POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es

Pablo Rodríguez Fernández

05 PROYECTO FIN DE CARRERA

INDUSTRIALES

PROYECTO FIN DE CARRERA

Development of Shape-Optimization Tools for the Aerodynamic Design of Turbomachinery Blades

PROYECTO FIN DE CARRERA
PARA LA OBTENCIÓN
DEL TÍTULO DE
INGENIERO INDUSTRIAL

MARZO **2015**

Pablo Rodríguez Fernández

DIRECTOR DEL PROYECTO:
Juan Luis Prieto Ortiz

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

Development of Shape-Optimization Tools for the Aerodynamic Design of Turbomachinery Blades

Pablo Rodríguez Fernández

March 12, 2015

Abstract

The aim of this work is to develop an automated tool for the optimization of turbomachinery blades founded on an evolutionary strategy. This optimization scheme will serve to deal with supersonic blades cascades for application to Organic Rankine Cycle (ORC) turbines. The blade geometry is defined using parameterization techniques based on B-Splines curves, that allow to have a local control of the shape. The location in space of the control points of the B-Spline curve define the design variables of the optimization problem. In the present work, the performance of the blade shape is assessed by means of fully-turbulent flow simulations performed with a CFD package, in which a look-up table method is applied to ensure an accurate thermodynamic treatment. The solver is set along with the optimization tool to determine the optimal shape of the blade. As only blade-to-blade effects are of interest in this study, quasi-3D calculations are performed, and a single-objective evolutionary strategy is applied to the optimization. As a result, a non-intrusive tool, with no need for gradients definition, is developed. The computational cost is reduced by the use of surrogate models. A Gaussian interpolation scheme (Kriging model) is applied for the estimated n-dimensional function, and a surrogate-based local optimization strategy is proved to yield an accurate way for optimization. In particular, the present optimization scheme has been applied to the re-design of a supersonic stator cascade of an axial-flow turbine. In this design exercise very strong shock waves are generated in the rear blade suction side and shock-boundary layer interaction mechanisms occur. A significant efficiency improvement as a consequence of a more uniform flow at the blade outlet section of the stator is achieved. This is also expected to provide beneficial effects on the design of a subsequent downstream rotor. The method provides an improvement to gradient-based methods and an optimized blade geometry is easily achieved using the genetic algorithm.

Acknowledgements

First, I have to thank my research advisor, Prof. Giacomo B. Persico. Without his guidance, involvement and ideas, this project would have never been accomplished. I also appreciate the reviewing and encouraging labor of my supervisor, Prof. Juan L. Prieto. I further extend my gratitude to Gabriela Scardine, for having kindly accepted to review this thesis. I cordially thank the Turbomachinery Group of Politecnico di Milano, for their warm reception and help during the development of this thesis.

To carry out this international project, I have enjoyed the financial support of the EAGLES fellowship for my studies in the US, and the Erasmus scholarship from the European Union for my research stay in Italy.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Nowadays, computational methods are employed to cover a wide range of engineering applications. In the field of turbomachinery, for example, several optimization tools have been developed to help in the design of improved machines, with higher efficiency, reduced losses, and better performance [1,2].

These optimization tools cannot always be developed using analytical techniques. Complex problems, such as non-linearities in the governing equations and non-practical solutions, have to be solved using approximations. As a consequence, heuristic techniques started to grow. These methods provide a wide range of solutions to a vast amount of complex problems. By definition, a heuristic technique is that which employs a practical methodology as the approach to problem solving. The methodology that they utilize is not guaranteed to reach the optimum, but the outcome is considered a reasonable approach to the real solution considering the problem difficulty. In particular, a genetic algorithm (GA) is a heuristic method that provides an interesting approach to solve problems, by mimicking how nature works. Genetic algorithms are built over the concept of natural selection proposed by the neo-Darwinism. The popularity and power of GAs is mainly based on the fact that the specifications of the problem are, hypothetically, of any kind [3].

In the last years, many optimization techniques coupled with computational fluid dynamics codes (CFD) have been developed. The advantage of using genetic algorithms along with CFD-based schemes are many and varied. For instance, they can easily approach multiobjetive optimization and their implementation is usually non-intrusive, which enables to build automatic tools to resolve complex engineering problems [1].

The field of turbomachinery is in constant development, and it presents many challenges that can benefit from those techniques. Turbomachinery design processes comprise optimization at several stages, from the preliminary design [1] to the blade shape definition [2], passing through the axisymmetric design (span-wise blade design). The optimization problems involve many objectives and constraints, and thus the use of evolutionary strategies can be widely found in literature.

1.2 Motivation

Many studies have been performed to improve optimization methods and to help in the design of innovative and enhanced components of turbomachinery. Also, CFD methods and their coupling with optimization techniques have arisen in the last decade. However, an automated tool (Fig. 1.1) for the shape-optimization of turbomachinery blades applicable to general cases has not been developed yet. This study aims to create a mechanism whose final purpose is to provide an easy way to build high-performing blade shapes both in the early steps of the design of new turbomachinery models and in the improvement of current configurations. Traditionally, developing new machines does not necessarily include the blade shape modeling. With this tool, this step can be easily overcome and can be included from the beginning of the process.

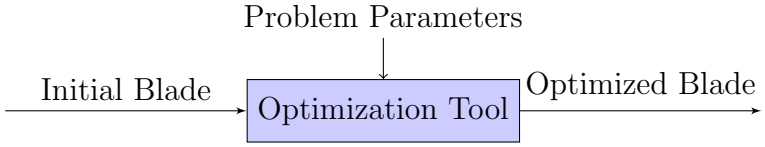


Figure 1.1: Black-Box Scheme of Optimization Tool[1].

The implicit purpose of this study is to create a tool that is able to extract the minimal physical data of the problem and work with a purely numerical optimization. By doing so, the optimization scheme can be easily implemented, as only a mathematical function is aimed to optimize. This enables to develop a tool for an extensively wide range of engineering applications.

In Chaper 2, the methodology employed in the development of the shape-optimization tool is described. For information about how to implement the algorithm, please see Chapter 3. In Chapter 4, a supersonic blade cascade of

[1]A black box function or process is a function or process that without been explicitly described and given a list of a finite number of points in the input space, corresponding outputs can be obtained.

an axial ORC turbine is optimized to validate this tool. Lastly, in Chapter 5, the main conclusions and outcomes of this work are discussed.

Chapter 2

Methodology

In this chapter we define the different methods used in the development of the shape-optimization tool. The procedure strongly relies on the combined used of four *modules*:

1. Geometry Parameterization.

2. Computational Fluid Dynamics (CFD) simulation codes.

3. Genetic Algorithms.

4. Surrogate Models.

For the different optimization steps, the main characteristics and theoretical founding will be defined in the following sections. In Section 2.1, the geometry parameterization technique used in this study is described; in Section 2.2, some key features of the CFD tool used for the evaluations of the high-fidelity model are shown; in Section 2.3, the genetic algorithm is presented; and in Section 2.4, the surrogate-based optimization strategy is illustrated. For information about how to connect the different parts and how to implement a robust algorithm, please see Chapter 3. Please notice that each section contains its own notation and symbols, which are described at the moment they are used. In Fig. 2.1 one can observe that all the different sections are interrelated in the optimization tool presented in this study.

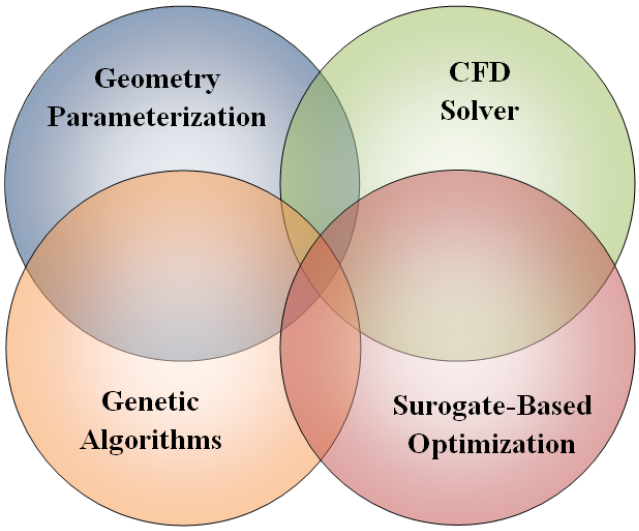


Figure 2.1: Components of the optimization tool.

2.1 Geometry Parameterization

Geometry Parametrization is the first module to be executed in the optimization tool developed in this research. It assumes that the end user has specified the initial shape of a blade, as described in the Black Box Scheme of Fig. 1.1. The objective of this module is to parameterize the geometry that defines the blade shape, so that it can be manipulated in the next modules. Throughout this dissertation, we will use the terms blade shape, geometry, and curve interchangeably.

The optimization strategy has been set to work with a limited number of design variables. This is because, in shape-optimization, the position of every point in the curve cannot be provided to the optimization algorithm as variables. Not only because that would consume a lot of computational time, but also the result would not be useful for a subsequent design. Therefore, our approach uses a method to create the shape of a blade as a function of a bounded number of variables. As a matter of fact, those variables have to provide local effects on the final shape of the curve.

The method that has been used in this work involves the use of a special type of 2-D curve called B-Spline, taking into account that a three-dimensional blade geometry can be constructed using several two-dimensional blade sections. The next subsection (2.1.1) introduces B-Spline curves and its properties, in order to explain why this method has been chosen to determine

the shape of the blade, and to lay the foundations for subsequent explanations on how this method is applied. For further information, a full account of the theory of B-Splines and the mathematical representation can be found in Refs. [4, 5]. The following subsection (2.1.2), Interpolation Procedure, expands on the explanation of how the initial blade geometry is reproduced using B-Spline curves. Finally, the last subsection about geometry parameterization (2.1.3), Automated Algorithm, describes how these concepts have been computationally applied.

2.1.1 B-Spline curves

Since the introduction of B-Splines in the nineteenth century , they have been recognized as powerful tools both in application and theory [5]. Nowadays, they are widely used in industry and design due to very interesting properties:

- They involve piecewise curves with components of degree n.
- They provide local support, so a change in the position of a control point only affects the curve in a given interval that can be specified by the degree of the individual curves.
- They can be defined to pass through the first and last control points.
- They are contained in the convex hull of their control polygon. This allows more control on the shape by changing the control points.
- Their smoothness and continuity can be controlled by the multiplicity of the knot sequence.
- An affine transformation applied to a B-Spline curve can be constructed from the affine images of its control points

To fully understand the mathematical insight of B-Splines, it is important to define the properties of the basic building blocks: linear interpolations. This will led the reader to the concept of Bézier curves that will be eventually generalized in B-Splines.

Linear Interpolation

The basics of any computation in geometrical modeling are the linear interpolations. Let p_0 and p_1 be two points in the space. The straight line between them has the form:

$$x(t) = (1 - t) \cdot p_0 + t \cdot p_1 \tag{2.1}$$

where t is a parameter. If $t = 0$ then $x(t) = p_0$, and if $t = 1$ then $x(t) = p_1$. Therefore, this can be conceived as an affine mapping from the interval $[0, 1]$ to the space (in our case, $\mathbf{R}^2$).

At this point we can define an important concept that will be very useful: the blossom[1], a n-variate function $b[t_1, t_2...t_n]$ that is completely characterized by the following axioms:

1. Symmetry - The order of the blossom argument does not matter.

$$b[t_1, t_2, ..., t_n] = b[t_{\sigma(1)}, t_{\sigma(2)}, ..., t_{\sigma(n)}]; \tag{2.2}$$

   where σ is any permutation of $\{1, 2...n\}$

2. Multiaffinity - The blossom is affine with respect to all its arguments.

$$b[\alpha\ r + \beta\ s, t_2...t_n] = \alpha\ b[r, t_2, ..., t_n] + \beta\ b[s, t_2, ..., t_n]; \tag{2.3}$$

3. Diagonality - If the arguments are equal, then the blossom draws a polynomial curve.

$$b[t, t, ..., t] = b[t^{<n>}] = b[t] \tag{2.4}$$

Taking a careful look at the blossom structure and the affine interpolation defined previously, it can be inferred that a blossom of the form $b[t]$ can represent a linear interpolation with parameter t between two points. The latter are given for the values of the blossom $b[0]$ and $b[1]$. In the following sections this concept will arise and become useful.

Bézier Curves

Given a set of points $p_0, p_1, ..., p_n$ and a parameter t we can define different sets of points given by:

$$p_i^r(t) = (1 - t) \cdot p_i^{r-1}(t) + t \cdot p_{i+1}^{r-1}(t); \tag{2.5}$$

where $r = 1, ..., n$ and $i = 0, ..., n - r$.

The set of initial points are called *Bézier points* (or *Control points*), and are written as $p_i = p_i^0(t)$. These points define the *Bézier polygon* (or *Control polygon*), and Eq. 2.5 is called the *de Casteljau Algorithm*. The points $p(t) = p_0^n(t)$ define the *Bézier curve*, whose parameter n is the degree of the curve.

[1] In numerical analysis, a blossom is a functional that can be applied to any polynomial, but this concept was invented for the use in Bézier and B-Spline curves mainly [6].

One can observe that each point of a generation r is generated by affine interpolation of two points of the previous generation $r-1$. If the points of the last generation n are written as a function of the original points and a parameter t only, then the Bézier curve is generated.

Another way of representing Bézier curves (but more abstract), is by the concept of *blossoming*. We can define any point involved in the de Casteljau Algorithm as the following:

$$p_i^r(t) = b[0^{<n-r-i>}, t^{<r>}, 1^{<i>}]; \tag{2.6}$$

where <> represents how many times the variable appears[2]. In this way we can express the Bézier points as blossom values: $p_i = b[0^{<n-i>}, 1^{<i>}]$. The same way, a point on the curve is defined by $p^n(t) = b[t^{<n>}]$.

A graphical idea of this concept can be extracted from Fig. 2.2.

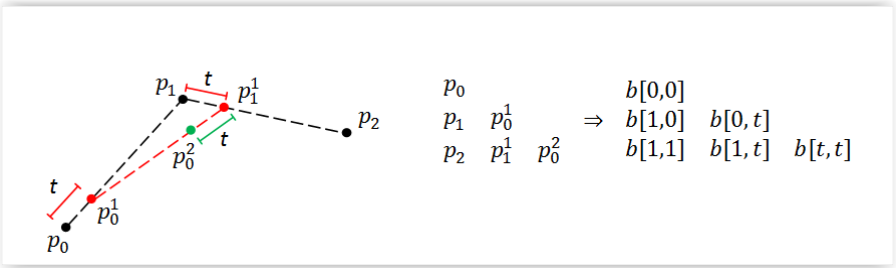


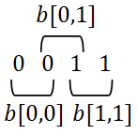
Figure 2.2: Bézier Process.

In this figure, one can observe what "blossoming" stands for. All the necessary points for the Bézier process can be generated from the previous points. Starting with p_0, p_1 and p_2 as the initial points, and considering a second-degree curve, one can define their blossoming formulations by taking into account that the only input to the blossom functional is the t parameter in each of the subsequent processes. For example, considering the blossom $b[1,0]$, the first parameter is 1, which means that in the linear interpolation between p_0 and p_1, and between p_1 and p_2 is $t=1$. Therefore the generated points are p_1 and p_2 respectively. The second parameter is 0, which means that in the linear interpolation between the previous points, i.e. p_1 and p_2, the parameter is 0. Therefore, the final point is p_1 .It can be shown that $b[1,0] = b[0,1]$, so the blossom symmetry property is complied. Following this process, all the intermediate points can be obtained by *blossoming* the mathematical flower of the successive interpolations.

[2]Recall: blossoms are symmetric

B-Spline Curves

B-Splines are a generalization of Bézier curves. In the latter, the control points were expressed in a blossom as $p_i = b[0^{<n-i>}, 1^{<i>}]$. The variables of this blossom were 0 and 1, both repeated depending on the degree and the position of the point. Then, each set of $\{0^{<n-i>}, 1^{<i>}\}$ can be obtained from a sequence of $\{0, 1\}$. From the second-order curve depicted in Fig. 2.2, the following sequence can be extracted as an example:



If we generalize this sequence we have the so-called *Knot Sequence*:

$$u = \{u_i\} \tag{2.7}$$

This sequence can contain any number, whose particularization to Bézier curves would be only 0's and 1's. Actually, as it will be explained in the following subsection, Interpolation Procedure, the spacing between the values is more important than the values themselves.

Hence, given a set of points $p_0, p_1, ..., p_n$ and a knot sequence $\{u_i\}$, we can define different sets of points over an interval $U = [u_m, u_{m+1}]$:

$$p_i^r(u) = b[u^{<r>}, U_i^{n-1-r}] \tag{2.8}$$

where U_i^r is a set of U that contains $r + 1$ successive knots and u_m is its $(r - i)$ th element[3].

Consequently, the set of control points is written as $p_i = b[U_i^{n-1}]$. Equation 2.8 is called the *de Boor Algorithm* and it is the generalization of the de Casteljau Algorithm. The points described as $x_0 = p_0^n(u)$ form the B-Spline curve.

In summary, an n-degree B-Spline consists of applying the de Boor Algorithm with a knot sequence $\{u_i\}_{i=0...K}$ to a control polygon $p_{j,j=0...L}$, such that $L = K - n + 1$. To have a better understanding, B-Splines are formed by dividing the knot sequence in intervals whose size depends on the degree of the curve. Then, a "*blossom*" is applied to the initial points in that interval

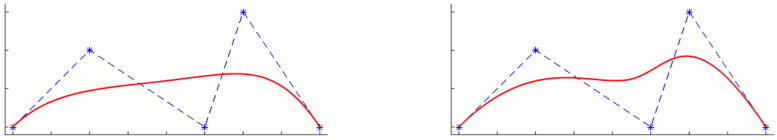
[3]Please notice a change in the notation. The parameter t (that defined the curve previously) is now written as u. This is more convenient since the parameter covers the range in the knot sequence $\{u_i\}$ and its influence on the final curve will depend on the subset U_i^r.

and many individual Bézier curves are created. These curves are intrinsically connected between each other because they share the initial blossoming points in a degree given by the multiplicity of the knot.

This is the reason why B-Splines can control the shape locally, without changing the degree and without modifying the continuity at the given point. One of the points can have more or less importance depending on the spacing in its knot sequence. The parameter used to build the curve (i.e. t or u, depending on the notation) would range from u_0 to u_{max}.

Observations and Comments

In Fig. 2.3 one can observe the difference between the construction of Bézier curves and B-Splines using the same control points. In B-Spline curves (2.3b),the shape is better defined by the control points. The B-Spline has an uniform knot sequence and multiplicity $m = 3$ in the ends. It can be observed the differences between Bézier curves and B-Splines. The same control points produce two different shapes. Note that the curve in 2.3b is more locally influenced by the relative position of the fourth control point.

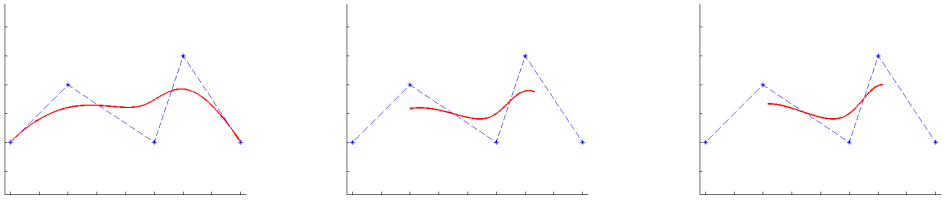


(a) Bézier Curve (b) B-Spline Curve

Figure 2.3: Differences between Bézier curves and B-Splines.

In Fig. 2.4 one can observe the influence of the multiplicity of the knot sequence in the ends. If the parameter u goes from the minimum to the maximum of the knot sequence, we need to establish a multiplicity of, at least, n in the ends so that the curve goes through them[4]. In this example, $n = 3$, and the internal knot sequence is uniform.

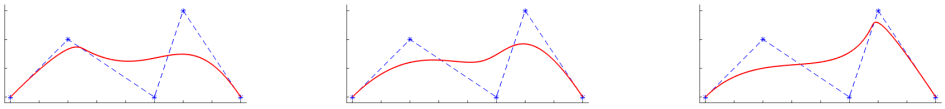
[4]This is commonly called Clamped Ends.



(a) $m = 3$ (b) $m = 2$ (c) $m = 1$

Figure 2.4: Ends multiplicity influence.

In Fig. 2.5 one can observe an example of controllability of the shape by changing the knots sequence and maintaining the control points. The knot sequence has the same structure in all of them: $\{0, 0, 0, u_i, 1, 1, 1\}$, while the value u_i changes in the different figures. The explanation for this is not straightforward. Analyzing how the points on the curve are generated, it can be inferred that, the smaller the interval in the knot sequence, the closer the curve is to the control polygon (blue dots) in that region. Therefore, the importance of the knots sequence is given by the interval between the knots and not by the values themselves. In Fig. 2.5c, the smallest knots interval is placed in the first half of the curve and therefore the curve attaches to the control polygon; Fig. 2.5b is the balanced case; and Fig. 2.5c gets the smallest interval at the end of the curve.

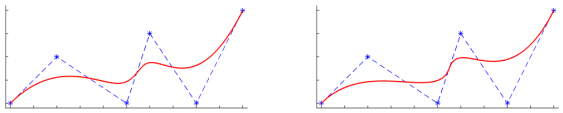


(a) $u_i = 0.1$ (b) $u_i = 0.5$ (c) $u_i = 0.9$

Figure 2.5: Knots sequence influence.

In Fig. 2.6 one can observe the influence of the multiplicity of an internal knot on the smoothness of the curve. In general, a knot of multiplicity r produces a local B-Spline with smoothness C^{n-r}. In this example, the knot sequence in Fig. 2.6a and Fig. 2.6b is $\{0, 0, 0, u_i, 0.5, 1, 1, 1\}$, differing only in the value u_i. In both figures, the degree of the curve is $n = 3$, and the end knots have multiplicity $r = 3$. Therefore, the smoothness of the B-spline curve in the end knots is the same: C^0. However, the smoothness in the inner knots differ in each case. In the first case, Fig. 2.6a, the inner knots have

multiplicity $r = 1$, resulting in a smoothness C^2. In Fig. 2.6b, a multiplicity of $r = 2$ results in a lower smoothness C^1.



(a) $u_i = 0.3$ (b) $u_i = 0.5$

Figure 2.6: Multiplicity influence in the smoothness.

In Fig. 2.7 one can observe the influence of the position of the control points on a B-Spline curve. In this example, B-Spline curves for different position of the fourth control point are depicted. This gives an idea of the local controllability, important feature of B-Splines.

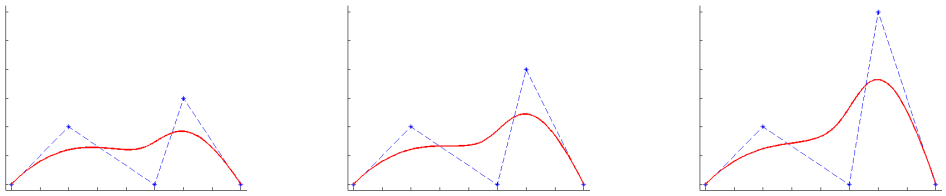
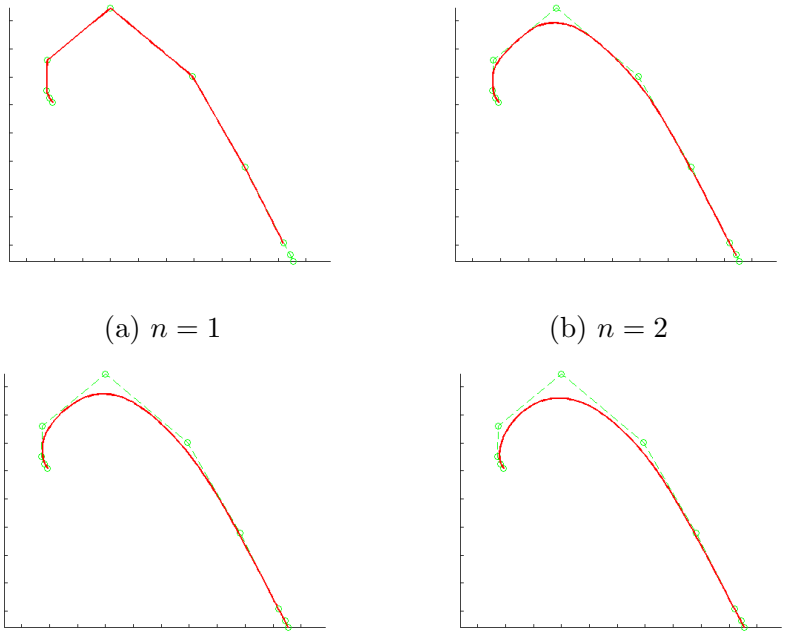


Figure 2.7: B-Spline curves for different position of one control point.

