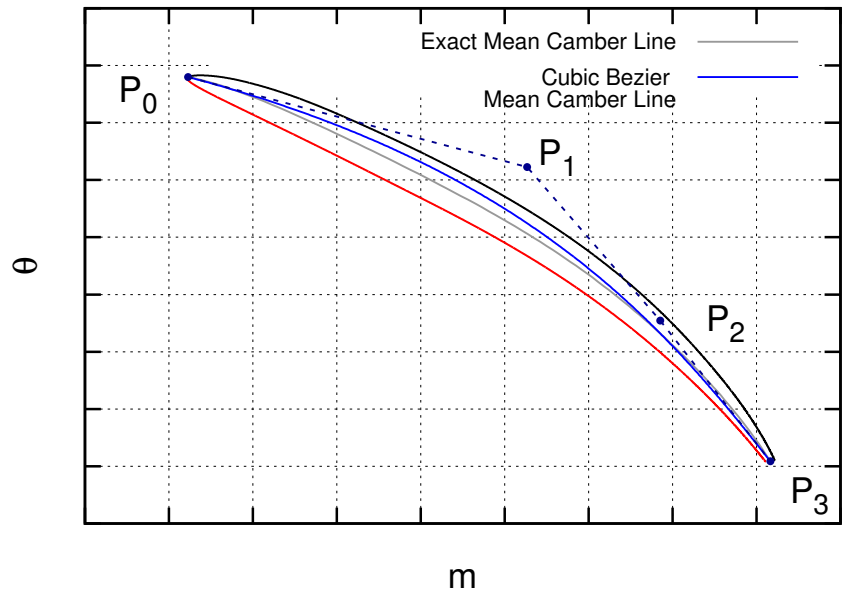


Figure 4.8: *The exact (grey) and cubic Bézier (blue) mean camber lines. The blade airfoil contour is also shown in black and red.*

Having computed the Cubic Bézier Mean Camber Line, hence P_0, P_1, P_2, P_3, the angles θ, β, δ are computed from their definitions (fig. 3.4):

$$\theta_{LE} = \mathbf{P_{0,\theta}} \quad \theta_{TE} = \mathbf{P_{3,\theta}} \tag{4.3}$$

$$\beta_{LE} = atan\Big(\frac{\mathbf{P_{1,\theta}} - \mathbf{P_{0,\theta}}}{\mathbf{P_{1,m}} - \mathbf{P_{0,m}}}\Big) \quad \beta_{TE} = atan\Big(\frac{\mathbf{P_{2,\theta}} - \mathbf{P_{3,\theta}}}{\mathbf{P_{2,m}} - \mathbf{P_{3,m}}}\Big) \tag{4.4}$$

$$\delta_{LE} = acos\Big(\frac{\vec{MP}_1 \cdot \vec{MP}_0}{|\vec{MP}_1||\vec{MP}_0|}\Big) \quad \delta_{TE} = acos\Big(\frac{\vec{MP}_2 \cdot \vec{MP}_3}{|\vec{MP}_2||\vec{MP}_3|}\Big) \tag{4.5}$$

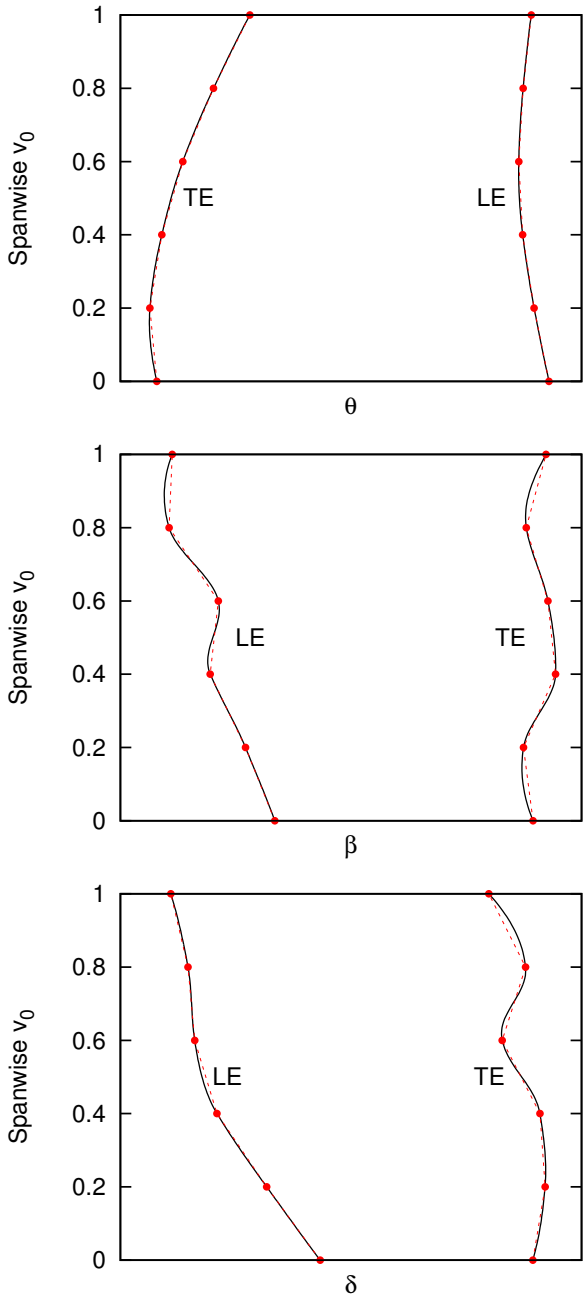


Figure 4.9: *Equations 4.3, 4.4, 4.5 applied to each spanwise 4 point mean camber line, result to spanwise distributions of the angles θ, β, δ. The discrete red points are extracted from the surface grid, for every spanwise position. Then, a NURBS curve interpolation provides the continuous spanwise distribution of the angles (black lines).*

Thickness Profiles:

The last step to the complete reparameterization of the blade grid is the computation of the thickness profiles, that are a combination of the non-dimensional streamwise thickness profiles for each spanwise position and the spanwise thickness factor distribution, described in section 3.3. Having the pressure and suction side NURBS representations and the mean camber line cubic Bézier representation, the normal distances of the mean camber line to both sides are computed, resulting to the two thickness profiles ($t(u)$) described in eq. 3.11. Then, dividing each profile with its maximum value that is the thickness factor of the profile, $t_f(v)$ (fig 4.11), leads to the non-dimensional thickness profiles ($\hat{t}(u)$) (fig 4.10).

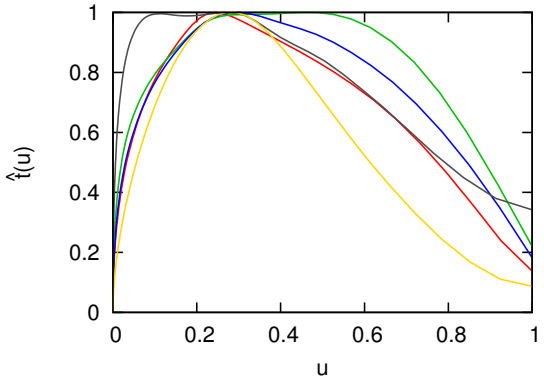


Figure 4.10: *Streamwise thickness profiles.*

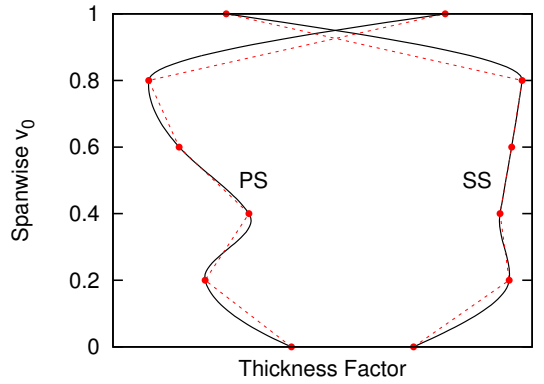


Figure 4.11: *Spanwise thickness factor distributions.*

Chapter 5

Grid Adaptation Tool (GAT)

The back-to-CAD method presented in chapter 4 generates a geometry that can be easily modified and optimized, simply by altering the CAD design variables of GMTurbo. However, through this transformation, the nodal representation is lost, when modifying the CAD parameters. To make it possible to perform CFD simulations on the modified CAD geometry a grid has to be generated around the geometry. To avoid mesh generation, the initial grid is exploited. A surface grid adaptation tool (GAT) that adapts the initial CFD grid, to the reparameterized CAD geometry is developed in this thesis. Using the 3D spring analogy technique, the initial volume grid is displaced to fit to the adapted surface grid resulted from the GAT. The final volume grid has the same number of nodes, same connecitvity and same quality as the initial grid.

The conversion from GMTurbo representation back to node-based representation is presented in this section.

5.1 Description of the GAT Method

The grid adaptation method developed in this thesis, adapts the initial surface grid to the CAD geometry. The method takes advantage of the availability of an initial grid (that is taken for granted in this thesis) to generate a new one around the CAD geometry. It is performed in two steps. First, the surface grid of the reparameterized wall patches is computed, by projecting the initial CFD surface nodes onto the reparameterized NURBS surfaces, for the various wall patches namely Hub, Shroud, Pressure Side, Suction Side, Trailing Edge (fig.5.1). By doing so, the structure and connectivity of the surface grid are maintained and only the coordinates of

the nodes are adapted to the reparameterized (and, consequently, slightly changed) surface shape. Then, the volume grid of the whole CFD domain is morphed w.r.t the displacement of the surface grids computed in the previous step.

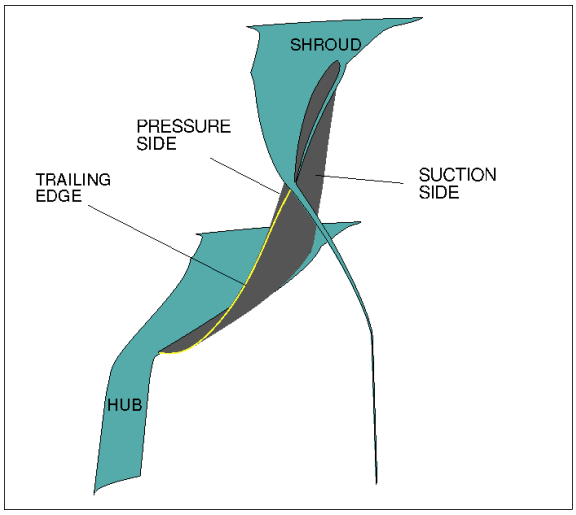


Figure 5.1: *Wall patches of a blade grid of a Francis Turbine.*

5.1.1 Step 1: Surface Grid Adaptation

To compute the surface grid of the reparameterized geometry of each patch, the wall patch nodes of the initial grid are projected onto the corresponding NURBS surfaces, obtaining a 2D representation of each surface. Then, in 2D coordinates the necessary morphing is performed and the morphed 2D grid is then transformed back to 3D coordinates.

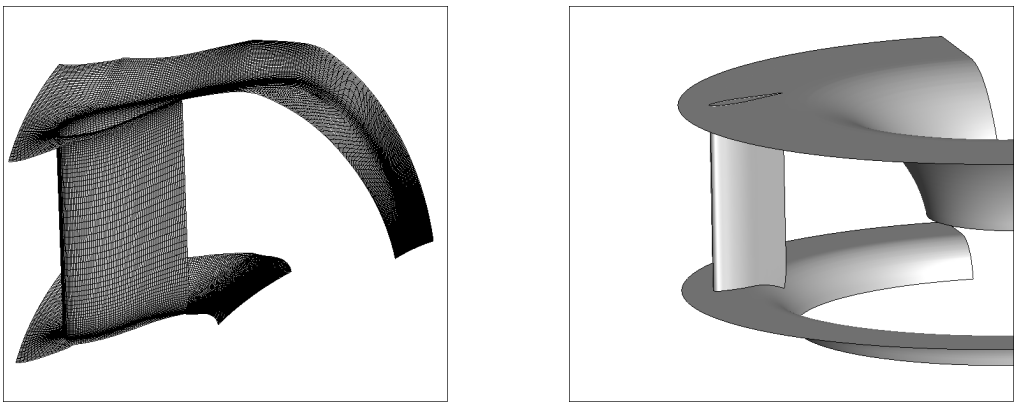


Figure 5.2: *The GAT uses the initial mesh (left) and the NURBS curves of the wall patches resulted from the GMTurbo(right) to create a new mesh with the same connectivity as the initial but node coordinates adapted to the parameterization surfaces.*

The patches are separated into two categories. In the first category, the Pressure Side, Suction Side and Trailing Edge Patches are NURBS surfaces, each point of which is represented by two parameters (u, v). The second category, includes the patches Hub and Shroud that can be represented from a NURBS revolved surface each point of which can be represented by the NURBS surface (u, v) parameters or the (m, θ) parameters through the conformal mapping. It is preferable to use the (m, θ) representation instead of the (u, v) representation of the NURBS revolved surface, since the first preserves the periodicity of the nodes that belong to the periodic patches. Two nodes in (x, y, z) coordinates, that have a periodic connection, have a specific angular (θ) pitch difference. This pitch is preserved when transforming into (m, θ) parameters, maintaining the periodicity of the nodes. Consequently, each one of the five wall patches corresponds to a parametric surface which, by definition, can be represented by 2 parameters ((u, v) or (m, θ)).

Projecting a single point onto a parametric surface, results the closest to this (x, y, z) point that also belongs to the parametric surface. The latter can also be discribed with two parameters (u, v), since it belongs to the parametric surface. Consequently, projecting all the nodes of a wall surface patch to its parametric surface produces a 2D grid of parameters (figs. 5.3, 5.4). Repeating the procedure for every one of the five wall patches, five 2D grids of parameters are computed.

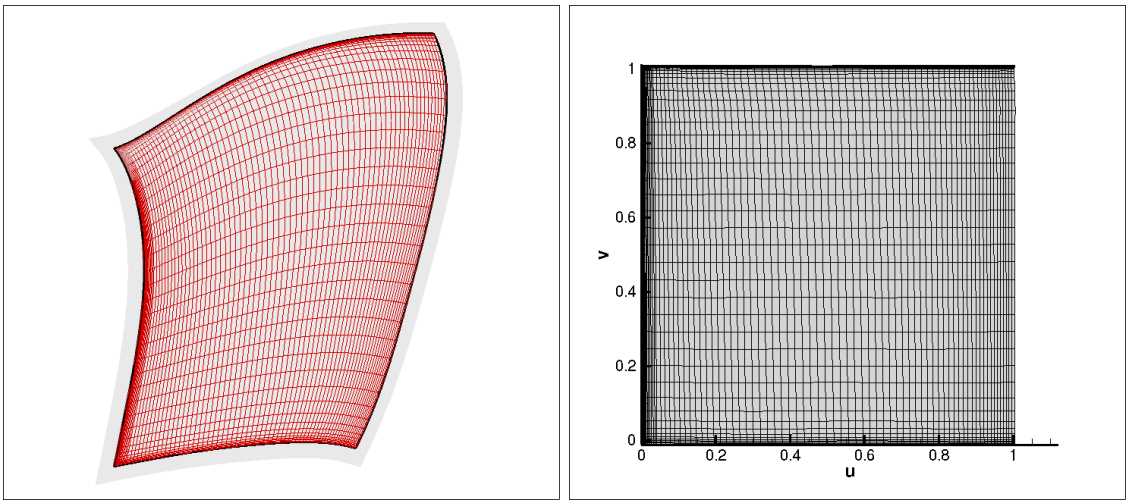


Figure 5.3: *The projection of the surface grid onto the NURBS reparameterized surface at the left can be transformed into 2D (u, v) points, resulting to a 2D grid (right figure).*

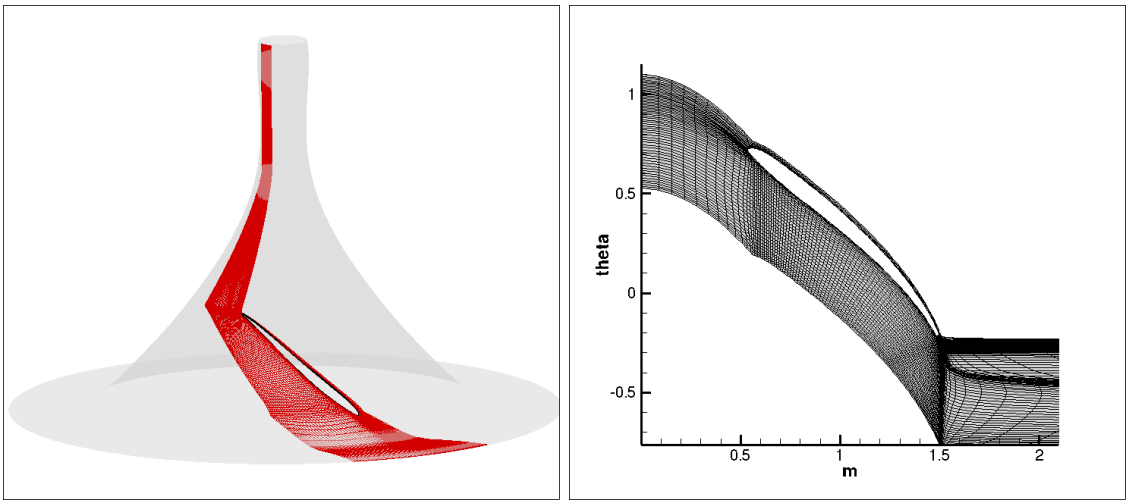


Figure 5.4: *The projection of the surface grid onto the NURBS reparameterized revolved surface at the left can be transformed into 2D (m, θ) points, resulting to a 2D grid (right figure).*

However, the edges of the grid are not projected onto the edges of the surface (fig. 5.5, 5.6). To force the surface grid to fit the NURBS surface, a deformation is applied to the 2D parameters of each wall patch computed earlier. The deformation is performed using the 2D Spring Analogy Technique developed and used at the PCOpt/NTUA [4]. This software takes a 2D grid (computed earlier by projecting the surface nodes onto parametric surfaces) and the 2D position of the edge nodes of the grid as input and distributes the internal nodes, w.r.t the edge positions.

The position of the edge nodes can be found with the following technique. The edges of the surfaces are 3D NURBS curves provided by the parameterization, since they are the intersections of the various NURBS surfaces resulting from GMTurbo (fig 5.5). The edge nodes of the adapted grid must belong to those edge curves. Thus, the edge nodes of the initial grid are distributed onto the 3D edge curve (using the distance distibution they had in the initial grid). Since they belong to the 3D edge curve, they also belong to the wall NURBS surface, thus they are represented by two parameters provided by the NURBS surface ((u, v) or (m, θ)) . These edge parameters are given as an input to the 2D spring analogy morpher. The moprhing practically slides the nodes on the surface to make the surface grid fit the edges of the NURBS surface.

After morphing the 2D grid, the displaced parameters of the surface nodes are found. It is easy to go back to 3D, using the equations of the corresponding surface ($\mathbf{S}(u, v)$ for pressure side, suction side and trailing edge and Φ^{-1} for hub and shroud), attaining the displaced (x, y, z) surface patches.

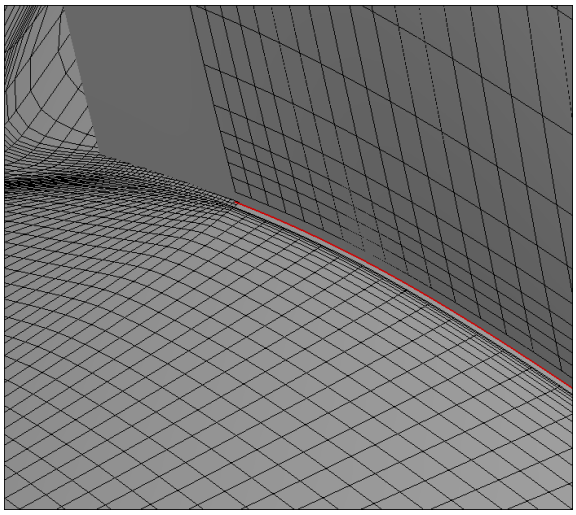


Figure 5.5: *The projection of the mesh onto the NURBS surfaces is not accurate, since the edge nodes are not projected onto the actual edges of the surface. The need to displace the surface mesh to fit the edges of the parameterization (red line) comes up.*

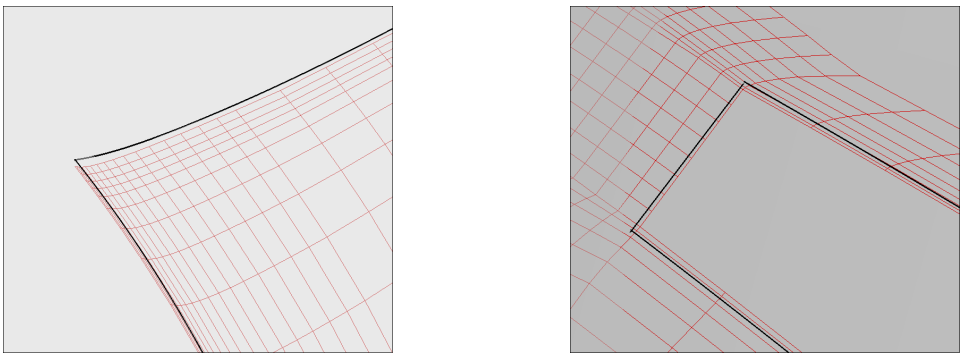


Figure 5.6: *The edges according to the reparameterization (black curves) are different than the projections of the edges of the initial grid onto the reparameterized surfces. Thus a 2D spring analogy morphing takes place to displace the projected grid (red grid) w.r.t the edge positions of the parameterization.*

5.1.2 Step 2: Volume Grid Adaptation

Using the surface wall patches computed on the previous step, a deformation to the initial volume grid can be applied using the 3D spring analogy technique [4], to adapt the internal volume grid, with respect to the position of the surface patches. This deformation results to a volume grid of the reparameterized CAD blade. This grid has the same structure and connectivity as the initial one but has been displaced in terms of coordinates.

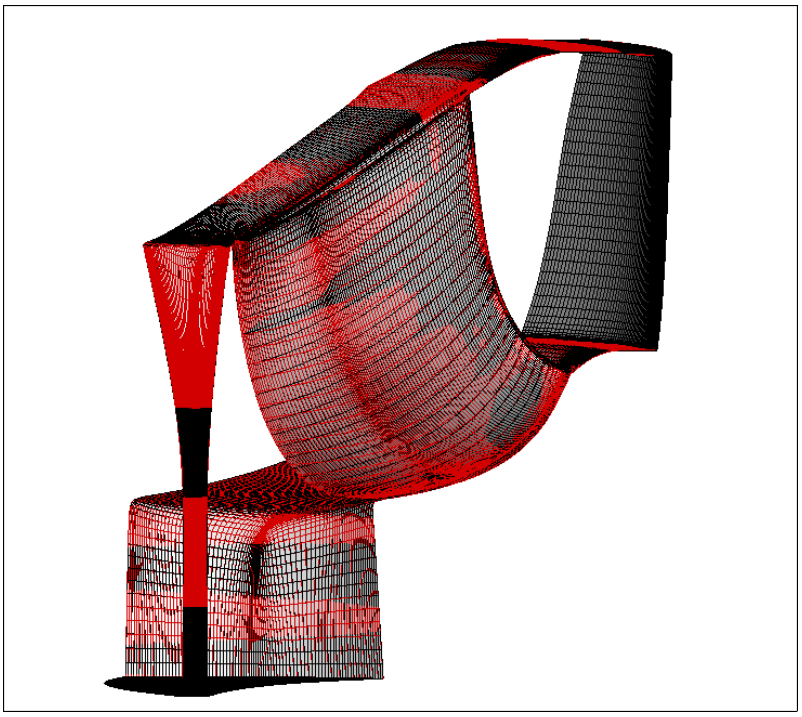


Figure 5.7: *The resulting grid is very close to the initial, depending on the user defined accuracy selected for the reparameterization (e.g. number of generatrices, number of control points in NURBS approximations etc).*

5.2 The GAT used in an EA Optimization

To generate a grid on the GMTurbo geometry, during an EA optimization, the method presented in section 5.1 is used. However, the method is separated; a part of it runs only once, before the optimization begins, as a pre-processing step and the rest is integrated into the optimization workflow and creates the grid for every candidate geometry.

To be more specific, the first part of the method, the projection of the surface nodes onto the NURBS surfaces of the parameterization, is common for every candidate geometry, since it just provides the connectivity of the various patches in 2D coordinates. Mapping back to the corresponding NURBS surface of the current EA evaluation, provides the new coordinates of the surface grid. Thus, the 2D parametric grid of each wall patch (fig. 5.1) is created once, at the begining of the optimization. It is morphed for each candidate geometry, according to the edge positions of that geometry. The rest of the method follows as presented in section 5.1 and is integrated to the EA workflow to be repeated for every candidate geometry.

Chapter 6

CFD Analysis

In this chapter, the governing equations with their boundary conditions that are solved in all hydrodynamic applications in this thesis using the CFD software PUMA developed by the PCOpt/NTUA in [9] [4] [3] are presented. The post processing that is performed is also presented. For more insight to the details of the various CFD methods that concern similar hydrodynamic cases and the pre and post processing of the CFD software used in these cases, the reader is referred to [10].

6.1 Hydraulic Turbomachines

Hydraulic Turbomachines, turbines and pumps, are used when energy is needed to be absorbed from or provided to a working fluid in liquid form. Hydroturbines are installed in places where water of high total pressure is available. Depending on the inlet flow conditions, hydroturbines are separated into three leading categories: Francis turbines, that are radial turbomachines, Kaplan (and propeller) turbines, that are axial turbomachines and Pelton turbines that are impulse-type turbines. In Kaplan turbines the pitch of the blades can be changed in order to increase performance in various operating conditions. Propeller turbine blades on the other hand, are build into their shaft allowing only one fixed pitch for any operating condition.

Operating Point: The operating point of the turbine is predefined by the installation site. The hydraulic head $H[m]$ of the installation and the volume flow rate of the water $Q[m^3/s]$, are given by the hydrological analysis of the site, as mean measurements for a long time period. (H,Q) points vary during the year, giving multiple operating points for a turbine to operate. The rotational speed n[RPM]

of the turbine is also predefned by the type of the generator that is connected to the turbine. Based on the design, each Operating Point on the (H,Q) diagram has a specific efficiency η, that is computed from the governing equations of the flow on the turbine. Using this operating point, the boundary flow conditions can be computed [10].

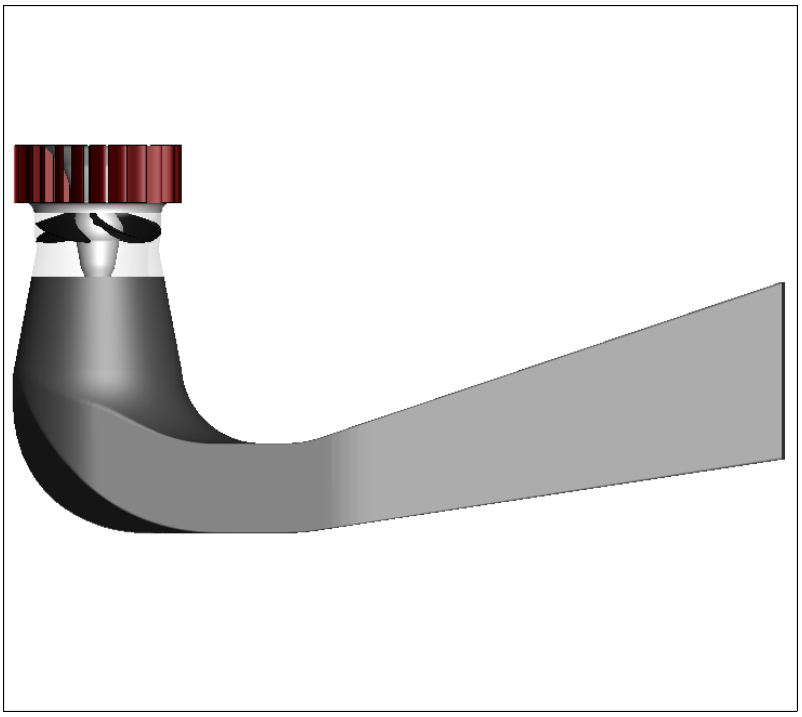


Figure 6.1: *Guide vanes, propeller runner and draft tube.*

Reaction Turbines:

Reaction turbines are the turbines for which the static pressure changes when the fluid passes through the rotor. Reaction rotors possess blades that use this pressure distribution to make them rotate. Since this thesis focuses on blades, reaction turbines are the main subject of interest.

Reaction turbine types are two, Francis and Kaplan (or propeller). Francis turbines are mixed flow reaction turbines that operate at average head values. Kaplan and propeller turbines are axial reaction turbines that operate at average to high head values.

The spiral casing and the guide vanes are the first components of the turbine, upstream of the rotor. The spiral casing, regulates the flow rate of the fluid (Q), so that it is equally distributed at the peripheral direction of the inlet to the rotor. The guide vanes can revolve at a certain angle around their axis and provide the inlet flow with a swirl, that creates a velocity profile at the inlet.

The rotor types vary, accroding to the flow of the fluid in the inlet and outlet planes. In Francis rotors the flow enters at the radial direction and exits axially, while in Kaplan and propeller rotors the inlet and outlet boundaries are both axial.

The outlet of the rotor is connected to the draft tube, in order to restore the static pressure of the fluid. To be more specific, the draft tube or diffuser is a duct that is placed at the outlet of the rotor (fig. 6.1). It decelerates the fluid that exits the rotor, increasing the static pressure, so that no back-flow occurs at the outlet of the draft tube. This gives the rotor the capability for lower outlet pressure, thus further energy absorption from the fluid, without the fear of backflows. The draft tube has a high contribution to the total efficiency of the turbine.

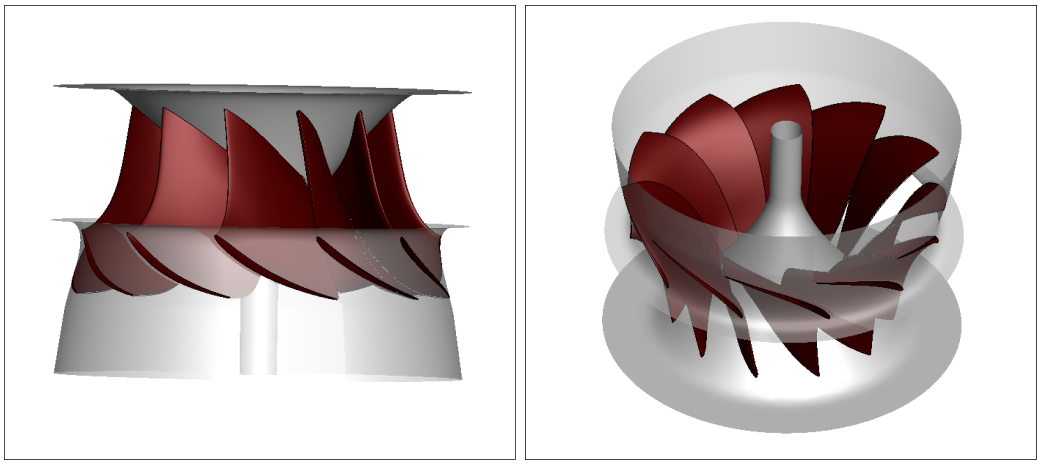


Figure 6.2: *The Francis turbine that is analyzed in chapter 7. Radial flow inlet and axial outlet.*

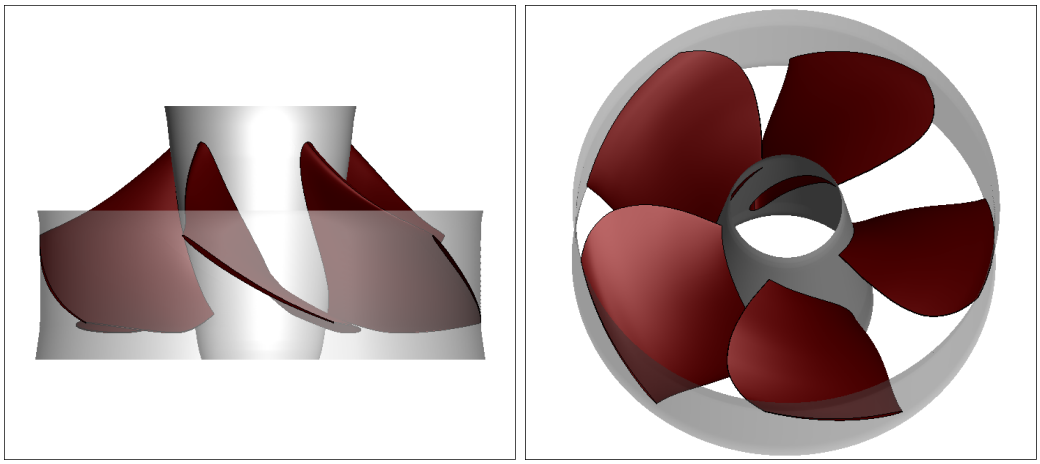


Figure 6.3: *The Propeller turbine that is analyzed in chapter 7. Axial inlet and outlet.*

6.2 The GPU-enabled CFD Solver PUMA

In order to examine the performance of a turbomachine, the incompressible GPU-enabled flow solver PUMA (Parallel Unstructured Multi-row Adjoint) developed by PCOpt/NTUA is used [3]. This software numerically solves the Navier-Stokes equations along with a turbulence modelling equation on a computational domain, using the vertex-centered, finite volume method on unstructured grids. Structured or block structured grids like the ones presented in this thesis, are treated by PUMA as unstructured. The implementation on GPUs provide a remarkable speed up in comparison with CPU implemented softwares, reducing the turnaround time of a CFD evaluation.

6.2.1 Flow Equations

The governing equations are the non-dimensional steady-state Navier-Stokes equations for incompressible fluid flows, written in a rotating frame of reference and using the artificial compressibility method [11], are expressed as

$$\Gamma^{-1}_{nm}\frac{\partial U_m}{\partial \tau} + \frac{\partial f^{inv}_{nk}}{\partial x_k} - \frac{\partial f^{vis}_{nk}}{\partial x_k} + S_n = 0 \tag{6.1}$$

Eq. 6.1 represents four equations with four unknown flow variables $U_m = \begin{bmatrix} \frac{p}{\rho} & u^A_1 & u^A_2 & u^A_3 \end{bmatrix}$ where $\frac{p}{\rho}$ is the kinematic pressure (since density is assumed constant) and $u^A_m (m = 1, 2, 3)$ the velocity components in the absolute reference frame. Variable τ is the pseudo-time step used to stabilize the system of PDEs. The inviscid f^{inv}_{nk}, viscous f^{vis}_{nk} fluxes and the source terms S_n are defined as:

$$f^{inv}_{nk} = \begin{bmatrix} u^R_k \\ u^A_1 u^R_k + p\delta_{1k} \\ u^A_2 u^R_k + p\delta_{2k} \\ u^A_3 u^R_k + p\delta_{3k} \end{bmatrix} \quad f^{vis}_{nk} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \end{bmatrix} \quad S_n = \begin{bmatrix} 0 \\ \varepsilon_{1lk}\omega_\ell u^A_k \\ \varepsilon_{2lk}\omega_\ell u^A_k \\ \varepsilon_{3lk}\omega_\ell u^A_k \end{bmatrix} \tag{6.2}$$

where δ_{ij} is the Kronecker symbol, $\varepsilon_{m\ell k}$ the Levi-Civita symbol and ω_k the rotational speed of the frame.

Relative (u^R_m) and absolute (u^A_m) velocities are linked through the rotating frame velocity u^F_m using the equation

$$\begin{aligned} u^A_m &= u^R_m + u^F_m \\ u^F_m &= \varepsilon_{m\ell k}\omega_k (x_k - x^C_k) \end{aligned} \tag{6.3}$$

The viscous stress tensor is given by

$$\tau_{km} = \frac{\nu + \nu_t}{Re_0}\left(\frac{\partial u_k^A}{\partial x_m} + \frac{\partial u_m^A}{\partial x_k}\right) \tag{6.4}$$

where ν is the kinematic viscosity, ν_t the eddy viscosity computed by the turbulence model and Re_0 the Reynolds number resulting from the non-dimensionalization of the equations. Finally, matrix Γ_{nm}^{-1} is the preconditioning matrix used to stabilize the system of PDEs and lead to a robust numerical solution, by giving appropriate values to the parameters α and β [12],

$$\Gamma_{nm}^{-1} = \begin{bmatrix} \dfrac{1}{\beta^2} & 0 & 0 & 0 \\ \dfrac{u_1^A + \alpha u_1^R}{\beta^2} & 1 & 0 & 0 \\ \dfrac{u_2^A + \alpha u_2^R}{\beta^2} & 0 & 1 & 0 \\ \dfrac{u_3^A + \alpha u_3^R}{\beta^2} & 0 & 0 & 1 \end{bmatrix} \tag{6.5}$$

6.2.2 The Spalart-Allmaras Turbulence Model

In this diploma thesis, all applications are simulated using the Spalart-Allmaras [13] turbulence model, implemented in the PUMA solver. In this model, the eddy viscosity is given by

$$\nu_t = \tilde{\nu} f_{v1} \tag{6.6}$$

$$\begin{aligned} &\frac{\partial \tilde{\nu}}{\partial \tau} + \frac{\partial (u_k^R \tilde{\nu})}{\partial x_k} - \frac{1}{Re_0 \sigma}\left\{\frac{\partial}{\partial x_k}\left[(\nu + \tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_k}\right] + c_{b_2}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}\right\} \\ &- c_{b_1}(1 - f_{t_2})\tilde{S}\tilde{\nu} + \frac{1}{Re_0}\left(c_{w_1} f_w - \frac{c_{b_1}}{\kappa^2} f_{t2}\right)\left(\frac{\tilde{\nu}}{\Delta}\right)^2 = 0 \end{aligned} \tag{6.7}$$

where Δ stands for the distance from the closest wall boundary. The parameters of eq. 6.7 are given by

$$
\begin{aligned}
&\chi = \frac{\tilde{\nu}}{\nu}, \quad f_{v_1} = \frac{\chi^3}{\chi^3 + c_{v_1}^3}, \quad f_{v_2} = 1 - \frac{\chi}{1 + \chi f_{v_1}}, \quad S = \sqrt{\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial u_m^A}{\partial x_\ell}\frac{\partial u_r^A}{\partial x_q}}, \\
&\tilde{S} = S + \frac{\tilde{\nu} f_{v_2}}{Re_0 \kappa^2 \Delta^2}, \quad f_w = g\left(\frac{1 + c_{w_3}^6}{g^6 + c_{w_3}^6}\right)^{\frac{1}{6}}, \quad g = r + c_{w_2}(r^6 - r), \\
&r = min\left(10, \frac{\tilde{\nu}}{Re_0 \tilde{S} \kappa^2 \Delta^2}\right), \quad \mu_t = \rho\tilde{\nu} f_{v_1}, \quad \tilde{\mu} = \rho\tilde{\nu}, \quad f_{t_2} = c_{t_3} e^{-c_{t_4}\chi^2}, \\
&c_{v_1} = 7.1, \quad c_{b_1} = 0.1355, \quad c_{b_2} = 0.622, \quad c_{w_1} = \frac{c_{b_1}}{\kappa^2} + \frac{1 + c_{b_2}}{\sigma}, \\
&c_{w_2} = 0.3, \quad c_{w_3} = 2.0, \quad \sigma = \frac{2}{3}, \quad \kappa = 0.41, \quad c_{t_3} = 1.2, \quad c_{t_4} = 0.5
\end{aligned}
\tag{6.8}
$$

6.2.3 Boundary Conditions

In order to solve the system of PDEs 6.1 and 6.7, appropriate boundary conditions should be defined and implemented. The GPU solver is equipped with a wide range of boundary condition options. In this section, the boundary conditions used in the numerical prediction of flows within turbomachine blade passages shown in chapter 7 are presented.

Wall Boundary Conditions:

In **low-Reynolds** turbulence modelling, the absolute velocity is set equal to the wall velocity

$$v_k^A = v_k^W, k = 1, 2, 3 \tag{6.9}$$

Variable $\tilde{v}$ is set equal to zero, i.e. $\tilde{v} = 0$.

In the case of the **high-Reynolds** approach, the Spalding's [14] expression is used. This equation models both the viscous sublayer and the logarithmic region of the turbulent boundary layer and is used for computing the non-dimensional velocity (u^+) given the non-dimensional height (y^+). The Spalding's expression reads,

$$y^+ = u^+ + e^{-\kappa B}\left[e^{-\kappa u^+} - 1 - \kappa u^+ - \frac{(\kappa u^+)^2}{2} - \frac{(\kappa u^+)^3}{6}\right] \tag{6.10}$$

where $\kappa = 0.41$ is the Von Kármán's constant , $B = 5.5$ and

$$\begin{aligned} y^+ &= \frac{yu_\tau}{\nu} \\ u^+ &= \frac{u}{u_\tau} \end{aligned} \tag{6.11}$$

In 6.11, y is the distance from the wall surface and u_τ is the friction velocity. The shear stresses on the wall are computed based on u_τ.

The mean flow equations boundary condition for the wall shear stress tensor in high-Reynolds grids cannot be computed by eq. 6.4 because the grid isn't sufficiently fine for the differentiation of the velocity. Thus, the wall function method is employed, that is based on the fact that the first node of the grid, next to the wall, is in the log-law region, for both vertex-centered and cell-centered grids. The friction velocity u_τ can be computed at this node, through eqs. 6.10 6.11 using the velocity value of the mean flow equation's previous iteration. The boundary condition of the wall shear stresses of the flow equations is computed as a function of u_τ.

Eq. 6.12 proposed by [15] is used to compute v_t and through eq. 6.6 the boundary condition for $\tilde{v}$ of the Spalart-Allmaras model is computed.

$$v_t = v\kappa e^{-\kappa B}[e^{\kappa u^+} - 1 - \kappa u^+ - \frac{(\kappa u^+)^2}{2}] \tag{6.12}$$

Inlet Boundary Conditions:

At the inlet of the domain, the three velocity components must be available, while pressure is extrapolated from the interior of the domain. The former are

$$\begin{aligned} v_1^A &= |v_\ell^A| sin\theta_1 \\ v_2^A &= |v_\ell^A| cos\theta_1 sin\theta_2 \\ v_3^A &= |v_\ell^A| cos\theta_1 cos\theta_2 \end{aligned} \tag{6.13}$$

Thus, angles θ_1, θ_2 and the velocity magnitude $|v_\ell^A|$ are specified by the user.

Outlet Boundary Conditions:

At the outlet, one quantity must be specified and three are extrapolated from the interior. This quantity is the (static) back pressure. This pressure is a reference pressure, thus the static pressure value at any grid node is relative to the back pressure.

6.3 Post-processing

In order to analyze the CFD results, some post processing scripts were programmed in this thesis that compute useful metrics from the discrete flow field data. The post-processing of the CFD results is an indispensable part of an EA optimization workflow, since it computes the objective or constraint functions of candidate solutions. The following are some metrics of great importance during hydroturbine design-optimization.

6.3.1 Pressure Coefficient

The pressure coefficient is a dimensionless quantity that describes the static pressure distribution at each point along an airfoil. In 3D geometries, it usually needs to be computed along isospan blade sections, from hub to shroud. In hydrodynamic applications, it is computed as

$$C_p = \frac{p - p_{REF}}{\rho g H_u} \tag{6.14}$$

where p is the static pressure on the blade surface, p_{REF} the reference pressure at the outlet, also used as a boundary condition (section 6.2.3), g the gravitational acceleration and H_u the theoretical hydraulic head that corresponds to the total hydraulic energy that is absorbed by the turbine.

6.3.2 Outlet Velocity Profile

During a hydraulic design, it is important to quantify the outlet conditions of the rotor. The outlet of the rotor is the inlet to the draft tube that is a component of significant impact in terms of efficiency. To make the two components compatible, the velocity profile at the outlet of the rotor is computed and applied to the inlet of the draft tube, to compute the total efficiency of the turbine.

The outlets of the rotors analyzed in this thesis are all axially directed, so the outlet patch has a constant z coordinate. Thus, having constant z, the CFD data at the outlet are placed on a 2D surface patch containing the peripheral and radial direction. The velocity cylindrical components are computed along discrete radial zones r_i (fig. 6.4), and integrated along the peripheral direction, to compute a single velocity vector for each radial zone $\vec{c}(r_i)$.

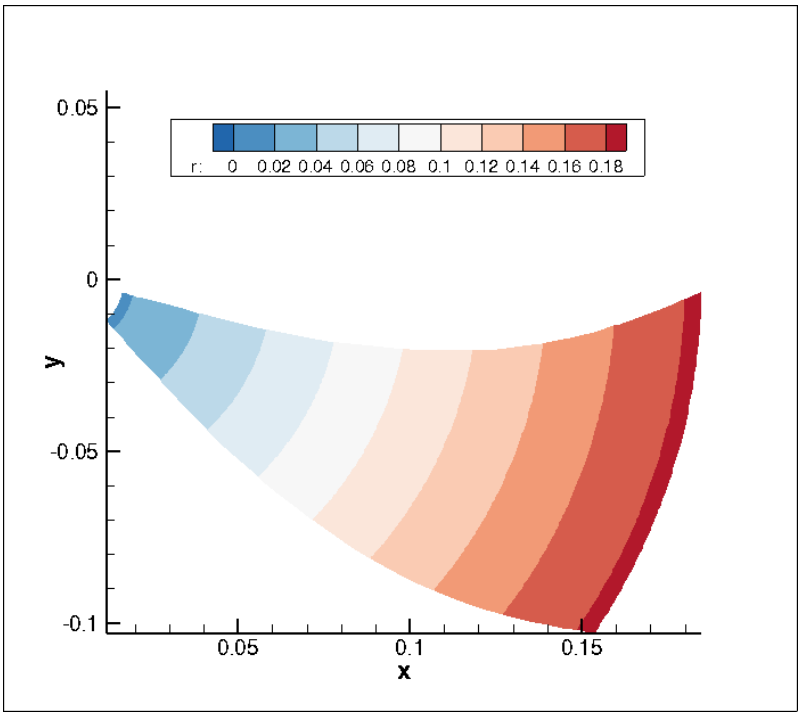


Figure 6.4: *The outlet patch of a runner is separated into radial zones r_i, so that a single velocity vector is computed for each zone $\vec{c}(r_i)$, by averaging the absolute velocity components of the CFD data that belong to these zones.*

To transform the Cartesian absolute velocity vector to cylindrical coordinates the cylindrical directions are computed

$$\begin{aligned} \vec{m} &= (0, 0, 1) \\ \vec{r} &= \frac{\vec{r}}{|\vec{r}|} \\ \vec{\theta} &= \vec{r} \times \vec{m} \end{aligned} \tag{6.15}$$

the projection of the computed absolute velocity vectors onto the unit vectors in the cylindrical coordinate system, given by the inner products of the velocity and the unit vectors of eq. 6.15 produce the absolute velocity cylindrical components, that are then non-dimensionalized by $\sqrt{2gH}$.

$$\begin{aligned} c_m &= \frac{\vec{u} \cdot \vec{m}}{\sqrt{2gH}} \\ c_r &= \frac{\vec{u} \cdot \vec{r}}{\sqrt{2gH}} \\ c_\theta &= \frac{\vec{u} \cdot \vec{\theta}}{\sqrt{2gH}} \end{aligned} \tag{6.16}$$

Integrating each velocity coordinate $k = 1, 2, 3$ along the peripheral direction, for each radial zone r_i, produces mean values of the velocity components at each zone,

$$\bar{c_k}(r_i) = \int_{r^i_{min}}^{r^i_{max}} \int_{u_{min}}^{u_{max}} c_k(u, r) du dr, \quad k = 1, 2, 3 \tag{6.17}$$

6.3.3 Efficiency

The efficiency of a turbomachine rotor is defined by the ratio of the power that is absorbed by the rotor shaft divided by the power that is provided by the working fluid

$$\eta = \frac{\vec{\omega} \cdot \vec{T}}{P_t, in - P_t, out} \tag{6.18}$$

Chapter 7

Validation of the RPT and GAT and Optimization

In this thesis, grid-to-CAD and CAD-to-grid blade geometry conversions take place. The Reverse Parameterization Tool (RPT) presented in chapter 4 converts a CFD grid to a CAD representation (GMTurbo representation), that can easily be optimized, since it consists of a small number of design variables. However, optimization using a CAD parameterization, requires the gemeration of a grid for the candidate geometry in order to compute the flow field and the objective functions using CFD software. The Grid Adaptation Tool (GAT) that is presented in chapter 5 was programmed to fulfill that need, creating a volume grid around the CAD geometry.

The RPT and GAT are tested and validated in this chapter. CFD runs are performed for both the initial and the reparameterized geometries; the latter has been generated with the RPT and the grid created by the GAT. In all cases, the CFD boundary conditions are computed according to the operating point and the solver uses the Spalart-Allmaras turbulence model with wall functions. The results demonstrate the level of accuracy both in geometry and flow solution terms. Three different geometries are studied and presented in order to highlight the capabilities of both the RPT and GAT.

7.1 Francis Runner

A Francis hydroturbine optimized below is a mixed flow type turbine the geometry of which can be seen in fig. 7.1. Reparameterizing the blade its runner leads to the

geometry of fig. 7.2, that needs to be as close to the initial as possible.

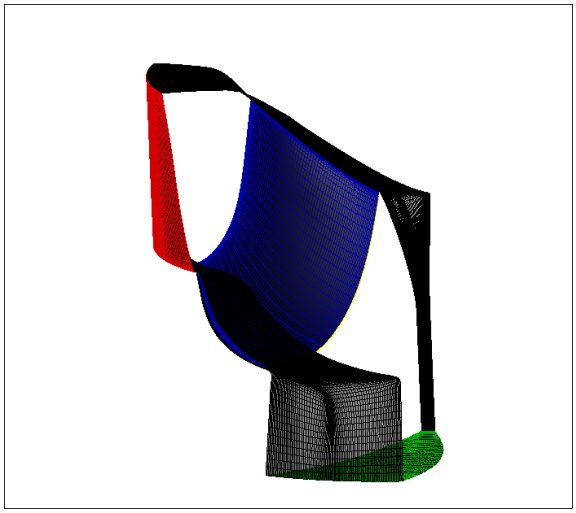


Figure 7.1: *Francis runner: Wall patches along with the inlet (red) and outlet (green) patches.*

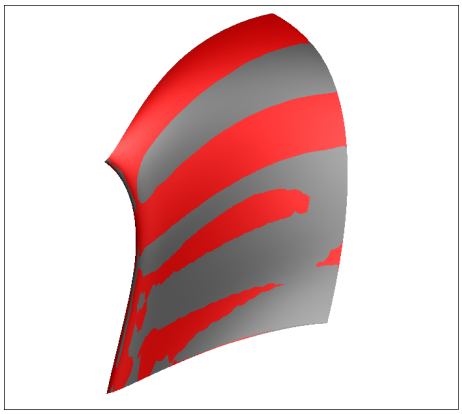


Figure 7.2: *Francis runner: Initial (grey) and reparameterized (red) geometries (using N=7 generatrices in the reparameterization), plotted together.*

In order to quantify the deviation between the two blades of fig 7.2, the field of the variables d given by

$$d = \sqrt{dx^2 + dy^2 + dz^2} \tag{7.1}$$

where dx, dy, dz is the x, y, z distance between initial nodal positions and the corresponding positions on the reparameterized blade. The results of such a comparison can be seen in fig. 7.3 and 7.4. It is obvious that the accuracy of the reparameterization is strongly connected to the number of generatrices chosen for the reparameterization.

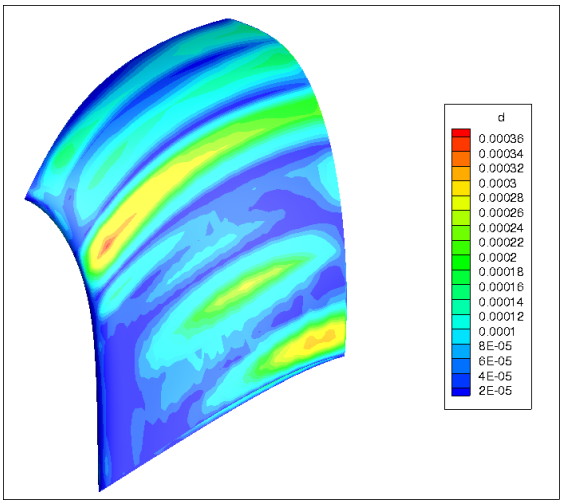


Figure 7.3: *Francis runner: Deviation of the two blades, using N=7 generatrices in the reparameterization.*

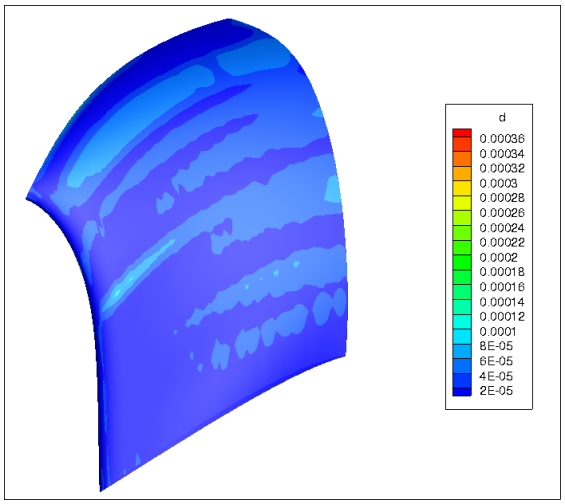


Figure 7.4: *Francis runner: Deviation of the two blades, using N=20 generatrices in the reparameterization. Much smaller deviations than in fig. 7.3 can be seen.*

In terms of CFD, runs for both cases, with the same initial conditions, were performed, and the post processing of the results is presented in figs.7.5 and 7.9. The reparameterized blade used in these CFD runs is approximated using N=10 generatrices. By comparing the flow profiles computed with the initial and reparameterized geometries, we easily conclude that they are very close to each other and their inevitable shape difference is so small that the flow practically does not "see" it.