In Fig. 2.8, one can observe the application of B-Splines to the design of the shape of a blade. In addition, one can observe the influence of the degree n of the curve in the final shape. In this example, the suction side of a blade is represented by B-Spline curves with the same control points (number of control points $L = 10$), a uniform knot sequence, but different degrees of the curve. This small difference produces very different blades. The first two, illustrated in Fig. 2.8a and 2.8b, do not present second order continuity. As a consequence, its use is not recommended to build blade geometries. Therefore, a tool to create proper blade shapes using B-spline curves requires a degree $n \geq 3$, yielding a second order continuity in all inner points. Figures 2.8c and 2.8d are considered admissible blade shapes, because they show, respectively, a cubic B-Spline and a quartic B-Spline.



(a) $n = 1$ (b) $n = 2$

(c) $n = 3$ (d) $n = 4$

Figure 2.8: Suction side of a blade for different curve degrees. Admissible blade shapes yield a second order continuity in all inner points. Because this only happens in curves whose degree $n \geq 3$, (a) and (b) are not admissible shapes, but (c) and (d) are.

2.1.2 Interpolation Procedure

Once the basics of B-Spline curves have been explained, this section explores how these curves have been used to represent blade shapes in our optimization tool. Throughout this dissertation, we will use the term data points to define the parameters that represent the initial shape of the blade, given by the user. Our optimization tool creates an approximate representation of this shape using a B-Spline curve, defined and manipulated by control points. In order to create this B-Spline, the data points need to be interpolated. Our optimization tool has used the least square interpolation method.

It is assumed that $P + 1$ data points p_i are given, with $i = 0...P$, and we seek to find the approximated B-Spline curve $x(u)$ of degree n and $K + 1$ knots u_k, with $k = 0...K$. This B-Spline curve will be defined by $L+1$ control points d_j, with $j = 0...L$, such that $L = K - n + 1$.

For this problem, the $P + 1$ data parameters w_i assigned to each given point need to be defined to find the least squares approximation. Then, the

error of the approximation for a given point can be expressed as $\|p_i - p(w_i)\|$. Therefore, the objective is to minimize the sum of all the approximation errors:

$$f(x) = \sum_{i=0}^{P} \|p_i - x(w_i)\| \tag{2.9}$$

In this application, it is useful to write a B-Spline in the form:

$$x(u) = \sum_{j=0}^{L} d_j N_j^n(u) \tag{2.10}$$

where $N_j^n(u)$ are called B-Splines and provide a local support. They can be defined recursively in the form:

$$N_j^k(u) = \frac{u - u_{j-1}}{u_{j+k-1} - u_{j-1}} N_j^{k-1}(u) + \frac{u_{j+k} - u}{u_{j+k} - u_j} N_{j+1}^{k-1}(u); \tag{2.11}$$

$$N_j^0(u) = \begin{cases} 0 & if \quad u_{j-1} \leq u < u_i, \\ 1 & if \quad otherwise \end{cases} \tag{2.12}$$

This notation refers to the general form of each of the individual B-Spline bases, providing the local support to each control point (Eq. 2.9). Once defined the B-Splines, we can rewrite:

$$f(\{d_j\}_0^L) = \sum_{i=0}^{P} \|p_i - \sum_{j=0}^{L} d_j N_j^n(w_i)\| \tag{2.13}$$

If a least squares approach is now carried out, the final formula for the $L+1$ normal equations (represented by the parameter $k = 1...L$) is derived (for more information, see [4]):

$$\sum_{j=0}^{L} d_j \sum_{i=0}^{P} N_j^n(w_i) N_k^n(w_i) = \sum_{i=0}^{P} p_i N_k^n(w_i); \tag{2.14}$$

This equation leads to the linear system:

$$\begin{bmatrix} m_{0,0} & m_{0,1} & \dots & m_{0,L} \\ m_{1,0} & m_{1,1} & \dots & m_{1,L} \\ \vdots & \vdots & \ddots & \vdots \\ m_{L,0} & m_{L,1} & \dots & m_{L,L} \end{bmatrix} \cdot \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_L \end{bmatrix} = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_L \end{bmatrix} \tag{2.15}$$

The linear system 2.15 can be written as $A \cdot x = B$, where A is a symmetric, square matrix. In this case, the system is solved by using *Cholesky*

Decomposition, since matrix A can be ill-conditioned (nearly singular), depending on the relative position of the points.

As matrix A is a symmetric, square, and real matrix, it is **positive definite**. It is known that Cholesky decomposition is unique when the matrix is of this kind:

$$A = L \cdot L^* \tag{2.16}$$

where L is lower triangular.

Equation 2.15 can be then written and solved as:

$$\begin{aligned} L \cdot L^* \cdot x &= B \\ L \cdot y = B &\Rightarrow y \\ L^* \cdot x = y &\Rightarrow x \end{aligned} \tag{2.17}$$

It is worth mentioning that the least squares approximation does not necessarily pass through any of the original data points, but an additional operation can be performed to guarantee that the resulting geometry passes through the first and last data points. A way to do that it is to fix the first and last control points of the interpolating curve. As we have decided to use a multiplicity of $m = n$ in the ends, we can assure that the B-Spline will pass through the first and last control points. Hence, by making them equal to the original points we can assure this to happen.

This connection is carried out simply by setting the first and last points in the least squares approximation matrices. Accounting for Eq. 2.15, we make the following modifications:

$$\begin{bmatrix} \hat{m_{0,0}} & 0 & \dots & 0 & 0 \\ 0 & m_{1,1} & \dots & m_{1,L-1} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & m_{L-1,1} & \dots & m_{L-1,L-1} & 0 \\ 0 & 0 & \dots & 0 & \hat{m_{L,L}} \end{bmatrix} \cdot \begin{bmatrix} P_{first} \\ d_1 \\ \vdots \\ d_{L-1} \\ P_{last} \end{bmatrix} = \begin{bmatrix} h_0 \\ \hat{h_1} \\ \vdots \\ \hat{h_{L-1}} \\ \hat{h_L} \end{bmatrix} \tag{2.18}$$

Hence, the new components turn out to be:

$$\begin{aligned} \hat{m_{0,0}} &= \tfrac{h_0}{P_{first}} \\ \hat{m_{L,L}} &= \tfrac{h_L}{P_{last}} \\ \hat{h_i} &= h_i - m_{i,0} \cdot P_{first} - m_{i,0} \cdot P_{last} \end{aligned} \tag{2.19}$$

Data Parameters

When dealing with a least squares interpolation, a sequence of data parameters $\{w_i\}$ has to be selected. From the different methods of determining the sequence, a centripetal parameterization ([7]) has been chosen, due to sharp turns that the blade may experience.

Let $w_0 = 0$ and $S = \sum_{i=0}^{P} \sqrt{\|p_i - p_{i-1}\|}$. Then we can write the rest of parameters:

$$w_i = w_{i-1} + \frac{\sqrt{\|p_i - p_{i-1}\|}}{S} \tag{2.20}$$

Knot Sequence

The sequence of knots $\{u_i\}$, $i = 0...K$, has to be decided as well. We need to remember that the influence of the knot sequence is given by the spacing between its points. Hence, another sequence of *spacing* $\{S_i\}$ defined from 1 to K can be defined. Then a knot sequence can be obtained:

$$\begin{aligned} u_0 &= 0 \\ u_i &= u_{i-1} + S_i \end{aligned} \tag{2.21}$$

This spacing function is very important for the interpolation. Above all when the points lay in a big gradient region, such as the trailing edge. Then, in that area, we can set a smaller spacing, so that the points are closer to the curve and the interpolation yields better results. Therefore, a linear distribution is set for the beginning and the end of the sequence, and an uniform spacing is set for the intermediate points, such as:

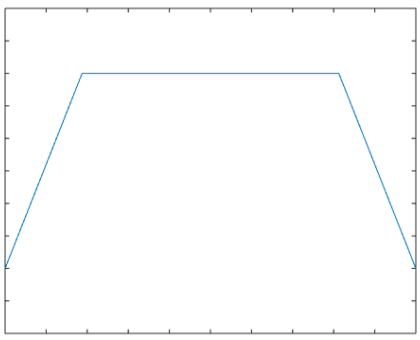


Figure 2.9: Example of a spacing distribution of a knot sequence, observing the restrictions adopted in the optimization tool.

where the uniform part of the spacing sequence goes from K/d to $K \cdot (1 - \frac{1}{d})$. In those cases, d is a number that can be used from two different perspectives. One perspective is making it simpler and defining this value at the beginning of the procedure. Or, as the method adopted in this work, the value of d can be optimized through an iterative method for a given number of control points.

Also, it should not be forgotten that in the end knots the multiplicity is equal to n, therefore the spacing needs to be recalculated with $n - 1$ *zeros* in the beginning and in the end.

NURBS improvement

Although the interpolation algorithm (see Chapter 3) has been designed for the use of B-Splines, it is worth mentioning that for complex geometries, the NURBS method has been employed to improve the interpolation and reduce the error in relation with the original curve. NURBS stands for non-uniform rational basis spline, and its mathematical implementation is very similar to a normal B-Spline, with the exception that we assign a weight to each B-Spline basis in the construction of the curve.

If we consider Eq. 2.10 and assign a weight to each basis, then we need to normalize the sum. Eventually we get:

$$x(u) = \frac{\sum_{j=0}^{L} d_j \omega_j N_j^n(u)}{\sum_{j=0}^{L} \omega_j N_j^n(u)} \tag{2.22}$$

To implement the interpolation code, we have made an evaluation with slightly different weights to generate an interpolated curve whose error has been reduced.

2.1.3 Automated Algorithm

One automated scheme has been developed to compute the B-Spline from given control points (Fig. 2.10), and another code has been developed to obtain the control points from a given curve (Fig. 2.11). The two algorithms use the theory described in sections 2.1.1 and 2.1.2 respectively.

Algorithm 1 (Fig. 2.10) has been designed using the blossoming expression of the B-Spline points. Each initial point is presented as a set of knots from the knots sequence (taking n knots), and we compute each new intermediate point using the de Casteljau Algorithm from Eq. 2.5. Eventually we find the points that belong to the B-Spline curve.

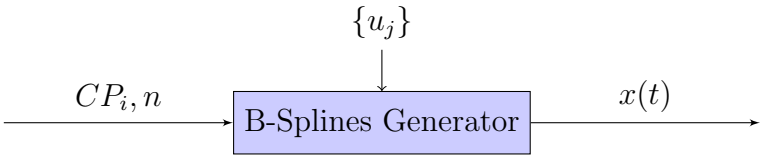


Figure 2.10: Algorithm 1 generates a B-Spline curve from given control points.

Similarly, Algorithm 2 (2.11) makes use of the B-Spline functions that have been discussed in section 2.1.2, along with a least squares approximation.

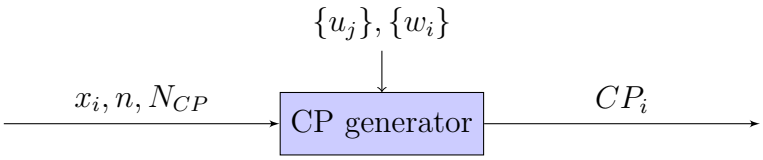


Figure 2.11: Algorithm 2 generates control points by interpolating the data points defining the original blade shape.

A third algorithm has been developed to compute the error of the interpolation method. If we assign to the points on the interpolated curve x_i the same data parameter w_i as to the given points p_i, then we can calculate the mean square error as follows:

$$MSE = \frac{\sum_{i=0}^{P} \|p(w_i) - x(w_i)\|^2}{P+1} \tag{2.23}$$

At this point, a tolerance for the interpolation can be set, and an algorithm that computes recursively the interpolating curve can be built, increasing the number of control points until it meets our tolerance criterion.

This third algorithm can be defined to improve 2.11. If we use the concept of tolerance and error described by Eq. 2.23 we can couple it with Fig. 2.10 to generate the curve and to check if it meets the required tolerance.

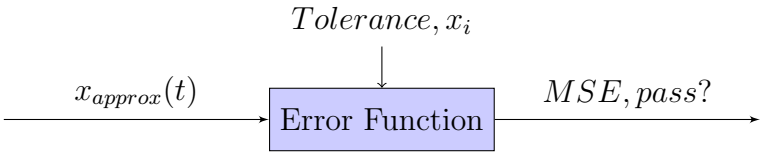


Figure 2.12: Algorithm 3 to compute the error.

The use of the algorithms described in this section will be discussed and validated in Chapter 4.

Big gradient region

The trailing edge of a blade is an area with a strong gradient. This area shall be separated when computing the B-Spline. In order to do that, the gradient in each point[5] is computed, so that the area with the strongest gradient can be separated. Once that happens, there are two possibilities:

1. To compute that region in the same way as the rest of the curve is computed. That means that a number of control points need to be interpolated using the algorithms described previously.

2. To consider that region as a common radius, which is used to connect the suction side with the pressure side of the blade.

There needs to be some discussion on both methods prior to their validation. On the one hand, the method of the common radius has many advantages, as it reduces the number of design variables by limiting the interpolation to the other region. This leads to an improvement of the computation speed. Moreover, it avoids the error from the connection of both parts. This will lead to a better interpolation and less mean square error. Furthermore, optimization is performed more accurately, since we can still play with the parameters defining the common radius, such as equal tangents on one side of the blade. On the other hand, this method does not provide much accuracy in the shape of the trailing edge. In the future, this automated optimization tool might be used for another blade shape whose trailing edge is not a common radius. Also, the relative position and size of the trailing edge might change during the optimization process when moving the first and last control points of the whole blade. Both methods have been tested out, and the final decision will be explained in Chapter 4.

[5]Discretized as the difference between the point and its neighbor.

2.2 CFD Solver

Fluid mechanics equations are solved for only a limited number of simplified flows that cannot be used for engineering and design purposes. Traditionally, the common approach is to perform a simplification of the equations, based on approximations and dimensionality, and the acquisition of parameters from experimental data. However, when the problem comprises a large number of dimensionless parameters, or when the experiments to be set are too complex, this approach is not enough. With the evolution of computer science, CFD techniques were developed to fill that gap. Computational Fluid Dynamics is a broad field that is widely used in engineering for design, analysis, and research [8]. CFD enables to simulate and understand fluids without performing experiments, and measuring flow variables at desired locations. However, CFD methods need to be considered with prudence, since solutions are always approximate and they do contain approximation errors:

- Modeling errors: approximations in the model, due to idealizations of the differential equations.
- Discretization errors: approximations in the discretization and in the solution methods.
- Convergence errors: approximations due to the iterative process.

In CFD there are many solution methods. We should take into account that a solution method needs to be: (a) Consistent - the numerical scheme must tend to the differential equation when the space and time discretization tend to zero; (b) Stable - the round-off error due to the finite algebra must remain bounded iteration after iteration; and (c) Convergent - the numerical solution must tend to the exact solution when the space and time discretization tend to zero. For sufficiently small grids, the rate of convergence depends on the order of the truncation error. Before performing a CFD simulation, one should define the time and length scales in which the flow system is modeled. The solution methods most widely used can be summarized in three categories:

- Finite Difference Method (FD)
- Finite Volume Method (FVM)
- Finite Element Method (FEM)

In particular, for the CFD techniques used in this study, the **Finite Volumes Method** uses the integral forms of the conservation equations,

and the solution domain is divided into small control volumes (CVs). Surface integrals for each CV are considered as the sum of integrals over the different CV faces. The value of each of them can be approximated either in terms of the value of the variable in different locations on the face, or in terms of the nodal values. Volume integrals are computed directly with the nodal values or with a higher-order approximation, taking into account more locations than just the center of the CV. In addition, boundary conditions need to be treated separately. Their values should be known (variable value, flux, or other conditions) and, since there are no nodes outside the boundary, sometimes we need to use one-sided interpolation or extrapolation to retrieve the value at the boundary. Moreover, the selection of a well-defined grid is essential in every CFD problem. The locations of the different evaluations along the process are defined by the numeric grid of the problem. We shall go from a continuous media (real world) to a discrete distribution of nodes (computation).

ANSYS-CFD package

ANSYS-CFD is a general fluid dynamics tool developed by *ANSYS*, a well-established engineering software developer that has been providing services to manage simulation processes and data for more than 40 years.

The package used in this study includes the tools for the different simulation stages, such as *TurboGrid* (geometry and mesh generator), *CFX-Pre*, *CFX-Solver* (preprocessor and solver respectively), and *CFD-Post* (post-processor).

Specifically, ANSYS-CFX [9,10] is a well-validated and widely used CFD analysis tool that provides efficient parallel calculation from different processing cores. The code is wrapped in an intuitive GUI user environment. In the following sections, the ANSYS-CFX Solver theory will be summarized. For more information about the physics behind the equations, please refer to fluid-dynamics books, such as [11].

Figure 2.13: ANSYS logo.

2.2.1 Governing Equations

In this section, we describe the general equations that are used to simulate the flow in the cases studied. For all fluid mechanics problems, the mass

(Eq. 2.24), momentum (Eq. 2.25), and total energy (Eq. 2.26) conservation equations must be obeyed. They are also called Navier-Stokes equations in their conservation form.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0 \tag{2.24}$$

$$\frac{\partial (\rho \mathbf{U})}{\partial t} + \nabla \cdot (\rho \mathbf{U} \times \mathbf{U}) = -\nabla p + \nabla \cdot \tau + \mathbf{S_M} \tag{2.25}$$

$$\frac{\partial \rho h_T}{\partial t} - \frac{\partial p}{\partial t} + \nabla \cdot (\rho \mathbf{U} h_T) = \nabla \cdot (\lambda \nabla T) + \nabla \cdot (\mathbf{U} \cdot \tau) + \mathbf{U} \cdot \mathbf{S_M} + \mathbf{S_E} \tag{2.26}$$

where ρ is the density, $\mathbf{U}$ is the velocity vector, $\mathbf{S}$ is the source vector (M stands for momentum and E for energy), λ is the thermal conductivity, p is the static pressure and T is the temperature. τ and h_T are the shear stress and the specific total enthalpy respectively, and are defined by:

$$\tau = \mu \left(\nabla \mathbf{U} + (\nabla \mathbf{U})^T - \frac{2}{3} \delta \nabla \cdot \mathbf{U} \right) \tag{2.27}$$

$$h_T = h + \frac{1}{2} U^2 \tag{2.28}$$

where μ is the dynamic viscosity and h is the specific enthalpy.

This set of equations represents the basic pillars of fluid mechanics and they need to be coupled with constitutive equations of state, that relate ρ and h with p and T. In the most general case, this can be written as[6]:

$$\begin{aligned} &\rho = \rho(p, T) \\ &dh = \frac{\partial h}{\partial T}|_p dT + \frac{\partial h}{\partial p}|_T dp = c_p dT + \left(v - T \frac{\partial v}{\partial T}|_p \right) dp \\ &c_p = c_p(p, T) \end{aligned} \tag{2.29}$$

where v is the specific volume and c_p is the specific heat capacity.

A wide range of constitutive equations can be found in any CFD solver. However, in this study, a real gas approach, with a look-up table (LuT) is employed, and therefore no further derivation is needed in this regard.

2.2.2 Turbulence Model

In most engineering applications and, above all, in the cases studied in this work, turbulence plays a very important role. Turbulence is a very complex flow feature that needs a model to be treated computationally.

[6]Remember that $v = 1/\rho$

In principle, the Navier-Stokes equations should describe turbulent flows. However, real cases involve length scales much smaller than the smallest finite volume mesh. For this reason, models need to be specifically developed to account for the effects of turbulence without recourse of a prohibitively fine mesh. There are several ways to handle it:

1. RANS-based models
2. Large Eddy simulations
3. Detached Eddy simulations
4. Hybrid models
5. Direct Numerical simulations

In general, turbulence models seek to modify the original Navier-Stokes equations by the introduction of averaged and fluctuating quantities and therefore generating the **Reynolds Averaged Navier-Stokes equations** (RANS). In this study, this type of model is used to treat turbulence. Simulations using RANS greatly reduce the computational effort. However, additional unknown terms containing products of fluctuating quantities appear. These quantities can be considered new stresses and they are commonly known as turbulent or Reynolds' stresses.

A variable, for example a component of the velocity vector U_i, is divided into an average component $\overline{U}_i$ and a time varying component u_i:

$$U_i = \overline{U}_i + u_i \tag{2.30}$$

The governing equations in this case turn out:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho U_j\right) = 0 \tag{2.31}$$

$$\frac{\partial \rho U_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho U_i U_j\right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\tau_{ij} - \rho\overline{u_i u_j}\right) + S_M \tag{2.32}$$

$$\frac{\rho h_T}{\partial t} - \frac{\partial p}{\partial t} - \frac{\partial}{\partial x_j}\left(\rho U_j h_T\right) = \frac{\partial}{\partial x_j}\left(\lambda \frac{\partial T}{\partial x_j} - \rho\overline{u_j h}\right) + \frac{\partial}{\partial x_j}\left(U_i\left(\tau_{ij} - \rho\overline{u_i u_j}\right)\right) + S_E \tag{2.33}$$

where:

$$h_T = h + \frac{1}{2}U_i U_i + k = h + \frac{1}{2}U_i U_i + \frac{1}{2}\overline{u_i^2} \tag{2.34}$$

Turbulence models complete the RANS equations by providing an expression to compute the Reynolds stresses and fluxes, as they provide a closure to the turbulence problem. In fact, there are three RANS-based turbulence models:

1. Linear Eddy Viscosity Model
2. Nonlinear Eddy Viscosity Model
3. Reynolds Stress Model

In this study, a **Linear Eddy Viscosity Model** is used. In it, the turbulence is considered as small eddies that are forming and dissipating, and in which Reynolds stresses are assumed to be proportional to mean velocity gradients. In this hypothesis, the Reynold stresses and Reynolds fluxes can be related through:

$$-\rho\overline{u_i u_j} = \mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\left(\rho k + \mu_t \frac{\partial U_k}{\partial x_k}\right) \tag{2.35}$$

$$-\rho\overline{u_i \phi} = \gamma_t \frac{\partial \phi}{\partial x_i} \tag{2.36}$$

where ϕ is a general scalar variable. μ_t and γ_t are the eddy viscosity (sometimes called turbulent viscosity) and eddy diffusivity, and are related by the turbulent Prandtl number ($\gamma_t = \frac{\mu_t}{Pr_t}$)

Among the linear eddy viscosity models, the κ-ω **model** is one of the most used, as it provides two extra transport equations to represent the turbulence of the flow. It then allows to account for the convection and diffusion of turbulent energy (history effects) of the fluid. κ stands for the first transported variable, the kinetic energy (turbulence energy). On the other hand, ω represents the specific dissipation and it determines the scale of the turbulence.

In this model, the turbulence viscosity is related with the turbulence kinetic energy and turbulent frequency as follows:

$$\mu_t = \rho \frac{\kappa}{\omega} \tag{2.37}$$

In this model, the parameters κ and ω are given by the following transport equations:

$$\frac{\partial \kappa}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho U_j \kappa\right) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\kappa}\right)\frac{\partial \kappa}{\partial x_j}\right] + P_\kappa - \beta' \rho \kappa \omega + P_{\kappa b} \tag{2.38}$$

$$\frac{\partial \omega}{\partial t} + \frac{\partial}{\partial x_j}(\rho U_j \omega) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\omega}\right)\frac{\partial \omega}{\partial x_j}\right] + \alpha\frac{\omega}{\kappa}P_\kappa - \beta\rho\omega^2 + P_{\omega b} \quad (2.39)$$

where β', β, α, σ_κ, σ_ω are model constants. $P_{\omega b}$ and $P_{\kappa b}$ are the additional buoyancy terms.