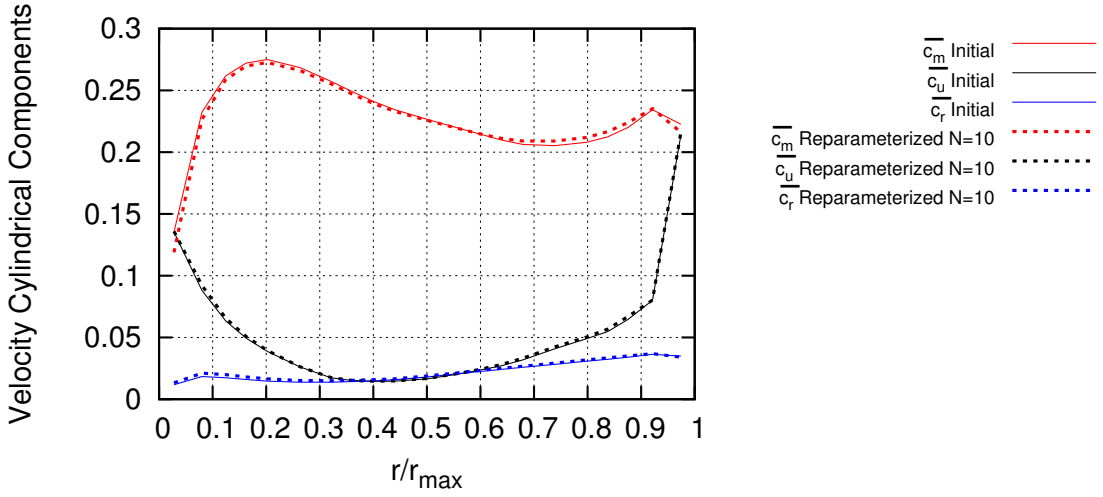


Figure 7.5: *Francis runner: Cylindrical components of the absolute velocity along the radius at the outlet of the initial and reparameterized runner.*

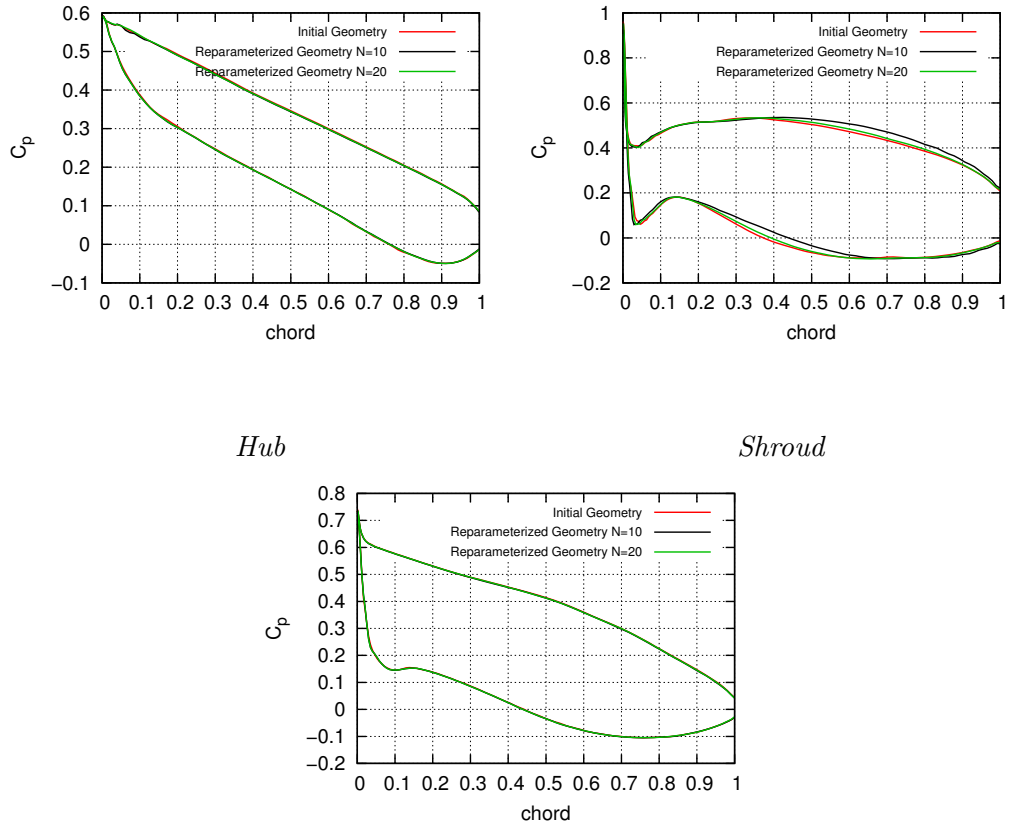


Figure 7.9: *Francis runner: Chordwise C_p profiles at three different blade spanwise postions of the initial two reparameterized runners, one for N=10 generatrices and one for N=20. The difference in the C_p profiles in shroud, is significantly reduced in the case of N=20.*

7.2 Propeller IGVs and Runner

This case is concerned with the flow field in a propeller hydroturbine, computed concurrently with the flow field around its inlet guide vanes, using the mixing plane method that is provided by the GPU solver when relative motion between the components of a turbine occurs (Rotor Stator Interface, RSI). This method circumferentially averages the values at the intersection plane computed from the one blade row, providing the other with a uniform in the circumferential direction flow field, achieving communication between the two domains.

The geometry of the two domains, guide vanes and the runner is seen in fig. 7.10 and 7.11 while the deviations of the two blades are seen in fig. 7.12, 7.13, 7.14, 7.15. Again, increasing the number of generatrices, increases the accuracy of the

reparameterization.

The CFD results presented in fig.7.19 and 7.20, concern the reparameterized geometries for both guide vanes and runner. The guide vanes are approximated with N=13 generatrices while the runner is approximated using N=11 generatrices. The comparison between the flow fields around the initial and the reparameterized blades again show that the flow is almost not affected by the slight geometry differences. It is interesting that this happens in this case as well since, in contrast to the Francis runner of section 7.1, there are two reparameterized blades (both rotor and stator).

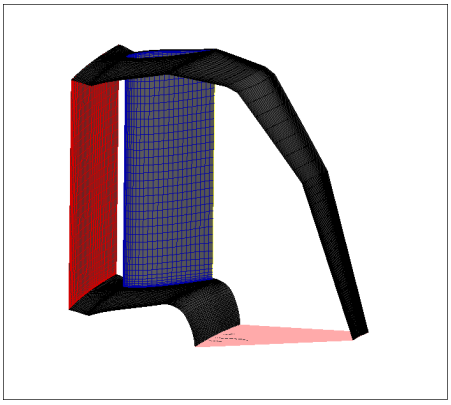


Figure 7.10: *Propeller IGVs and runner: Geometry of the guide vanes upstream to the runner.*

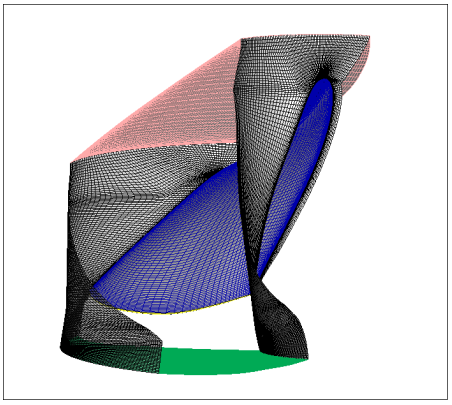


Figure 7.11: *Propeller IGVs and runner: Geometry of the runner.*

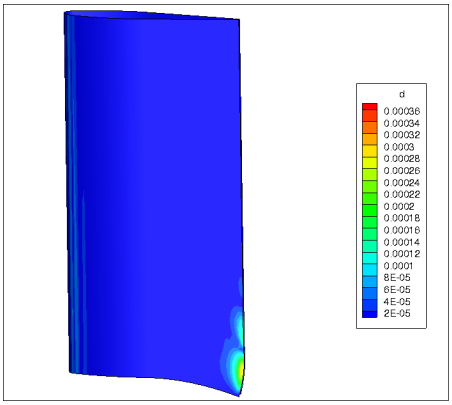


Figure 7.12: *Propeller IGVs and runner: Deviation of the two guide vanes, using N=8 generatrices in the reparameterization.*

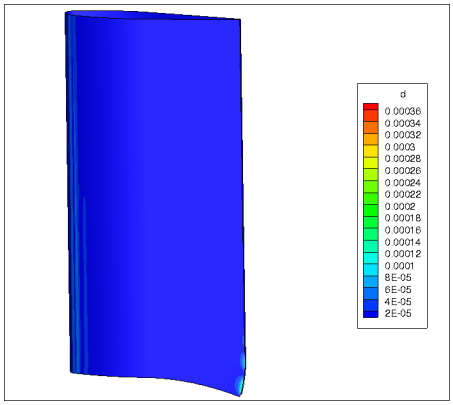


Figure 7.13: *Propeller IGVs and runner: Deviation of the two inlet guide vanes, using N=18 generatrices in the reparameterization. Much better agreement between the two geometries compared to the one obtained in fig. 7.12 with N=8 generatrices.*

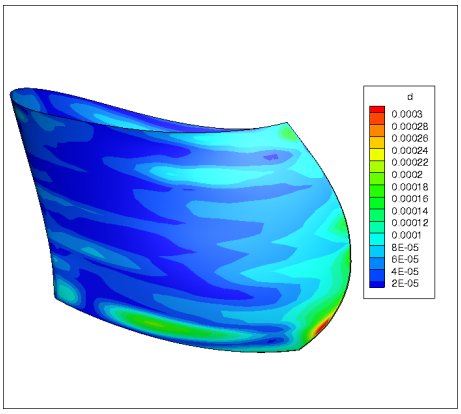


Figure 7.14: *Propeller IGVs and runner: Deviation of the two runner blades, using N=8 generatrices in the reparameterization.*

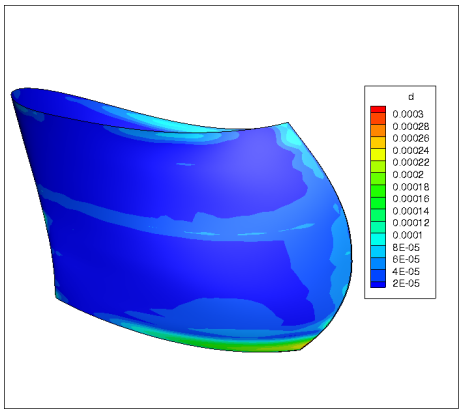
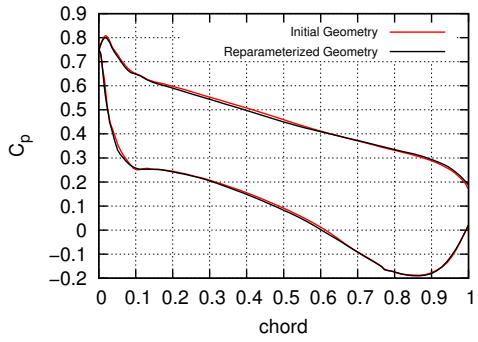
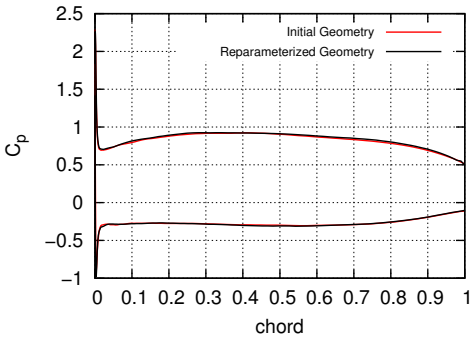


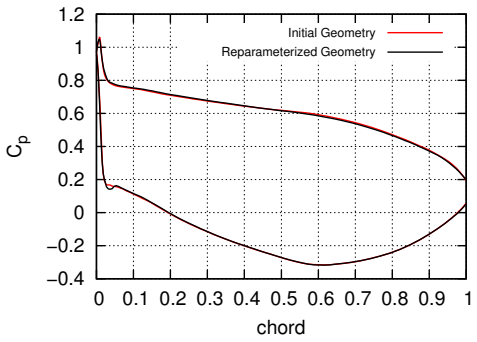
Figure 7.15: *Propeller IGVs and runner: Deviation of the two runner blades, using N=18 generatrices in the reparameterization.*



Hub



Shroud



Mid-span

Figure 7.19: *Propeller IGVs and runner: Chordwise C_p profiles at three different blade spanwise postions for the initial and reparameterized geometries.*

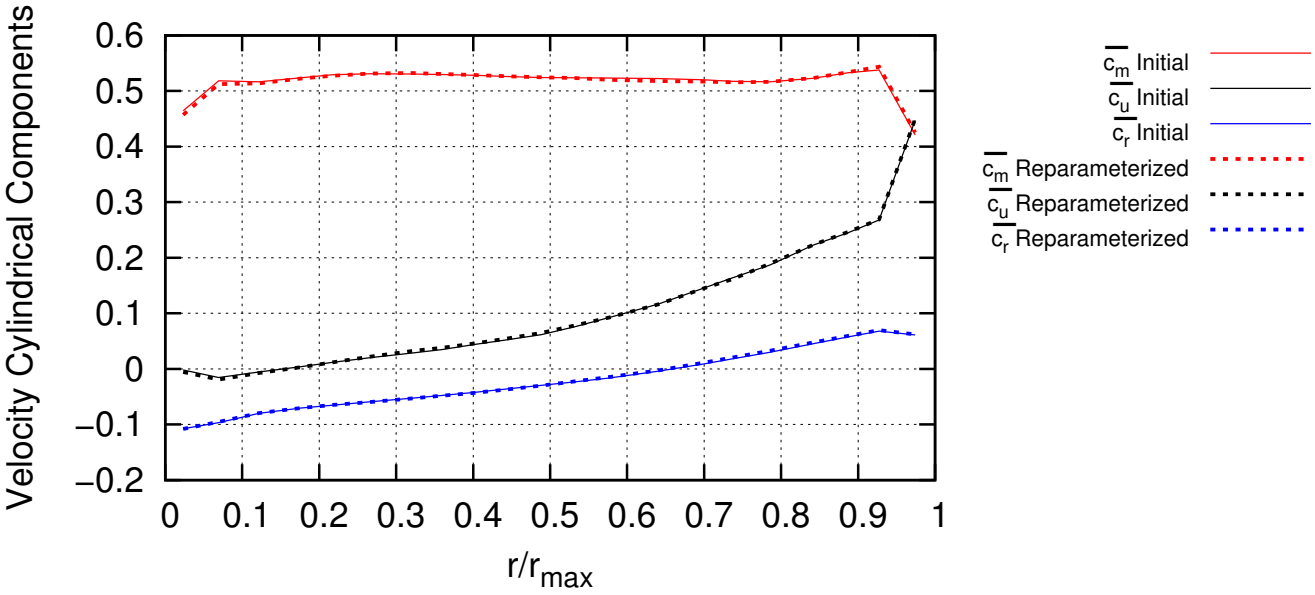


Figure 7.20: *Propeller IGVs and runner: Cylindrical components of the absolute velocity along the radius at the outlet of the runner.*

7.3 Optimization

The RPT, except from the flexibility it provides to alter the parameterization from a node-based to a CAD-based form, was mainly developed to support an optimization loop as a pre-processing step. It makes it possible to optimize a blade geometry, given in a node-based form, using the GMTurbo software. The benefits of this procedure are the following; first, the use of GMTurbo geometry generator within the optimization loop is advantageous in terms of design variable number because the geometry is described by a small number of parameters that also have great physical significance and are ideal for an optimization. Second, no need for extra geometry modifications is required (for example, optimizing the blade shape using the node-based parameterization must be followed by a smoothing software in order to restore a smooth geometry). However, the need of a grid generator or morpher to create a grid around the CAD geometry is indispensable.

In the previous section, it is shown that, in terms of geometry, the RPT and GAT provide a grid that is very close to the initial. Also, in terms of CFD, the results appear to be sufficiently close. Having validated the two softwares, their integration within an EA based optimization workflow is safe in terms of results accuracy.

In this section, three optimization runs of the case presented in section 7.2 are carried out, using different sets of design variables. The objective function in all runs is

the maximization of the efficiency of the Guide Vanes - Runner domain as a whole (defined in section 6.3.3). A $(\mu, \lambda) = (10, 20)$ EA with 10 parents and 20 offspring assisted by metamodels is used. The latter (Radial Basis Function Networks) are activated after the first 30 CFD evaluations have been completed and stored in the DB. Then, on each generation the offspring population members are approximated based on a local metamodel and the top 2 promising members (based on the metamodel prediction) are re-evaluated on the CFD tool. A termination criterion of 200 evaluations on the CFD was set for the first two runs and a criterion of 250 evaluations for the third. Each evaluation has a total duration of approximately 15 min. and the CFD solver runs on two NVIDIA Tesla K40 GPUs.

7.3.1 Guide Vane Optimization: 16 design variables

In the first optimization, the design variables are 16: 8 spanwise control points for the β distribution of the LE and 8 for the β distribution of the TE. The convergence history of the optimization is presented in fig. 7.21 and the resulting geometry compared to the baseline one can be seen in 7.22.

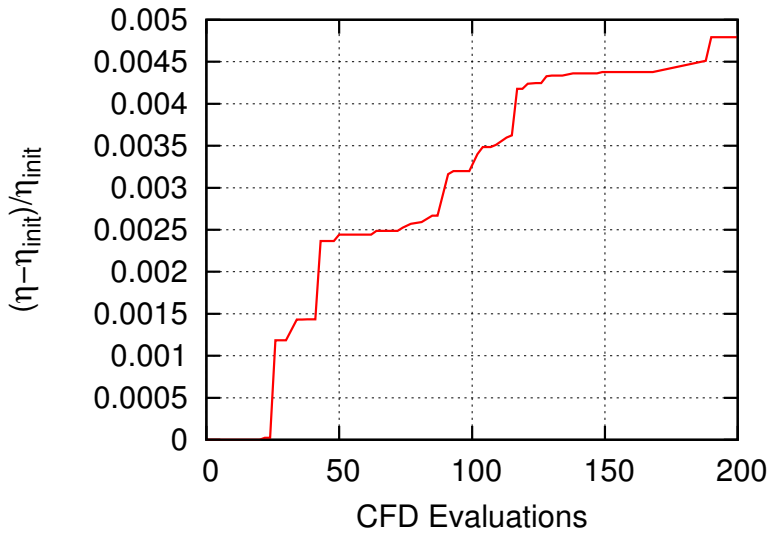


Figure 7.21: *EA-based optimization of the IGV blades of the propeller hydroturbine (16 design variables): Increase in the efficiency in terms of the number of evaluations of the EA.*

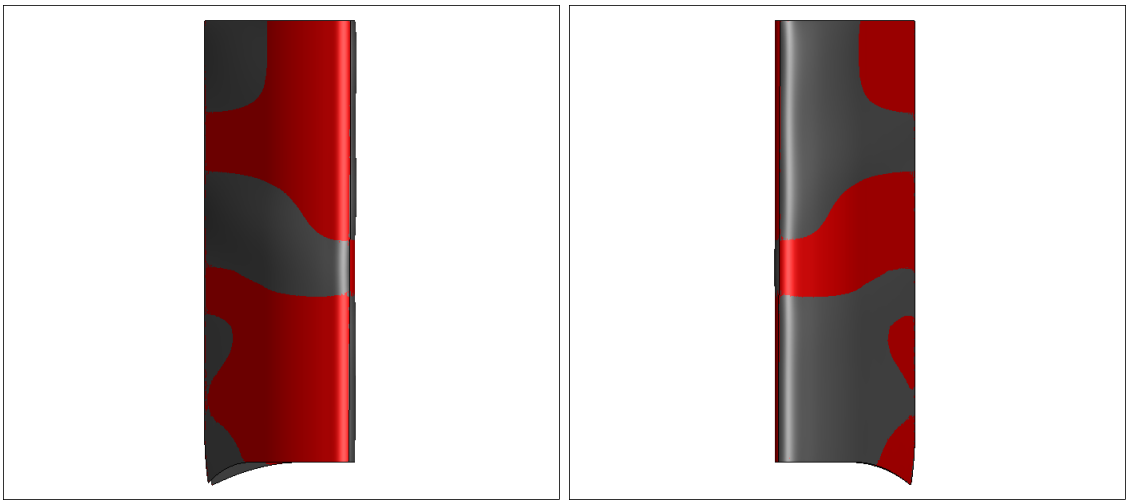


Figure 7.22: *EA-based optimization of the IGV blades of the propeller hydroturbine (16 design variables): Initial (red) and optimized (gray) IGV blades.*

7.3.2 Guide Vane Optimization: 32 design variables

In the second optimization, the number of the design variables is increased. The thickness factors along the span of the blade are added as design variables resulting to a total of 32 design variables ($8 \times \beta_{LE}$, $8 \times \beta_{TE}$, $8 \times t_{f,PS}$, $8 \times t_{f,SS}$). The convergence history of the optimization is presented in fig. 7.23. The optimized blade shape (fig. 7.24) is further deformed, in comparison with the previous optimization.

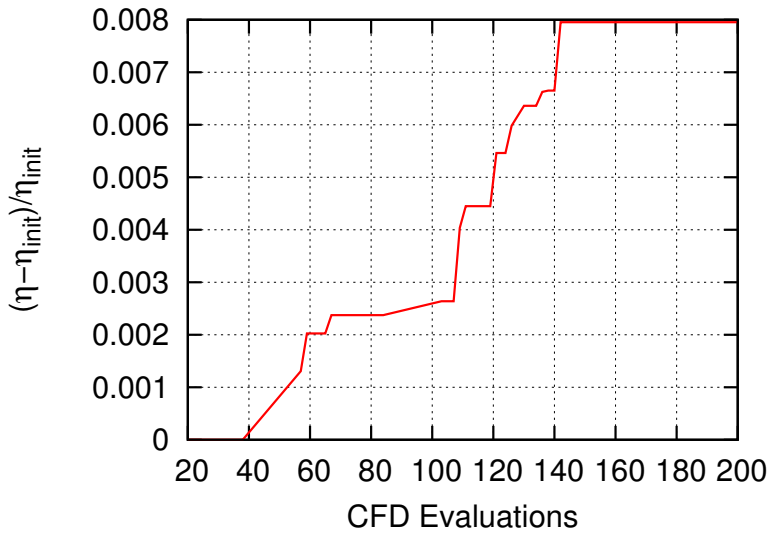


Figure 7.23: *EA-based optimization of the IGV blades of the propeller hydroturbine (32 design variables): The increase in the efficiency in terms of the number of evaluations of the EA.*

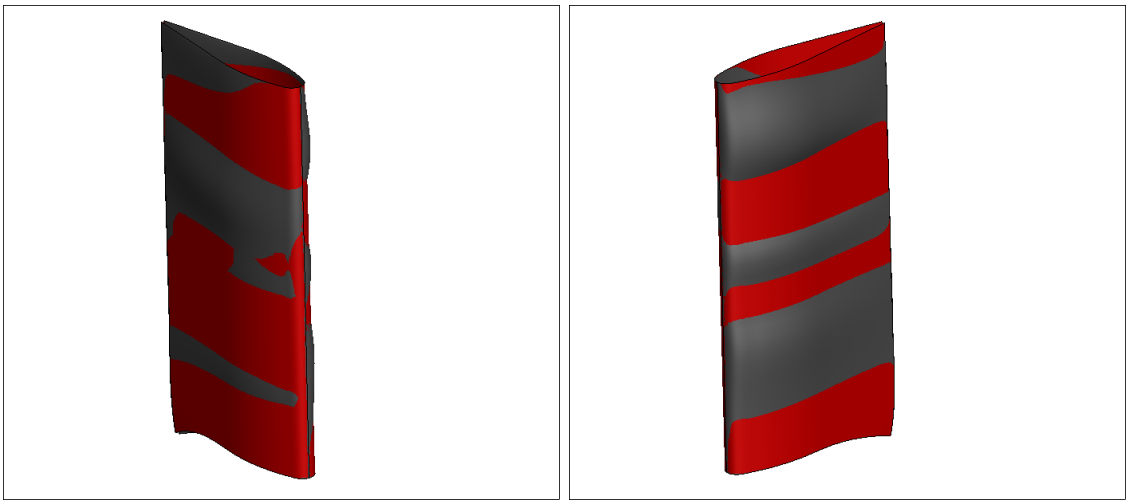


Figure 7.24: *EA-based optimization of the IGV blades of the propeller hydroturbine (32 design variables): Initial (red) and the optimized (gray) IGV blades.*

Comparing the two optimized geometries as resulted after the same number of evaluations one can observe that the increase in the objective function value is almost doubled when the design variable number is increased. The increase in the design variables practically increases the EA search space and makes it possible to capture a better result.

The optimized geometries appear to have a spanwise displacement because the design variables are spanwise parameters of the blade. Any manufacturing constraint can be applied to the geometry shape by bounding the shape deformation, that EASY can handle. Also, by choosing a suitable set of design variables, the desirable shape can result. The great value of the GMTurbo parameterization software, within an optimization, is highlighted here. The design variables have a physical sense that can provide the designer with the ability to roughly predict the displacements the blade will undergo during the stochastic optimization and, thus, chose to optimize those variables that are significant for the application.

7.3.3 Guide Vane and Runner Optimization: 32 design variables

In the third case, shape optimization of both the guide vanes and the runner occurs. The design variables concern both blades. Specifically, a set of 16 design variables ($8 \times\beta_{LE}$, $8 \times\beta_{TE}$) are used to optimize the Guide Vane geometry, and the same 16 design variables ($8 \times\beta_{LE}$, $8 \times\beta_{TE}$) are used for the runner. The convergence history of the optimization is presented in fig. 7.25. The optimized guide vane and runner blade are also shown in figs. 7.26 and 7.27.

Comparing this optimization with the optimization presented in section 7.3.1 it is noted that for equivalent number of evaluations, the shape optimization of both the runner and the guide vanes, provides a better result. The contribution of the runner however is not very drastic in terms of efficiency increase. That is due to the fact that the limits of the guide vane design variables are set to $\pm 5^{\circ}$ of the initial geometry values, while the limits of the runner design variables are set to $\pm 2^{\circ}$ of the initial geometry values. Consequently, in this optimization the guide vanes have a greater ability to deform and thus, increase the efficiency.