The direct use of this model commonly leads to an over-prediction of the eddy-viscosity. To account for that, the **Shear Stress Transport** (SST) model is employed. In this model, the proper behavior can be obtained by a limiter to the formulation of the eddy-viscosity:

$$\nu_\tau = \frac{a_1 \kappa}{max(a_1\omega, S \cdot F} \quad (2.40)$$

where $\nu_\tau = \frac{\mu_\tau}{\rho}$,$F$ is a blending function, and S is an invariant measure of the strain rate.

2.2.3 Flow Near the Wall

It is important to mention the way in which the flow is modeled near to a no-slip wall. Actually, in the cases that are considered in this study, the boundary layer-turbulence interaction will be of primary importance. As an extension of the method of *Launder and Spalding*, the near wall velocity is given by:

$$u^+ = \frac{1}{\kappa}\ln(y^+) + C \quad (2.41)$$

where u^+ and y^+ are, respectively, the near wall velocity and the dimensionless distance from the wall and can be expressed as:

$$u^+ = \frac{U_\tau}{u_\tau} = \frac{U_\tau}{\left(\frac{\tau_\omega}{\rho}\right)^{1/2}} \quad (2.42)$$

$$y^+ = \frac{\rho \Delta y u_\tau}{\mu} \quad (2.43)$$

where u_τ is the friction velocity, U_τ is the known velocity tangent to the wall at a distance of Δy, τ_ω is the wall shear stress, κ is the Karman constant and C is a log-layer constant (function of the wall roughness).

An automatic wall treatment, which is based on $\Delta y = \Delta n$, will be employed. The specification of a small number of mesh elements next to the wall (Δn) is crucial to produce accurate results, as it maintains a low y^+. As a rule of thumb, a y^+ value around 1 or less than 1 is recommended for highly accurate simulations, like heat transfer predictions.

2.2.4 Solution Theory

As it was described at the beginning of this section, analytical solutions to the Navier-Stokes equations are only available for very simple flows and conditions. A numerical approach is adopted for the rest of cases and the exact Navier-Stokes equations are replaced by algebraic approximations that can be solved using a numerical method.

As it was mentioned, the finite volume methodology is used in this work, and therefore, the conservation equations for mass, momentum and energy[7] (Eq. 2.24, Eq. 2.25, and Eq. 2.26) are integrated over each control volume. The divergence theorem is applied to convert volume integrals -with divergence and gradient operators- to surface integrals.

Once this formulation is correctly described, the volume integrals and surface integrals are discretized within each element sector and at integration points of each surface, respectively.

For its part, the **Convection Term** (sometimes referred as Advection Term) requires approximated values of the variable (ϕ in general form) at points other than the nodes. Depending on the scheme used, the robustness, computational load and accuracy will vary. The general formulation is:

$$\phi_{ip} = \phi_{up} + \beta \nabla\phi \cdot \Delta\overline{r} \tag{2.44}$$

where ϕ_{ip}, ϕ_{up} are the values at the integration point and the upwind node respectively, and $\overline{r}$ is the vector from up to ip.

The selection of the values of β and $\nabla\phi$ yields different schemes (1st Order Upwind, Central Difference Scheme and High Resolution Scheme).

On the other hand, the **Diffusion Terms** formulation follows the standard finite-element approach, using shape functions to evaluate spatial derivatives. The linear set of equations that arise by applying the finite volume approach can be written in the form:

$$\sum_{nbi} a_i^{nb} \phi_i^{nb} = b_i \tag{2.45}$$

where i identifies the control volume or node, and nb, the neighbor.

Instead of a segregated approach (that solves in a "guess-and-correct" nature[8]), ANSYS CFX uses a coupled solver, which solves for u, v, w, p as a single system. In this approach, fully implicit discretization at any given time step is employed. For steady-state problems, a time-step scheme is equally used to guide the solution to a steady state.

[7] Or any other scalar quantity to which the transport equation is applied.

[8] In this approach, the momentum equations are solved with a guessed pressure. Then, a correction equation for the pressure is obtained.

In addition, ANSYS CFX uses a Multigrid accelerated Inomplete Lower Upper (MG ILU) factorization technique for solving the discrete system of linearized equations.

2.3 Genetic Algorithms

For the last decades, the interest in single and multi-objective optimization has grown dramatically. A wide range of heuristic methods has been developed as a result. These techniques are designed to solve problems quickly, normally at the expense of accuracy. An important application of heuristic methods is the design of Evolutionary Algorithms (EAs) [3], among others. These have become very interesting for a quite large number of applications, due to many advantages that make them outperform other optimization methods:

1. Possibility of dealing with oscillating or smooth-less objective functions.
2. Identification of a number of possible solutions.
3. Capability to deal in an easy manner with multi-objective optimization.
4. Capability to consider a wide range of configurations.
5. They can be used according to non-intrusive methods.
6. They can be very general and easy to adapt to different problems, as well as different evaluating functions and optimization strategies.

Furthermore, in contrast to local optimization methods, evolutionary algorithms are global optimization methods. Hence, EAs are best suited to optimization problems with multiple local optima, and where gradient methods are too computationally expensive or not readily available. In many optimization problems, EAs quickly identify promising regions of the design space where the global optimum might be located. Therefore, hybrid optimization strategies can be used. Those combine the global search capabilities of EAs with gradient-based algorithms that efficiently perform a local search.

Genetic Algorithms (GAs) are evolutionary algorithms that do not consider any gradient information. This fact makes them capable of moving out of a local minimum. A gradient-based method usually remains within an interval near a local suboptimal solution, and a well-designed GA can likely reach the true optimum.

Genetic Algorithms base their operation on the natural selection and evolution described by the neo-Darwinian theory. It is very interesting to understand how a genetic algorithm imitates nature to perform the optimization. In Fig. 2.14 one can understand how a genetic algorithm interprets a member of the solutions population.

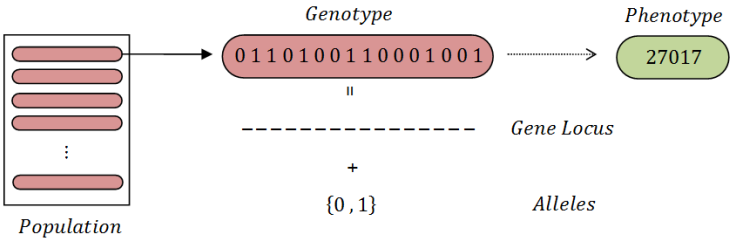


Figure 2.14: Formulation of a GA.

An initial database containing a given number of random solutions represents the initial population of living beings. Each of the members of the population has its own and unique features (phenotype). In optimization language, each member of the database has a mathematical meaning, e.g. a numerical value. This phenotype is codified into a unique genotype, which corresponds to a binary representation of the number[9]. The genotype comprises a set of genes that can take different values, as it occurs in the binary codification.

As it can be seen in Fig. 2.15, once each member is represented as a string of alleles, the evolution process takes place. First, two members of the population chosen at random (or with a dependency on a fitness function) reproduce and give birth to a certain number of children. As it occurs in nature, the offspring will have features from both parents. In order to do so mathematically, the binary strings are cut off and combined. This assures that offspring is formed from parenthood passing on certain genes[10]. Later, each member of the population may suffer random mutation in their genes, to maintain diversity. This helps explore different areas of the design space and search for the global optimum. Finally, the members of the population are confronted and the selection of the best individuals takes place. Only those that comply with an specified criteria will pass to the next generation. This process is then repeated until the convergence criterion is met.

2.3.1 JEGA Library

JEGA (Java Engine for Genetic Algorithms) is a framework that offers mechanisms to work with genetic algorithms. JEGA provides a flexible and exten-

[9]One can come up with a different codification, but binary numbers are commonly used to represent the genotype in genetic algorithms.

[10]Actually, the reality is another. Each member of the population is formed by the genotype of n design variables, and a genetic algorithm can be set to either produce different combinations of design variables or change the variables themselves.

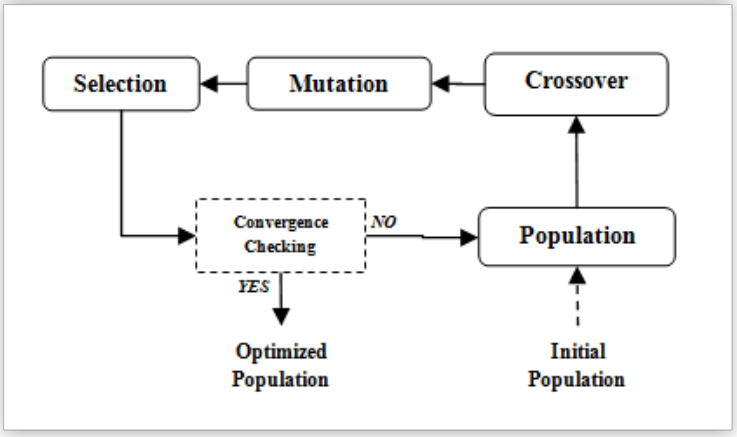


Figure 2.15: GA Process.

sible problem-solving environment for computational models. The database was written by John Eddy [12] and it contains two main genetic algorithms: SOGA and MOGA, for single and multiple-objective problems respectively.

Each design variable is represented, when required, as a single 32 bits signed integer. This saves a great deal of computational time. It is worth mentioning that in those cases where the genetic algorithm works with this representation, the code multiplies the real number by 10^6 and then truncates. This only allows a precision of six decimal places. The initial population is created from random numbers within the design variable bounds.

A check of duplicity is performed to assured uniqueness among members, to encourage diversity. This "clone testing" is carried out always after creating a new member. In this way we avoid wasting both computation time and memory, and clustering within a suboptimal solution. The fitness function is assigned to each set of objective functions, applying the L_2 norm.

When crossover takes place, a hierarchic distribution according the fitness is established (merit function). To assure that the best individuals are more likely to reproduce, they appear more times in the mating pool. Each chosen mating pair produce a given number of children, by randomly selecting a crossover point in their chromosomes. For mutation, children are randomly selected and different methods can be used. A bit can be negated or the value of the design variable can be mutated to a value inside the bounds. The selection and children insertion into the population is given by the Pareto dominance or their fitness, depending on the selection method used.

The different methods and parameters of the genetic algorithm will be chosen according to the specific application and thus will be commented in Chapter 4. However, some of the possibilities are worth mentioning.

For the initialization type we have:

- Simple Random. It creates design with random variable values according to a uniform distribution.
- Unique Random. It behaves as Simple Random, but it does not allow repetition.
- Specified population.

For the crossover type we have:

- Multipoint Binary. It performs switching chromosomes of parents at N crossover points, over any design variable.
- Multipoint Parameterized Binary. It performs switching chromosomes of parents at N crossover points, but in each design variable individually.
- Multipoint Real. This is a special case, since it does not convert the number (design variable) into binary, but it computes the crossover using the real value.
- Shuffle Random. It creates a random shuffle of the parent design variable values.

For the mutation type we have:

- Replace Uniform. It chooses a random valid value for a randomly chosen variable.
- Bit Random. It just convert 0 into 1 and viceversa in the binary coding of a randomly chosen variable of a randomly chosen design.
- Offset Normal. Introduces a random offset from a Gaussian distribution of mean of zero and an specified standard deviation.
- Offset Cauchy. Similar, but with a Cauchy random variable.
- Offset Uniform. Uniformly distributed random offset.

For the replacement type we have:

- Roulette-Wheel. Each design is assigned a portion of the wheel proportional to it fitness relative to the others. Portions of the wheel are chosen at random. A design can be selected a number of times (regular Roulette-Wheel) or only once (Unique Roulette-Wheel).

- Below-Limit. Keep all designs for which the fitness is below a certain limit. For this type, also a shrinkage percentage is specified to assure that a given population size is maintained (for diversity issues).

- Elitist. This method simply chooses a given number of designs, taking the most fit.

- Favor Feasible. This method first checks for feasibility (when constraints are specified). If no winner appears, then it computes the fitness.

For the convergence type we have:

- Average Fitness Tracker. Keep track of the average fitness of the population and assess if it does not change in a given percent over a given number of generations.

- Best Fitness Tracker. It works in a similar manner, but it tracks the best fitness of the population instead of the average.

To have a further understanding, some of the parameters will be studied for an optimization example (see Appendix A.1).

2.4 Surrogate-Based Optimization

The optimization performed in this tool needs to be limited in time. CFD codes are excellent tools but the time required for each evaluation is not desirable. A regular evaluation can easily take from minutes for simple problems to many hours or even days for complex applications. Furthermore, a genetic algorithm requires a large amount of objective function evaluations, ranging from tens to hundreds generations, with dozens of individuals in each one. This would be too taxing and a new method needed to be developed.

The optimization tool developed in this study is the perfect example of the use of surrogate models [13, 14]. As a summary, a surrogate model (also known as meta-model or response surface) is a function that relates design variables with performance (objective function) in an approximate way, to reduce the overall computational cost of the optimization process. A mathematical representation is selected for the objective function with no relation with the physical phenomena of the real problem. The parameters of this function are updated within the optimization process, as it evolves (surrogate training). An initial database is sometimes required but the computational cost is considerably reduced.

Surrogate models theory is very extensive and many schemes have been developed. In this study, the Kriging model is used as the mathematical approximate objective function or non-linear constraint, Latin Hypercube Sampling (LHS) technique is employed, and different optimization approaches are considered and tested for the betterment of the shape-optimization tool.

2.4.1 Surrogate Model

In this study, the Kriging statistical model will be employed. Kriging [15] is a set of interpolation methods, sometimes called Gaussian Processes, that were initially developed for geostatistics problems and nowadays they are widely used in many fields. The most common form of a Kriging model is as follows:

$$\tilde{f}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x})^T \boldsymbol{\beta} + \boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{R}^{-1} (\boldsymbol{f} - \boldsymbol{G}\boldsymbol{\beta}) \tag{2.46}$$

where $\boldsymbol{x}$ is the current point vector, composed by n design variables; $\boldsymbol{g(x)}$ is the vector of evaluations of the trend basis functions; $\boldsymbol{\beta}$ is the vector of the least squares estimates of the trend basis function coefficients; $\boldsymbol{r(x)}$ is the correlation vector with the data points; $\boldsymbol{R}$ is the correlation matrix for all of the data points; $\boldsymbol{f}$ is the response values vector; and $\boldsymbol{G}$ is the matrix of the trend basis functions evaluations at all data points.

The terms in the correlation vector ($\boldsymbol{r(x)}$) and correlation matrix ($\boldsymbol{R}$)

are computed using a Gaussian correlation function and depend on a vector $\boldsymbol{\theta}$ of correlation parameters, which are computed using a MLE procedure[11].

The Kriging model is guaranteed to pass through all the response data values, and the predicted response values $\tilde{f}(\boldsymbol{x})$ decay to the trend function $\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{\beta}$ when the point $\boldsymbol{x}$ is far from any of the data points. The mathematical model of Kriging can be understood as the linear combination of a known function of the variables and the realization of a stochastic process:

$$\tilde{f}(x) = y(x) + Z(x) \tag{2.47}$$

where $\tilde{f}(x)$ is the unknown deterministic response, $y(x)$ is the known function of x and, $Z(x)$ is a realization of a stochastic process with zero mean and non-zero covariance. For more details about the mathematics of Kriging, please refer to [14, 16]

Approximation Type

To determine the Kriging parameters, the Surfpack library, developed by [17], is employed. In this case, it uses a global search method called *DiRect* algorithm (dividing rectangles), a derivative free global optimization method that balances global search in promising regions with global search in unexplored regions. For ill-conditioning cases, it selects points that meets a constraint on the condition number and that provides useful information for the correlation lengths. The latter are obtained via maximum likelihood estimate. Then, the set of points used to build the model is the one associated with the most likely set of correlation lengths.

However, some other characteristics of the model should be taken into account. There exist different methods to determine correlation parameters (those governing the mean and correlation functions), apart from the DiRect method. The Surfpack library provides three more approximation types: Loca, Sampling and None. The *Local* optimization method makes use of the *ConMin* public library (constrained function minimization) of nonlinear programming optimizers. *Sampling* stands for a method in which several random guesses are generated and the one with greatest likelihood is picked. *None* stands for a method with no optimization, as it just picks the center of the feasible region. On the other hand, the trend function can be built in different ways: *constant*, *linear*, *quadratic*, and *reduced quadratic*. A *reduced quadratic* trend function is such that includes the main effects, but

[11]The Maximum Likelihood Estimation (MLE) is a method to find the value of the parameters of an statistical model which assigns to the known sample distribution the maximum likelihood in the model.

not mixed/interaction terms. In this study, a reduced quadratic expression is employed.

Sampling Plan

As it has been mentioned before, Latin Hypercube Sampling (LHS) is employed as the sampling strategy. LHS method divides uniformly the design space for each factor and combines randomly these levels to specify the final sampling points.

Among several sampling plans, LHS has been favored by many authors [14], since it efficiently samples large design spaces and provide sets of points whose projections onto each variable axis are uniform.

2.4.2 Surrogate-Based Local Framework

In a surrogate-based local optimization (SBLO or SBLM), also called Trust Region or Move Limits optimization technique, the optimization algorithm operates directly on a surrogate model, which can be generated through different methods. Since the surrogate model has a limited range of accuracy, this algorithm checks the goodness of the approximation by comparing with the high-fidelity expensive tool and updating its parameters. The main feature of the local optimization is the use of a trust region approach, which defines de extent of the approximation.

SBLO method needs to generate and update the data fit in each trust region, performing high-fidelity evaluations over a design of experiments. Although it is a local approach, each sampling in the trust region can be performed globally, which allows to extract the relevant global design trends. The global data fit (using Kriging model, for instance) can manage poorly-behave response variations, and it generates smooth, differentiable surrogates. SBLO methods are not intended for smooth continuous optimization problems, since direct gradient-based optimization can be more efficient for such applications. On the other hand, SBLO with global data fits is better for most engineering problems, where response quantities may be discontinuous, non-smooth, or with multiple local optima.

If we formulate the optimization problem as follows:

$$\min_{x_l \leq x \leq x_u} f(x) \tag{2.48}$$

The surrogate-based problem can be described:

$$\min_{\|x - x_c^k\|_\infty \leq \triangle^k} \tilde{f}^k(x) \tag{2.49}$$

where $\tilde{f}^k(x)$ is the surrogate objective function, k refers to the sequence of approximate optimization subproblems subject to a trust region constrain $\triangle^k$. This surrogate-based approach based on the objective function (direct surrogate approach) is attractive both from its simplicity and potential for improved performance. After each of the k iterations, the predicted value is assessed by computing the trust region ratio ρ^k, which can be defined as follows:

$$\rho^k = \frac{f(x_c^k) - f(x_{opt}^k)}{\tilde{f}(x_c^k) - \tilde{f}(x_{opt}^k)} \tag{2.50}$$

where x_c is the center of the trust region (which was the optimum of the previous step) and x_{opt}^k is the optimum in the current step. Therefore, ρ^k is a measure of the ratio between the improvement in the truth model and the improvement in the surrogate model.

Once ρ^k is computed, this value is used to define the step acceptance and trust region size of the next iteration $\triangle^{k+1}$. These are defined by the contraction and expansion thresholds and factors. If the typical values of 0.75 and 0.25 are applied for the expansion and contraction thresholds respectively, the trust region ratio logic becomes as in Table 2.1

TR Ratio	Accuracy	Acceptance	next TR Size
$\rho^k \leq 0$	Poor	Reject	Shrink
$0 \leq \rho^k \leq 0.25$	Marginal	Accept	Shrink
$0.25 \leq \rho^k \leq 0.75$	Moderate	Accept	Retain
$0.75 \leq \rho^k \leq 1.25$	Good	Accept	Expand
$1.25 \leq \rho^k$	Moderate	Accept	Retain

Table 2.1: Trust Region Ratio Logic

A negative value of ρ^k always means a poor accuracy because an improvement in the truth model means an impairment in the surrogate model or vice-versa.

In this way, the calculated step is accepted or rejected based on the trust region ratio. In our case, also a filter is applied. As still we are not working with constrains, the only filter applied is the concept of Pareto optimality to the objective function and only accept the step if it is better than any previous step.

SBLO might be thought to be guaranteed to find local instead of global optima. However, this method can often find global minimum, due to the random sampling used in each trust region, which solves the problem a little differently each time.

For SBLO, the initial and maximum size of the trust region should be considered carefully, as well as the contraction and expansion factors. Contraction and expansion thresholds are also indicated. The stopping criteria is double. On the one hand, a maximum number of SBLO iterations is set, but also a soft convergence criterion can be used to stop the process. This criterion limits the number of consecutive iterations with improvement less than the convergence tolerance (the tolerance value and the iterations number must be specified).

The process of a SBLO can be summarized in the following steps:

1. Generates trust region (upper, lower bounds of design variables) in the design space.

2. Performs high-fidelity evaluations over the specified trust region and generates Kriging parameters. The number is determined by the sample size and it has a minimum defined by the number of design variables[12]

3. Performs optimization of the Kriging mathematical model.

4. Evaluates the optimum with the high-fidelity tool.

5. Generates new trust region by setting the previous optimum as the center if the step is accepted or recalculating the step shrinking the trust region.

6. Begin process until a maximum number of iterations or the soft convergence criteria is met.

The goodness of this strategy and some of the parameters will be studied for an optimization example (see Appendix A.2). For its implementation, please refer to Section 3.1.

2.4.3 Surrogate-Based Global Framework

In this work, the influence of different surrogate models will be studied. There are some cases in which a global approach needs to be taken into account. For example, for future applications to mutliple-objective optimization.

In a surrogate-based global optimization (SBGO), the optimization algorithm is not supported by a trust-region approach. It starts from an initial sample of points and the optimizer operates on that surrogate. This approach should be used carefully, as there is no guarantee of convergence. It

[12]As a rule of thumb, a minimum of 5 times the number of design variables normally holds.

should be used either when there exists the need of using an initial database or when the surrogate needs, somehow, to be updated globally.

This global approach was initially designed for multiobjective genetic algorithms. Instead of creating one set of surrogates for the individual objectives, the idea would be to select points along the Pareto frontier, which will be then used to supplement the existing set of points that was used to construct the model. The surrogate becomes more accurate as the iterations progress.

The process of a SBGO can be summarized in the following steps:

1. Generates initial random sample, using a design of experiments method.
2. Evaluates all members of the sample with high-fidelity tool and construct mathematical model.
3. Obtains the optimum of the mathematical model.
4. Evaluates optimum with high-fidelity tool and adds it to the sample.
5. Updates mathematical model (recalculates parameters) with the new sample.
6. Repeats the process as many times as specified.

The goodness of this strategy and some of the parameters will be studied for an optimization example (see Appendix A.3). For its implementation, please refer to Section 3.1.

Chapter 3

Implementation

Once defined the methodology used in each module of the optimization, a way to implement them computationally needs to be developed. In the next page, the main scheme of the optimization tool is shown. In the following sections, each different module of the tool is explained in detail.

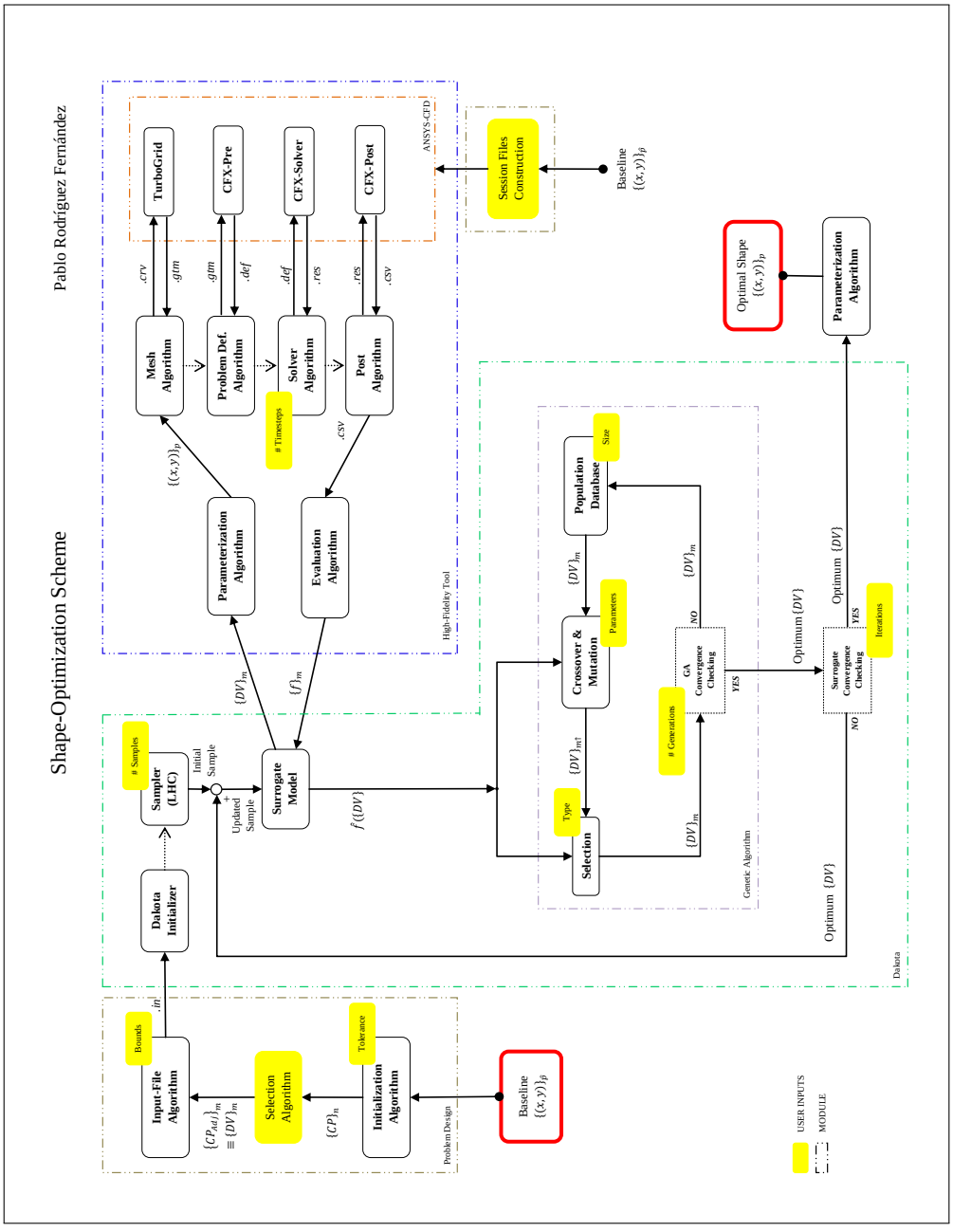
For this study, an environment for the optimization is provided. Dakota is a General Public License (GPL) software developed by Sandia National Laboratories, two major US Department of Energy R&D national laboratories. Dakota Toolkit is defined by the authors as "a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis" [16].

Specifically in this work, Dakota provides the optimization algorithms, i.e. the single-objective and multi-objective genetic algorithms (JEGA library), as well as the surrogate models (Surfpack library) and schemes. Special care must be taken to interconnect Dakota with the high-fidelity CFD codes and the parameterization algorithms.

Figure 3.1: Dakota logo.

3.1 Main Scheme

For surrogate-based **global** optimization, the following approach is applied:



The figure explains in detail the whole process when a global strategy is followed. From left to right, following the arrows, one can understand the evolution of the optimization procedure. The yellow boxes stand for the user

main inputs, necessary for operation. All of them need to be specified before the optimization process is triggered. Most should be provided in the *Input-File Algorithm*, right before Dakota is initialized. However, in the scheme, the yellow boxes are placed in those spots which they provide information to.

All the process starts with the Baseline specification, providing the points of the blade geometry to be optimized. The number of points is arbitrary[1], and they should be provided in a two columns file (two coordinates). Once the *Problem Design* module is provided with this information, the *Initialization Algorithm* can operate as described in Section 3.2.1, providing the control points with which the optimization tool will work. The *Selection Algorithm* (Section 3.2.2) generates the design variables and subsequently the design space in which the optimization will be performed. *Input-File Algorithm* stands for a simple code that generates, from user specifications, the input file that Dakota can interpret. It is at this point when the optimization is set in motion and therefore special care must be taken to avoid unexpected computational errors (Section 3.2.2).

Once Dakota is provided with the required input file, the optimization process is initialized and the sampler based on the Latin Hypercube generates a set of points that will be used by the surrogate scheme to estimate the meta-model parameters. Each member of this initial sample is evaluated using the *High-fidelity Tool* module (Section 3.3). Only the design variables are provided to the module and the only output is the objective function value. This is only applicable when zero-constraints single-objective functions are used. When using the code for multi-objective or when constraints must be placed, the output of the High-fidelity tool (a single file) must contain all the information. For further reading, please refer to [16]. Both input and output will be provided using a single file containing the information according to [16]. Once the surrogate model has a real value for each individual in the sample, the mathematical estimated function is built. From this moment on, the optimization process forgets about the physical content of the problem and focuses only on the optimization of the estimated function.

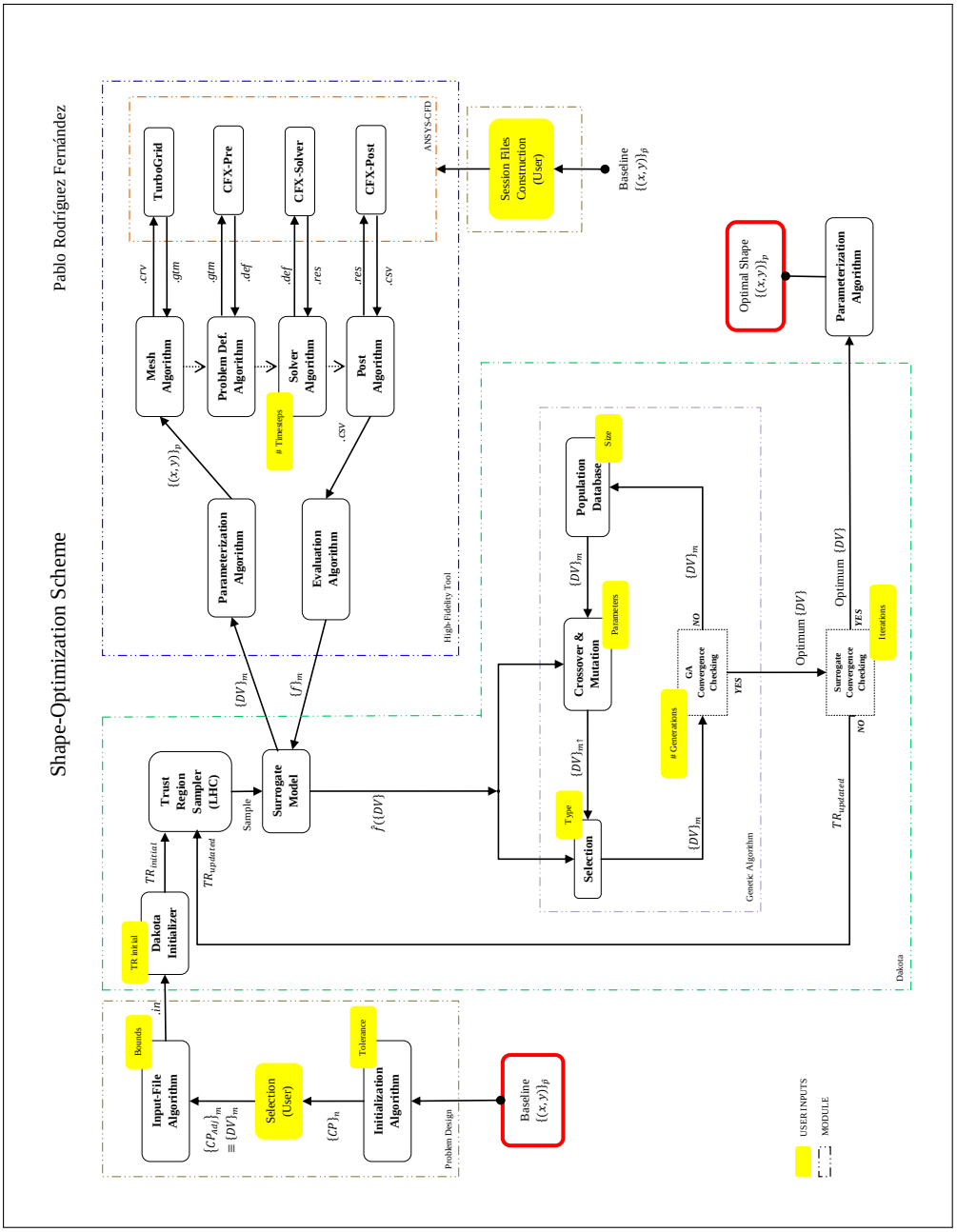
A genetic algorithm, as it was explained in Section 2.3, only needs the value of the objective functions and constraints to work. These values are used to assess the fitness of the individuals in the population, to determine both their probability for crossover (in some evolutionary strategies) and, more importantly, their chances to survive to the next generation. Once convergence has been reached, the best individual is sent to the surrogate model for assessment. Convergence is attained when the specified number

[1] A number ranging in the hundreds is recommended

of generations is reached and when the rate of evolution between the last and the next-to-last is below a certain limit (for some types of evolutionary strategies). The surrogate model will be then checked for convergence after a given number of maximum iterations. If convergence has not been attained yet, the optimum provided by the genetic algorithm is included in the initial population, performing a new high-fidelity evaluation and re-estimating the mathematical surrogate parameters.

If the surrogate model convergence is reached, then the best individual found in the last "surrogate-iteration" is then returned as the optimized value. The parameterization algorithm (Section 2.1) converts the optimized design variables into a geometry that will be considered the result of the optimization.

On the other hand, for surrogate-based **local** optimization, the following approach is applied:



The operation of this scheme is almost the same as for the global optimization. The only difference falls on which individuals are used by the surrogate model to estimate the mathematical parameters. In the previous case, the sample was obtained right from the beginning and it was updated

and trained with a single individual in each "surrogate iteration"[2]. In this new case, as it is described in Section 2.4.2, the region in which the sampler provides the random individuals depends on the current trust region. These "moving limits" are updated in each surrogate iteration and new individuals are evaluated with the high-fidelity tool. In this case, the surrogate convergence assessment module not only checks to terminate the process but it also computes the trust region ratio (Equation 2.50) to determine the trust region size and position in the next iteration.

3.2 Problem Design Algorithms

It should not be forgotten that Dakota environment is designed for purely optimization purposes. An input file containing the design variables and their limits, and defining the high fidelity tool to evaluate the fitness should be provided to Dakota. Therefore, previous to start the optimization process, the desired design space and control points of an interpolated geometry should be defined and included into Dakota's input file. Also, errors may appear during the optimization process that should be prevented. For that reason, a set of problem design algorithms needed to be performed.

3.2.1 Initialization Algorithm

If Algorithms 2.10, 2.11, 2.12 are coupled together, a tool that generates the interpolated control points, evaluates the precision of the interpolation and generates the approximated B-Spline is developed. If the knot sequence is optimized, and the data parameters and degree of the curve are fixed as indicated, a complete algorithm (Fig. 3.2) that computes the required number of control points for a specified blade shape is built.

This algorithm is used at the beginning of the optimization tool to determine the number of required control points and the best knot sequence to optimize a given blade shape.

Once the control points, data parameters, knot sequence and weights are obtained as an output of the previous algorithm, one can wonder if the spacing between control points might be modified for a more precise control of the design variables of the problem. This is, indeed, very useful for the *Selection of Design Variables* (see next section), as it might be valuable to have a more dense distribution of the control points in the fixed part of the blade, as it is not desired a variation of the geometry in that part as a

[2]Actually, it could be updated with more than one individual, for multi-objective optimization above all (Pareto front).

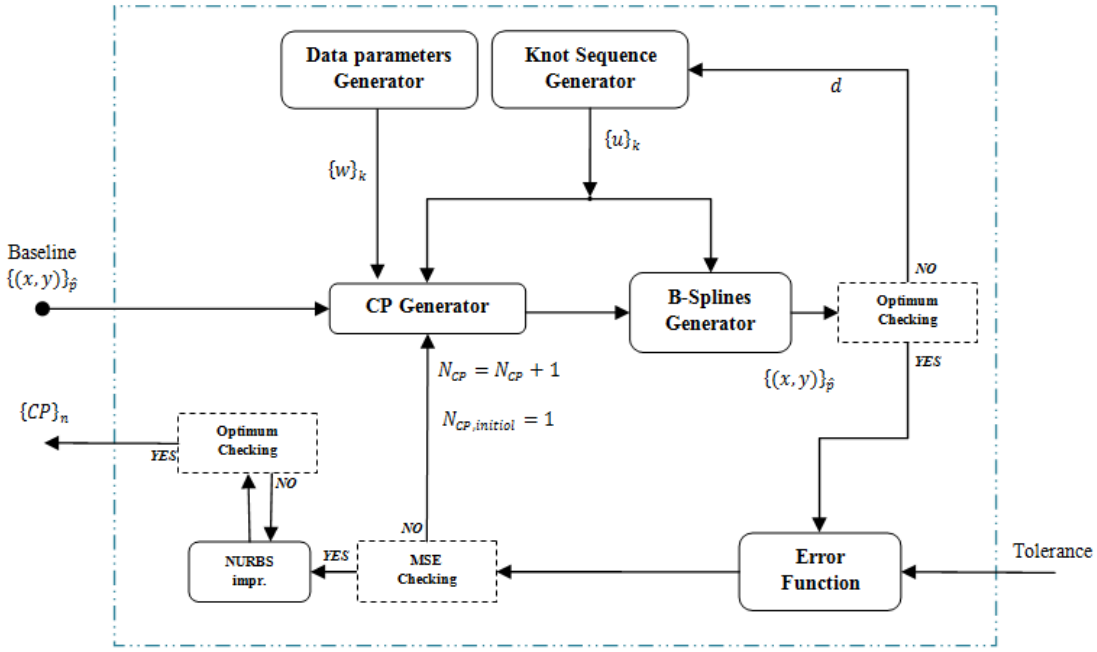


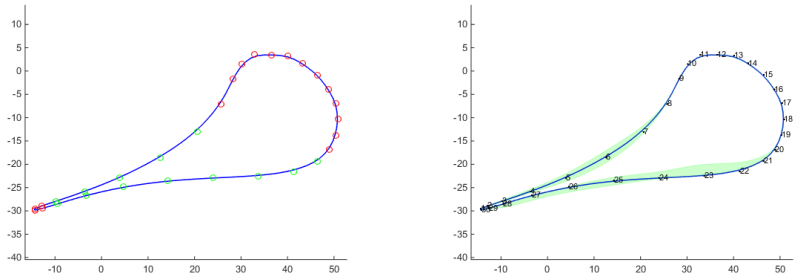
Figure 3.2: Initialization Algorithm.

consequence of the different location of the adjustable control points. In case the user chooses to modify the spacing, the knot sequence obtained via the Algorithm in Fig 3.2 will not be longer used for the interpolation.

3.2.2 Selection of Design Variables

Algorithm 3.2 is very useful to construct the geometry by interpolating for a given tolerance. However, in this problem, the number of design variables must be chosen according to the user criteria. This will be done by observing the optimization results that have been obtained previously [2] and comparing them with the results of the previous initialization algorithm for a large number of control points. We will set then the position of the *Throat* and we will divide the geometry into two parts: fixed, and adjustable. The control points belonging to the fixed part will not be modified and they will be a constant during all the optimization process. The adjustable control points will become the design variables for the optimization algorithm.

As explained in the previous section, the user can modified the spacing of the control points as desired. This can be done graphically through the Selection of Design Variables module.



(a) Points Selection. (b) Design Space Selection.

Figure 3.3: Example of Selection Algorithm results.

Automatic Topology and Meshing

The shape-optimization tool developed in this work needs to have a high level of automation. The mesh definition for each geometry must be defined automatically. The so-called Automatic Topology and Meshing (ATM) from ANSYS Turbogrid is employed. This method enables to create high-quality hexahedral meshes and topologies with no need to adjust manually the control points. It provides with mesh creation tailored specifically to the needs of the given bladed geometry. The ATM Optimized topology is an H-O grid and works well with rounded leading and trailing edges.

Failure Mitigation

This tool is expressively designed to cover a very wide range of blades. As new turbines and blade designs evolve, geometries become more complex and well-rounded, and shape blades are no longer used. Geometries can be very sharp or with sudden turns. The ATM Optimizer is a very powerful tool, but it has limitations. Our experience indicates that negative volumes may appear in the grid when the topology becomes too complex for the ATM Optimized. This is due to a very large width difference between leading and trailing edge. In this present work, the space design will be limited so that this problem does not appear. For further research, new topology and meshing algorithms will be proposed.

Taking into account the long time that the shape-optimization tool takes to get a final result, it is important to make sure that the different geometries that are evaluated in the surrogate model do not yield any error. In order to do that, a simple program has been created (Fig. 3.4). This algorithm creates the mesh of all the future evaluations and tells the user if the optimization tool will find any error in the process.

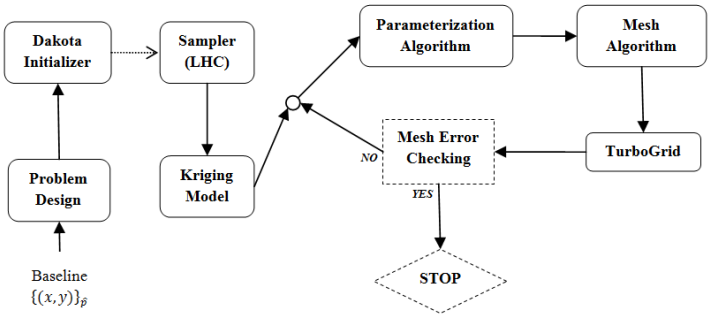


Figure 3.4: Mesh-Checking Algorithm.

It is important to notice that this algorithm can be used when all the following conditions are met:

- A surrogate-based GLOBAL strategy is used. This is due to the fact that a local strategy uses different trust regions along the optimization process, therefore only the first trust region could be analyzed with this algorithm.
- The LHS sampler is used WITH a seed specification.

When the Failure Mitigation Tool cannot be employed, a script has been programmed to detect mesh errors and propose ways to solve them. For example, our experience indicates that the ATM Optimized proposed by ANSYS sometimes performs poorly the B-Spline interpolation that is initially proposed. In the case that an error of this kind is detected, a Piecewise-Linear interpolation will be assigned to the geometry. In other cases, the initialization file differs too much from the given geometry and, when using a coarse mesh, it can produce a blow-up. In this case, the script will indicate to the Solver not to specify a initialization file.

3.2.3 Number of Cells Selection

In every algorithm that works with CFD tools, the number of cells to be imposed in the mesh is a critical issue. It can mean accuracy, convergence and computational time. To find a compromise between these variables, several number of cells have been computed for a given geometry. In blade optimization, entropy generation is very important and therefore this value should be compared for different mesh sizes. This study will be performed before each application.

In Figure 3.5 one can observe an example of this study. The entropy generation between the inlet and the outlet of a transonic blade is depicted.

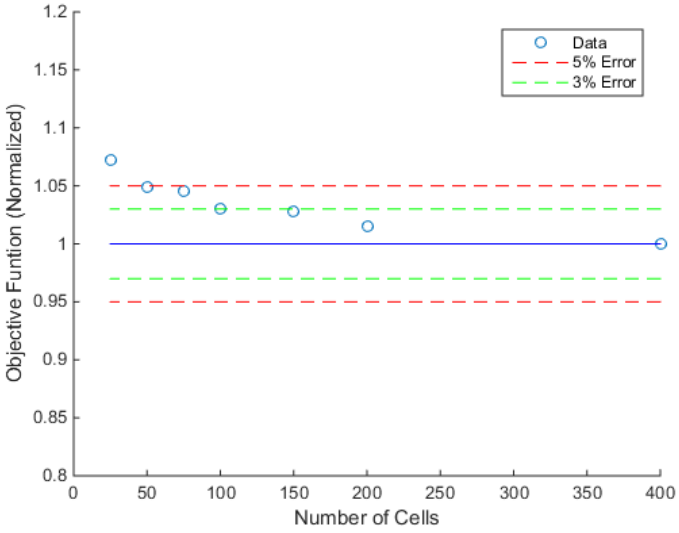


Figure 3.5: Entropy Generation discrepancy.

As an extra plot to be considered, the Mach Number distribution in a given position downstream has been depicted and compared for different number of cells (Fig. 3.6).

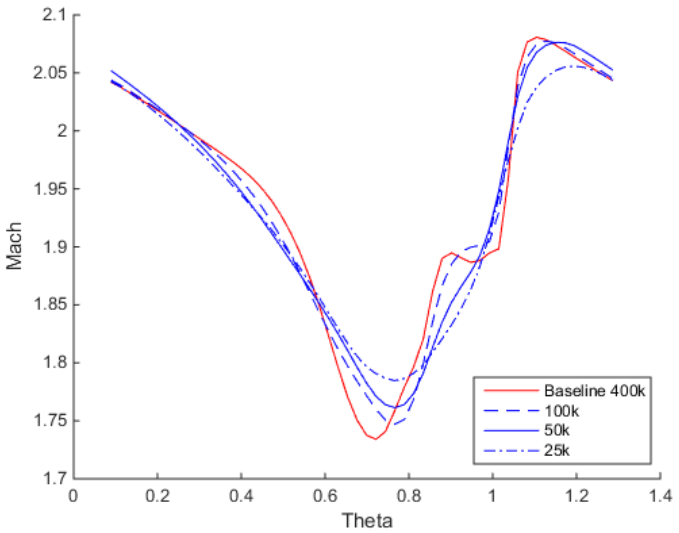


Figure 3.6: Mach Distribution discrepancy.

In this example, the value for 400,000 cells has been considered the true value. As a result of the previous study, a value of 50,000 cells seems to be enough for the successive optimization evaluations. This will define two different objective function assessments: Low-Fidelity for the coarse mesh and High-Fidelity with the fine mesh with large number of iterations.

Apart from the Mach distribution and the entropy production, it is interesting to check the influence on the objective function values that are used in this study. As it will be described in Chapter 4, the standard deviation of the pressure distributions half chord downstream will be commonly the parameter to optimize. Therefore, in Fig. 3.7 one can see an example of the discrepancy using a different number of cells.

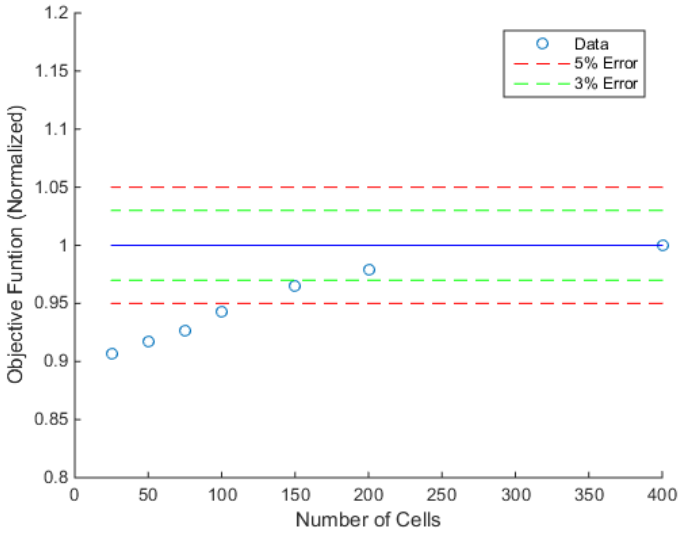


Figure 3.7: Pressure Standard Deviation discrepancy.

Although the pressure seems to depend a little stronger on the number of cells, it is still allowed its use with 50k cells for optimization. A further conclusion on this topic will be provided in Section 4.4, once real optimization results are available for assessment.

3.2.4 Timesteps Selection

In this work, it is critical to find an optimum number of iterations for each high-fidelity evaluation that takes into consideration and finds the compromise between accuracy and timing. It should be considered that the algorithm will need a number ranging from 100 to 200 evaluations for the global surrogate scheme, and about 500 to 1000 evaluations when the local scheme is needed. Hence, depending on the power of the machine in which the shape-optimization tool is launch we should determine a maximum time for each evaluation. In order to do that, a similar study to one in the previous section should be perfomed. As an example, the value of a objective function, in this case, the standard deviation of the Pressure distribution half chord downstream a transonic blade, has been computed for different values of cells and for different number of timesteps.