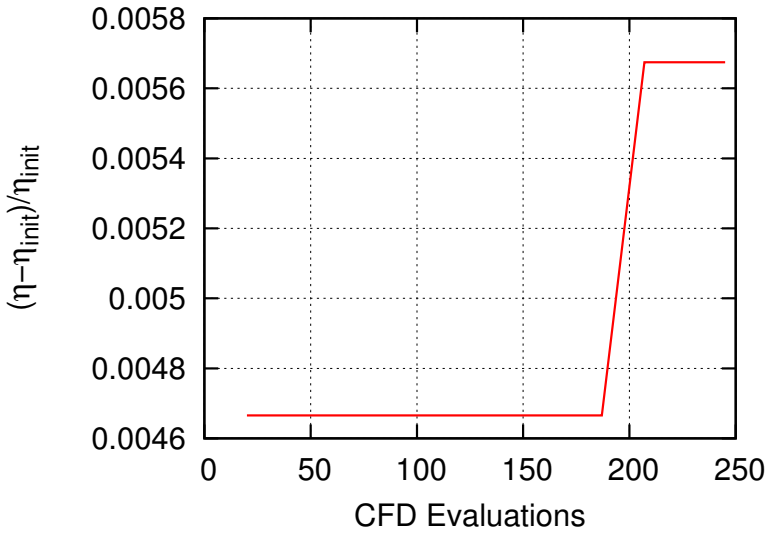


Figure 7.25: *EA-based optimization of both the IGV blades and propeller hydroturbine (32 design variables): The increase in the efficiency in terms of the number of evaluations of the EA.*

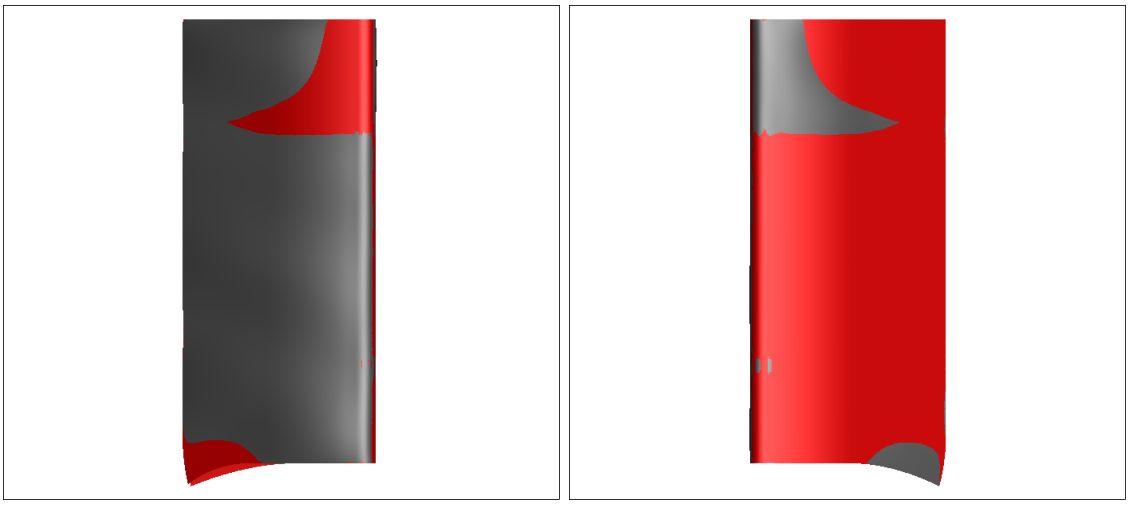


Figure 7.26: *EA-based optimization of both the IGV blades and propeller hydroturbine (32 design variables): Initial (red) and the optimized (gray) guide vanes.*

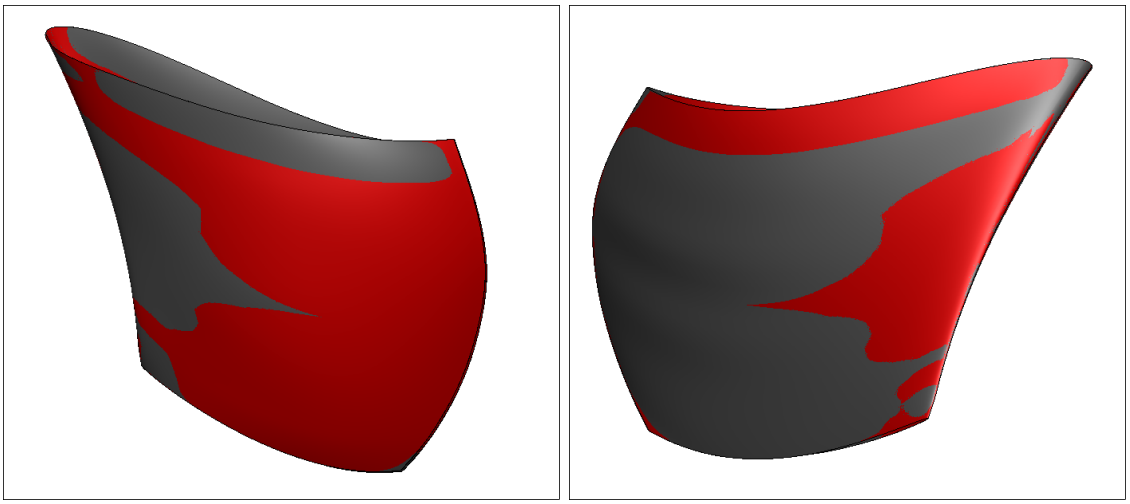


Figure 7.27: *EA-based optimization of both the IGV blades and propeller hydroturbine (32 design variables): Initial (red) and the optimized (gray) runner blades.*

Chapter 8

Overview and Conclusions

8.1 Overview

In this diploma thesis, software that transforms blade geometries from one parameterization method to another is developed, programmed and tested. The distinction between node-based and CAD-based methods is highlighted. A node-to-CAD method is programmed to transform a CFD grid into a GMTurbo form and is referred to as the Reverse Parameterization Tool (RPT). To create a grid around the GMTurbo geometry, the Grid Adaptation Tool (GAT) is programmed that adapts the initial CFD grid to the GMTurbo surfaces. **Using the RPT and GAT, it is made possible to optimize the shape of a blade, that is initially provided in a node-based (CFD grid) form, using the GMTurbo parameterization, that was not possible so far.**

Benefits are as follows:

- Most geometries are exchanged via a CFD grid, because it is a parameterization that is compatible with any CFD software. Thus, to optimize such geometries, the node-based parameterization has to be used, that is sometimes expensive in terms of design variables and requires extra software to smoothen the geometry roughness. Having the geometry in CAD form is much more general/portable and, thus, more convenient during design.
- The GMTurbo parameterization is ideal for use in optimization workflows, since the parameters it uses are much less than those of a nodal parameterization and have physical meaning. Thus, any optimization that can be carried out using GMTurbo is preferable.

To be more specific, in this diploma thesis, the following were presented, developed

and validated:

• First, an introduction to the basics of the parametric geometry was made that is used extensively in Computer Aided Design. NURBS curves and surfaces are the main form of representation in most CAD techniques, including the GMTurbo software of PCOpt/NTUA. Thus, a detailed introduction to the mathematical background of these geometric notions is indispensable. Since the software programmed in this thesis is implemented in the GMTurbo parameterization software, the latter is described in detail in this thesis.

• The main topic of the thesis is the development and programming of the RPT and GAT. The RPT is the software that converts a blade geometry given in CFD grid format, to a GMTurbo format. That way the geometry can be optimized through the optimization workflow that is used in PCOpt/NTUA, using the GMTurbo geometry generation software. The RPT performs various geometrical operations on the grid geometry to produce the final GMTurbo format, that are described in chapter 4 and in more detail in the appendices.

The Grid Adaptation Tool (GAT) is also developed in order to perform CFD simulations on the GMTurbo reparameterized geometry. It adapts the initial grid to the GMTurbo geometry using the 2D and 3D Spring Analogy Morphers.

• To validate the RPT and GAT, three different turbine blades are reparameterized. A Francis blade, a propeller blade and a Guide Vane are all transformed to GMTurbo and Grid Adaptation is performed around these reparameterized blades using GAT. Comparisons between the initial and reparameterized geometries, in terms of geometry and CFD results are performed showing that the results are similar, thus the reparameterization is accurate. After validating the RPT and the GAT, the GAT is integrated to the optimization workflow, and the hydrodynamic optimization of the of a node-based parameterized guide vane, using the GMTurbo software is performed, proving that the optimization of a geometry given in CFD grid is now possible using the RPT.

8.2 Proposals For Future Work

During the development of this diploma thesis, ideas for further expansion of the presented tools came up, that can enhance their abilities and make them compatible with even more applications. Some of these are

- The development of the RPT for grids that have a different structure than the one presented in section 4.1. The RPT is programmed for meshes with trailing edges of a specific type (blunt). Thus, different TE types can be included in the method. It would also be useful to adapt the method to unstructured grids.

- The development and integration of an extended RPT into the GMTurbo for other components, like ducts, the parameterization of which is also included in GMTurbo.
- The development-extension of the RPT for conversions of other parametric types to GMTurbo form (e.g. NURBS Surfaces to GMTurbo).

Appendix A

Conformal Mapping

For a mapping of a surface $\mathbf{S}(u,v)$ to a surface $\mathbf{P}(u,v)$ through the transformation function $\Phi : \mathbf{S}(u,v) \mapsto \mathbf{P}(u,v)$ to be conformal, it is sufficent to prove that there exists a function $c(u,v) > 0$ such that

$$\bar{g}_{ij}(u,v) = c(u,v)g_{ij}(u,v), \quad for i,j \in 1,2 \tag{A.1}$$

where g_{ij} and $\bar{g}_{ij}$ the coefficients of the first fundamental form of $\mathbf{S}(u,v)$ and $\mathbf{P}(u,v)$ [8]. The most important property of every conformal mapping is the angle preservation property. *Conformal mappings preserve the magnitude and direction of the angle between two curves.* [8]

The mapping of a surface of revolution $\mathbf{S}(u,\theta) = (r(u)cos\theta, r(u)sin\theta, z(u))$ to a (m,θ) plane is given by the transformation

$$\Phi : (r(u)cos\theta, r(u)sin\theta, z(u)) \mapsto (m(u), \theta) \tag{A.2}$$

with

$$m(u) = \int_0^u \frac{\sqrt{r_u(t)^2 + z_u(t)^2}}{r(t)} dt \tag{A.3}$$

The mapping A.2 is conformal according to eq A.1.

For a point $\mathbf{s}$ that lies on the revolved surface $\mathbf{S}(u,v) = (r(u)cos\theta, r(u)sin\theta, z(u))$ the partial derivatives w.r.t u and θ are given by

$$\begin{aligned} \frac{\partial \mathbf{s}}{\partial u} &= \Big(\frac{\partial r(u)}{\partial u} cos\theta, \frac{\partial r(u)}{\partial u} sin\theta, \frac{\partial z(u)}{\partial u} \Big) \\ \frac{\partial \mathbf{s}}{\partial \theta} &= \Big(-r(u)sin\theta, r(u)cos\theta, 0 \Big) \end{aligned} \tag{A.4}$$

and for a $\mathbf{p}$ point on the (m, θ) plane $\mathbf{p}(u, \theta) = (m, \theta)$ the partial derivatives w.r.t u and θ are

$$\begin{aligned} \frac{\partial \mathbf{p}}{\partial u} &= \Big(\frac{\sqrt{r_u(t)^2 + z_u(t)^2}}{r(t)}, 0 \Big) \\ \frac{\partial \mathbf{p}}{\partial \theta} &= \Big(0, 1 \Big) \end{aligned} \tag{A.5}$$

the first fundamental coefficients of $\mathbf{s}$ are given by

$$\begin{aligned} E_s &= \frac{\partial \mathbf{s}}{\partial u} \cdot \frac{\partial \mathbf{s}}{\partial u} = \Big(\frac{\partial r(u)}{\partial u} \Big)^2 + \Big(\frac{\partial z(u)}{\partial u} \Big)^2 \\ F_s &= \frac{\partial \mathbf{s}}{\partial u} \cdot \frac{\partial \mathbf{s}}{\partial \theta} = 0 \\ G_s &= \frac{\partial \mathbf{s}}{\partial \theta} \cdot \frac{\partial \mathbf{s}}{\partial \theta} = r^2(u) \end{aligned} \tag{A.6}$$

while the first fundamental form coefficients of $\mathbf{p}$, by

$$\begin{aligned} E_p &= \frac{\partial \mathbf{p}}{\partial u} \cdot \frac{\partial \mathbf{p}}{\partial u} = \frac{\Big(\frac{\partial r(u)}{\partial u} \Big)^2 + \Big(\frac{\partial z(u)}{\partial u} \Big)^2}{r^2(u)} \\ F_p &= \frac{\partial \mathbf{p}}{\partial u} \cdot \frac{\partial \mathbf{p}}{\partial \theta} = 0 \\ G_p &= \frac{\partial \mathbf{p}}{\partial \theta} \cdot \frac{\partial \mathbf{p}}{\partial \theta} = 1 \end{aligned} \tag{A.7}$$

from eqs. A.6 and A.7 the relationship between the coefficients

$$\begin{aligned} E_p &= \frac{E_s}{r^2(u)} \\ F_p &= \frac{F_s}{r^2(u)} = 0 \\ G_p &= \frac{G_s}{r^2(u)} \end{aligned} \tag{A.8}$$

that prove that the mapping A.2 is conformal with conformal factor $\dfrac{1}{r^2(u)}$. The square of the distance between two points $\mathbf{s}$ and $\mathbf{s} + d\mathbf{s}$ on surface $\mathbf{S}$ is given by the

first fundamental form of $\mathbf{s}$

$$I_s = \ell_s^2 = d\mathbf{s} \cdot d\mathbf{s} = E_s du^2 + 2F_s du d\theta + G_s d\theta^2 \tag{A.9}$$

and the square of the distance between two points $\mathbf{p}$ and $\mathbf{p} + d\mathbf{p}$ on the (m, θ) plane is given by the first fundamental form of $\mathbf{p}$

$$I_p = \ell_p^2 = d\mathbf{p} \cdot d\mathbf{p} = E_p du^2 + 2F_p du d\theta + G_p d\theta^2 \tag{A.10}$$

Combining eq. A.9 and eq. A.10, it is easily proved that the relationship between the magnitudes on the two surface representations $\mathbf{S}(u, \theta)$ and (m, θ) is given by

$$\ell_s = r(u)\ell_p \tag{A.11}$$

Consequently, the mapping used to transform a revolved surface into a (m, θ) plane, given by eq. A.2, being conformal, preserves angles and scales the magnitudes by a factor of $r(u)$ (eq. A.11).

Appendix B

Surface Structured Grid-NURBS Surface Intersection

During reverse parameterization, the need to separate the blade to iso-span sections from hub to shroud comes up. To do this, the intersection between the blade surface grid, and the iso-span revolved surfaces has to be computed.

The intersection points between a NURBS revolved surface and the structured surface quad grid (of a blade) are computed. First, the grid nodes of the blade are placed into a 2D matrix $[i, j]$, using the structured connectivity. The lines and rows of the matrix correspond to spanwise (constant-i) or streamwise(constant-j) sets of nodes as demonstrated in fig.B.1.

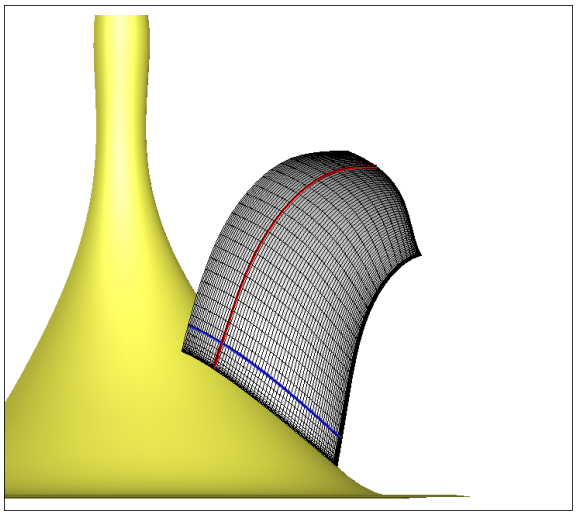


Figure B.1: *One spanwise set of nodes in blue and one streamwise set of nodes in red.*

These sets of nodes create lines that are considered to be composed of straight

line segments between the nodes (1^{st} degree interpolation). The algorithm that has been developed searches for an intersection of every line segment of a spanwise line (**AB** in fig.B.1) with the revolved surface. Once it is found, the intersection for the next iso-stream line is sought. Therefore, the problem of grid-surface intersection is converted into a line-surface intersection problem. The line-surface intersection problem is described below.

Let **A** and **B** be two neighbouring nodes along a spanwise line (fig. B.2). The analytical equation of the **AB** line in 3D space is:

$$\left\{\begin{array}{l} f(x) = \dfrac{x - x_A}{x_B - x_A} \\ g(y) = \dfrac{y - y_A}{y_B - y_A} \\ h(z) = \dfrac{z - z_A}{z_B - z_A} \end{array}\right\} \tag{B.1}$$

$$f(x) = g(y) = h(z)$$

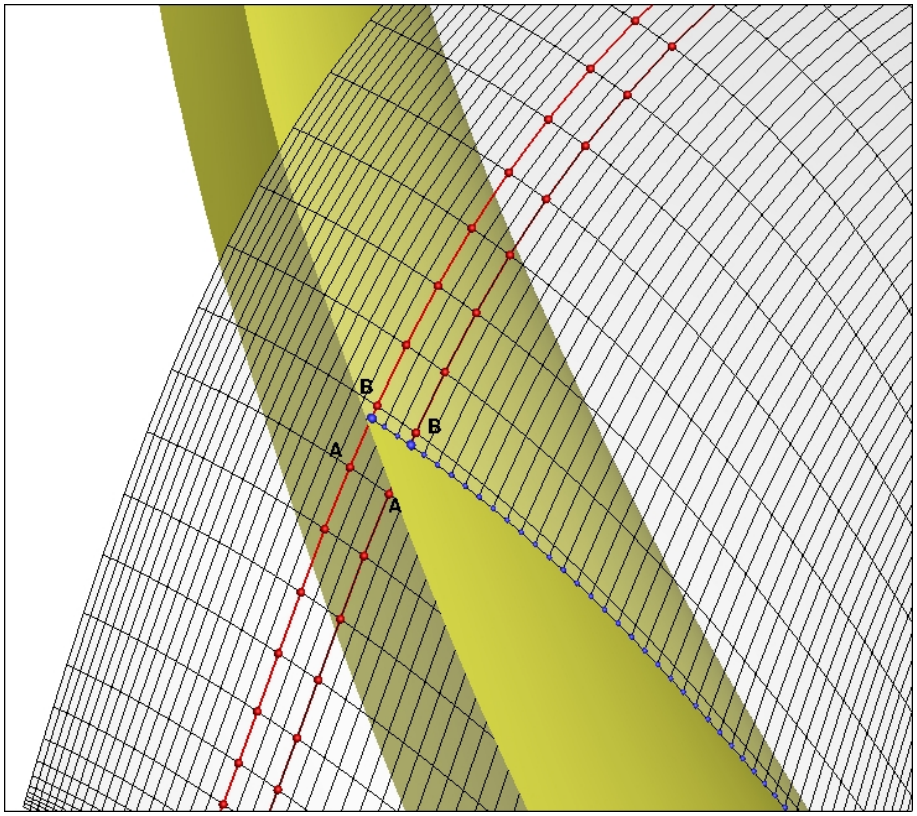


Figure B.2: *Grid Line - Surface intersection*

Also, if x, y and z belong to the revolved surface $\mathbf{S}(u, v)$ they can be written as

$x(u,v)$, $y(u,v)$, $z(u,v)$ and, thus, $f(x)$ becomes $f(u,v)$, $g(y)$ becomes $g(u,v)$ and $h(z)$ becomes $h(u,v)$. This transformation along with equation B.1 produces the non-linear two equation system:

$$\begin{Bmatrix} f(u,v) - g(u,v) = 0 \\ f(u,v) - h(u,v) = 0 \end{Bmatrix} \rightarrow \begin{Bmatrix} F(u,v) = 0 \\ G(u,v) = 0 \end{Bmatrix} \tag{B.2}$$

with the unknowns $\vec{u} = [u, v]$. Newton-Raphson iterative method is used to solve the non linear system. Thus, the jacobian of the matrix B.2 w.r.t. the unknowns is needed:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial F}{\partial u} & \dfrac{\partial F}{\partial v} \\ \dfrac{\partial G}{\partial u} & \dfrac{\partial G}{\partial v} \end{bmatrix} \begin{bmatrix} \dfrac{\partial x}{\partial u}\dfrac{1}{x_B - x_A} - \dfrac{\partial y}{\partial u}\dfrac{1}{y_B - y_A} & \dfrac{\partial x}{\partial v}\dfrac{1}{x_B - x_A} - \dfrac{\partial y}{\partial v}\dfrac{1}{y_B - y_A} \\ \dfrac{\partial x}{\partial u}\dfrac{1}{x_B - x_A} - \dfrac{\partial z}{\partial u}\dfrac{1}{z_B - z_A} & \dfrac{\partial x}{\partial v}\dfrac{1}{x_B - x_A} - \dfrac{\partial z}{\partial v}\dfrac{1}{z_B - z_A} \end{bmatrix} \tag{B.3}$$

The derivatives of the surface coordinates x, y, z w.r.t. u or v are computed using the equation of the partial derivatives of a NURBS surface w.r.t. its parameters that is found in litterature [6].

Appendix C

Constrained NURBS Approximation of Pressure and Suction Sides

Least Squares Curve Approximation:
The standard least-squares NURBS curve approximation algorithm is presented herein [6]. Let $\mathbf{Q_0}, ..., \mathbf{Q_m}$ be the set of $m+1$ 2D points to be approximated by a NURBS curve of a user defined degree p and a user defined set of $n+1$ control points given by

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P_i}}{\sum_{i=0}^{n} N_{i,p} w_i}, \quad u \in [0,1] \tag{C.1}$$

The $\mathbf{Q_0}, ..., \mathbf{Q_m}$ points are approximated in the least-squares sense, defining a function to minimize

$$F = \sum_{k=0}^{m+1} \Big(\mathbf{Q_k} - \mathbf{C}(\bar{u}_k) \Big)^2 \tag{C.2}$$

To avoid the nonlinear problem, only the control points $\mathbf{P_i}$ are sought. Weights w_i are set equal to 1, the parameters $\bar{u_k}$ that correspond to the approximation points are computed by normalizing the distance of the $\mathbf{Q_i}$ points to a $[0,1]$ interval (eq.C.3) and the knot vector is precomputed using methods that are proven to be appropriate for the approximation.

$$\begin{aligned} \bar{u_0} &= 0 \\ \bar{u_i} &= \bar{u_{i-1}} + \frac{|\mathbf{Q_i} - \mathbf{Q_{i-1}}|}{\sum_{i=0}^{m} |\mathbf{Q_i} - \mathbf{Q_{i-1}}|} \quad i = 1, ..., m \end{aligned} \tag{C.3}$$

The endpoint interpolation constraint naturally yields the first and last control

points:

$$\mathbf{P_0} = \mathbf{Q_0}$$
$$\mathbf{P_n} = \mathbf{Q_m} \tag{C.4}$$

and the system of $m+1$ equations is reduced by two. Thus by defining the function

$$\mathbf{R_k} = \mathbf{Q_k} - N_{0,p}(\bar{u_k})\mathbf{Q_0} - N_{n,p}(\bar{u_k})\mathbf{Q_m} \quad k = 1, ..., m-1 \tag{C.5}$$

$\mathbf{F}$ becomes

$$\begin{aligned}
\mathbf{F} &= \textstyle\sum_{k=0}^{m+1} \Big(\mathbf{Q_k} - \mathbf{C}(\bar{u}_k)\Big)^2 = \sum_{k=1}^{m-1} \Big(\mathbf{R_k} - \sum_{i=1}^{n-1} N_{i,p}(\bar{u_k})\mathbf{P_i}\Big)^2 \\
&= \textstyle\sum_{k=1}^{m-1} \Big(\mathbf{R_k} - \sum_{i=1}^{n-1} N_{i,p}(\bar{u_k})\mathbf{P_i}\Big) \cdot \Big(\mathbf{R_k} - \sum_{i=1}^{n-1} N_{i,p}(\bar{u_k})\mathbf{P_i}\Big) \\
&= \textstyle\sum_{k=1}^{m-1} \Big[\mathbf{R_k} \cdot \mathbf{R_k} - 2\sum_{i=1}^{n-1} N_{i,p}(\bar{u_k})(\mathbf{R_k} \cdot \mathbf{P_i}) \\
&\quad + \Big(\sum_{i=1}^{n-1} N_{i,p}(\bar{u_k})\mathbf{P_i}\Big)\Big(\sum_{i=1}^{n-1} N_{i,p}(\bar{u_k})\mathbf{P_i}\Big)\Big]
\end{aligned} \tag{C.6}$$

To minimize $\mathbf{F}$ with respect to the unknowns $\mathbf{P}_\ell, \ell = 1, ..., n$, derivatives $\dfrac{\partial \mathbf{F}}{\partial \mathbf{P}_\ell}$ are computed and set equal to zero, so

$$\begin{aligned}
\frac{\partial \mathbf{F}}{\partial \mathbf{P}_\ell} &= \textstyle\sum_{k=1}^{m-1} \Big(- 2N_{\ell,p}(\bar{u_k})\mathbf{R_k} + 2N_{\ell,p}(\bar{u_k}) \sum_{i=1}^{n-1} N_{i,p}(\bar{u_k})\mathbf{P_i}\Big) = 0 \\
&- \textstyle\sum_{k=1}^{m-1} N_{\ell,p}(\bar{u_k})\mathbf{R_k} + \sum_{k=1}^{m-1} \sum_{i=1}^{n-1} N_{\ell,p}(\bar{u_k})N_{i,p}(\bar{u_k})\mathbf{P_i} = 0 \\
&\textstyle\sum_{i=1}^{n-1} \Big(\sum_{k=1}^{m-1} N_{\ell,p}(\bar{u_k})N_{i,p}(\bar{u_k})\Big)\mathbf{P_i} = \sum_{k=1}^{m-1} N_{\ell,p}(\bar{u_k})\mathbf{R_k}
\end{aligned} \tag{C.7}$$

which is a two linear systems of $n-1$ equations of $2 \times n - 1$ unknown coordinates of the control points $\mathbf{P}_\ell$.

Constrained Least Squares Curve Approximation:
To approximate the pressure and suction side points of an airfoil with two distinct NURBS curves (one for each side), the first derivative continuity constraint at the LE point (which is the first point of both curves) emerges. If the constraint isn't included, the two curves will have point continuity (since leading edge belongs to both curves), but the slopes will be different at this point resulting to a non feasible geometry that will be pointy in the leading edge. To achieve a smooth blade there, the control points of the two curves are computed within the same system of equations, adding an extra constraint equation that implies equality of the first

derivatives at the leading edge for the two curves.