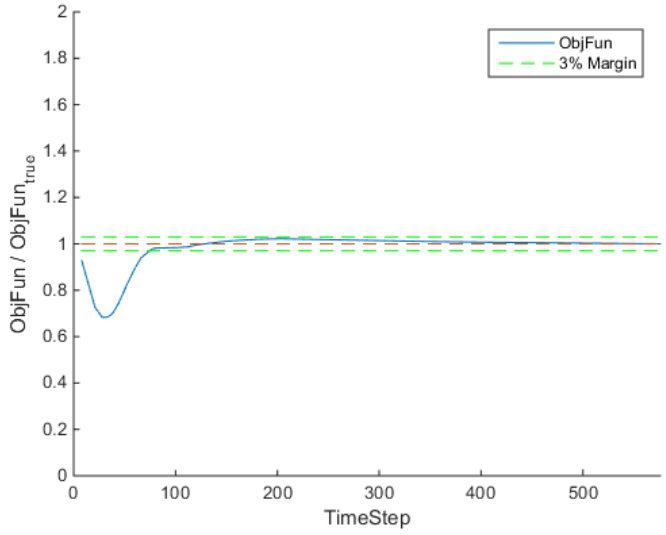


Figure 3.8: Objective Function as a function of the timesteps for 400k cells.

As a result, it can be identified a critical value of 150-200 iterations for each evaluation. The goodness of this selection will be checked once the algorithm is applied to a given geometry to optimize (Chaper 4). For further information about time-steps and computational cost, please refer to Section 5.1.

Apart from the previous analysis, it is important to take into account that the simulations that will be performed will use an initialization file, which means that the Solver will start solving the current problem interpolating from given results, which always means a faster convergence, above all in our

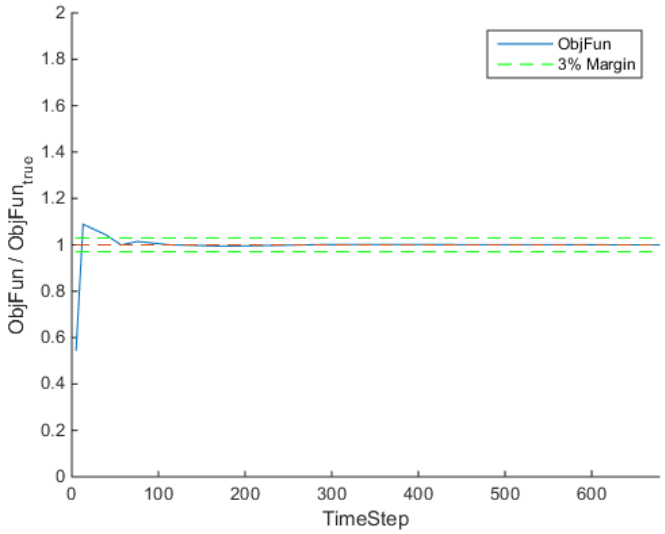


Figure 3.9: Objective Function as a function of the timesteps for 50k cells.

case, in which new geometries differ very little from each other.

3.3 High-Fidelity Tool

As it was explained in Section 2.2, a high-fidelity, fully turbulent and thermodynamically accurate simulations will be performed to solve the fluid-mechanics problems. These evaluations serve to assess the fitness function of the different members of the genetic algorithm population. Specifically for those cases with transonic applications, high-fidelity simulations that take into account real gas effects are of the greatest importance [2].

Special attention should be taken to provide correct instructions to the different parts of the simulation code during the Dakota evaluations[3]. The optimization environment calls the simulation code when needed and the results are returned to the environment. No gradients nor further information are required, so only one objective function is supplied. Any other data of the simulations will be updated along the optimization process, but it is not stored, fact that saves memory space and reduces optimization costs.

The process that the high-fidelity tool performs can be summarized as follows:

1. When the optimization process needs to evaluate the fitness of one

[3]Python has been chosen as the language to work with in this study.

member of the sample, it will call the simulation code via a python script.

2. The script first calls the parameterization algorithm. This algorithm is entirely written in Python. The environment provides a *.in* file that contains the design variables. The parameterization algorithm retrieves the fixed control points from a *.txt* file that the Problem Design Algorithm has generated after the initial interpolation. The adjustable control points are defined with the values of the design variables. With the complete set of control points, the parameterization algorithm generates the B-Spline curves that define the blade geometry. This set of points is stored in a *.crv* file that the mesh generator will be able to interpret.

3. The script then calls the mesh generator. Through an already generated topology (from a previous case or the Baseline), it uploads the curve through the *.crv* file generated previously. Once the new points are inserted and the topology is defined, the mesh is generated and stored in a *.gtm* file.

4. The preprocessor is called and the new mesh is uploaded to an already created *.pre* file with the problem conditions. The solver input file is generated from this step.

5. The solver is triggered, specifying the number of processors working in parallel, the wall clock time per evaluation or maximum number of iterations, the initialization file and the solver precision. After the specified time is completed, the results file *.res* is then stored for post-processing. In some cases, this file will also be used as the initialization solution for the next evaluation.

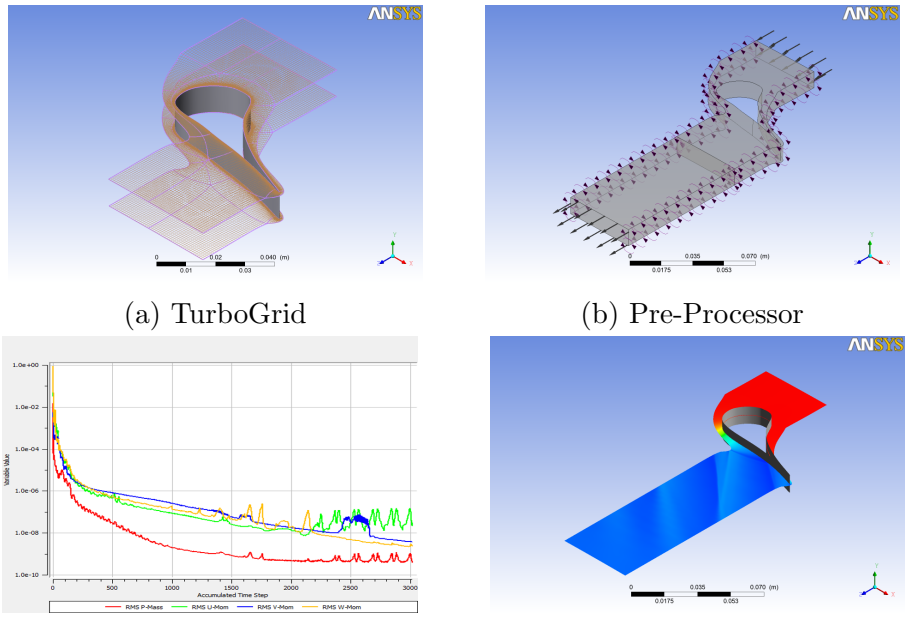
6. The post-processor, specific for turbomachinery problem will initialize the turbine components and will generate a *.csv* file with the required information (the desired objective function).

7. In some cases, the data provided by the post-processor is not directly employed for optimization. An evaluation script is then applied. For instance, this script can compute the standard deviation of a given distribution, when uniformity of a given flow variable is sought.

8. The value of the objective function is retrieved to the optimization environment, which will interpret this value, normalize it and correctly assign a fitness function value to the member of the sample.

All the simulations are performed in batch mode, using session files that need to be generated before the optimization is started. Those session files are short Perl scripts that the CFD toolkit is able to interpret as user inputs.

In Fig. 5.4, one can have a graphical idea of what is happening in each step of the high-fidelity tool.



(a) TurboGrid (b) Pre-Processor

(c) Solver (d) Post-Processor

Figure 3.10: High-Fidelity Tool Example.

Chapter 4

Application: Supersonic ORC turbine

As an application of the shape-optimization tool, the optimization of the convergent-divergent blade of a supersonic ORC turbine is performed (Figure 4.1).

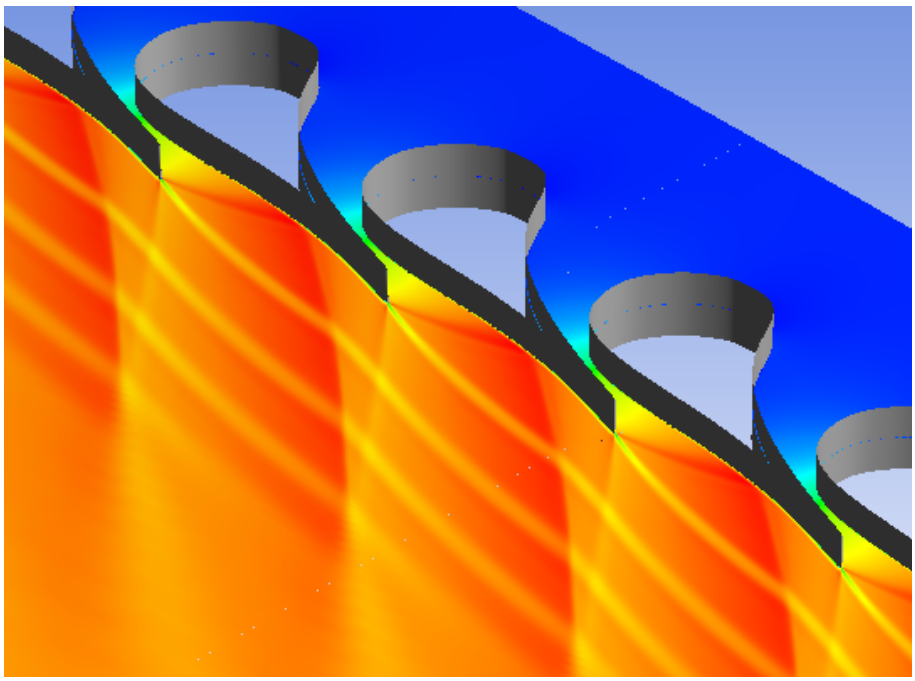


Figure 4.1: Supersonic blades (Mach Distribution).

Organic Rankine Cycles (ORC) have been proved to be a promising technology for small scale heat recovery systems. The main difference of an ORC

system with a convectional Rankine cycle is the working fluid, which in the case of ORC is a high-molecular organic fluid, instead of steam, commonly used in Rankine cycles. The use of this working fluid enables the achievement of high efficiencies, since it takes advantage of the dry expansion and high pressure differences.

An ORC turbine commonly involves highly supersonic flows, as a high-molecular fluid has a low speed of sound. The design of ORC turbines is then strongly characterized by real gas effects of the working fluid, and it should be taken into account when optimizing blade geometries, which are crucial in the improvement of the efficiency [2]. These features are easily supported by the shape-optimization algorithm that has been developed in this work and therefore its application to the improvement of the blade geometry of an ORC turbine stator is studied for assessment.

Strong real gas effects are considered in high-fidelity simulations of the blade-to-blade flow using an accurate CFD code. This can only be achieved by the development of a non intrusive tool. The evolutionary strategy coupled with surrogate models turns out to be excellent in this regard.

The optimization process for different parameters is performed. However, in the first place, the validation of the interpolation method needs to be carried out. In order to do so, the CFD solver is launch with the real geometry and with the interpolated or approximated B-Spline obtained with a limited number of control points and the least squares approximation. Once the interpolation method is validated, the optimization is ready to start. Ultimately, we will discuss the optimization results in the next chapter.

4.1 Shock waves and Entropy Production

A converging-diverging blade cascade of the stator of a supersonic ORC turbine is studied. When observing the 2D distribution of the Mach Number downstream the blade, in Fig. 4.1, one can notice shock waves patterns in the flow.

An analysis of the phenomena that take place in the flow is carefully considered. In Fig. 4.2 the investigation of the flow features is performed. They can be divided into three categories:

- **Wake Pattern.** In fluid mechanics, a wake is the region of the flow that is disturbed downstream of a moving body. They are produced because the two flows from the pressure and suction side of the blade get together. Specifically this effect comes from the boundary layers interaction on the two sides of the profile. Wakes spread outward from the

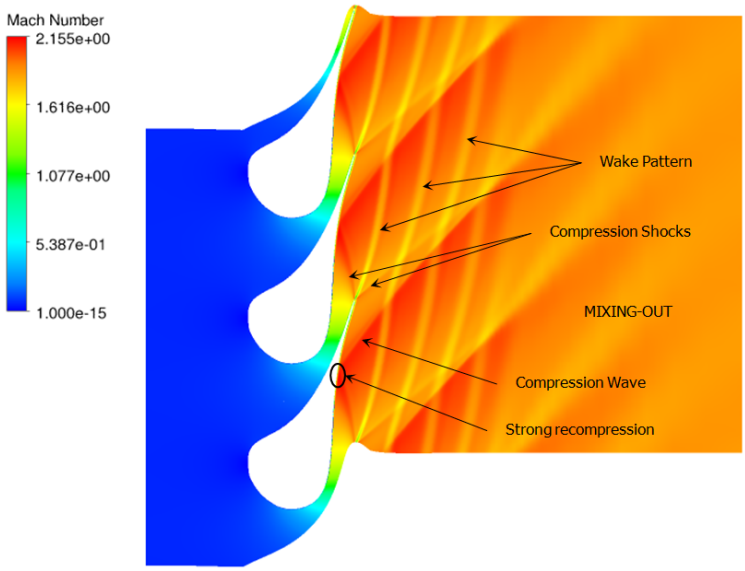


Figure 4.2: Shock Waves Distribution.

body until its energy is overcome or lost. In this case, the wake spread after the trailing edge of the blade until it mixes out and diffuses. It can be understood as a velocity defect and it will always happen. They can be therefore be seen very clearly in a Mach Number or Velocity distribution.

- **Fishtail Shock Pattern.** When the two jets in the trailing edge get together and collide, two reflecting shocks are produced due the acceleration of the supersonic flow. Two oblique shocks are formed at the trailing edge of the blade. It is indeed a compression shock wave because the low base pressure formed immediately behind the trailing edge makes the flow expand around it and then it is recompressed by a strong shock wave at the point where the suction and pressure side flows meet.

- **Compression Wave.** The generated compression wave is produced by the effect on the blade wall of the pressure side leg of the fishtail shock pattern generated at the adjacent blade trailing edge. The curvature of the blade in the suction side intensifies the effect and the boundary layer - shock wave interaction is of greatest importance. The strong local recompression in the collision point provokes an abrupt enlargement of the boundary layer. Although the shock wave might be weak when

arriving to the wall, an extra dissipation is likely to occur within and downstream of the point. Even if the boundary layer was laminar, the interaction will certainly cause transition, as the reflected shock will cause a significant increase of boundary layer momentum thickness.

These shock waves result in the non-uniformity of the flow downstream, affecting the efficiency of the stage. The two shock waves produced downstream of the blade mix out after propagating for some distance and this produces losses due to a large entropy generation that takes place in mixing and diffusion processes. In fact, the mechanisms for entropy generation are:

- **Viscous Friction.** It can occur in boundary or free shear layers, including mixing processes (leakage jet), due to the fact that the rate of shear strain is not the same as the vorticity and so viscous dissipation is not confined to boundary layers.

- **Non-equilibrium Processes.** Like the ones in very rapid expansions or shock waves. The entropy creation occurs due to the heat conduction and high viscous normal stresses within the shock wave.

- **Heat Transfer.**

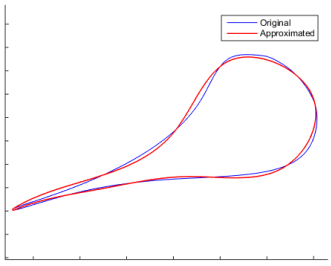
Actually, for most machines the flow is closely adiabatic and hence the entropy is generated only as a result of irreversibilities, which contributes significantly to the loss of efficiency [18], such as viscous effects in boundary layers, viscous effects in mixing processes and shock waves.

Therefore, main aim of the shape-optimization is to produce a more uniform flow downstream, which leads to weaker shock waves and reduction of the pressure losses. Moreover, the interaction between the generated shock waves and the boundary layer will be also attenuated, which will reduce the risk of boundary layer separation and will lead to further reduction of viscous losses. Moreover, a more uniform flow in the outlet will eventually help improve the efficiency of the subsequent rotor of the turbine stage.

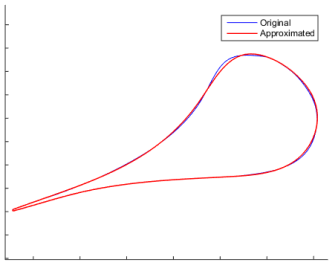
4.2 Validation of Parameterization Algorithms

In Fig. 4.3 the interpolation result for different number of control points is depicted. The assessment of the interpolation method using the convergent-divergent blade is very appropriate, as the ratio between the leading and trailing heads is large.

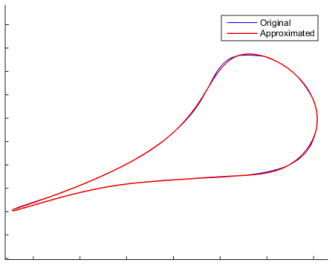
To validate the interpolation method, we first use the algorithm described previously in Fig. 3.2. This algorithm is simplified as in Fig. 4.4.



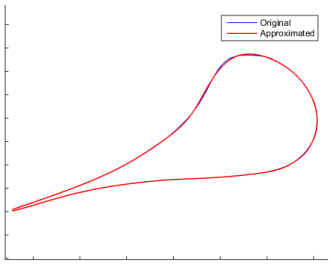
(a) $N_{CP} = 10$



(b) $N_{CP} = 12$



(c) $N_{CP} = 15$



(d) $N_{CP} = 20$

Figure 4.3: Different Number of Control Points.

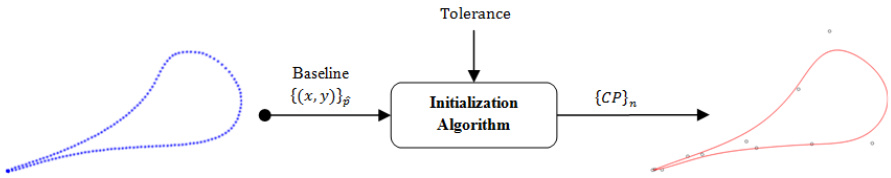
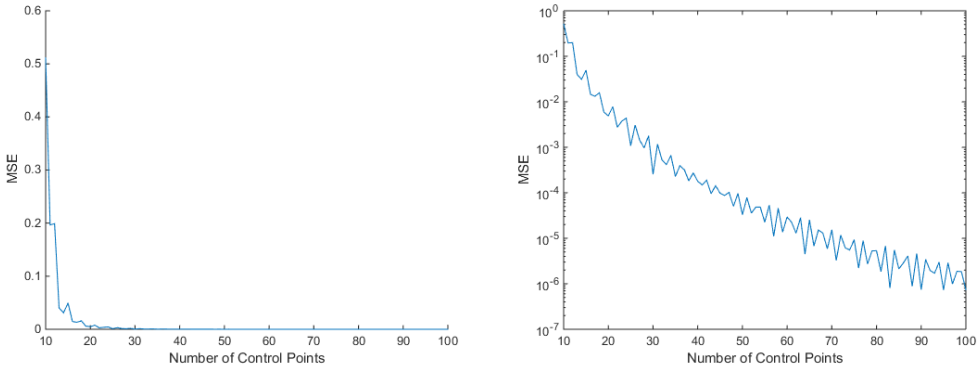


Figure 4.4: Simplified Initialization Process.

Only the baseline geometry points and the specified tolerance are provided to the algorithm. As a outcome of the process, an interpolated geometry, with its respective control points, is generated. To have a more graphical idea of the interpolation error, the code has been running for a large number of control points and the MSE has been computed[1]. The MSE as a function of the number of control points is depicted in Fig. 4.5a.

As expected, the error of interpolation decreases with the number of control points defining the curve. One can observe that the error strongly decays

[1]Recall from Section 2.1.3: $MSE = \frac{\sum_{i=0}^{P} \|p(w_i) - x(w_i)\|^2}{P+1}$

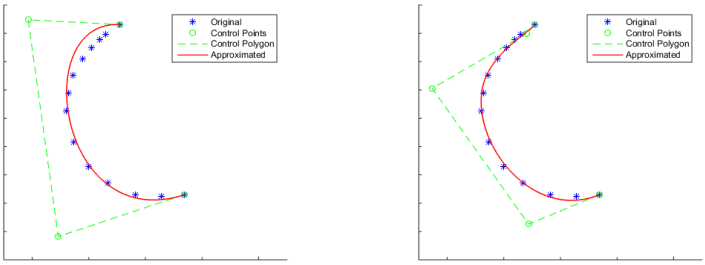


(a) Gross. (b) Logarithmic

Figure 4.5: Error of Interpolation.

at the beginning, remaining roughly steady after reaching, approximately, 25 control points.

At this point, the validation of the "trailing edge issue" has to be performed, as it was discussed in Section 2.1.3. We came across two different methods to deal with the big gradient region. In Fig 4.6 we can observe interpolations for different number of control points, using the first method (*Trailing edge as a B-Spline*):



(a) $N_{CP} = 4$ (b) $N_{CP} = 5$

Figure 4.6: Different Number of Control Points for the big gradient region.

On the other hand, this region can be defined by a circumference, if we consider the second method described (*Traling edge as a Common Radius*):

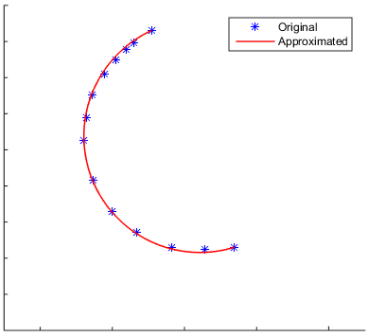


Figure 4.7: Trailing Edge as a common radius

Therefore, it can inferred that a Common Radius should be used for this blade geometry. The common radius method that will be used from this point on provides equal tangent in the pressure side of the blade, allowing for a much better pressure profile in the trailing edge (Fig. 4.8).

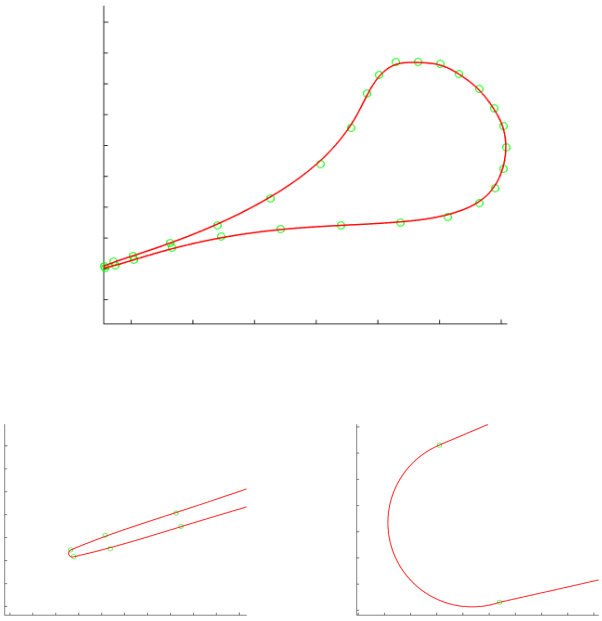


Figure 4.8: Detail of the Trailing Edge.

At this point, the number of control points used in the optimization and design variables needs some discussion. By observing Fig. 4.5b a starting

rough number of 30 control points seems appropriate. Furthermore, the knot sequence has been modified to assign the desired number of control points to each area. It is important to provide a good interpolation in the sonic and geometrical throats, since even very small changes affects to the pressure distribution on the blade[2]. Therefore, a knot sequence with a larger number of control points in those areas of interest is chosen. Once the interpolation result is obtained, the "cutting" control points are decided and the geometry is divided into different parts, as it can be seen in Fig. 4.10. It is important to find the compromise between the interpolation accuracy and the number of design variables. A large amount of the latter would yield to random shape variations in the genetic algorithm and surrogate model, which would find hard to optimize and to improve substantially the population in each cycle.