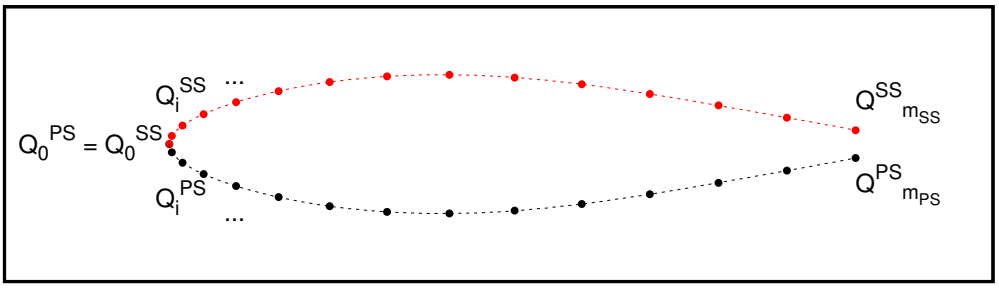


Figure C.1: *The PS and SS points to be approximated. The LE is common for the two sides, while the TE in each side is different, because the TE type of the 3D blade is "blunt" thus the blade is cut-off at the trailing edge, resulting to an open 2D airfoil.*

The $m_{PS}+1$ pressure side points $\mathbf{Q_0^{PS}}, ..., \mathbf{Q_{mPS}^{PS}}$ and the $m_{SS}+1$ suction side points $\mathbf{Q_0^{SS}}, ..., \mathbf{Q_{mSS}^{SS}}$ are going to be approximated. Let $n_{PS}+1$ be the number of control points of the pressure side, and $n_{SS}+1$ the number of control points of the suction side. Also, p_{PS} and p_{SS} the corresponding degrees. The endpoint interpolation gives the equations

$$\mathbf{P_0^{PS}} = \mathbf{P_0^{SS}} = \mathbf{Q_0^{PS}} = \mathbf{Q_0^{SS}} = LE$$

$$\mathbf{P_{nPS}^{PS}} = \mathbf{Q_{mPS}^{PS}} \tag{C.8}$$

$$\mathbf{P_{nSS}^{SS}} = \mathbf{Q_{mSS}^{SS}}$$

The first derivative of a p^{th} degree NURBS curve at the first point, $u = 0$, is given by the equation

$$\mathbf{C}'(u) = \frac{p}{u_{p+1}}(\mathbf{P_1} - \mathbf{P_0}) \tag{C.9}$$

Consequently, the constraint of first derivative continuity is given by the equation

$$\mathbf{C}'_{PS}(0) = -\mathbf{C}'_{SS}(0) \tag{C.10}$$

or

$$\frac{p_{PS}}{u_{pPS+1}}(\mathbf{P_1^{PS}} - \mathbf{P_0^{PS}}) = -\frac{p_{SS}}{u_{pSS+1}}(\mathbf{P_1^{SS}} - \mathbf{P_0^{SS}}) \tag{C.11}$$

In the least-square sense, k functions $\mathbf{F_k}$ are defined, the sum of which is minimized to provide the coordinates of the control points of the two curves, satisfying the above mentioned continuity constraint. For $k \in [1, m_{PS}+1]$ functions $\mathbf{F_k}$ concerns the approximation of the points of the pressure side ,for $k \in [m_{PS}, m_{PS}+m_{SS}-1]$ the same for suction side, and for $k = m_{PS}+m_{SS}$, $\mathbf{F_k}$ is the extra constraint equation

$$\mathbf{F_k} = \left(\mathbf{R_k^{PS}} - \mathbf{C^{PS}}(\bar{u}_k^{PS})\right)^2 \qquad 1 \le k \le m_{PS}+1$$

$$\mathbf{F_k} = \left(\mathbf{R_k^{SS}} - \mathbf{C^{SS}}(\bar{u}_k^{SS})\right)^2 \qquad m_{PS} \le k \le m_{PS}+m_{SS}-1$$

$$\mathbf{F_k} = \frac{p_{PS}}{u_{p_{PS}+1}}(\mathbf{P_1^{PS}} - \mathbf{P_0^{PS}}) + \frac{p_{SS}}{u_{p_{SS}+1}}(\mathbf{P_1^{SS}} - \mathbf{P_0^{SS}}) \qquad k = mps + mss \tag{C.12}$$

$$\mathbf{f} = \textstyle\sum_{k=1}^{m_{PS}+m_{SS}} \mathbf{F_k} =$$

$$\textstyle\sum_{k=1}^{m_{PS}-1}\left(\mathbf{R_k^{PS}} - \mathbf{C^{PS}}(\bar{u}_k^{PS})\right)^2 + \sum_{k=m_{PS}}^{m_{PS}+m_{SS}-1}\left(\mathbf{R_k^{SS}} - \mathbf{C^{SS}}(\bar{u}_k^{SS})\right)^2 \tag{C.13}$$

$$+\frac{p_{PS}}{u_{p_{PS}+1}}(\mathbf{P_1^{PS}} - \mathbf{P_0^{PS}}) + \frac{p_{SS}}{u_{p_{SS}+1}}(\mathbf{P_1^{SS}} - \mathbf{P_0^{SS}})$$

and, by computing the derivatives $\dfrac{\partial \mathbf{f}}{\partial \mathbf{P}_\ell}$ and setting them equal to zero, two systems (2D points, one system for each coordinate) of $n_{PS}+n_{SS}-2$ equations with $n_{PS}+n_{SS}-2$ unknowns each, that can readily be solved.

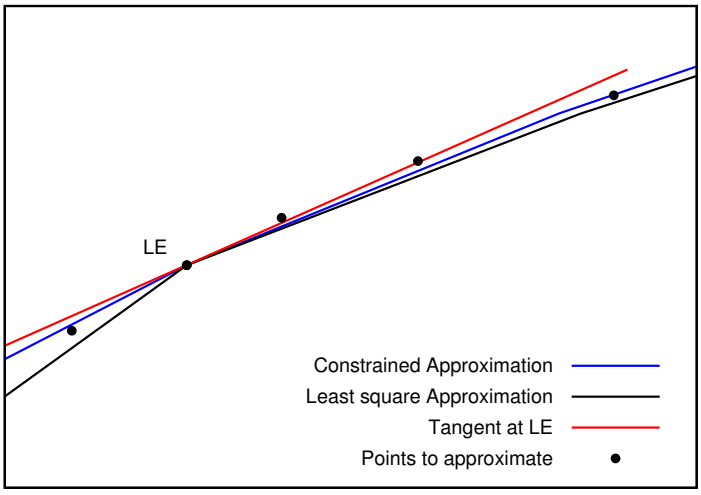


Figure C.2: *The same set of points are approximated with both standard and constrained approximation methods, presented with the same number of control points and degrees. The tangent equality at the leading edge, with the constrained method is obvious.*

Appendix D

Computation of the Mean Camber Line of an Existing Airfoil

The mean camber line is the line joining the leading and trailing edges of an airfoil, equidistant from the upper and lower sides. To compute the mean camber line of an existing airfoil, a geometric method was developed. This method requires the parametric representation of the two sides of the airfoil, pressure and suction (two NURBS curves) and provides a set of ordered points, from the LE to the TE, that lie in the middle of the two sides.

Let N be the number of the points of the mean camber line to be computed. One of the airfoil sides (whichever) is divided into N points ($\mathbf{A_0}...\mathbf{A_N}$) and the opposite, to $k \times N$ points ($\mathbf{B_0}...\mathbf{B_{k\times N}}$) where k is a positive integer. The attempt to produce a set of $\mathbf{M}_0, ..., \mathbf{M}_N$ points that belong to the mean camber line is made, with a "trial and error" method. The first point on the mean camber line ($\mathbf{M_0} = \mathbf{A_0} = \mathbf{B_0}$) is, of course, the leading edge. Starting from point $\mathbf{A_1}$(fig. D.1) of the N-discretized side, line ($\mathbf{A_1B_1}$) connecting $\mathbf{A_1}$ to the first point on the opposite side $\mathbf{B_1}$ is created. The midpoint $\tilde{\mathbf{M}}$ of this line belongs to the mean camber line, only if $\mathbf{A_1B_1}$ is orthogonal (within a certain margin) to the tangent of the mean camber line at point $\tilde{\mathbf{M}}$. The inner product

$$\left(\frac{\mathbf{M_0}\tilde{\mathbf{M}}}{|\mathbf{M_0}\tilde{\mathbf{M}}|}\right) \cdot \left(\frac{\mathbf{A_1B_1}}{|\mathbf{A_1B_1}|}\right) < e \tag{D.1}$$

where e is a very small number. If this statement is true, point $\tilde{\mathbf{M}}$ becomes point $\mathbf{M_1}$ and $i = i + 1$,i the counter of the points $\mathbf{A}_i$. Else, if orthogonality isn't reached yet, $j = j + 1$, where j is the counter of the points $\mathbf{B}_j$ and the same step is repeated.

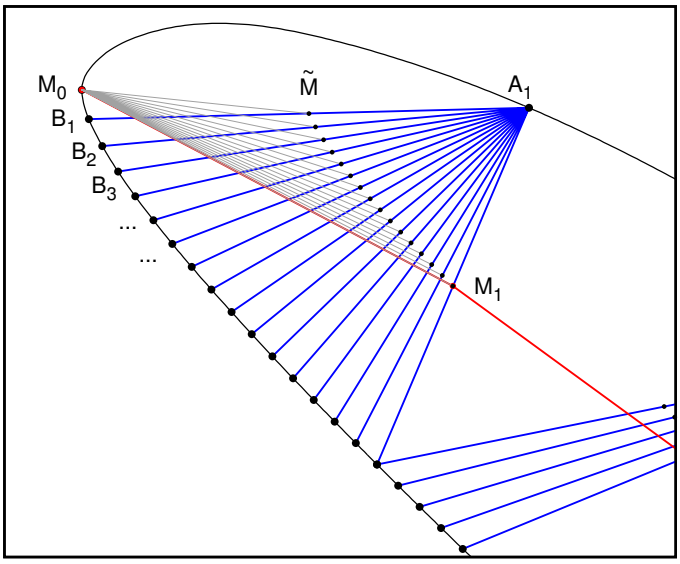


Figure D.1: *The proposed algrithm. For every* $\mathbf{A}_i$*, a* $\mathbf{B}_j$ *for which* $\vec{AB} \cdot \vec{M_{i-1}\tilde{M}} = 0$ *is sought.*

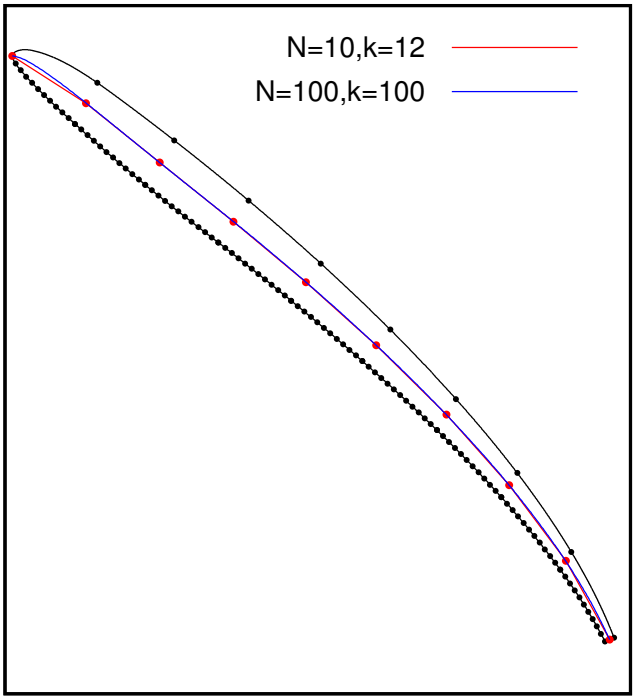


Figure D.2: *The resulting mean camber lines. The mean camber line in red results from a coarser discretization and is less accurate than the blue line.*

Appendix E

Constrained Point Approximation with Cubic Bézier Curve

Having two NURBS curves for pressure and suction side of the airfoil ($PS(u)$ and $SS(u)$, $u \in [0,1]$), and a set of $\mathbf{Q_0}, ..., \mathbf{Q_k}$ exact mean camber line points (computed in appendix Z), the cubic Bézier representation of the mean camber line is sought, because the cubic or 4-point Bézier representation of the mean camber line is required to compute angles θ , β, δ of the GMTurbo parameterization. The approximation is carried out by imposing point interpolation at the LE and TE, tangent interpolation at the same endpoints and, finally, minimizing the distance between the curve and the $\mathbf{Q_i}$ points.

First, the LE and TE are interpolated by points $\mathbf{P_0}$ and $\mathbf{P_3}$

$$\begin{aligned} \mathbf{P_0} &= \mathbf{PS}(0) = \mathbf{SS}(0) = LE \\ \mathbf{P_3} &= \frac{\mathbf{PS}(1) + \mathbf{SS}(1)}{2} = TE \end{aligned} \tag{E.1}$$

The tangents at the LE and TE ($\vec{T_1}$ and $\vec{T_2}$) are predefined by the shape of the airfoil

$$\begin{aligned} \mathbf{T_1} \cdot \mathbf{PS}'(0) = 0 \\ \mathbf{T_2} \cdot (\mathbf{PS}(1) - \mathbf{SS}(1)) = 0 \end{aligned} \tag{E.2}$$

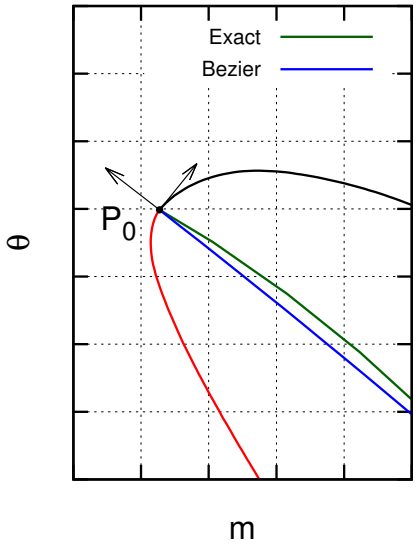


Figure E.1: *The tangent to the airfoil at the LE is orthogonal to the tangent to the mean camber line at the LE.*

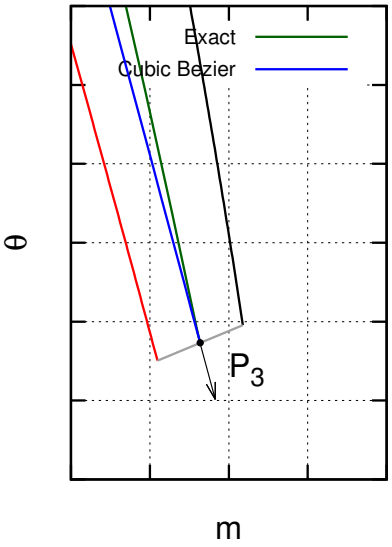


Figure E.2: *The tangent to the airfoil at the TE is orthogonal to the tangent to the mean camber line at the TE.*

thus points $\mathbf{P_1}$ and $\mathbf{P_2}$ are given by

$$\begin{aligned} \mathbf{P_1} &= \mathbf{P_0} + a\mathbf{T_1} \\ \mathbf{P_2} &= \mathbf{P_3} + b\mathbf{T_2} \end{aligned} \tag{E.3}$$

The cubic Bézier curve that is sought is given by the equation

$$\mathbf{C}(u) = \mathbf{P_0}s^3 + 3s^2u\mathbf{P_1} + 3su^2\mathbf{P_2} + \mathbf{P_3}u^3 \tag{E.4}$$

where $s = 1 - u$. Combining eq. E.3 and eq.E.4

$$\mathbf{C}(u) = \mathbf{P_0}(s^3 + 3s^2u) + \mathbf{T_1}(3s^2u)a + \mathbf{T_2}(2su^2)b + \mathbf{P_3}(u^3 + 3su^2) \tag{E.5}$$

To minimize the distance between points $\mathbf{Q_0}..\mathbf{Q_k}$ and curve $\mathbf{C}(u)$, a function $\mathbf{F}$ is defined

$$\mathbf{F} = \sum_0^k \Big(\mathbf{Q_i} - \mathbf{C}(\bar{u_i})\Big) = 0 \tag{E.6}$$

the parameter $\bar{u_i}$ is assigned to each exact mean camber line point $\mathbf{Q_i}$ with respect

to the point's normalized arc length

$$\begin{aligned} \bar{u_0} &= 0 \\ \bar{u_i} &= \bar{u_{i-1}} + \frac{|\mathbf{Q_i} - \mathbf{Q_{i-1}}|}{\sum_{i=0}^{k} |\mathbf{Q_i} - \mathbf{Q_{i-1}}|} \quad i = 1, ..., k \end{aligned} \tag{E.7}$$

eq. E.6 is a linear system of two equations with two unknowns a and b, that can readily be solved.

Bibliography

[1] Verstraete, T.: *Introduction to optimization and multidisciplinary design*, May 2016.

[2] Tsiakas, KT, Gagliardi, F, Trompoukis, XS, and Giannakoglou, KC: *Shape optimization of turbomachinery rows using a parametric blade modeller and the continuous adjoint method running on gpus*. June 5-10 2016.

[3] Tsiakas, K.: *Development of shape parameterization techniques, a flow solver and its adjoint, for optimization on GPUs. Turbomachinery and external aerodynamics applications*. PhD thesis, Laboratory of Thermal Turbomachines, NTUA, Athens, in progress.

[4] Trompoukis, X.: *Numerical solution of aerodynamic-aeroelastic problems on GPUs*. PhD thesis, Laboratory of Thermal Turbomachines, NTUA, Athens, 2012.

[5] *Easy webpage*. `http:/velos0.ltt.mech.ntua.gr/EASY`.

[6] Piegl, L. and Tiller, W.: *The NURBS Book*. Springer-Verlag, New York, NY, USA, second edition, 1996.

[7] Gagliardi, F.: *Shape Parameterization and Integrated Constrained Optimization Loops for Turbomachinery Applications*. PhD thesis, Laboratory of Thermal Turbomachines, NTUA, Athens, in progress.

[8] Rossgatterer, M, Jüttler, Bert, Kapl, Mario, and Della Vecchia, Giovanni: *Medial design of blades for hydroelectric turbines and ship propellers*. Computers & Graphics, 36(5):434–444, 2012.

[9] Kampolis, I.: *Parallel, Multilevel Algorithms for the Aerodynamic Optimization In Turbomachines*. PhD thesis, Laboratory of Thermal Turbomachines, NTUA, Athens, 2009.

[10] Tsopelas, I.: *Integration of the GPU-Enabled CFD solver PUMA into the workflow of a Turbomachinery Industry. Validation*. Diploma Thesis. Laboratory of Thermal Turbomachines, NTUA, 2018.

[11] Chorin, A.: *A numerical method for solving incompressible viscous flow problems*. Journal of Computational Physics, 2(1):12–26, 1967.

[12] Turkel, E.: *Preconditioned methods for solving the incompressible and low speed compressible equations.* Journal of Computational Physics, 72:277–298, 1987.

[13] Spalart, P. and Allmaras, S.: *A one-equation turbulence model for aerodynamic flows.* AIAA Paper 1992-439, 30th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, January 6–9 1992.

[14] Spalding, D. B.: *A single formula for the law of the wall.* Journal of Applied Mechanics, 28:455–457, 1961.

[15] Frink, Neal: *Assessment of an unstructured-grid method for predicting 3-d turbulent viscous flows.* In *34th Aerospace Sciences Meeting and Exhibit*, 1996.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Εργαστήριο Θερμικών Στροβιλομηχανών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής
& Βελτιστοποίησης

Προγραμματισμός ενός 'Πίσω-σε-CAD' Εργαλείου για την Ανάλυση και Βελτιστοποίηση Πτερυγώσεων Στροβιλομηχανών. Βιομηχανικές Εφαρμογές.

Διπλωματική Εργασία

Μαρκέλλα Ζορμπά

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου , Καθηγητής ΕΜΠ

Αθήνα, 2018

Ακρωνύμια

ΕΜΠ	Εθνικό Μετσόβιο Πολυτεχνείο
ΕΘΣ	Εργαστήριο Θερμικών Στροβιλομηχανών
ΜΠΥΡ&Β	Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης
ΥΡΔ	Υπολογιστική Ρευστοδυναμική

CAD	Computer Aided Design

Περιεχόμενα

Περιεχόμενα **i**

1 Εισαγωγή **1**

1.1 Αναπαράσταση πτερυγίων στροβιλομηχανών και βελτιστοποίηση 1

1.2 Αντικείμενο της Διπλωματικής Εργασίας 3

2 GMTurbo **5**

2.1 Μεσημβρινό Επίπεδο 5

2.2 Μέση Γραμμή Κυρτότητας 6

2.3 Κατανομή Πάχους 7

3 Λογισμικό Αντίστροφης Παραμετροποίησης (Reverse Parameterization Tool) **9**

3.1 Υπολογισμός Μεσημβρινής Τομής 9

3.2 Υπολογισμός Πλευρών Υπερπίεσης και Υποπίεσης 10

3.3 Υπολογισμός Μέσης Γραμμής Κυρτότητας 11

3.4 Υπολογισμός Κατανομής Πάχους 12

4 Λογισμικό Προσαρμογής Πλέγματος (Grid Adaptation Tool) **14**

4.1 Προσαρμογή Επιφανειακού Πλέγματος 15

4.2 Παραμόρφωση Ογκικού Πλέγματος 17

5 Εφαρμογές και βελτιστοποίηση **18**

5.1 Εφαρμογή σε υδροστρόβιλο Francis 18

5.2 Εφαρμογή σε υδροστρόβιλο τύπου Προπέλας με οδηγά πτερύγια 20

5.3 Βελτιστοποίηση Μορφής 21
5.3.1 Βελτιστοποίηση οδηγών πτερυγίων (16 μεταβλητές σχεδιασμού) 22
5.3.2 Βελτιστοποίηση οδηγών πτερυγίων (32 μεταβλητές σχεδιασμού) 23
5.3.3 Βελτιστοποίηση οδηγών πτερυγίων και πτερωτής 24

6 Συμπεράσματα **26**

Κεφάλαιο 1

Εισαγωγή

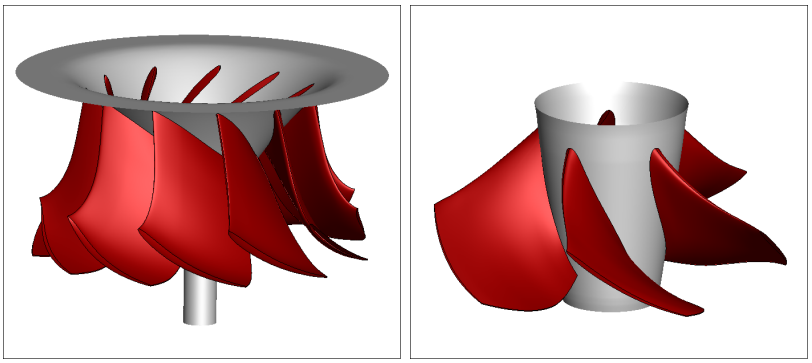
1.1 Αναπαράσταση πτερυγίων στροβιλομηχανών και βελτιστοποίηση

Ο αεροδυναμικός και υδροδυναμικός σχεδιασμός πτερυγώσεων στροβιλομηχανών αποτελεί αντικείμενο έρευνας μεγάλης σημασίας. Οι εφαρμογές των στροβιλομηχανών είναι πάρα πολλές, συνεπώς η ποικιλία που συναντάται ως πρός τη μορφή τους είναι μεγάλη. Για το λόγο αυτό, είναι σημαντική η επιλογή του κατάλληλου τρόπου περιγραφής - παραμετροποίησης των πτερυγίων της στροβιλομηχανής, ώστε να εξυπηρετεί τις ανάγκες του σχεδιασμού. Στη διπλωματική αυτή εργασία διακρίνονται και αναλύονται δύο τρόποι περιγραφής της γεωμετρίας ενός πτερυγίου. Ο πρώτος είναι η περιγραφή του πτερυγίου ως πλέγμα υπολογιστικής ρευστοδυναμικής (ΥΡΔ) και ο δεύτερος μέσω CAD.

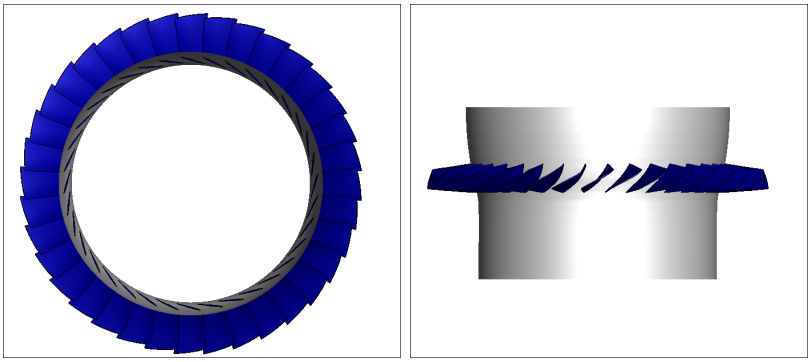
Η περιγραφή σε μορφή πλέγματος ΥΡΔ, είναι από ένα σύνολο κόμβων που αναπαριστούν διακριτά τη γεωμετρία και δίνει τη δυνατότητα για άμεση ανάλυση της ροής με τη χρήση λογισμικού ΥΡΔ, προσφέροντας πληροφορίες για τα ροϊκά μεγέθη γύρω απο το πτερύγιο.

Με τον όρο CAD εννοείται οποιοδήποτε λογισμικό λαμβάνει ώς είσοδο κάποιες παραμέτρους η φύση των οποίων ορίζεται απο το ίδιο το λογισμικό και παράγει τη γεωμετρία. Τα CAD λογισμικά διευκολύνουν τη διαδικασία του σχεδιασμού και της βελτιστοποίησης, αφού οι παράμετροι που χρησιμοποιούν έχουν συχνά γεωμετρική σημασία και η μεταβολή τους εγγυάται ένα πτερύγιο ομαλό χωρίς ασυνέχειες.

Στην διπλωματική εργασία αυτή το CAD λογισμικό που χρησιμοποιείται είναι το λογισμικό GMTurbo που έχει αναπτυχθεί από τη ΜΠΥΡ&Β. Αυτό χρησιμοποιεί παρα-



Σχήμα 1.1: *Υδροστρόβιλοι τύπου Francis και τύπου Propeller.*

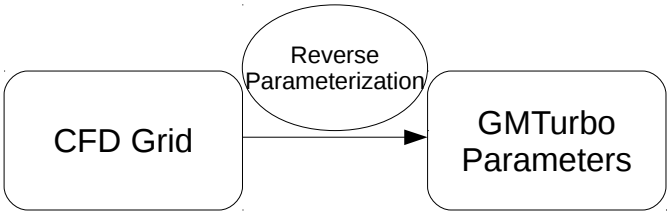


Σχήμα 1.2: *Αξονικός συμπιεστής με 36 πτερύγια.*

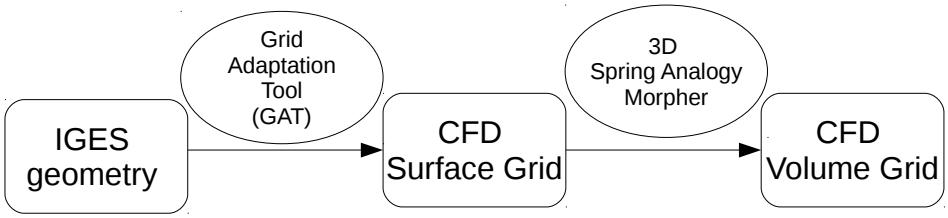
μέτρους, η φύση των οποίων σχετίζεται άμεσα με τη θεωρία των στροβιλομηχανών (μεσημβρινή τομή, γωνίες μετάλλου κλπ) για να κατασκευάσει το πτερύγιο. Είναι ιδανική για το σχεδιασμό και τη βελτιστοποίηση πτερυγίων δεδομένου ότι οι παράμετροι που χρησιμοποιεί έχουν φυσική σημασία και είναι περιορισμένου αριθμού, κάτι που αποτελεί μεγάλο όφελος σε μία βελτιστοποίηση με εξελικτικό αλγόριθμο.