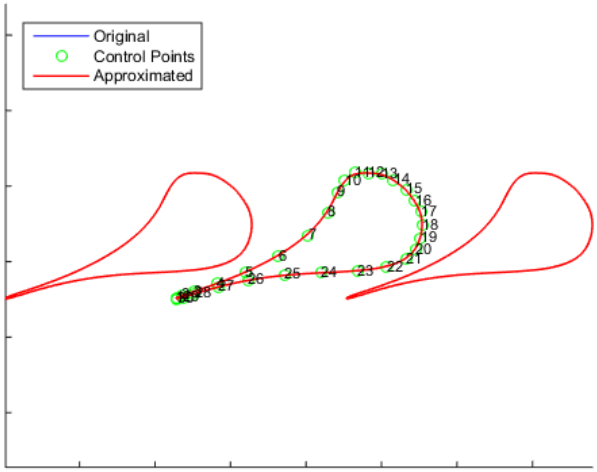


Figure 4.9: Interpolation result for 30 control points.

Now, the areas subjected to optimization are identified. Normally, the exclusion of the leading and trailing edge is recommended. In Fig. 4.10 the division has been established and the number of design variables is decided.

It is worth noticing that the trailing edge is set to move freely. In order to do that, we set a constrain such as the trailing edge width remains fixed, allowing it to move downwards during the optimization process. By doing so, only one design variable is needed to describe the trailing edge location.

[2]It has been proved that a 1% error in the geometry yields a 4% error in the Pressure Distribution.

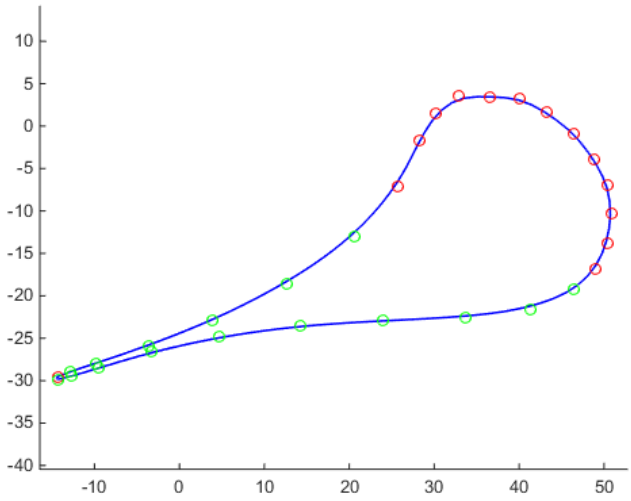


Figure 4.10: Division in fixed (red) and adjustable (green) areas.

4.2.1 CFD validation

The validation of the interpolation code is performed by carrying out simulations with the CFD tool. A supersonic turbine blade cascade for an ORC is computationally studied. A converging-diverging *Bière* blade with 6.5 cm of cascade pitch is considered. The original diverging part of its geometry is designed by means of the method of characteristics (MOC) and the leading-edge portion is constructed with a high smoothness [2]. The original blade performance is compared with the resulting blade from Algorithm 3.2.

Problem Conditions

As only blade-to-blade effects are of interest, quasi-3D calculations are performed and planar 2D profiles are considered. To achieve these conditions, a large number of blades is provided, in order for the radial effects to disappear (large circumference). For the boundary conditions, the design requirements are used.

The total pressure and total temperature at the inlet are fixed to 8.0 bar, 272°C respectively, and a weak boundary condition with average static pressure of 1.078 bar is placed at the outlet. The working compound is the siloxane MDM[3], whose thermophysical properties are calculated using a Lookup Table (LuT) that was generated considering a Span-Wagner model

[3]Octamethyltrisiloxane, $C_8H_{24}O_2Si_3$

(see Fig. 4.11). The importance of accurate fluid thermophysical descriptions in ORC turbines has been demonstrated in several studies [2, 19, 20]. These works showed that simulation methods not considering real gas effects are expected to mislead the design indications where strong non-ideal effects appear. In fact, the operating conditions of the present supersonic turbine lead to relevant real-gas effects. For a investigation of the advantages offered by a fully real-gas adjoint-based design methodology compared to approaches based on simplified equations of state, please refer particularly to [2].

In Fig. 4.11 it can be observed how the thermodynamic diagram has been divided with a grid according to the required accuracy. The data points have been included into a LuT that is called by the code when the gas properties are needed. Also, it can be observed how the problem conditions are far from those of ideal gas simplification.

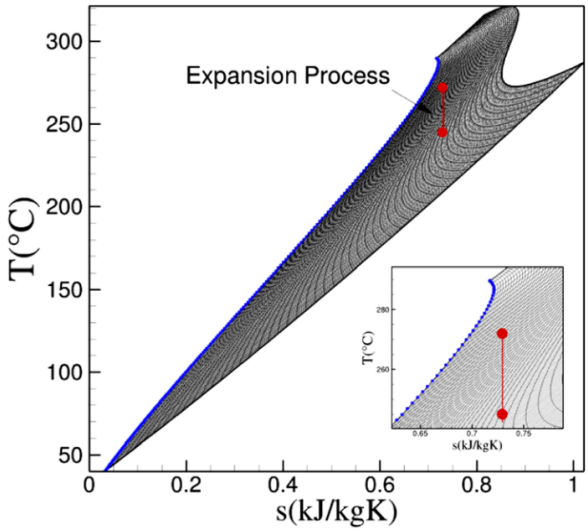


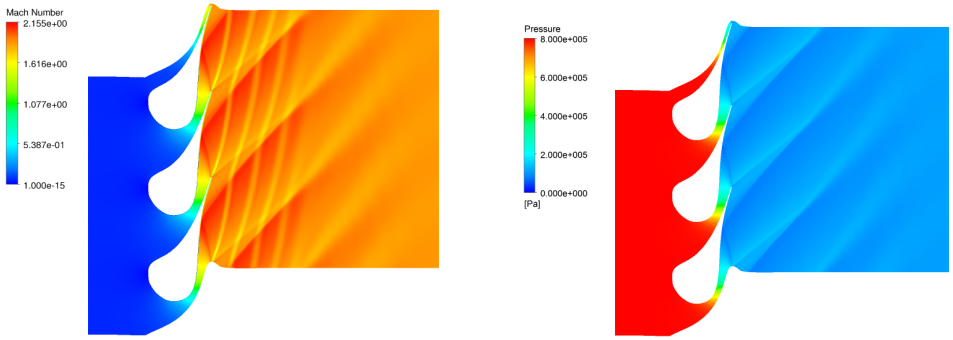
Figure 4.11: Thermodynamic diagram for the Siloxane MDM.

Turbulence effects are modeled using the $\kappa - \omega$ SST model and ensuring a y^+ below unity along the blade. This turbulence model is widely used in turbomachinery and blade design, as boundary layer and shock waves influence is of primary concern. As described previously, Siloxane MDM is treated as a real gas via LuT interpolation method suitably defined (Fig. 4.11). An outflow domain is placed at a distance of about three axial chords from the trailing edge, allowing for the flow features to fade out. The flow is considered purely axial at the inlet and turbulence quantities are fixed (5%, medium intensity).

Original Case Results

First, a simulation with the original points, extracted for a general case using MOC[4] method, is performed. The goodness of the interpolation is measured computing the pressure distribution as a function of the axial coordinate.

In Fig. 4.12a it is depicted the Mach Number contour plot of the original distribution of points. In Fig. 4.12b, the pressure distribution is plotted.



(a) Mach number distribution. (b) Pressure distribution.

In Fig. 4.13, pressure on the blade as a function of the axial coordinate can be observed for the Original distribution of points.

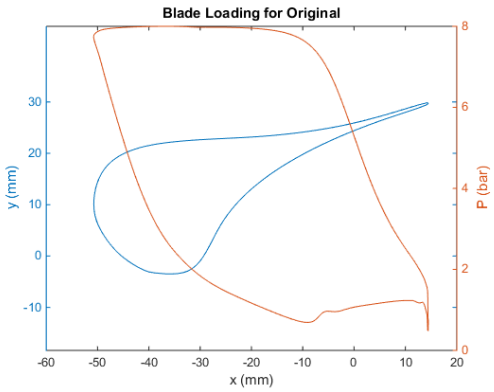


Figure 4.13: Pressure distribution (Original Case)

Comparison of Interpolated Case

For the interpolated blade shape, the results of the pressure distribution along the axial coordinate are compared and the accuracy of the interpolation

[4]The Method of Characteristics (MOC) is a commonly used analytical method to design supersonic blades by solving ordinary differential equations on a suitable hypersurface.

is estimated. In Fig. 4.14 the difference in the pressure distribution along the axial coordinate can be observed. Some differences are found but the compromise with the number of design variables determines its acceptance.

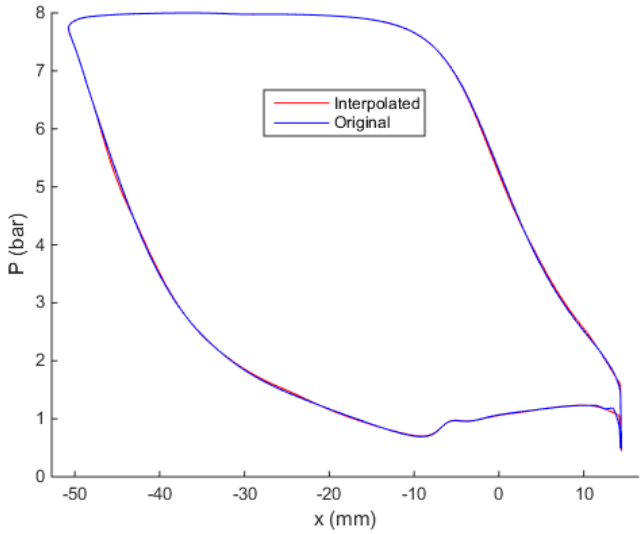


Figure 4.14: Pressure distribution comparison.

4.3 Optimization Process

A surrogate-based local method (trust regions method) will be applied to assure convergence[5], along with a Pressure-based objective function (standard deviation of the pressure profile in a circumferential line half chord downstream).

The design space is depicted in Fig. 4.15. The trailing edge is allowed to move, almost freely, downwards. The optimization parameters are those indicated in Table 4.1.

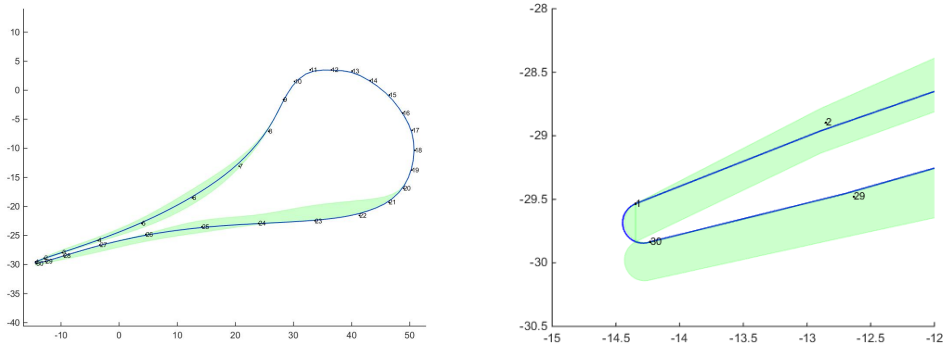


Figure 4.15: Design Space.

With the parameters described in Table 4.1, the optimization process is started, yielding the results depicted in Fig 4.16. It can be observed a rapid reduction of the standard deviation of the pressure distribution half chord downstream. In the next section, the consequences of this reduced standard deviation will be studied. In the following graphs, *Pini M. (2015)*'s reference line is the value obtained through the Adjoint Method in a previous study of this research group [2]. The same blade was computationally studied and optimized with a gradient-based method. This present work aims to provide an automated tool for a wide range of blade geometries, and therefore the optimization results will be evaluated and compared following already known results.

[5]As it will be seen in the Conclusions, a global scheme will yield results in a faster manner. However, convergence to the optimum is not assured and therefore for this thesis, the result obtained with the local optimizer are shown. For further discussion, please refer to Section 5.3.

Genetic Algorithm			
Generations		150	
Population Size		50	
Crossover Rate		0.8	
Crossover Points		2	
Mutation Rate		0.2	
Mutation Type		Replace Uniform	
Surrogate-Based Method			
Method Type		Local	
Sampling Type		LHS	
Sample Size		80	
Iterations		20	
Surrogate Model	Type	Global - Kriging	
	Trend Function	Reduced Quadratic	
	Correlation Params,	Global	
	Correlation Lengths	Internally Computed	
	Correction	No Crrection	
	Trust Region	Initial Size	0.8
		Minimum Size	10^{-5}
		Contract Threshold	0.25
		Expand Threshold	0.75
		Contraction Factor	0.5
		Expansion Factor	2

Table 4.1: Optimization Parameters

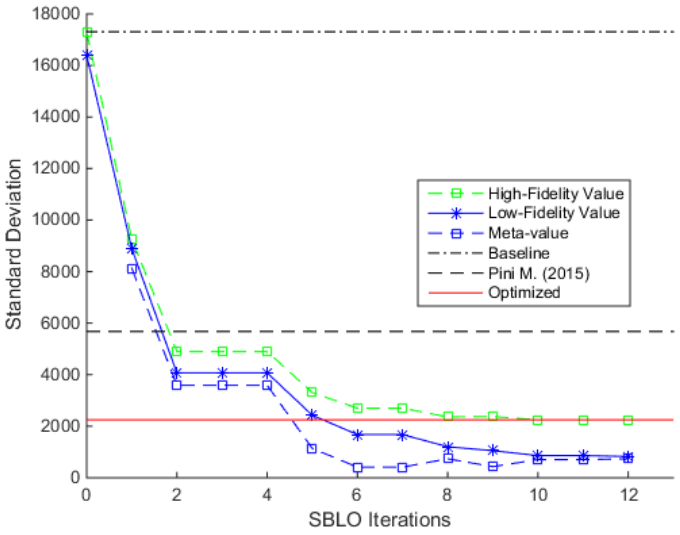


Figure 4.16: Optimization Process.

Along with Fig. 4.16, the evolution of the trust region ratio in Fig 4.17 should be study to fully understand the evolution of the optimization process. As it is observed, the reason of the flat region in iterations 2, 3 and 4 in Fig. 4.16 is due to the poor behavior of the trust region accuracy during those iterations. As it was explained in Table 2.1, the current step is rejected either because of it poorly accuracy or because the Pareto-filter, which excludes new values that do not improve an already-found optimum. Clearly, iterations 2, 3 and 4 yield the same optimum value. This trust region updating process ensures convergence.

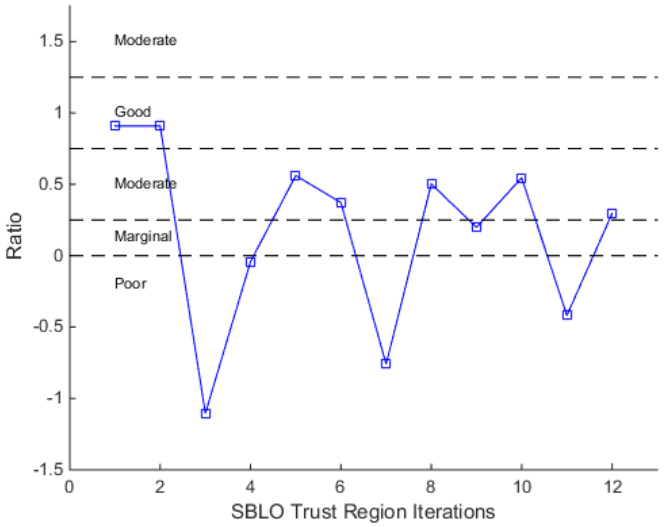


Figure 4.17: Trust Region Ratio.

At the end of the process, a very nice reduction of the shock waves generated can be observed. The Pressure distribution, as it is shown in Fig. 4.18, has much less standard deviation. To evaluate the goodness of the pressure-based objective function, the Mach distribution is also plotted in Fig. 5.9, to assess the new intensity of the shock waves.

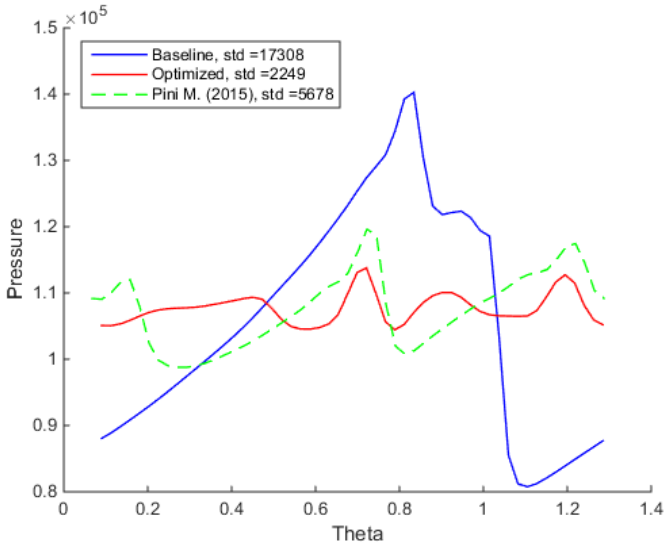
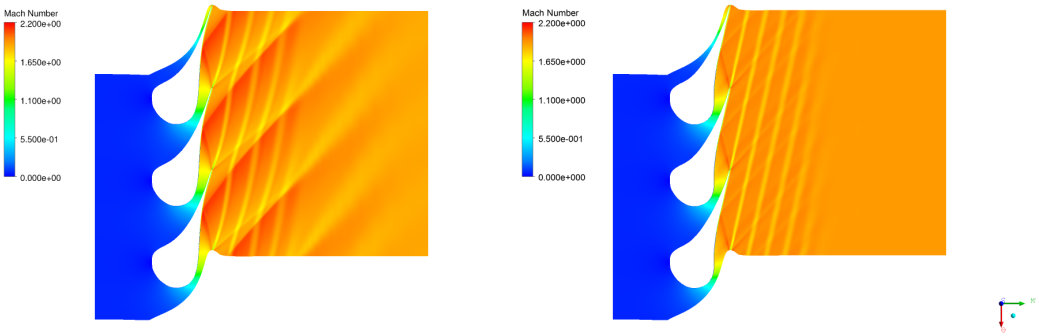


Figure 4.18: Pressure distribution half chord downstream.



(a) Baseline (b) Optimized

Figure 4.19: Mach Distribution.

It is of great importance to notice that the shock waves in this new case are much weaker than in the Baseline geometry. Optimization has worked as expected, using a surrogate-based local Kriging model, a genetic algorithm and a B-Splines parameterization.

4.4 Optimization Results

In this section, the physical consequences of the blade optimization are shown. For comments about the optimization process itself, please refer to Chaper 5 or Section 5.2.

As a result of the optimization process, the Mach number distribution turns out to be as in Fig. 4.20.

Figure 4.20: Optimized - 2D Mach Distribution.

For this blade geometry, the flow is much more uniform downstream, which will simplify the design of the subsequent rotor. The entropy production and efficiency are improved due to the reduction of the shock waves generated by the transonic flow. In the following section the study of the efficiency is performed.

4.4.1 Analysis of Performance

The new performance of the new blade needs to be assessed to confirm that, by assuring uniformity of the flow, a good way for optimization is provided. For further information about the relation between entropy generation and efficiency of turbines please see Appendix B.1.

A way to assess the new efficiency of the blade is by computing the total pressure losses (also known as Stagnation Pressure Loss Coefficient), through Eq.4.1. This is the most common way to define the losses for individual blade rows and, in our case, this provides a good way to evaluate the performance of the stator. In this regard, $P_{T,in} - P_T$ measures the pressure energy loss

employed to increase the dynamic pressure through $P_{T,in} - P$. The expression turns out:

$$Y = \frac{P_{T,in} - P_T}{P_{T,in} - P} \tag{4.1}$$

where $P_{T,in}$ stands for the total pressure (mass flow averaged) at the inlet position, P_T is the total pressure at the given position, and P refers to the static pressure (area averaged) at the same given position.

In Fig. 4.21 one can observe the pressure losses for the baseline and that of the Adjoint Method [2], to compare with the new geometry proposed in this study. The two optimization results show a similar tendency. However, the optimized geometry returns a lower value of the losses, increasing the efficiency of the stage.

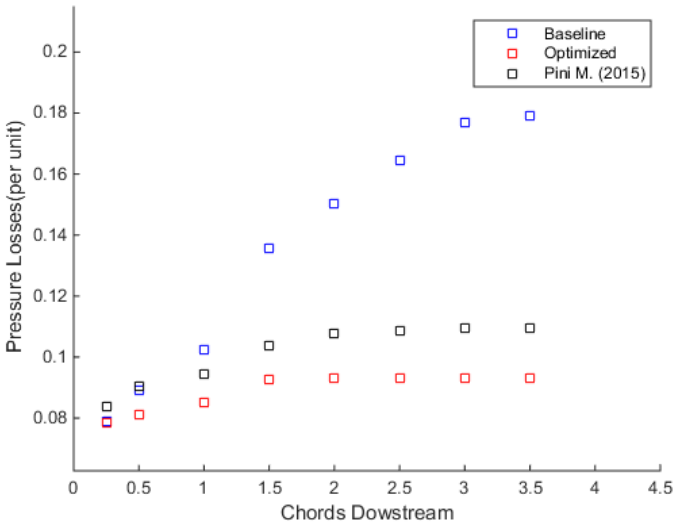


Figure 4.21: Optimized - Pressure Losses Coefficient.

The pressure losses have a very close relation with the entropy production, being the latter the only source of losses in the blade, as no heat is added and the viscous effects increase the entropy in the fluid (see Section 4.1)

By analyzing the entropy generation on the single blade row of the stator of the turbine one can get an idea of how the whole performance of the stage is. As it can be observed in Fig. 4.22, the entropy production ($\Delta S = \overline{S} - \overline{S}_{in}$) is almost constant downstream, showing a value much lower than that of the Baseline geometry. The latter keeps growing as the mixing of shock waves occurs.

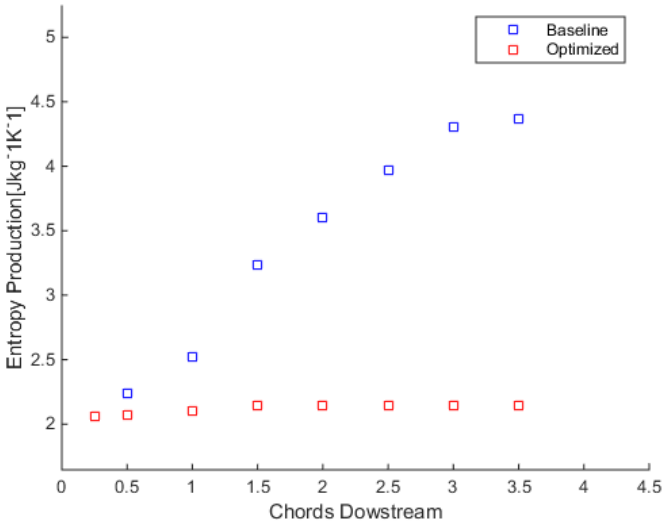
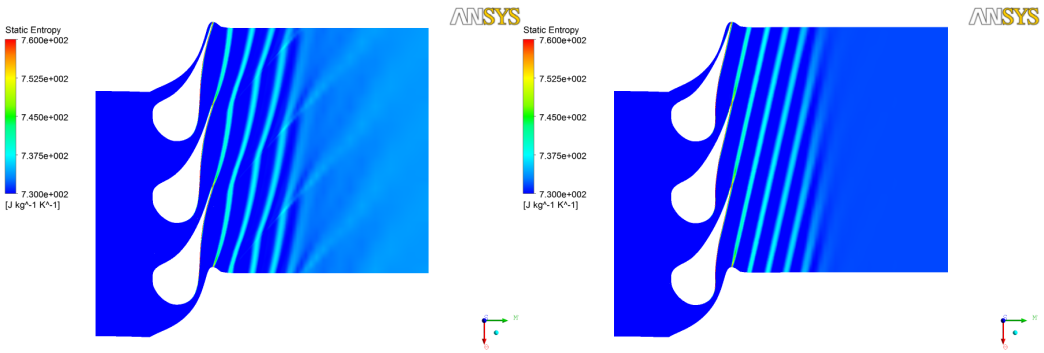


Figure 4.22: Optimized - Entropy Production.

In fact, we can see in Fig. 4.22 that the entropy generation is uniform downstream of the blade. On the contrary, for the Baseline the gradients are large and there is a successive production of entropy along the wake (Fig. 4.23a). When the shock waves are mixed out downstream, the diffusion generates a large entropy production that compromises efficiency for the Baseline geometry.



(a) Baseline (b) Optimized

Figure 4.23: Entropy Distribution.

Entropy is generated within the flow whenever a deleterious and irreversible process takes place. This “excess” of entropy is convected downstream through the machine and diffuses into the surrounding flow. The

entropy generated and thus "concentrated" at the exit of the machine includes a contribution from every source within the machine and the loss of machine efficiency is proportional to the average "concentration" of entropy at the exit [18].

For the optimized case (Fig. 4.23b) the entropy production is uniform and in straight lines from the trailing edge, which gives an image that the pressure waves are much weaker, so that the optimization aim has been achieved. As a consequence, shock losses are greatly minimized and the cascades is more efficient.

4.4.2 Blade Shape

The blade shape after the optimization process turns out as in Fig. 4.24.

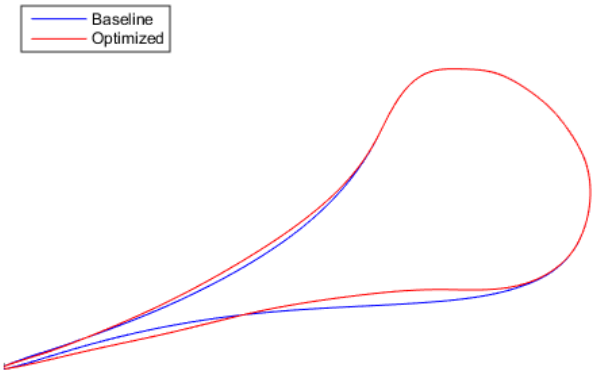


Figure 4.24: Optimized Blade Shape.

One can easily understand the goodness of this tool to optimize supersonic ducts, in which a very small deviation on the baseline blade geometry yields a great variation on the flow variables. The blades can be conceived as convergent-divergent ducts. Therefore, a deep analysis of them, with an approximate location of the throat can be depicted as in Fig. 4.25.

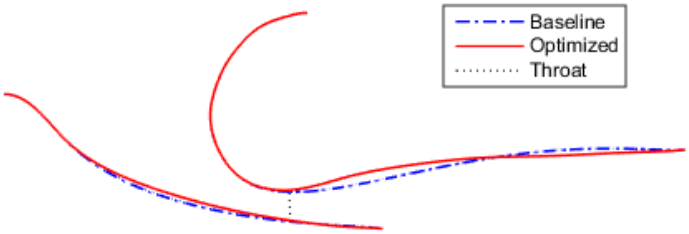


Figure 4.25: Optimized Duct Throat.

As observed in the previous section, the shock waves are strongly weakened in the optimized configuration. This is a direct consequence of the shape of the new suction side of the blade. Thanks to the almost straight shape of the rear suction side, no significant over-speed is observed in the semi-bladed region (part of the blade that is bounded by only one blade wall).

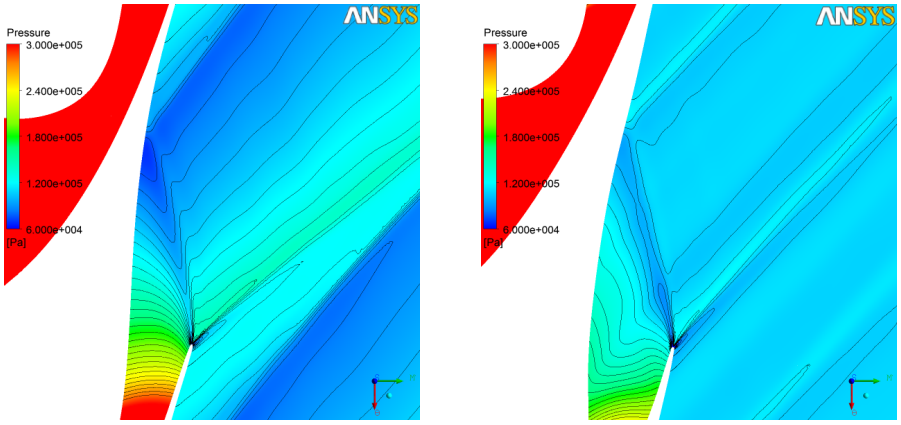
4.4.3 Analysis of Shock Waves

As a measure of the shock waves, the pressure distribution on the blade is a good measure. In Fig. 4.26, one can observe the pressure distribution over the baseline configuration (Fig. 4.26a), the optimized geometry using the Adjoint method described by [2] (Fig. 4.26b), and the optimized geometry using genetic algorithms as it was described in this study ((Fig. 4.26c)).

(a) Baseline (b) Adjoint method (c) Genetic Algorithm

Figure 4.26: Compression Fan Waves.

The main effect appears in the trailing edge area, as it is depicted in Fig. 4.27.



(a) Baseline (b) Genetic Algorithm

Figure 4.27: Reflected compression shock waves.

In the optimized case in Fig. 4.27, the straight rear shape mitigates the shock generation. In that case, the downstream shock is just produced by the shock reflected on the blade wall. The shock reflection will cause a local drop in the isentropic Mach Number, which will certainly reduce the flow velocity along the rear suction side. Also, the boundary layer thickness is not enlarged, as it happened in the Baseline configuration due to the strong recompression. This leads to weaker shocks in the trailing edge, as the cross section available for the fluid will not suffer a reduction because of an increment in the boundary layer thickness.

4.4.4 Flow Rate

For any optimization performed, the variation of the mass flow should be carefully considered. It should not be forgotten that the optimization is performed over a current design. A large variation of the mass flow downstream means that the blade span needs to be modified consequently so that the nominal flux is complied. This would lead to an abrupt change on the design of the whole turbine. The mass flow downstream could be modified due to a change in the velocity flow angle.

The mass flow rate as a function of the blade span is as follows:

	Baseline	Optimized
$\dot{m}$ $(\frac{kg}{s \cdot m})$	15.88	15.27
$\dot{m}_{rel}$	1	0.96

For future work, a flow rate constraint will be applied to the optimization.

Chapter 5

Conclusions and Future Work

It has been shown that genetic algorithms can efficiently be used for shape-optimization. They have many advantages over gradient-based methods, as genetic algorithms do not precise of more information than the value of the objective function. This allows the user to use the tool for a wide range of applications with the minimal variation in the original code. Not only did genetic algorithms provide a non-intrusive tool, but also the final optimum improved that of a gradient-based method (Adjoin method). Genetic algorithms are very efficient to search for global optimum regions, being useful when non-smooth, oscillating or/and non-regular functions need to be studied.

5.1 Important Aspects

The curve-parameterization using B-Splines meant an intuitive way to convert physical phenomena into solely-mathematical variables. The local support of B-Splines allowed a detailed shape-optimization, exploring alternatives otherwise unperceived using standard blade geometries or analytical approaches.

A surrogate-based local strategy successfully provides a low computational cost for shape-optimization in comparison to a direct utilization of genetic algorithms with CFD codes. The randomness intrinsic to sampling methods involved in the construction of effective trust regions (Latin Hypercube in our case) allows for global search over the design space, as it was shown when comparing with the global scheme.

The use of high-fidelity simulations that take into account real-gas effects, fully-turbulent flow has been proved to successfully mimic the real transonic flow of ORC turbines, providing an easy way for shape-optimization.

Computational Cost

In computational physics and engineering it is very important to assess the efficiency of an algorithm in terms of its computational cost. A tool can provide great results but it might be too expensive in time, power or computer resources.

It is worth mentioning that almost all computational time is consumed by the CFD simulations, being the required time for the optimization via genetic algorithm of the Kriging mathematical model almost negligible.

The computational cost would be extracted from the following formula:

$$Time \approx i * s * te \tag{5.1}$$

where i is the number of iterations of the SBLO, s is the number of samples in each trust region, and te is the time per evaluation.

If we consider the minimum number of samples for Kriging to accurately represent the objective function, $s = 5 * DV$, where DV is the number of design variables. te can be considered a linear function of the required iterations per evaluation (time-steps), ts, resulting $te \approx k * ts$, where k is a parameter that depends on the machine employed to run the code.

Formula 5.2 represents approximately the time consumed.

$$\begin{gathered} Time \approx K_{machine} * i * DV * ts \\ K_{machine} = 5 * \frac{time}{iteration} \end{gathered} \tag{5.2}$$

For each different problem and prior to the optimization we need to determine DV (as in Section 3.2.2), ts (as in Section 3.2.4) and $K_{machine}$. The latter is a function of the machine, but also it depends on the number of cells employed in the mesh generation (as in Section 3.2.3). The only unknown of the problem after launching the tool would be the number of iterations for the SBLO (i), as it is common in any optimization problem based on iterations. It has been proved, though, that roughly 10 iterations provide a more than acceptable result for all the cases studied via the surrogate-based local approach.

5.2 Optimization Strategy Comparison

In the optimization case discussed in Section 4.3, the optimization strategy followed was a surrogate-based *local* scheme.

However, once a good set of optimization parameters has been found (Table 4.1) for the surrogate-based local method, the comparison with a *global* scheme is provided, as in Fig. 5.1. A global scheme does not assure convergence, although the final result is, for this case, the same. Hence, it can be concluded that the local optimization scheme does search for a global optimum, although its trust region approach.

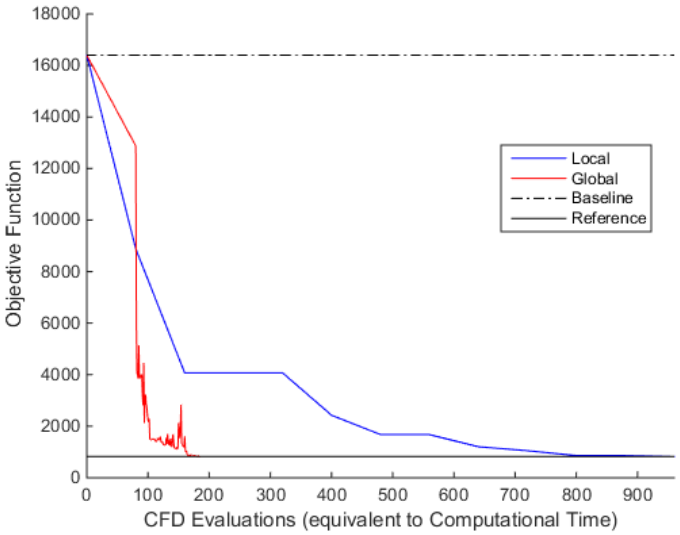


Figure 5.1: Comparison between local and global schemes.

In Fig. 5.1 it can be observed the rapid evolution of the global optimization strategy. In this case, the initial model is built using 80 samples, so for the first 80 evaluations there is no optimization. From that point on, the global scheme evolves very quickly and it reaches almost the minimum for less than 100 surrogate iterations.

The values represented in the figure correspond to the low-fidelity (coarse mesh) CFD simulations. It can be implied that the high-fidelity values will follow a similar pattern.

5.3 Publications

As an outcome of this work, several alternatives will be explored. As in Chapter 4, surrogate-based local optimization with genetic algorithms is very appropriate for application to ORC transonic turbines. As a result, a paper for the ASME ORC 2015 Conference is submitted, resulting in the following abstract:

AUTOMATIC DESIGN OF SUPERSONIC TURBINE PROFILES BY EVOLUTIONARY ALGORITHMS

Giacomo Persico, Pablo Rodriguez Fernandez

Laboratorio di Fluidodinamica delle Macchine
Politecnico di Milano
Via Lambruschini 4, I-20156 Milano, Belgium
e-mail: giacomo.persico@polimi.it
Web page: http://www.lfm.polimi.it

ABSTRACT

In this work an automated design tool for Organic Rankine Cycles (ORC) turbines is presented. Supersonic flows and real-gas effects feature ORC turbines and complicate significantly their design. From this perspective the design of ORC turbine blades cannot rely on simplified rules but needs the application of dedicated and automated tools to achieve an optimal blade shape. This study proposes a complete method to perform shape optimization of ORC turbine blades, constructed as a combination of a generalized geometrical parametrization technique, a high-fidelity Computational Fluid Dynamic (CFD) solver (including real gas and turbulence models) and an evolutionary algorithm.

The blade geometry is defined using parameterization techniques based on B-Splines curves, that allow to handle both global and local control of the shape. The space location of the control points of the B-Spline curve define the design variables of the optimization problem. The performance of the blade shape is assessed by means of fully turbulent flow simulations performed with a CFD package, in which a look-up table method is applied to ensure an accurate thermodynamic treatment. The solver is set in parallel along with the optimization tool to determine the optimal shape of the blade. As only blade-to-blade effects are of interest in this study, quasi-3D calculations are performed, and a single-objective evolutionary strategy is applied to the optimization. As a result, a non-intrusive tool, with no need for gradients definition, is developed. The computational cost is reduced by the use of surrogate models. A Gaussian interpolation scheme (Kriging model) is applied for the estimated n-dimensional function, and a surrogate-based local optimization strategy is proved to yield an accurate way for optimization.

Application to ORC turbines optimization has been proved to be successful, also in comparison to previous approaches based on inviscid flows, resulting in a comprehensive method for a very wide range of applications. In particular the present optimization scheme has been applied to the re-design of a supersonic stator cascade of an axial-flow turbine. In this design exercise very strong shock waves are generated in the rear blade suction side and shock-boundary layer interaction mechanisms occur. A significant efficiency improvement as a consequence of a more uniform flow at the blade outlet section of the stator is achieved. This is also expected to provide beneficial effects on the design of a subsequent downstream rotor. The method provides an improvement to gradient-based methods and an optimized blade geometry is easily achieved using the genetic algorithm. It is confirmed that only a couple of iterations of the surrogate-based local optimizer are enough to enhance the performance of the blade, with a limited computational cost of the proposed methodology.

5.4 Research Lines

This study opened a wide range of research lines on the development of shape-optimization tools for turbomachinery blades.

Specifically for the optimization of converging-diverging ORC blades:

1. Perform optimization using mass-flow constraint.
2. Perform multi-objective optimization to achieve high efficiency and desired flow features.
3. Perform sensitivity analysis on operating conditions.

For the development of improved tools:

1. Write own genetic algorithm and Kriging model, to avoid the use of external libraries and implement improvements and state-of-art evolutionary strategies.
2. Write own optimization environment.
3. Create graphical user interface for the problem design algorithms.
4. Improve Topology and Meshing Design, to allow complex geometries and large trailing-leading edges width differences.
5. Update parameters of the code along with the evolution of the process, such as the mesh-size.

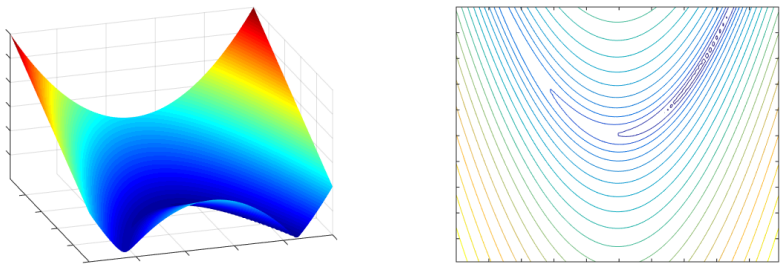
Appendix A

Instructive Examples

A.1 Genetic Algorithm

In order to fully understand how a genetic algorithm works, a single-objective example has been performed.

The Rosenbrock function (Fig. A.1) has been widely used as a performance test for optimization algorithms. In this function, the global minimum is inside a narrow parabolic shaped valley. Its importance in testing optimization methods is due to the high difficulty of converging to the global minimum. This has been proved to be located exactly at point $(1, 1)$.

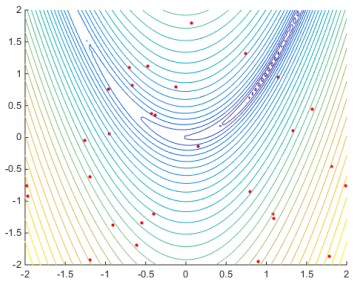


(a) 3D surface (b) Contours

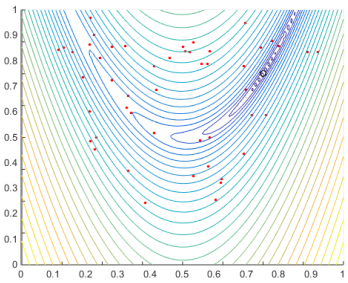
Figure A.1: Rosenbrock function.

We perform the optimization of the Rosenbrock function by using the single-objective genetic algorithm provided by the JEGA database. A population size of 30 random individuals has been chosen. This size will be maintained all over the evolution. Crossover is performed with a rate of 1. The mutation is defined as offset normal and the scale has been set to 1. The

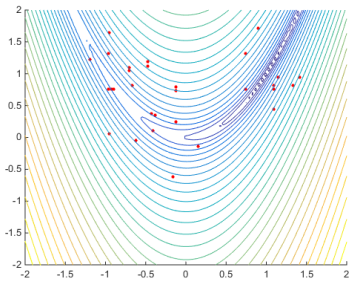
crossover carries out a 2 points cutting of the parenthood genotype. The replacement is elitist and the design variables (x, y) are bounded within the interval $[-2, 2]$ for both of them. The result of the optimization process is shown in Fig A.2. It can be observed that the method converges to an area near the optimum very fast (roughly 20 generations), with very little computational cost. The genetic algorithm does not get stuck in the parabolic valley due to the maintenance of diversity through successive mutations.



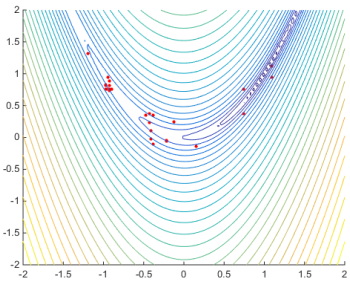
(a) Initial Population



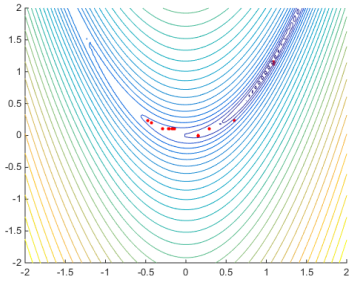
(b) Generation 2



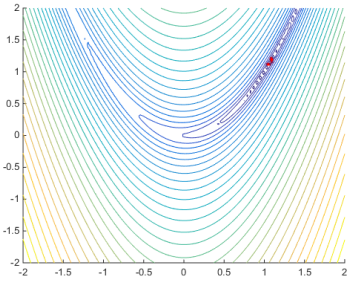
(c) Generation 3



(d) Generation 5



(e) Generation 10



(f) Generation 20

Figure A.2: Genetic Algorithm Optimization.

The different parameters available for evolutionary strategy optimization are very varied and their influence strongly differs from one case to another.

However some example of this variance should be taken into account.

For the first optimization, the size of the population should be considered and varied.

Genetic Algorithm	
Number of Generations	50
Crossover Rate	0.8
Crossover Points	1
Mutation Rate	0.2
Mutation Type	Replace Uniform

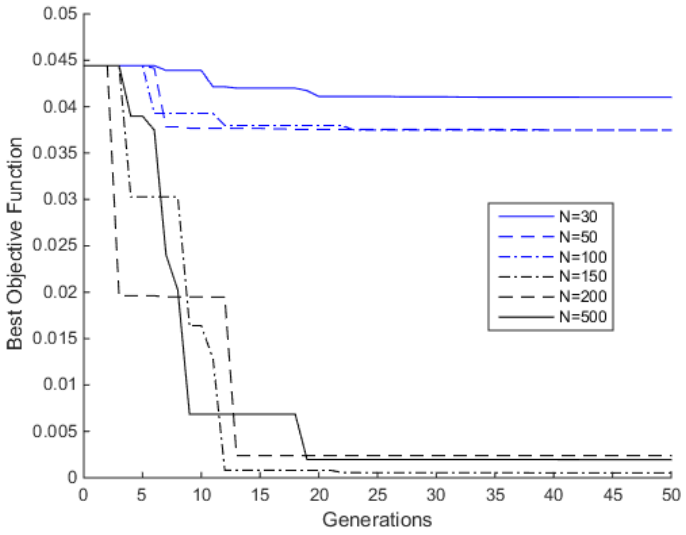


Figure A.3: Population Size influence in the convergence.

Empirical results from many authors suggest that population sizes as small as 30 are quite adequate in many cases. An initial population of about 50 should contain sufficient alleles for the genetic algorithm to make useful progress. In the previous graph, larger sizes produce an apparent better performance. This is due to the small design space and high mutation rate. Randomly, the process with such large number of members investigate more regions of the space, but a greater computational cost, which can be avoided with a size of 30 or 50 samples.

Regarding parenting, crossover rate value generates a great controversy. It cannot be found a pattern for this value, which ranges from 0 to 1. For different values we get:

Genetic Algorithm	
Number of Generations	50
Population Size	50
Crossover Points	1
Mutation Rate	0.2
Mutation Type	Replace Uniform

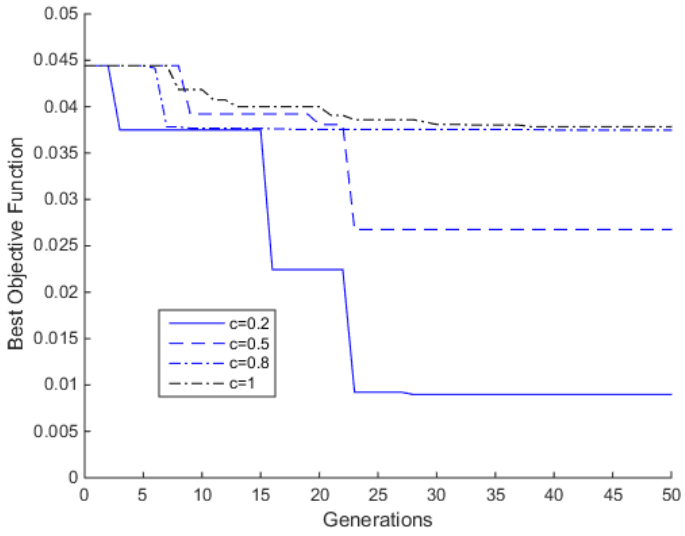


Figure A.4: Crossover rate influence in the convergence.

An inexperience researcher may set a 100% crossover rate. However, as it can be observed, the latter value produce the worse performance. The mutation-crossover relation is of greatest importance. Some authors think that mutation is more important than crossover, since it allows to search into regions unreachable from an only-crossover approach.

In crossover, the number of points in which the parents are divided to produce offspring is important:

Genetic Algorithm	
Number of Generations	50
Population Size	50
Crossover Rate	0.8
Mutation Rate	0.2
Mutation Type	Replace Uniform

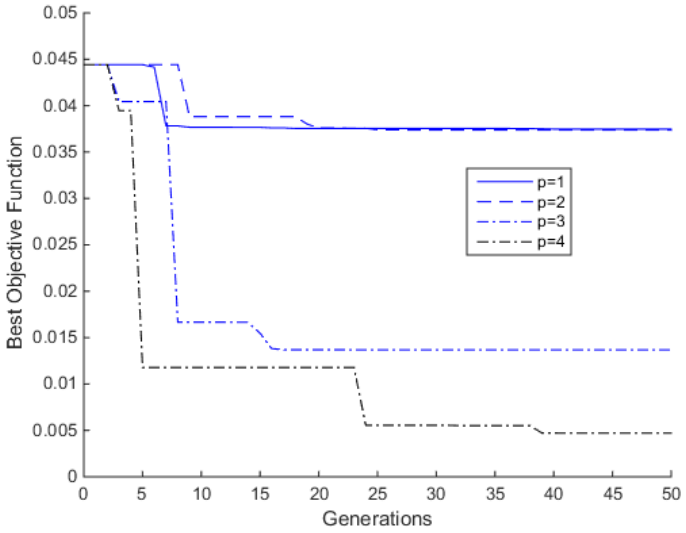


Figure A.5: Crossover points number influence in the convergence.

The effect observed in Fig. A.5 is similar to the one in Fig. A.3. A larger number of crossover points necessarily involves more randomness, a very important thing to take into account when designing a robust genetic algorithm.

Among all factors, the mutation rate is one of the most important:

Genetic Algorithm	
Number of Generations	50
Population Size	50
Crossover Rate	0.8
Crossover Points	1
Mutation Type	Replace Uniform

Mutation not only has the role of helping to investigate further regions of the design space, but also a secondary function should be considered. They help to preserve a reasonable level of diversity. Not all authors agree and the balance between crossover and mutation if often a problem.