Είναι σημαντικό λοιπόν να υπάρχει η δυνατότητα της ενσωμάτωσης του GMTurbo σε κάθε βελτιστοποίηση (σχ.1.5). Στην περίπτωση που η αρχική γεωμετρία δίνεται απο ένα πλέγμα ΥΡΔ, η βελτιστοποίηση με GMTurbo είναι αδύνατη. Για το λόγο αυτό στο πλαίσιο της διπλωματικής αυτής εργασίας, αναπτύσσεται Λογισμικό Αντίστροφης Παραμετροποίησης (Reverse Parameterization Tool (RPT)) που πραγματοποιεί τη μετατροπή ενός πλέγματος ΥΡΔ σε παραμετροποίηση GMTurbo (σχ. 1.3).

Οι CAD γεωμετρίες παρόλο που διαθέτουν μεγάλη ευελιξία σε θέματα σχεδίασης και τροποποίησης της γεωμετρίας, έχουν ένα βασικό μειονέκτημα. Χρειάζονται πλεγματοποίηση προκειμένου να εφαρμοσθεί οποιοσδήποτε κώδικας ΥΡΔ γύρω από τη γεωμετρία. Στη διπλωματική αυτή εργασία αναπτύσσεται Λογισμικό Προσαρμογής Επιφανειακού Πλέγματος (Grid Adaptation Tool (GAT)) γύρω απο GMTurbo γεωμετρία, σύμφωνα με το οποίο το αρχικό πλέγμα προσαρμόζεται στην νέα γεωμετρία με τη χρήση του 2Δ και 3Δ Λογισμικού Παραμόρφωσης Πλέγματος με τη μέθοδο των Γραμμικών Ελατηρίων που έχει αναπτυχθεί στη ΜΠΥΡ&Β (σχ. 1.4).



Σχήμα 1.3: *Το λογισμικό RPT.*

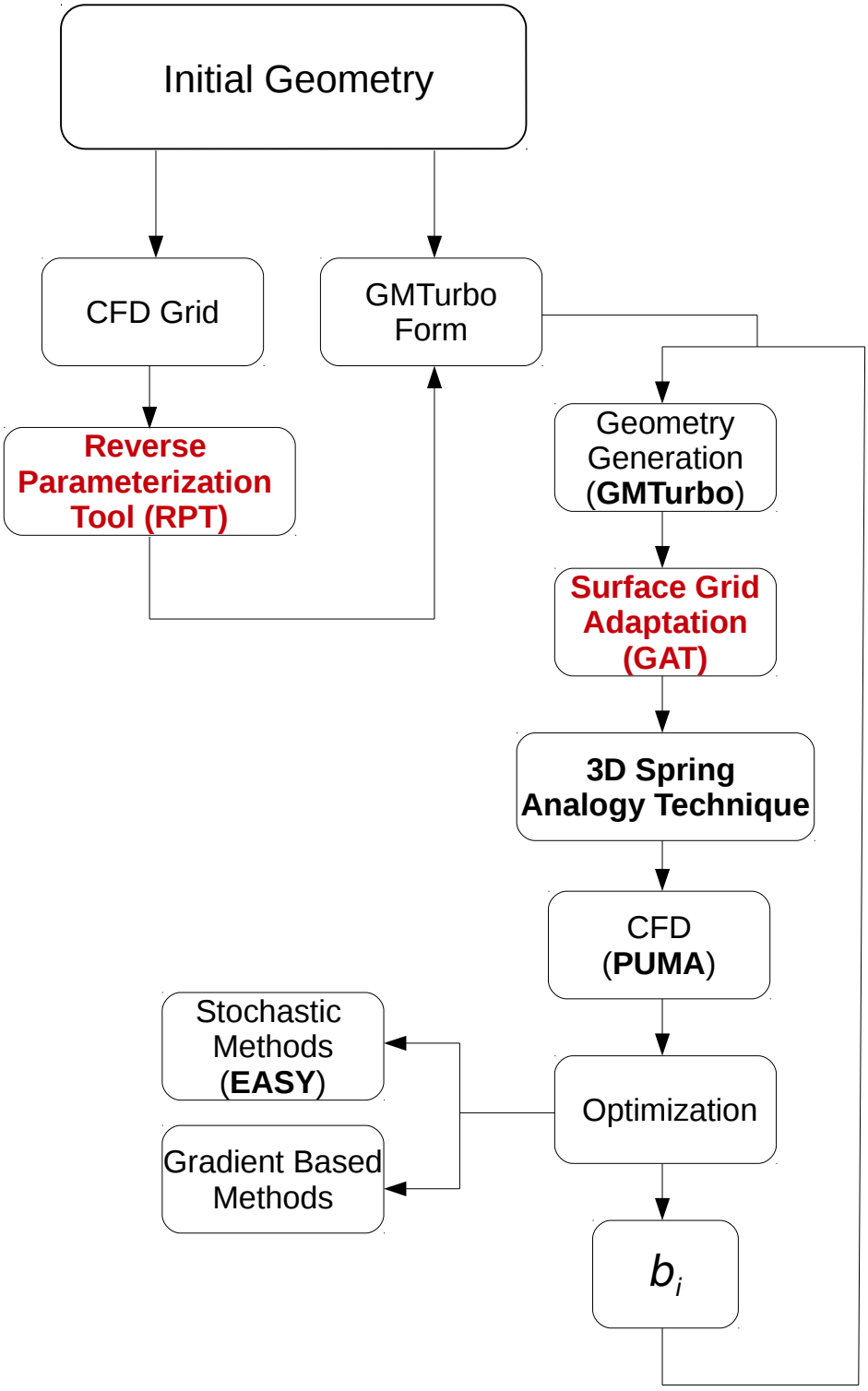


Σχήμα 1.4: *Το λογισμικό GAT.*

1.2 Αντικείμενο της Διπλωματικής Εργασίας

Στη διπλωματική αυτή εργασία προγραμματίζονται:

- Λογισμικό Αντίστροφης Παραμετροποίησης RPT (C++) το οποίο μετατρέπει ένα πλέγμα ΥΡΔ σε παραμέτρους συμβατές με το CAD λογισμικό GMTurbo για τη διευκόλυνση του σχεδιασμού και της βελτιστοποίησης.
- Λογισμικό Προσαρμογής Πλέγματος GAT (C++) που δημιουργεί πλέγμα γύρω απο την GMTurbo γεωμετρία.
- Μετεπεξεργαστές που επεξεργάζονται τα αποτελέσματα του λογισμικού ΥΡΔ (Fortran 77). Συγκεκριμένα, υπολογίζουν τη κατανομή του αδιάστατου συντελεστή πίεσης κατά μήκος ισο-γραμμών του πτερυγίου, τις κυλινδρικές συντεταγμένες της ταχύτητας στην έξοδο του ρότορα κατά μήκος της αδιάστατης ακτίνας και τον βαθμό απόδοσης της στροβιλομηχανής.



Σχήμα 1.5: *Για την εκκίνηση μίας βελτιστοποίησης με τη χρήση του αλγορίθμου που φαίνεται στο σχήμα για την αξιολόγηση είναι απαραίτητο η αρχική γεωμετρία να βρίσκεται σε μορφή συμβατή με το CAD λογισμικό, εδώ το GMTurbo. Σε περίπτωση που μόνο το πλέγμα ΥΡΔ είναι διαθέσιμο για την αναπαράσταση της αρχικής γεωμετρίας, αυτό πρέπει να μετατραπεί σε ισοδύναμη GMTurbo μορφή με το Λογισμικό Αντίστροφης Παραμετροποίησης (RPT) που προγραμματίζεται. Για τη βελτιστοποίηση, εναρμονίζονται τα λογισμικά που φαίνονται στο σχήμα. Προγραμματίζεται Λογισμικό Προσαρμογής Πλέγματος (GAT) για τη δημιουργία πλέγματος πάνω στη GMTurbo γεωμετρία.*

Κεφάλαιο 2

GMTurbo

Το λογισμικό GMTurbo βασίζεται στη χρήση βασικών εννοιών από τη θεωρία των στροβιλομηχανών για τη παραμετροποίηση ενός πτερυγίου. Η κατασκευή του πτερυγίου περιλαμβάνει τα εξής:

2.1 Μεσημβρινό Επίπεδο

Αρχικά ορίζονται οι NURBS καμπύλες στο rz επίπεδο που περιγράφουν τις μεσημβρινές των hub, shroud, εισόδου, εξόδου και των ακμών πρόσπτωσης και εκφυγής του πτερυγίου (σχ. 2.1). Με τον τρόπο αυτό ο χρήστης ορίζει τη μεσημβρινή τομή του πτερυγίου στο μεσημβρινό επίπεδο. Γραμμική παρεμβολή των σημείων ελέγχου των καμπυλών hub και shroud, δημιουργεί ενδιάμεσες καμπύλες στο μεσημβρινό επίπεδο οι οποίες είναι οι γενέτειρες του πτερυγίου. Κάθε γενέτειρα στο rz επίπεδο αποτελεί μια εκ περιστροφής επιφάνεια στον 3Δ χώρο αν περιστραφεί γύρω από τον άξονα z της στροβιλομηχανής.

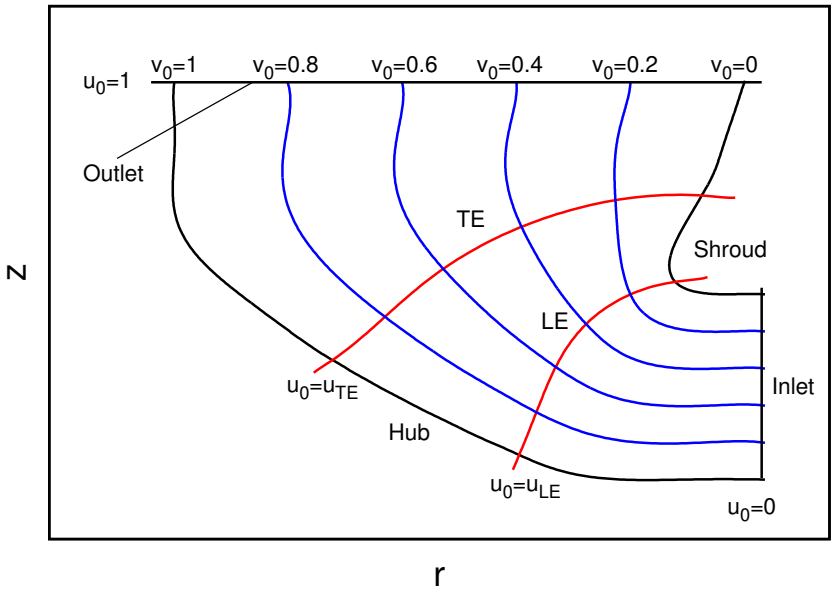
Στο σημείο αυτό εισάγεται η έννοια του σύμμορφου μετασχηματισμού. Μία εκ περιστροφής επιφάνεια με γενέτειρα μια καμπύλη rz μπορεί να μετασχηματιστεί στο (m, θ) επίπεδο σύμφωνα με τη συνάρτηση

$$\Phi(v_0) : (r(u, v_0)cos\theta, r(u, v_0)sin\theta, z(u, v_0)) \mapsto (m(u, v_0), \theta) \tag{2.1}$$

όπου η συνάρτηση $m(u, v_0)$ δίνεται από

$$m(u, v_0) = \int_0^u \frac{\sqrt{r_u(t, v_0)^2 + z_u(t, v_0)^2}}{r(t, v_0)} dt \tag{2.2}$$

Έτσι, αφού θεωρείται ότι κάθε ενδιάμεση καμπύλη του σχ. 2.1 αποτελεί εκ περιστροφής επιφάνεια που μετασχηματίζεται σύμμορφα στο (m, θ) επίπεδο, η ανάλυση των μεγεθών σε κάθε εκ περιστροφής επιφάνεια, ανάμεσα στα hub και shroud, μπορεί να μεταφερθεί στο 2Δ επίπεδο (m, θ).



Σχήμα 2.1: *Ο χρήστης αρχικά ορίζει τη μεσημβρινή τομή του πτερυγίου ορίζοντας τις 6 μεσημβρινές καμπύλες (hub, shroud, είσοδος, έξοδος, ακμή πρόσπτωσης, ακμή εκφυγής). Ο χώρος κατά μήκος του πτερυγίου διακριτοποιείται με N (αριθμός οριζόμενος από το χρήστη) ισαπέχουσες καμπύλες που προκύπτουν από γραμμική παρεμβολή μεταξύ των hub και shroud.*

2.2 Μέση Γραμμή Κυρτότητας

Η μέση γραμμή κυρτότητας, περιγράφεται στο επίπεδο (m, θ) με μία καμπύλη Bézier 4 σημείων (κυβική Bézier). Τα 4 σημεία ελέγχου, ορίζονται με τη χρήση των σχετικών μεταξύ τους γωνιών (6 γωνίες: β_{LE}, β_{TE}, θ_{LE}, θ_{TE}, δ_{LE}, δ_{TE}) (σχ. 2.2), με σκοπό να γίνει πιο σαφής η γεωμετρική σημασία των παραμέτρων. Αυτές ορίζονται ώς εξής

$$\mathbf{P_0} = (m_{LE}, \theta_{LE}) \tag{2.3}$$

$$\mathbf{P_3} = (m_{TE}, \theta_{TE}) \tag{2.4}$$

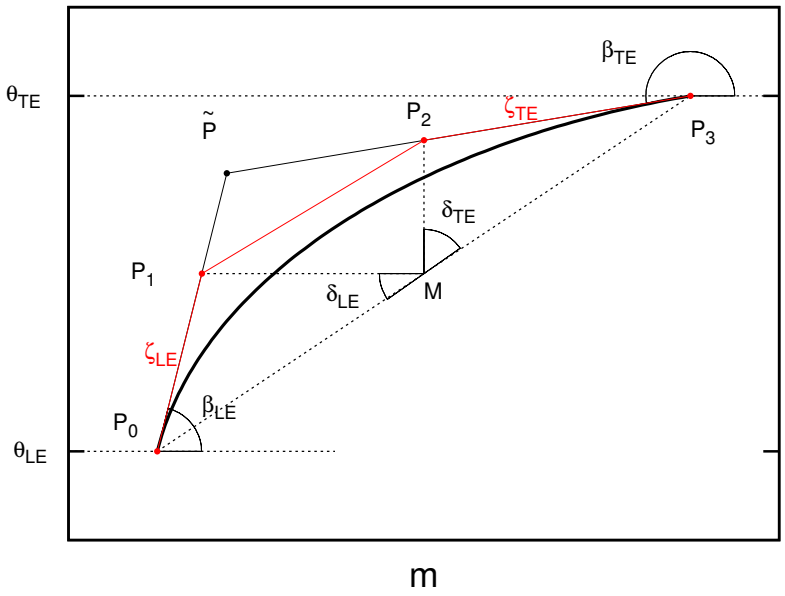
Το σημείο $\tilde{\mathbf{P}}$ είναι το σημείο τομής των εφαπτόμενων στις ακμές πρόσπτωσης και

εκφυγής, που ορίζονται από τις γωνίες β_{LE} και β_{TE}. Τα σημεία $\mathbf{P_1}$ και $\mathbf{P_2}$ είναι συναρτήσεις των δ_{LE} και δ_{TE} σύμφωνα με τις εξισώσεις

$$(\mathbf{P_1} - \mathbf{M}) \cdot (\mathbf{P_0} - \mathbf{M}) = cos\delta_{LE} ||\mathbf{P_1} - \mathbf{M}|| \cdot ||\mathbf{P_0} - \mathbf{M}|| \tag{2.5}$$

$$(\mathbf{P_2} - \mathbf{M}) \cdot (\mathbf{P_3} - \mathbf{M}) = cos\delta_{TE} ||\mathbf{P_2} - \mathbf{M}|| \cdot ||\mathbf{P_3} - \mathbf{M}|| \tag{2.6}$$

όπου $\mathbf{M}$ είναι το μέσο της $\mathbf{P_0P_3}$ χορδής.



Σχήμα 2.2: *Τα σημεία ελέγχου της μέσης γραμμής κυρτότητας και οι σχετικές γωνίες μεταξύ τους. Αναφέρεται ενδεικτικά ότι οι γωνίες β_{LE} και β_{TE} είναι οι γωνίες μετάλλου στις ακμές πρόσπτωσης και εκφυγής και οι γωνίες θ_{LE} και θ_{TE} είναι οι περιφερειακές θέσεις των ακμών αυτών.*

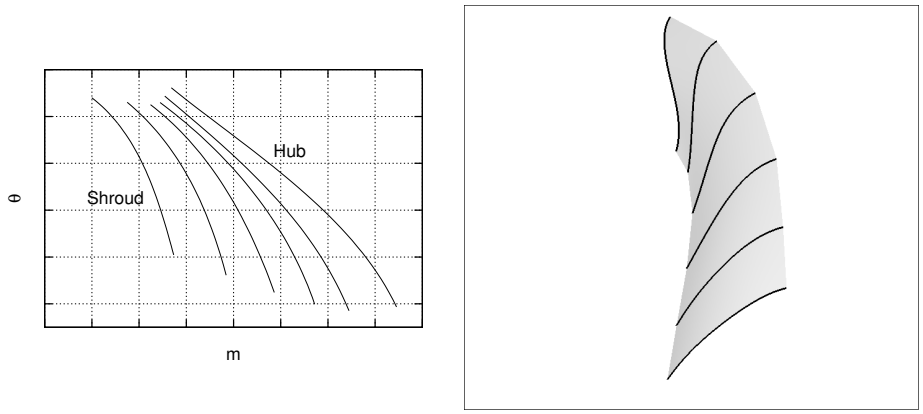
2.3 Κατανομή Πάχους

Έχοντας ορίσει μία μέση γραμμή κυρτότητας σε κάθε εκ περιστροφής επιφάνειας, έχει δημιουργηθεί το πτερύγιο μηδενικού πάχους. Ακολουθεί η υπέρθεση κατανομών πάχους στις δύο πλευρές, υπερπίεσης και υποπίεσης, κατά μήκος του πτερυγίου, που ορίζεται με τη χρήση δύο μεγεθών ($t(u, v_0) = \hat{t}(u)t_f(v_0)$). Το πρώτο είναι η κατά μήκος της χορδής αδιάστατη κατανομή πάχους του πτερυγίου $\hat{t}(u)$. Η κατανομή αυτή πολλαπλασιάζεται με έναν συντελεστή πάχους $t_f^{PS}(v_0)$ και $t_f^{SS}(v_0)$ για να πάρει

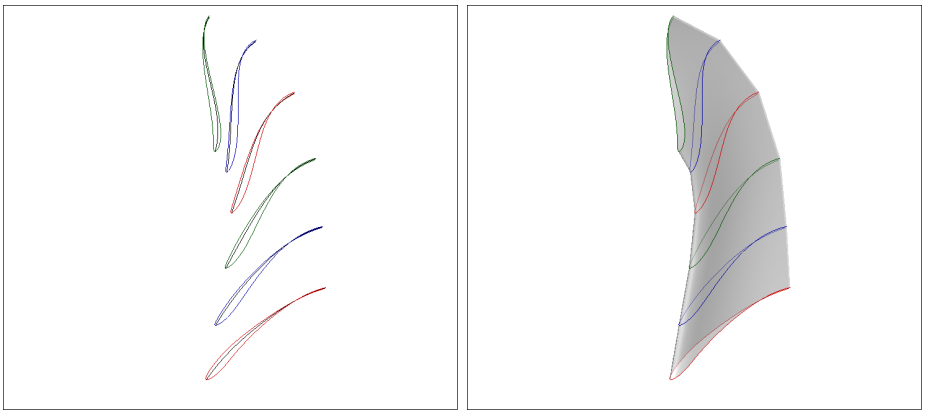
διαστάσεις. Απομένει η παρεμβολή με δύο NURBS επιφάνειες των κατά μήκος του πτερυγίου αεροτομών, για την ολοκλήρωση της κατασκευής του πτερυγίου. Η κατανομή πάχους υπερτίθεται στις δύο πλευρές της μέσης γραμμής κυρτότητας σύμφωνα με τον τύπο:

$$\mathbf{c}_{m\theta}^{side_i}(s, v_0) = \boldsymbol{\mu}_{m\theta}(s, v_0) \pm \mathbf{n}_{m\theta}(s, v_0) \frac{t^{side_i}(s, v_0)}{r^2(m(s, v_0))} \tag{2.7}$$

όπου $r(m(s, v_0))$ είναι η αντίστοιχη ακτίνα του σημείου $\mu_{\mathbf{m}\theta}(\mathbf{s}, \mathbf{v_0})$ της μέσης γραμμής κυρτότητας, και χρησιμοποιείται για να μετασχηματίσει το μήκος $t(s, v_0)$ του 2Δ χώρου σε αντίστοιχο μήκος στο (m, θ) επίπεδο.



Σχήμα 2.3: *Ορίζεται μία μέση γραμμή κυρτότητας σε κάθε (m, θ) επίπεδο (αριστερά). Το πτερύγιο μηδενικού πάχους προκύπτει επιστρέφοντας στο 3Δ χώρο με τον αντίστροφο σύμμορφο μετασχηματισμό Φ^{-1} (δεξιά).*



Σχήμα 2.4: *Υπερτίθενται κατανομές πάχους σε κάθε μέση γραμμή κυρτότητας στις δύο πλευρές σύμφωνα με την εξίσωση 2.7 (αριστερά). Οι καμπύλες πάχους, παρεμβάλλονται από δύο NURBS επιφάνειες, μία για την πλευρά υπερπίεσης και μία για την πλευρά υποπίεσης (δεξιά).*

Κεφάλαιο 3

Λογισμικό Αντίστροφης Παραμετροποίησης (Reverse Parameterization Tool)

Προκειμένου να είναι δυνατή η επεξεργασία μίας γεωμετρίας με τη χρήση του GMTurbo, η γεωμετρία αυτή πρέπει να βρίσκεται σε συμβατή με τον GMTurbo μορφή. Να περιγράφεται δηλαδή από τις παραμέτρους που παρουσιάστηκαν στο κεφάλαιο 2.

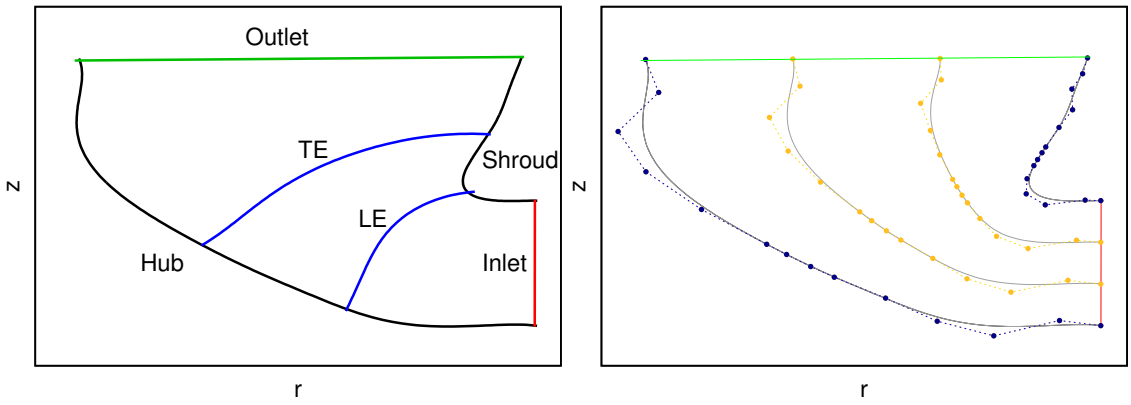
Ένα πτερύγιο μπορεί να έχει σχεδιαστεί με οποιοδήποτε CAD λογισμικό. Προκειμένου να βελτιστοποιηθεί με το λογισμικό GMTurbo, οφείλει να μετατραπεί σε κατάλληλη μορφή. Πολύ συχνά μία γεωμετρία που μπορεί να προέρχεται από τη βιομηχανία περιγράφεται σε μορφή πλέγματος. Η ανάπτυξη λογισμικού που μετατρέπει ένα πλέγμα ΥΡΔ σε GMTurbo παραμετροποίηση αναπτύσσεται σε αυτή τη διπλωματική εργασία και ενσωματώνεται στο λογισμικό GMTurbo.

Η μέθοδος που ακολουθείται περιγράφεται από τα εξής βήματα:

3.1 Υπολογισμός Μεσημβρινής Τομής

Αρχικά πραγματοποιείται ο υπολογισμός της μεσημβρινής τομής του πλέγματος. Ο υπολογισμός των μεσημβρινών καμπυλών, γίνεται με την προβολή των σημείων που ανήκουν στις αντίστοιχες 3Δ καμπύλες του πλέγματος, στο rz επίπεδο. Αφού προβληθούν όλοι οι κόμβοι, στη συνέχεια παρεμβάλλονται από κατάλληλες NURBS καμπύλες. Ενδιαμέσως των καμπυλών hub και shroud που υπολογίστηκαν, υπολογίζονται

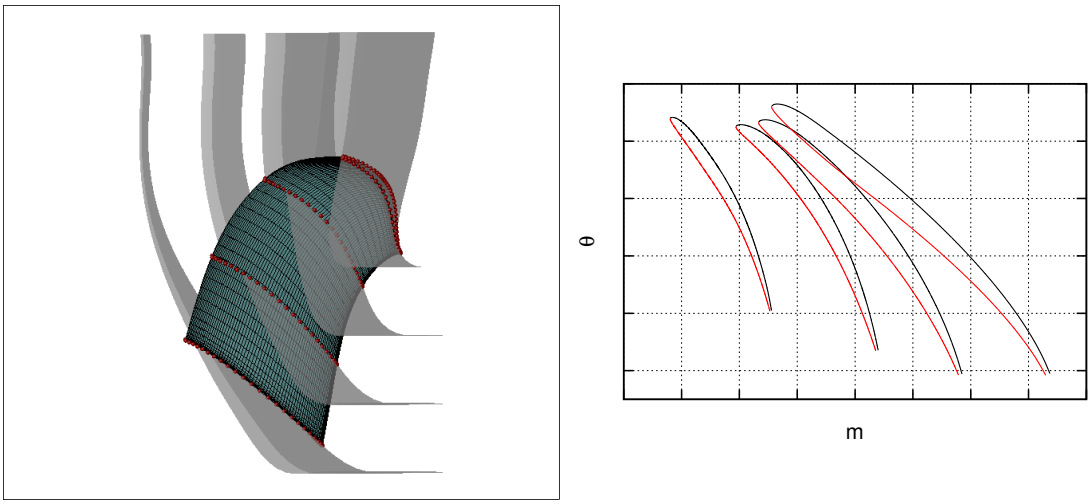
N (αριθμός οριζόμενος από το χρήστη) καμπύλες, ως NURBS καμπύλες, τα σημεία ελέγχου των οποίων προκύπτουν από γραμμική παρεμβολή των σημείων ελέγχου των hub και shroud καμπυλών. Οι καμπύλες αυτές μπορούν να αποτελέσουν γενέτειρες εκ περιστροφής επιφάνειας γύρω από τον z άξονα, κατ'αναλογία με αυτές του GMTurbo. Σε αυτές τις εκ περιστροφής επιφάνειες θα υπολογιστούν τα δεδομένα από το πλέγμα του πτερυγίου.



Σχήμα 3.1: *Υπολογίζεται αρχικά η μεσημβρινή τομή του πλέγματος και, στη συνέχεια, δημιουργούνται οι ενδιάμεσες ισαπέχουσες γενέτειρες.*

3.2 Υπολογισμός Πλευρών Υπερπίεσης και Υποπίεσης

Για τον υπολογισμό των πλευρών υπερπίεσης και υποπίεσης προγραμματίστηκε αλγόριθμος που βρίσκει την τομή μεταξύ δομημένου επιφανειακού πλέγματος (πτερύγιο) και NURBS εκ περιστροφής επιφάνειας. Οι τομές του πτερυγίου με τις εκ περιστροφής επιφάνειες που υπολογίστηκαν δεδομένου ότι ανήκουν σε μία εκ περιστροφής επιφάνεια, μπορούν να περιγραφούν από (m, θ) συντεταγμένες. Επιτυγχάνεται, λοιπόν, ο υπολογισμός των κατά μήκος του πτερυγίου αεροτομών σε 2Δ αναπαράσταση. Απομένει κάθε μία απο αυτές τις αεροτομές να εκφραστεί κατά GMTurbo, δηλαδή ώς μία μέση γραμμή κυρτότητας (σε κυβική Bézier μορφή) και δύο κατανομές πάχους.



Σχήμα 3.2: *Προγραμματίζεται αλγόριθμος που υπολογίζει την τομή μεταξύ επιφάνειας που δίνεται σε διακριτή μορφή (δομημένο επιφανειακό πλέγμα) και NURBS εκ περιστροφής επιφάνειας.*

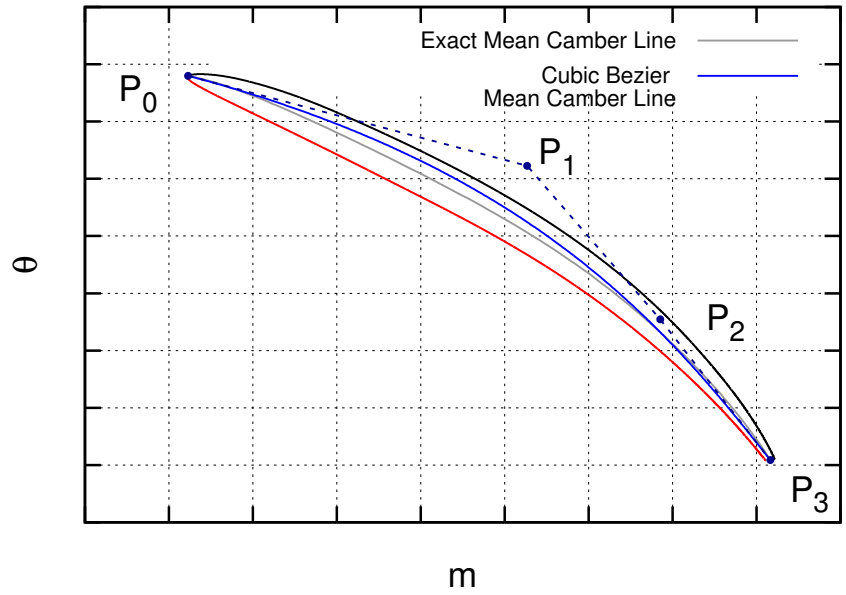
3.3 Υπολογισμός Μέσης Γραμμής Κυρτότητας

Η μέση γραμμή κυρτότητας υπολογίζεται σε δύο βήματα. Αρχικά υπολογίζονται σημεία ενδιαμέσως (Exact Mean Camber Line) των δύο πλευρών στο (m, θ) επίπεδο που υπολογίστηκε στο προηγούμενο βήμα. Στη συνέχεια αυτά τα σημεία προσεγγίζονται από μία κυβική καμπύλη Bézier. Η προσέγγιση γίνεται με περιορισμούς στα ακραία σημεία και στις εφαπτόμενες των ακραίων σημείων. Η κυβική Bézier καμπύλη οφείλει (1) να περνάει από τα σημεία πρόσπτωσης και εκφυγής, (2) να είναι κάθετη στην εφαπτόμενη στην ακμή πρόσπτωσης και (3) κάθετη στο πτερύγιο στην ακμή εκφυγής. Έτσι, υπολογίζονται τα σημεία ελέγχου της κυβικής Bézier που περιγράφει την μέση γραμμή κυρτότητας, και μετατρέπονται στις αντίστοιχες γωνίες της παραμετροποίησης σύμφωνα με τους τύπους:

$$\begin{aligned} \theta_{LE} &= \mathbf{P_{0,\theta}} \\ \theta_{TE} &= \mathbf{P_{3,\theta}} \end{aligned} \tag{3.1}$$

$$\begin{aligned} \beta_{LE} &= atan\Big(\frac{\mathbf{P_{1,\theta}} - \mathbf{P_{0,\theta}}}{\mathbf{P_{1,m}} - \mathbf{P_{0,m}}}\Big) \\ \beta_{TE} &= atan\Big(\frac{\mathbf{P_{2,\theta}} - \mathbf{P_{3,\theta}}}{\mathbf{P_{2,m}} - \mathbf{P_{3,m}}}\Big) \end{aligned} \tag{3.2}$$

$$\delta_{LE} = acos\Big(\frac{\vec{MP_1} \cdot \vec{MP_0}}{|\vec{MP_1}||\vec{MP_0}|}\Big)$$
$$\delta_{TE} = acos\Big(\frac{\vec{MP_2} \cdot \vec{MP_3}}{|\vec{MP_2}||\vec{MP_3}|}\Big) \tag{3.3}$$

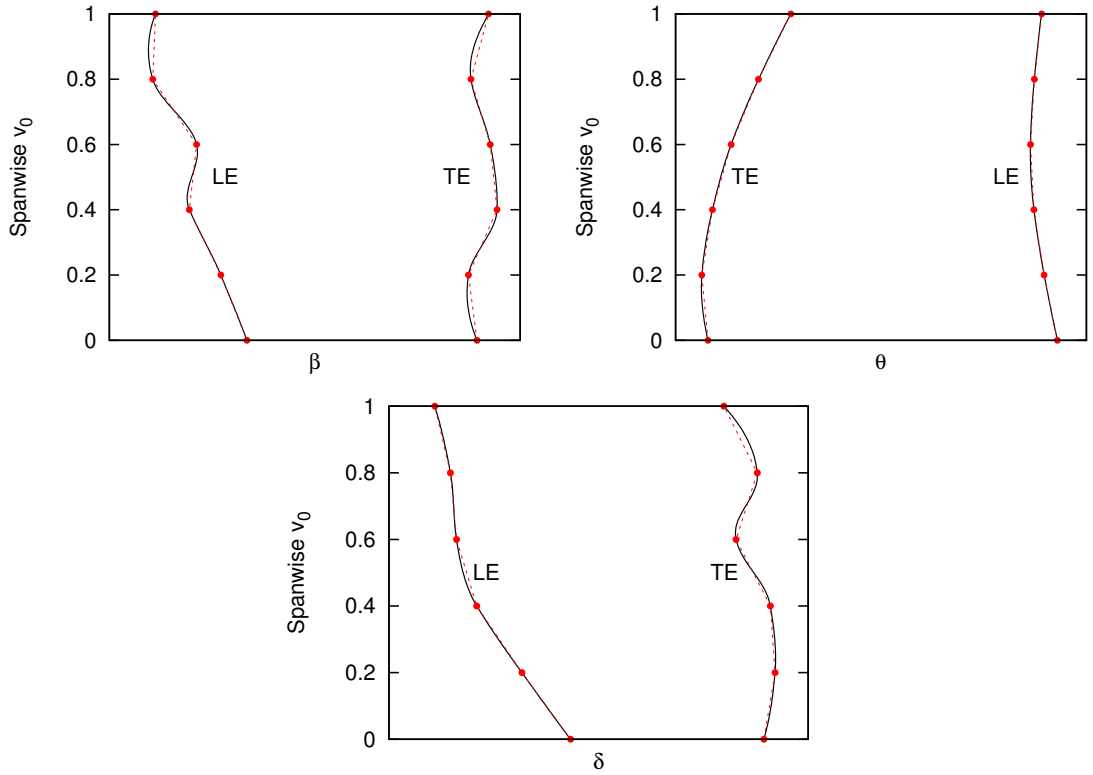


Σχήμα 3.3: *Η μέση γραμμή κυρτότητας όπως προκύπτει από τον υπολογισμό σημείων που ισαπέχουν από τις πλευρές υπερπίεσης και υποπίεσης (με γκρι χρώμα) και η προσέγγιση της από μία κυβική Bézier καμπύλη (μπλε γραμμή) που αποτελεί τη GMTurbo μέση γραμμή κυρτότητας.*

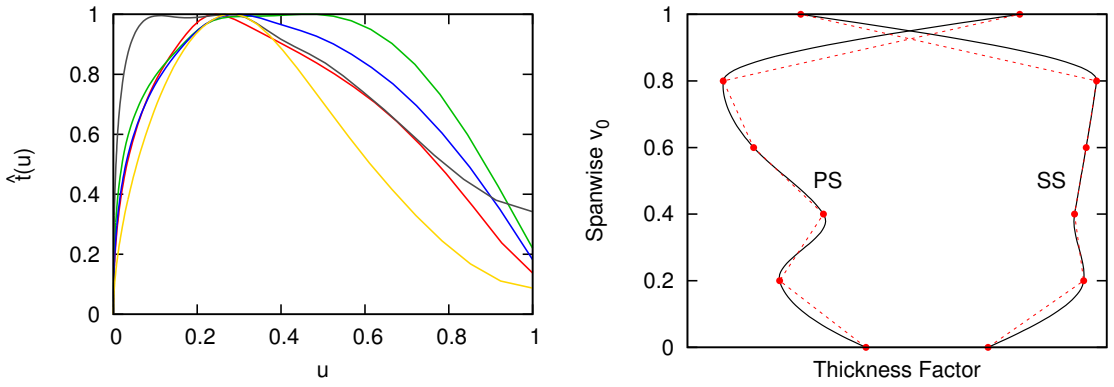
3.4 Υπολογισμός Κατανομής Πάχους

Οι κάθετες αποστάσεις της μέσης γραμμής κυρτότητας, από τις αεροτομές που υπολογίστηκαν δίνουν την κατανομή πάχους κατά μήκος της χορδής του πτερυγίου. Διαιρώντας την κατανομή αυτήν με τη μέγιστη τιμή της (που αποτελεί το συντελεστή πάχους), προκύπτει η αδιάστατη κατανομή πάχους.

Συνεπώς έχουν υπολογισθεί, η μεσημβρινή τομή της στροβιλομηχανής, κατανομές για τις γωνίες β, θ, δ (σχ. 3.4) κατά μήκος του πτερυγίου που δίνουν τη μέση γραμμή κυρτότητας κατά GMTurbo και κατανομές πάχους (σχ.3.5) για τις πλευρές υπερπίεσης και υποπίεσης σε κάθε κατά μήκος θέση. Ολοκληρώνεται έτσι η κατά GMTurbo παραμετροποίηση του πτερυγίου.



Σχήμα 3.4: *Οι κατανομές των γωνιών κατα μήκος του πτερυγίου.*



Σχήμα 3.5: *Κατανομές πάχους κατά μήκος της χορδής (αριστερά) και η κατανομή των συντελεστών πάχους κατά μήκος του πτερυγίου (δεξιά).*

Κεφάλαιο 4

Λογισμικό Προσαρμογής Πλέγματος (Grid Adaptation Tool)

Μετατρέποντας ένα πλέγμα ΥΡΔ σε μορφή συμβατή με το GMTurbo με τη χρήση του Λογισμικού Αντίστροφης Παραμετροποίησης, προκύπτει η πληροφορία της CAD αναπαράστασης της γεωμετρίας, που είναι πολύ πιο εύχρηστη σε θέματα σχεδιασμού και βελτιστοποίησης. Ωστόσο, με αυτήν τη μετατροπή, χάνεται η πλεγματική αναπαράσταση. Αυτό συμβαίνει επειδή οι δύο γεωμετρίες, πλέγμα ΥΡΔ και GMTurbo πτερύγιο, δεν είναι γεωμετρικά ταυτόσημες, αλλά το GMTurbo πτερύγιο που προκύπτει από το Λογισμικό Αντίστροφης Παραμετροποίησης είναι μία προσέγγιση των σημείων του αρχικού επιφανειακού πλέγματος.

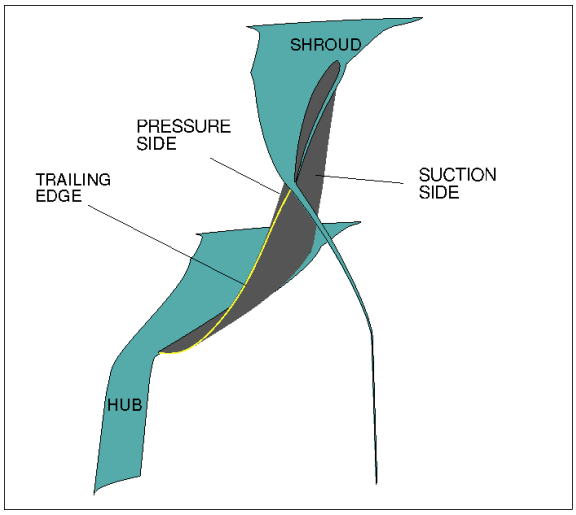
Για τον λόγο αυτόν, χρειάζεται δημιουργία πλέγματος πάνω στη GMTurbo γεωμετρία, προκειμένου να μπορεί να επιλυθεί η ροή πάνω σε αυτή. Έτσι αναπτύσσεται Λογισμικό Προσαρμογής Πλέγματος το οποίο υπολογίζει πλέγμα πάνω στη GMTurbo γεωμετρία, εκμεταλλευόμενο το αρχικό πλέγμα ΥΡΔ. Το πλέγμα που προκύπτει είναι ίδιου αριθμού κόμβων, ίδιας ποιότητας και ίδιας δομής με το αρχικό.

Η δημιουργία πλέγματος λαμβάνει χώρα σε δύο βήματα. Αρχικά δημιουργείται επιφανειακό πλέγμα πάνω στη GMTurbo γεωμετρία και, στη συνέχεια, το αρχικό ογκικό πλέγμα παραμορφώνεται βάσει του επιφανειακού.

4.1 Προσαρμογή Επιφανειακού Πλέγματος

Για τη δημιουργία επιφανειακού πλέγματος πάνω στην GMTurbo γεωμετρία, προσαρμόζεται το αρχικό επιφανειακό πλέγμα, πάνω στα στερεά όρια της γεωμετρίας. Τα στερεά όρια, υπάρχουν ταυτόχρονα σε μορφή επιφανειακού πλέγματος και σε μορφή GMTurbo. Αυτά είναι (σχ. 4.1):

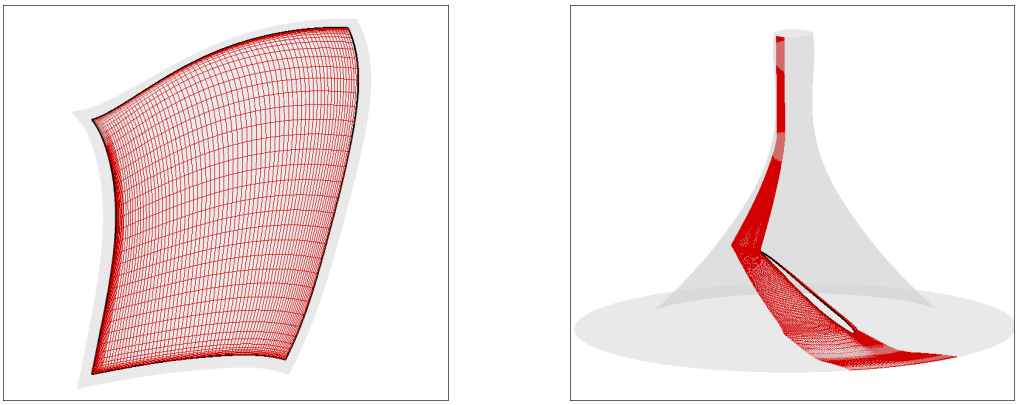
- Πλευρά Υπερπίεσης
- Πλευρά Υποπίεσης
- Hub
- Shroud
- Πάχος του πτερυγίου στην ακμή εκφυγής



Σχήμα 4.1: *Τα στερεά όρια των γεωμετριών που μελετώνται.*

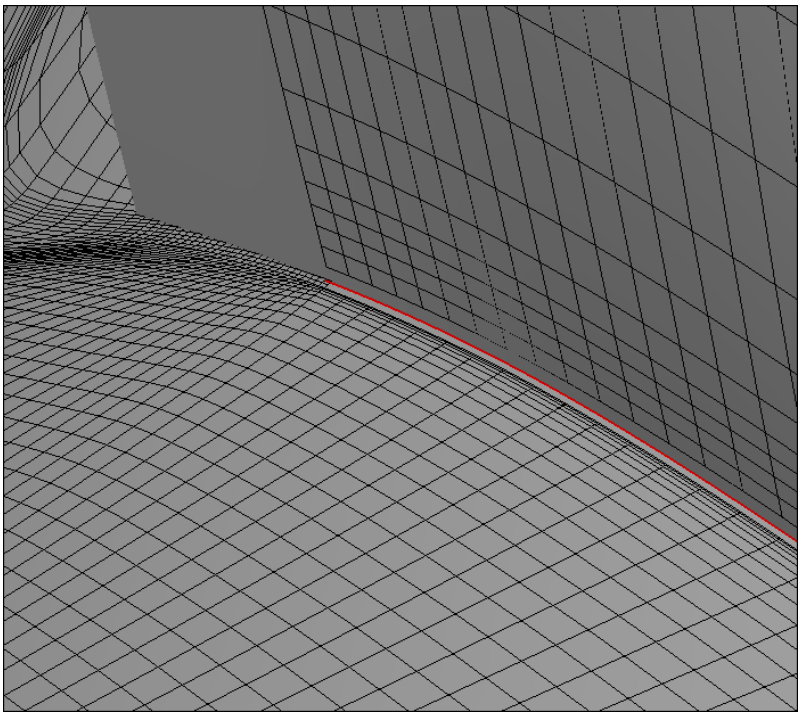
Για κάθε στερεό όριο πραγματοποιούνται τα εξής βήματα:

- Το αρχικό επιφανειακό πλέγμα προβάλλεται στη NURBS επιφάνεια που προκύπτει από το GMTurbo. Προκύπτει, λοιπόν, επιφανειακό πλέγμα που έχει την ίδια δομή με το αρχικό επιφανειακό πλέγμα και οι κόμβοι του βρίσκονται πάνω στην NURBS επιφάνεια του στερεού ορίου (σχ. 4.2). Το πρόβλημα που προκύπτει από αυτή τη προβολή είναι ότι τα όρια του πλέγματος δεν ταυτίζονται με τα όρια της επιφάνειας (σχ. 4.3).



Σχήμα 4.2: *Η προβολή του αρχικού πλέγματος πάνω στην NURBS επιφάνεια δημιουργεί επιφανειακό πλέγμα οι κόμβοι του οποίου βρίσκονται πάνω στην NURBS επιφάνεια.*

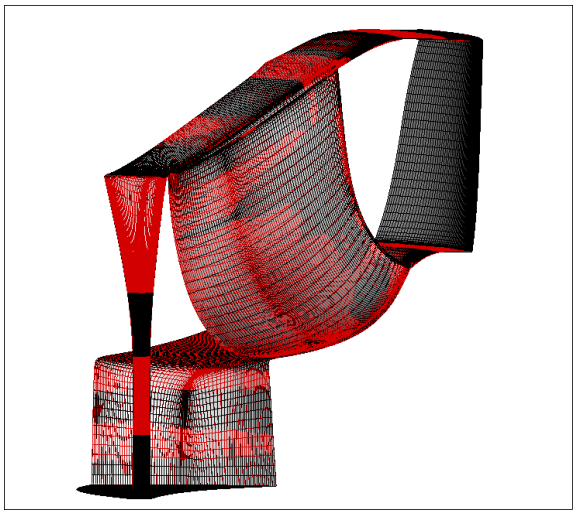
- Πραγματοποιείται παραμόρφωση του προβεβλημένου πλέγματος ώστε αυτό να εφαρμόζει στα όρια της επιφάνειας, και να αποτελεί τη διακριτή περιγραφή της. Η παραμόρφωση γίνεται, αφού εκφραστούν οι συντεταγμένες των κόμβων στη διπαραμετρική (u, v) ή (m, θ) περιγραφή των παραμέτρων της NURBS επιφάνειας. Σε αυτά τα 2Δ πλέγματα εφαρμόζεται παραμόρφωση με τη χρήση της μεθόδου των γραμμικών ελατηρίων.



Σχήμα 4.3: *Οι ακμές του πλέγματος δέν ταυτίζονται με τα όρια της επιφάνειας (κόκκινη γραμμή).*

4.2 Παραμόρφωση Ογκικού Πλέγματος

Αφού υπολογιστούν τα έγκυρα επιφανειακά πλέγματα πάνω στα GMTurbo στερεά όρια (σχ. 4.4), αυτές οι θέσεις των επιφανειακών ορίων, χρησιμοποιούνται ως οριακή συνθήκη για να μετατοπίσουν το εσωτερικό πλέγμα με τη χρήση ενός κώδικα παραμόρφωσης 3Δ πλέγματος με τη μέθοδο των γραμμικών ελατηρίων. Έτσι, προκύπτει ένα έγκυρο ογκικό πλέγμα πάνω στην GMTurbo γεωμετρία, στο οποίο μπορεί να επιλυθεί η ροή με κάποιο λογισμικό ΥΡΔ.



Σχήμα 4.4: *Με τη διαδικασία της προσαρμογής πλέγματος προκύπτει επιφανειακό πλέγμα όλων των στερεών ορίων που βρίσκεται πάνω στη GMTurbo γεωμετρία.*

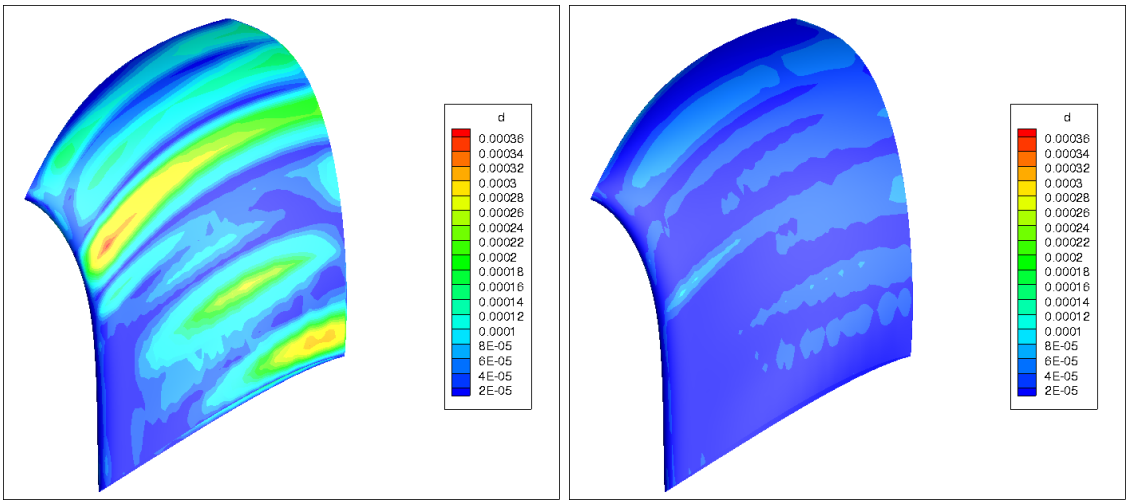
Κεφάλαιο 5

Εφαρμογές και βελτιστοποίηση

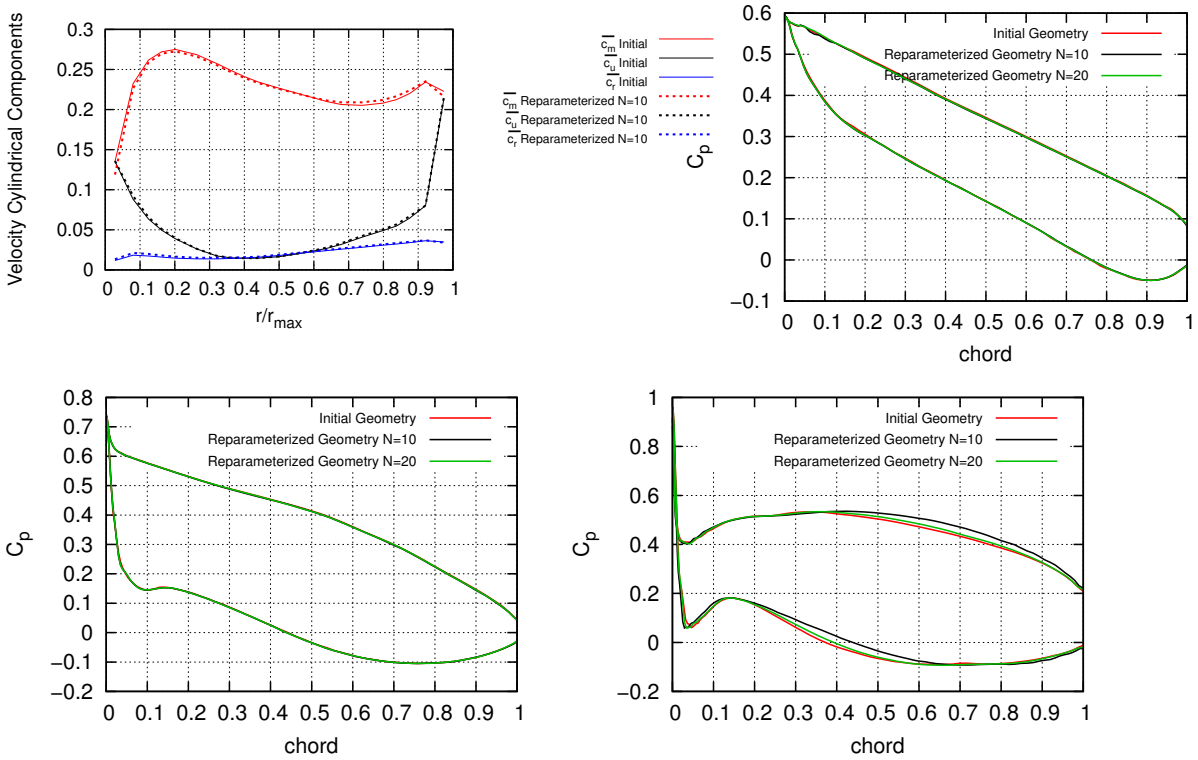
Για την αξιολόγηση της πιστότητας των δύο λογισμικών, παρουσιάζονται τρεις διαφορετικές γεωμετρίες πτερυγίων υδροστροβίλων που αρχικά βρίσκονται σε μορφή πλέγματος, στα οποία εφαρμόζονται τα λογισμικά αντίστροφης παραμετροποίησης (για τη δημιουργία της ισοδύναμης GMTurbo γεωμετρίας) και, στη συνέχεια, το λογισμικό προσαρμογής πλέγματος (για τη δημιουργία πλέγματος επί της GMTurbo γεωμετρίας). Οι τρεις γεωμετρίες είναι διαφορετικές μεταξύ τους προκειμένου να ερευνηθεί η ακρίβεια των δύο εργαλείων (RPT&GAT) σε διαφορετικές γεωμετρίες. Η αξιολόγηση της ροής γίνεται με τον επιλύτη των εξισώσεων Navier-Stokes, γνωστό ως PUMA που έχει αναπτυχθεί στη ΜΠΥΡ&Β.

5.1 Εφαρμογή σε υδροστρόβιλο **Francis**

Η πρώτη εφαρμογή αφορά υδροστρόβιλο μεικτής ροής Francis. Για τη σύγκριση της γεωμετρίας υπολογίζεται η απόσταση μεταξύ των κόμβων του αρχικού επιφανειακού πλέγματος του πτερυγίου και του πλέγματος που προκύπτει μετά την εφαρμογή των RPT&GAT (σχ. 5.1). Πραγματοποιείται επίσης σύγκριση των πεδίων ροής. Τα πεδία αυτά μοντελοποιούνται σε τέσσερις κατανομές. Η πρώτη αφορά τις κυλινδρικές συντεταγμένες της ταχύτητας στην έξοδο του ρότορα αδιαστατοποιημένες με έναν σταθερό αριθμό και οι υπόλοιπες, τις κατανομές του συντελεστή πίεσης κατά μήκος της χορδής σε τρεις κατά μήκος του πτερυγίου θέσεις, hub, shroud και την ενδιάμεση θέση (σχ. 5.2)



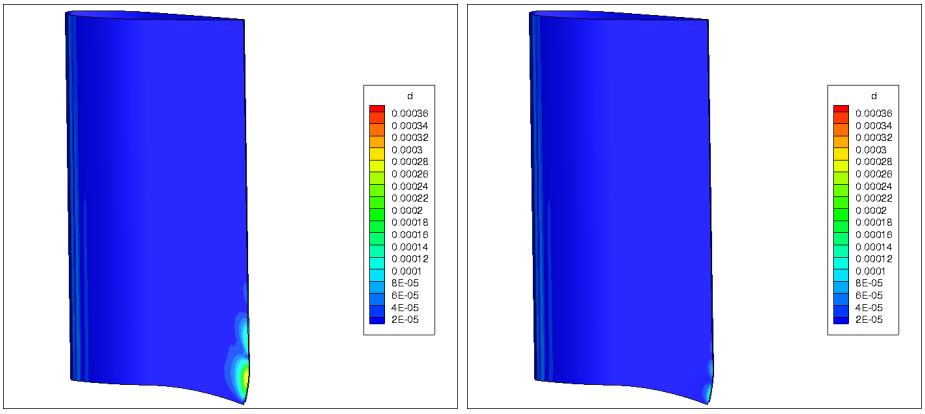
Σχήμα 5.1: *Εφαρμογή σε υδροστρόβιλο Francis: Η απόκλιση των δύο γεωμετριών, αρχικής και παραμετροποιημένης, για N=7 γενέτειρες (αριστερά) και N=20 γενέτειρες (δεξιά). Σαφώς αυξάνοντας τον αριθμό των γενετειρών αυξάνεται η ακρίβεια της προσέγγισης.*



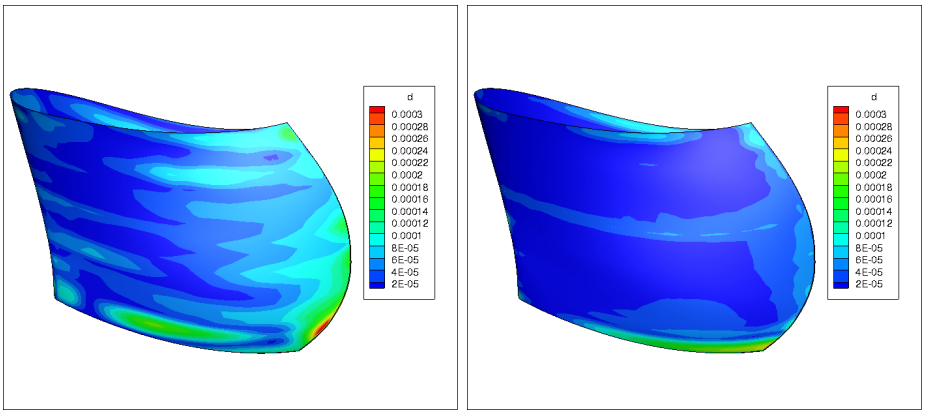
Σχήμα 5.2: *Εφαρμογή σε υδροστρόβιλο Francis: Οι διάφορες κατανομές των ροϊκών μεγεθών, για N=10 γενέτειρες, και στην περίπτωση των κατανομών συντελεστών πίεσης για N=20 γενέτειρες.*

5.2 Εφαρμογή σε υδροστρόβιλο τύπου Προπέλας με οδηγά πτερύγια

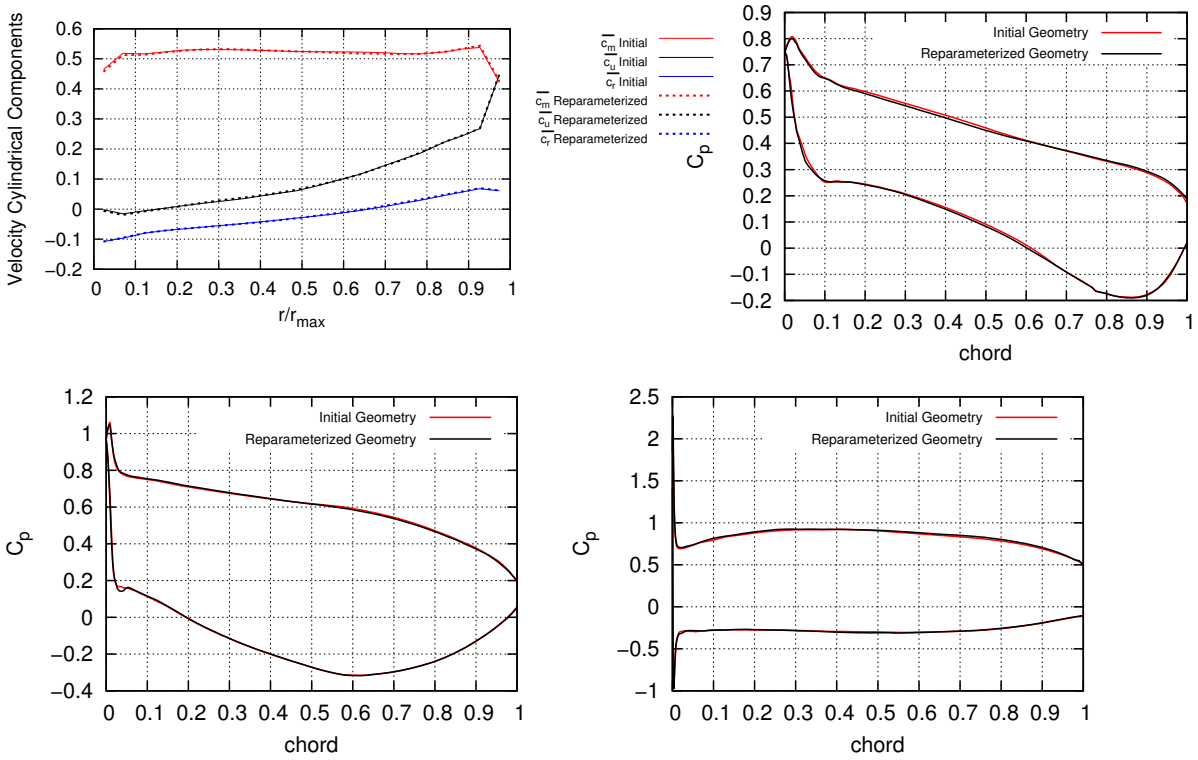
Η δεύτερη εφαρμογή αφορά έναν αξονικό στρόβιλο τύπου προπέλας με οδηγά πτερύγια. Εφαρμόζονται τα λογισμικά RPT&GAT και στα σταθερά οδηγά πτερύγια που είναι ακτινικά και στον αξονικό ρότορα. Η σύγκριση των γεωμετριών φαίνεται στα σχ. 5.3 και 5.4. Παρατηρώντας τα αποτελέσματα ΥΡΔ μπορεί κανείς να θεωρήσει ότι οι δύο γεωμετρίες είναι ροϊκά σχεδόν ισοδύναμες (σχ. 5.5).



Σχήμα 5.3: *Εφαρμογή σε υδροστρόβιλο τύπου Προπέλας με οδηγά πτερύγια: Η απόκλιση των δύο γεωμετριών των οδηγών πτερυγίων, αρχικού και παραμετροποιημένου, για N=8 γενέτειρες (αριστερά) και N=18 γενέτειρες (δεξιά). Στη συγκεκριμένη γεωμετρία N=8 γενέτειρες είναι αρκετές, εκτός από τα σημεία όπου η γεωμετρία έχει ιδιαιτερότητες οπότε χρειάζονται περισσότερες γενέτειρες για να εξαλειφθεί η απόκλιση.*



Σχήμα 5.4: *Εφαρμογή σε υδροστρόβιλο τύπου Προπέλας με οδηγά πτερύγια: Η απόκλιση των δύο γεωμετριών του ρότορα, αρχικής και παραμετροποιημένης, για N=8 γενέτειρες (αριστερά) και N=18 γενέτειρες (δεξιά). Αυξάνοντας τον αριθμό των γενετειρών αυξάνεται η ακρίβεια της προσέγγισης.*



Σχήμα 5.5: *Εφαρμογή σε υδροστρόβιλο τύπου Προπέλας με οδηγά πτερύγια: Οι διάφορες κατανομές των ροϊκών μεγεθών, για N=10 γενέτειρες, και στην περίπτωση των κατανομών συντελεστών πίεσης για N=20 γενέτειρες.*

5.3 Βελτιστοποίηση Μορφής

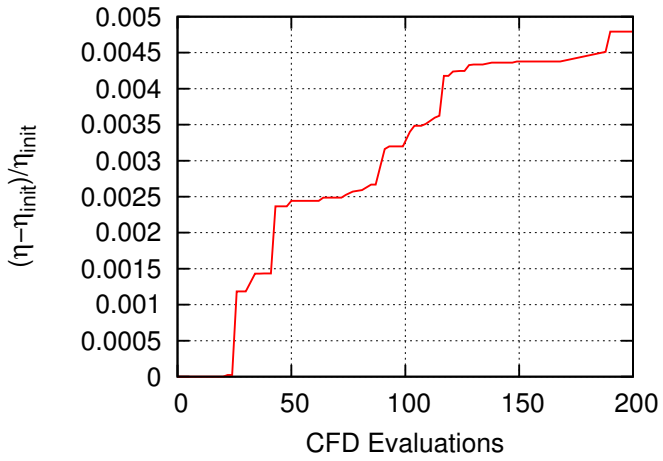
Αφού τα δύο λογισμικά RPT&GAT αξιολογήθηκαν, και κρίθηκε οτι η GMTurbo γεωμετρία μπορεί να θεωρηθεί καλή προσέγγιση της αρχικής γεωμετρίας του πτερυγίου που δίνεται σε μορφή πλέγματος, μπορεί να χρησιμοποιηθεί ώς αρχική λύση για βελτιστοποίηση με Εξελικτικούς Αλγορίθμους. Το λογισμικό που χρησιμοποιείται για τη βελτιστοποίηση είναι το λογισμικό EASY το οποίο έχει αναπτυχθεί από τη ΜΠΥΡ&Β.

Για τη βελτιστοποίηση μίας γεωμετρίας που αρχικά δίνεται σε μορφή πλέγματος πραγματοποιούνται κάθε φορά οι εξής ενέργειες. Αρχικά το πλέγμα μετατρέπεται σε GMTurbo γεωμετρία με τη χρήση του RPT. Στο σημείο αυτό, έχουν υπολογιστεί οι GMTurbo παράμετροι και επιλέγονται από το χρήστη κάποιες από αυτές ώς μεταβλητές σχεδιασμού καθώς και τα όρια τους. Έτσι η βελτιστοποίηση δύναται να εκκινήσει.

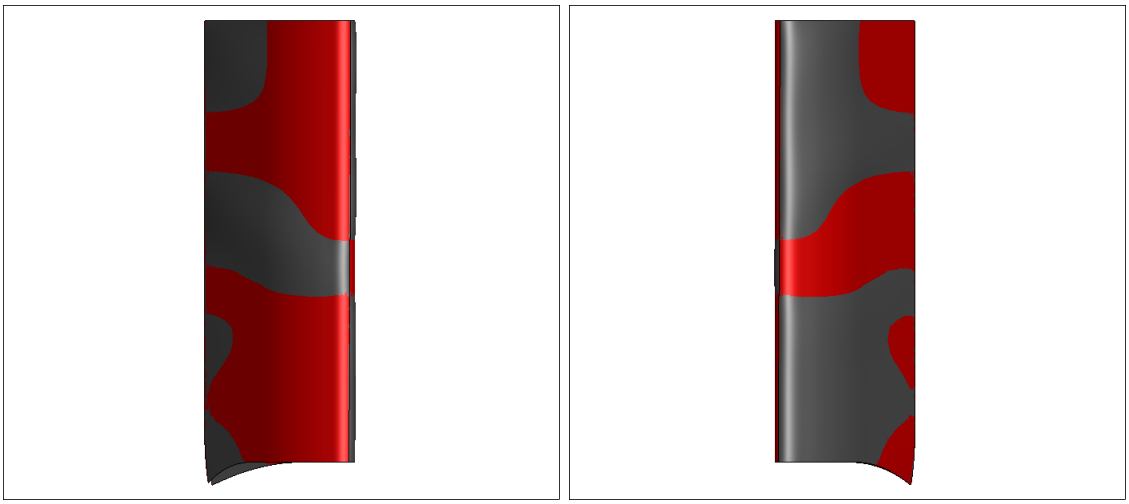
Οι τρείς περιπτώσεις που παρουσιάζονται αφορούν τη βελτιστοποίηση μορφής της εφαρμογής της ενότητας 5.2 με αντικειμενική συνάρτηση τον βαθμό απόδοσης της συνολικής διάταξης.

5.3.1 Βελτιστοποίηση οδηγών πτερυγίων (16 μεταβλητές σχεδιασμού)

Στην πρώτη περίπτωση, πραγματοποιείται βελτιστοποίηση μορφής μόνο των οδηγών πτερυγίων με 16 μεταβλητές σχεδιασμού που αφορούν τις γωνίες μετάλλου β_{LE} και β_{TE} σε 8 κατά μήκος θέσεις του πτερυγίου. Η σύγκλιση φαίνεται στο σχ. 5.6 και η βελτιστοποιημένη γεωμετρία στο σχ. 5.7.



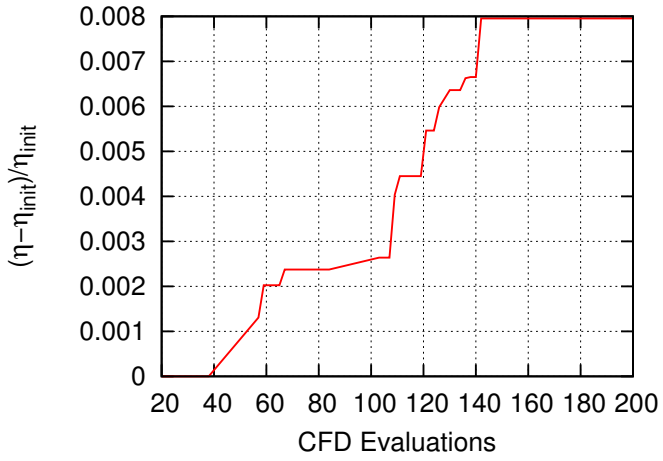
Σχήμα 5.6: *Βελτιστοποίηση οδηγών πτερυγίων: 16 μεταβλητές σχεδιασμού : Σύγκλιση.*



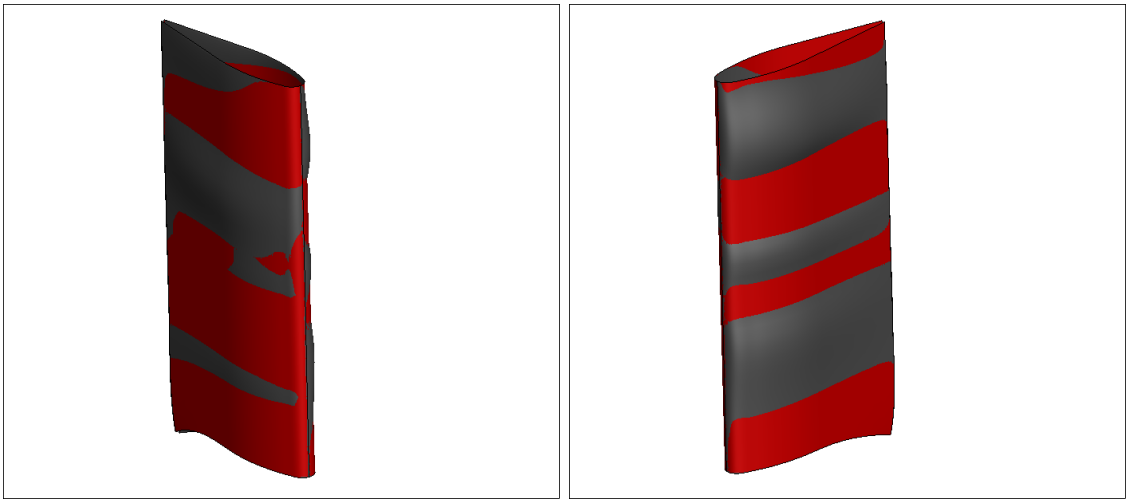
Σχήμα 5.7: *Βελτιστοποίηση οδηγών πτερυγίων (16 μεταβλητές σχεδιασμού): Η αρχική (κόκκινο χρώμα) και η βελτιστοποιημένη (γκρίζο χρώμα) γεωμετρία.*

5.3.2 Βελτιστοποίηση οδηγών πτερυγίων (32 μεταβλητές σχεδιασμού)

Στην δεύτερη περίπτωση, πραγματοποιείται βελτιστοποίηση μορφής και πάλι μόνο των οδηγών πτερυγίων, αλλά με 32 μεταβλητές σχεδιασμού. Οι πρώτες 16 αφορούν όπως στην ενότητα 5.3.1 τις γωνίες μετάλλου β_{LE} και β_{TE} σε 8 κατά μήκος θέσεις του πτερυγίου, ενώ οι υπόλοιπες 16 τους συντελεστές πάχους t_f^{PS} και t_f^{SS} στις ίδιες κατά μήκος θέσεις. Η σύγκλιση φαίνεται στο σχ. 5.8 και η βελτιστοποιημένη γεωμετρία στο σχ. 5.9.



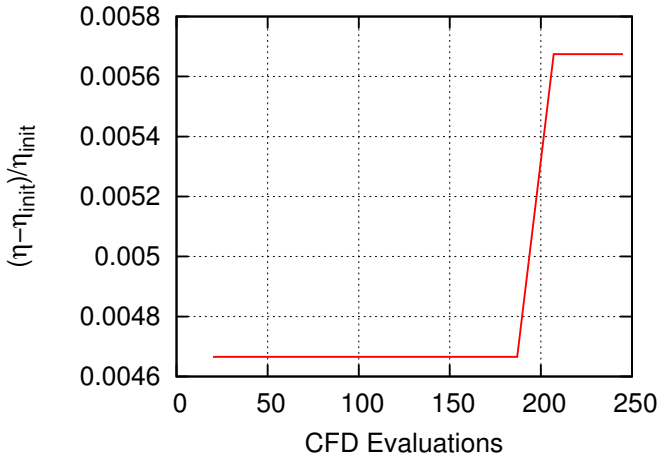
Σχήμα 5.8: *Βελτιστοποίηση οδηγών πτερυγίων (32 μεταβλητές σχεδιασμού): Σύγκλιση.*



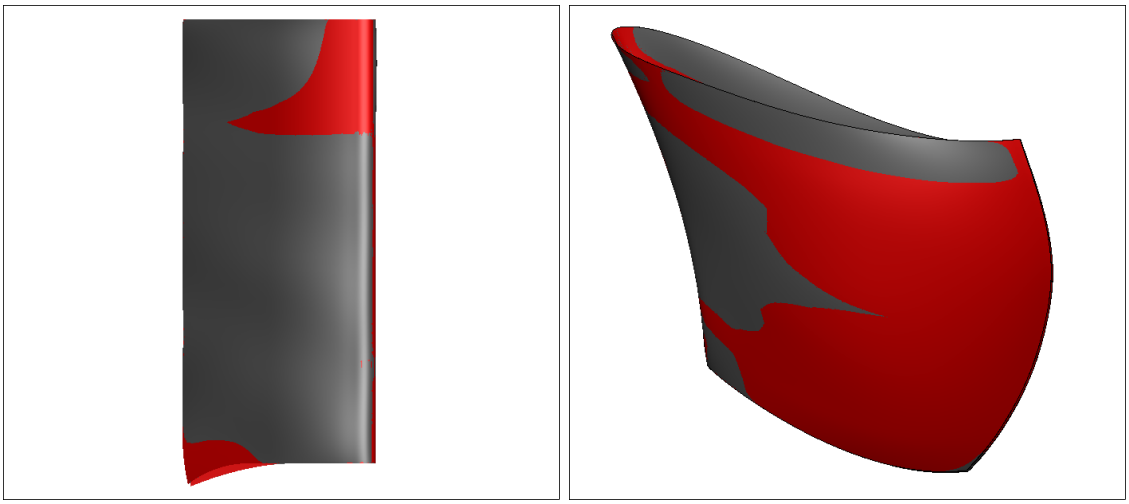
Σχήμα 5.9: *Βελτιστοποίηση οδηγών πτερυγίων (32 μεταβλητές σχεδιασμού): Η αρχική (κόκκινο χρώμα) και η βελτιστοποιημένη (γκρίζο χρώμα) γεωμετρία.*

5.3.3 Βελτιστοποίηση οδηγών πτερυγίων και πτερωτής

Η τρίτη περίπτωση αφορά τη ταυτόχρονη βελτιστοποίηση των οδηγών πτερυγίων και των πτερυγίων της πτερωτής. Οι μεταβλητές σχεδιασμού είναι 32 και αφορούν όλες τις γωνίες μετάλλου β_{LE} και β_{TE} σε 8 κατά μήκος θέσεις των οδηγών πτερυγίων και σε 8 κατά μήκος θέσεις της πτερωτής. Η σύγκλιση φαίνεται στο σχ. 5.10 και η βελτιστοποιημένη γεωμετρία στο σχ. 5.11 και 5.12.



Σχήμα 5.10: *Βελτιστοποίηση οδηγών πτερυγίων και πτερωτής: Σύγκλιση.*



Σχήμα 5.11: *Βελτιστοποίηση οδηγών πτερυγίων και πτερωτής: Η αρχική (κόκκινο χρώμα) και η βελτιστοποιημένη (γκρίζο χρώμα) γεωμετρία των οδηγών πτερυγίων και της πτερωτής.*

Κεφάλαιο 6

Συμπεράσματα

Σε αυτήν τη διπλωματική εργασία προγραμματίστηκε λογισμικό αντίστροφης παραμετροποίησης που μετατρέπει πλέγμα ΥΡΔ σε συμβατή με το GMTurbo μορφή. Επίσης προγραμματίστηκε Λογισμικό Προσαρμογής Πλέγματος το οποίο υπολογίζει πλέγμα γύρω από τη GMTurbo γεωμετρία. **Συνεπώς, είναι πλέον εφικτό να πραγματοποιηθεί βελτιστοποίηση γεωμετρίας που δίνεται αρχικά σε μορφή πλέγματος με τη χρήση του GMTurbo**. Συγκεκριμένα, σε αυτήν την εργασία, αναλύθηκαν, προγραμματίστηκαν και αξιολογήθηκαν τα εξής:

- Αναλύθηκε η μέθοδος που ακολουθείται απο το λογισμικό GMTurbo για τη παραμετροποίηση πτερυγίων στροβιλομηχανών
- Προγραμματίστηκαν δύο λογισμικά. Το RPT και το GAT. Το RPT μετατρέπει πλέγμα ΥΡΔ σε παραμέτρους GMTurbo. Το GAT δημιουργεί πλέγμα πάνω στη GMTurbo γεωμετρία. Το μεν RPT επιτρέπει την εκκίνηση της βελτιστοποίησης αφού η GMTurbo γεωμετρία που παράγει κατασκευάζεται άμεσα από τις μεταβλητές σχεδιασμού. Το GAT ενσωματώνεται στον αλγόριθμο αξιολόγισης του Εξελικτικού Αλγορίθμου μαζί με άλλα λογισμικά (σχ. 1.5) για τη δημιουργία πλέγματος πάνω στην CAD γεωμετρία σε κάθε αξιολόγηση.
- Τέλος αξιολογήθηκε η πιστότητα των δύο λογισμικών. Για τρείς διαφορετικές γεωμετρίες που προέρχονται από τη βιομηχανία, πραγματοποιήθηκαν συγκρίσεις του αρχικού πλέγματος με την γεωμετρία που προκύπτει μετά από την εφαρμογή των RPT&GAT. Οι συγκρίσεις που έγιναν είναι γεωμετρικές και συγκρίσεις ροϊκών μεγεθών. Τέλος πραγματοποιήθηκαν τρείς βελτιστοποιήσεις γεωμετριών που αρχικά βρίσκονταν σε μορφή πλέγματος, αποδεικνύοντας ότι ο αρχικός στόχος επετεύχθη.