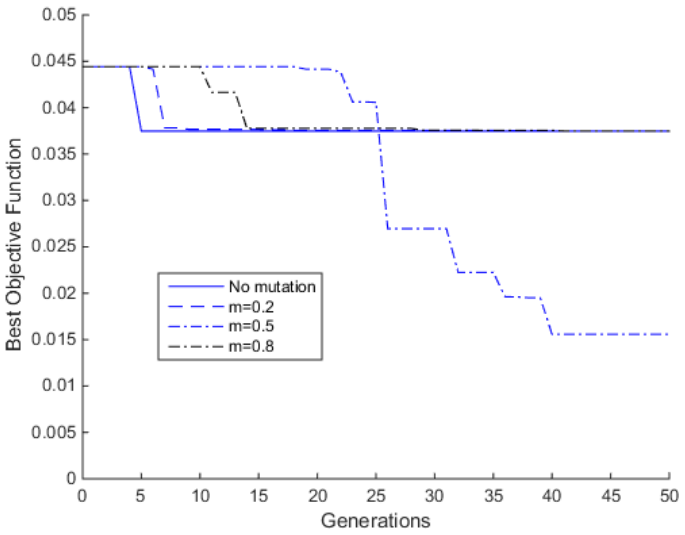


Figure A.6: Mutation rate influence in the convergence.

As it can be observed, both a small and a large mutation rate yield undesirable performances. A balance must be found.

Not only the mutation rate is a parameter in the algorithm, but also the mutation type can be considered. The "Replace" (uniform) method introduces random variation by first randomly choosing a design variable of a randomly selected design and reassigning it to a random valid value for that variable. The "Offset" (normal) mutator introduces random variation by adding a Gaussian random amount to a variable value. The random amount has a specified standard deviation.

Genetic Algorithm	
Number of Generations	50
Population Size	50
Crossover Rate	0.8
Crossover Points	1
Mutation Rate	0.2

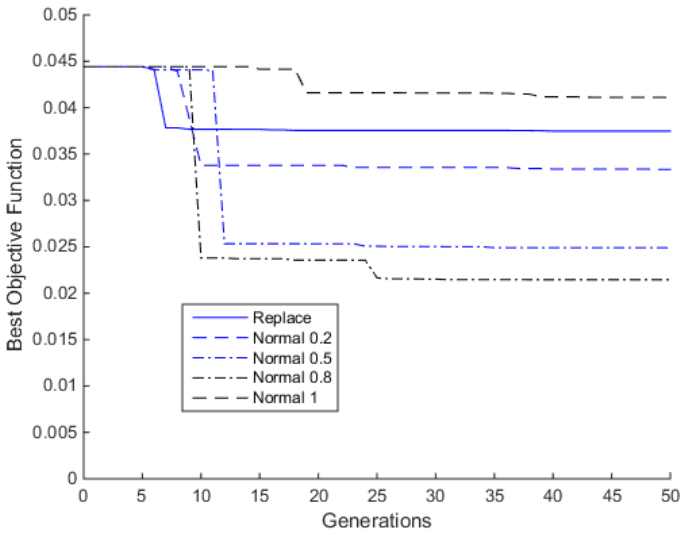


Figure A.7: Mutation type influence in the convergence.

Although the "Offset" method may produce better performance for different values of the standard deviation. However, observing inside the evolutionary process, it can be observed that the mutated values often go outside the bounds of the design space. Then, they need to be corrected, which makes the dependency of the mutation parameters confusing. Personally, for this work, a "Replace" method was used in most of the cases.

A.2 Surrogate-based local method

In order to fully understand how a surrogate-based local optimization model works, an example with the Rosenbrock function and a genetic algorithm is performed. The GA remains the same for the different cases:

Genetic Algorithm	
Number of Generations	100
Population Size	30
Crossover Rate	0.8
Crossover Points	1
Mutation Rate	0.2
Mutation Type	Replace Uniform

We will start observing the evolution of the optimum-search process. As a beginning, a sample size of 20 individuals will be sufficient, since the Rosenbrock problem comprises only 2 design variables. Also, a local approach for the correlation parameters may be used. No correction will initially be needed.

Surrogate-Based Method			
Method Type		Local	
Sampling Type		LHS	
Sample Size		20	
Iterations		20	
Surrogate Model	Type	Global - Kriging	
	Trend Function	Quadratic	
	Correlation Parameters	Global	
	Correlation Lengths	Internally Computed	
	Surrogate Correction	No	
	Trust Region	Initial Size	0.4
		Minimum Size	10^{-10}
		Contract Threshold	0.25
		Expand Threshold	0.75
		Contraction Factor	0.5
		Expansion Factor	2

As it can be seen in Fig. A.8, the optimization process easily reaches the optimum. Only 8 iterations (8 trust region generations) are needed for an accurate value of optimization. The goodness of the surrogate-based local model can be assessed by means of the trust region ratio, which is the actual decrease in the truth model divided by the predicted decrease in the truth model in the current trust region. In Fig. A.9, it can be observed such evolution. For iterations 1, 2, 3, 4, 8 and 10 there is an excellent accuracy. For excellent accuracy iterations, the model accepts the step and it iterates in the trust region interior, retaining the trust region size. In iteration 5, the model has a marginal accuracy and it starts reducing the trust region size. In iterations 6 and 11, the model has a satisfactory accuracy, retaining the trust region size. For iteration 7, 9, 12 and 13, SBLO has poor accuracy. For

these cases, it rejects the current step and it reduces the trust region size. For iteration 9, it has poor accuracy again, it rejects the current step and it reduces the trust region size.

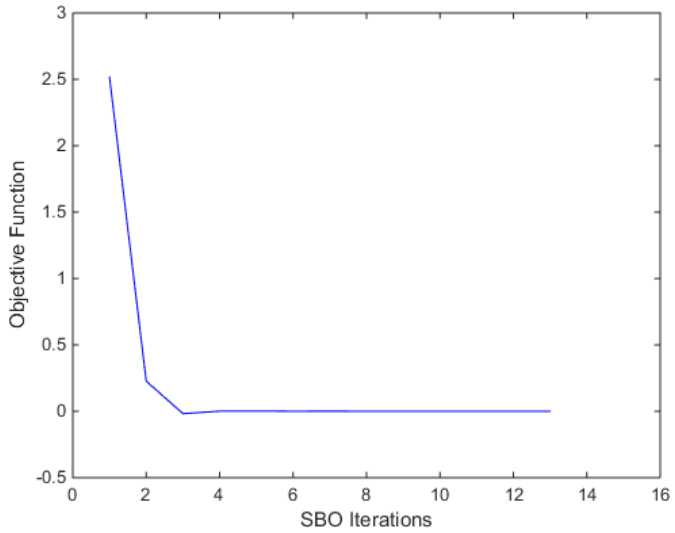


Figure A.8: SBLO process - Objective Function.

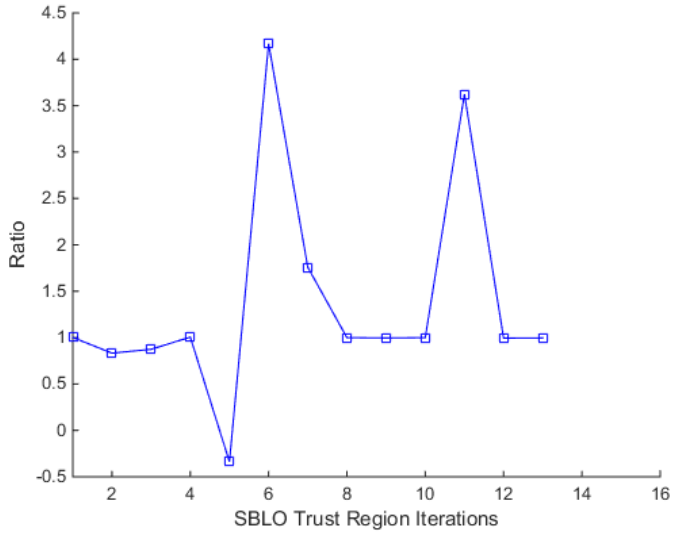
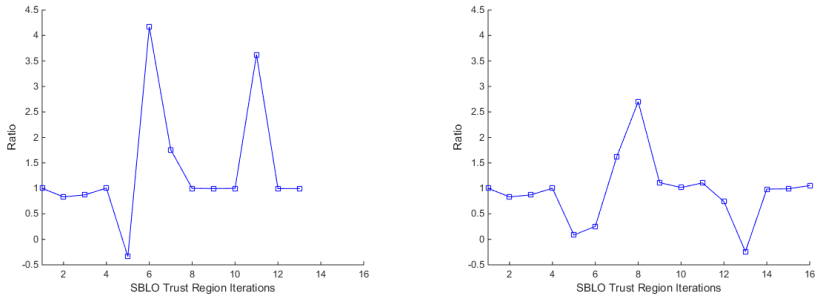


Figure A.9: SBLO process - Trust Region Ratio.

In this case, the initial trust region - design space ratio was set to be 40%.

A minimum size of the trust region was $1 * 10^{-10}$. The contract threshold was set to 0.25. This value has influence in the trust region iterations process.



(a) $c_{th} = 0.25$ (b) $c_{th} = 0.5$

Figure A.10: Contract threshold influence.

A.3 Surrogate-based global method

In order to fully understand how a surrogate-based global optimization model works, an example with the Rosenbrock function is performed.

A Kriging algorithm has been studied. In this first example, the following parameters have been established:

Surrogate-Based Method		
Method Type	Global	
Sampling Type	LHS	
Sample Size	20	
Iterations	20	
Surrogate Model	Type	Global - Kriging
	Trend Function	Quadratic
	Correlation Parameters	Global
	Correlation Lengths	Internally Computed
	Surrogate Correction	Zeroth-order additive

The Kriging model that is considered in this study takes a initial sample (generated by using the Latin Hypercube) and computes an initial set of parameters. Then, using the optimization method (genetic algorithm), the optimum of that given Kriging model is computed. Afterwards, the optimum is evaluated with the real function and the value is then added to the Kriging database, so that it is progressively updated. It is worth noticing that each update for Kriging is performed in the surroundings of the optimum, which makes the surrogate model more accurate in that area. In Fig. A.11 it can be observed the discrepancy of the surrogate model value with the real value, for each iteration (each update).

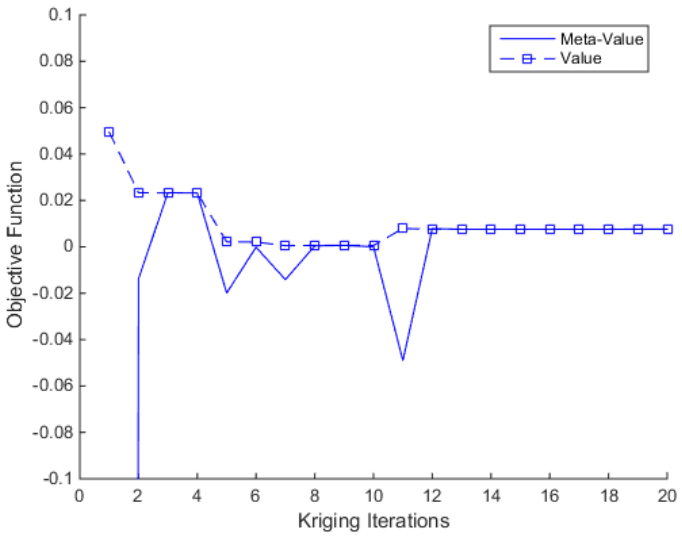


Figure A.11: Kriging Process.

In this case, after around 15 iterations, the surrogate model accurately simulates de exact value of the Rosenbrock function. This is very interesting, since the approximating function provided by the Kriging model is easily optimized using the genetic algorithm, with very little computational cost. The genetic algorithm used in this example comprises the following parameters:

Genetic Algorithm	
Number of Generations	100
Population Size	30
Crossover Rate	0.8
Crossover Points	1
Mutation Rate	0.2
Mutation Type	Replace Uniform

In Fig. A.12 it can be observed the influence of the number of the genetic algorithm generations on the optimum value, which is the best fitness function found in the given population. After 40 generations, a further improvement in the population cannot be observed.

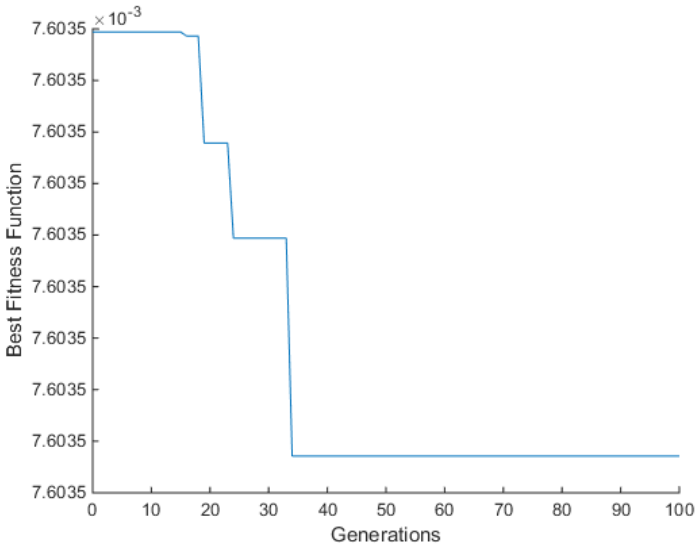


Figure A.12: Surrogate-Based Genetic Algorithm.

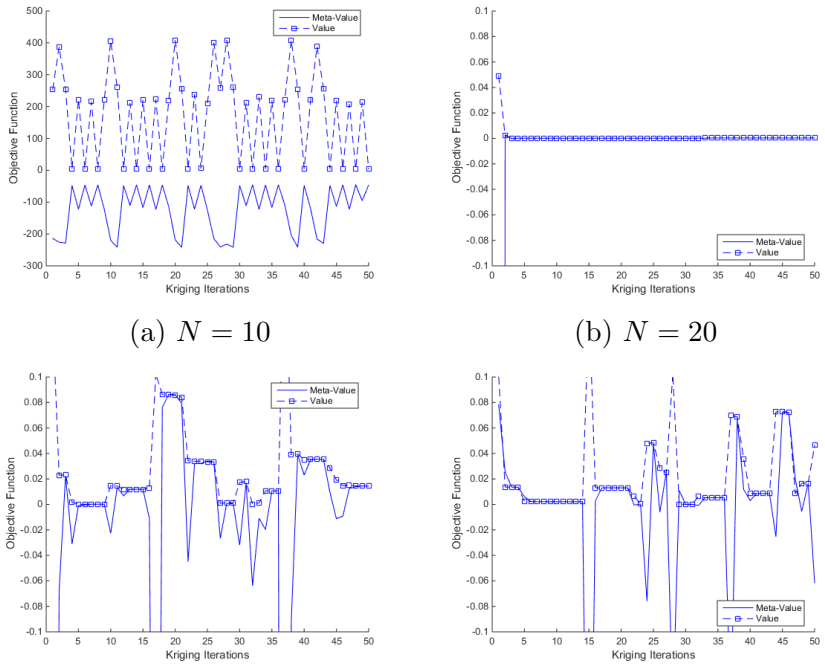
It is interesting to observe the evolution of the meta-function during the successive iterations. The trend basis function evolution can be observed in Table A.1. It has been checked that after the iteration #15 the trend basis functions do not change.

$\boldsymbol{\beta}^T g(x) = k + c_0 * x_0 + c_1 * x_1 + c_2 * x_0^2 + c_3 * x_1^2$

	k	c_0	c_1	c_2	c_3
1	-7.019	-1.98935	-4.01441	20.1636	1.96377
2	-2.05748	-0.708016	-1.73948	13.0873	0.5934
3	-2.14219	-0.650096	-1.76705	13.3625	0.596203
4	-2.14221	-0.650122	-1.76704	13.3624	0.596229
5	-2.1422	-0.650124	-1.76704	13.3625	0.596228
6	-2.05785	-0.715655	-1.73592	13.0671	0.598957
7	-2.05743	-0.714577	-1.73607	13.0683	0.598247
8	-2.05723	-0.714156	-1.73614	13.0687	0.597948
9	-2.06282	-0.688705	-1.74566	13.1377	0.584822
10	-2.06281	-0.688686	-1.74566	13.1377	0.584821
11	-2.06952	-0.68316	-1.74882	13.1644	0.583785
12	-2.07257	-0.678155	-1.75012	13.1783	0.582866
13	-2.07258	-0.678197	-1.75012	13.1782	0.582886
14	-2.07258	-0.67818	-1.75012	13.1782	0.582881
15	-2.07258	-0.678179	-1.75012	13.1783	0.582881
16	-2.07258	-0.678179	-1.75012	13.1783	0.582881
...	...	...	...	...	...

Table A.1: Trend basis functions for surrogate-based global optimization.

As an exercise, we observe the influence of the number of initial samples on the convergence of the surrogate model in Fig. A.13. Please notice the different scale of the y-axis for $N = 10$.



(a) $N = 10$ (b) $N = 20$

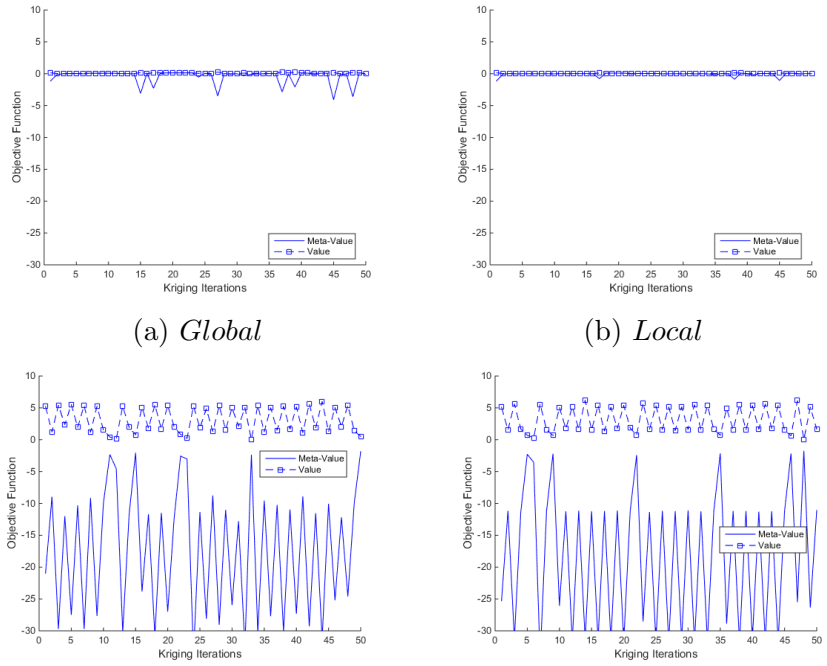
(c) $N = 30$ (d) $N = 40$

Figure A.13: Initial sample size influence in the convergence.

It can be concluded that it exists a minimum size of the initial sample. For lower values, the model is not triggered in the right direction and it will never get redirected. Also, for high values of the initial sample, the model finds

hard to adapt to the optimization surroundings, as it was initially defined in the whole area.

Depending how the data in the surrogate mathematical model is computed we can have the comparison in Fig. A.14. From this set of figures, it can be concluded that both DiRect and ConMin algorithms provide good solutions. For non-optimized methods, the meta-values differ too much from the real ones.



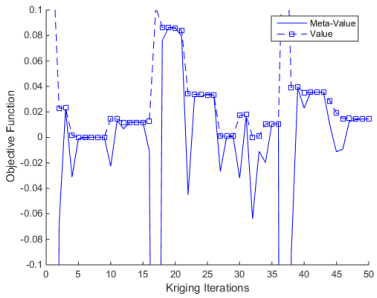
(a) *Global* (b) *Local*

(c) *Sampling* (d) *None*

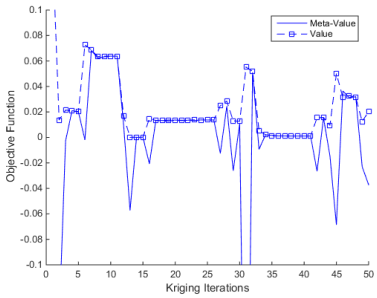
Figure A.14: Optimization type influence in the convergence (N=30).

Also, the trend functions can be computed using a quadratic approach, as they were obtained in all previous example. Let's take a look at other approach, as in Fig. A.15. "Reduced Quadratic" stands for a method that uses quadratic polynomials but only taking into account main effects.

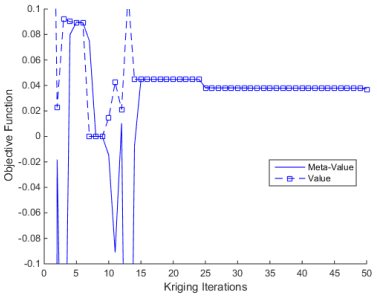
Kriging surrogate model supports the use of correction factors that improve the local accuracy. This factors will correct the approximation to match truth values, gradients and/or hessians. The correction can be performed by adding an offset or by the multiplication by a factor. If both methods are used, the approximation usually matches the previous correcting point as well. For an extensive discussion about correction in surrogate models, please refer to [21]. In Fig. A.16 one can get a rough idea of this application.



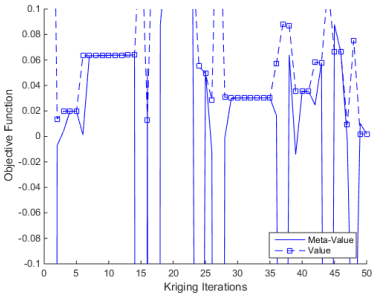
(a) Quadratic



(b) Reduced Quadratic

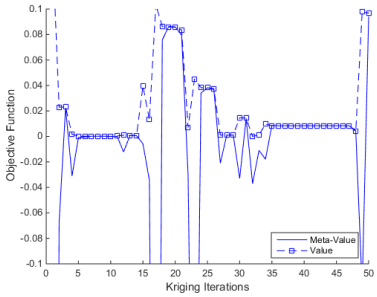


(c) Constant

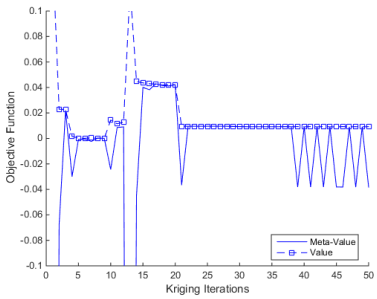


(d) Linear

Figure A.15: Trend function type influence in the convergence (N=30).



(a) Additive-Multiplicative Zeroth Order Correction



(b) No correction

Figure A.16: Correction influence (N=30).

Appendix B

Derivations

B.1 Efficiency

A deep analysis on the turbine efficiency must be followed to assess the goodness of the proposed solution. In order to do that, with permission of the reader, we recall the basics of fluidmechanics. For notation purposes we define the substantial derivative as the derivative following a particle:

$$\frac{d}{dt} = \frac{D}{Dt} = \frac{\partial}{\partial t} + \overline{v} \cdot \nabla \tag{B.1}$$

The extended integral of a fluid volume is used in fluidmechanics to evaluate the variation of a given property ϑ over a fixed volume:

$$\frac{d}{dt}\int_{V_f} \vartheta dV = \int_{V_f} \frac{\partial \vartheta}{\partial t} dV + \int_{S_f} \vartheta(\overline{v} \cdot \overline{n}) dS \tag{B.2}$$

If we evaluate the energy $\rho(e + \frac{v^2}{2})$ in Eq. B.2, we get the integral form of the energy conservation equation for a fluid volume:

$$\frac{d}{dt}\int_{V_f} \rho(e + \frac{v^2}{2}) dV = \int_{V_f} \frac{\partial \rho(e + \frac{v^2}{2})}{\partial t} dV + \int_{S_f} \rho(e + \frac{v^2}{2})(\overline{v} \cdot \overline{n}) dS \tag{B.3}$$

Taking into account the sources of energy variation on the fluid volume:

$$\begin{aligned} \frac{d}{dt}\int_{V_f} \rho(e + \frac{v^2}{2}) dV == & \int_{S_f} (\overline{n} \cdot \overline{\overline{\tau}})\overline{v} dS + \int_{V_f} \rho\overline{f_m} \cdot \overline{v} dV - \\ & \int_{S_f} (\overline{q} \cdot \overline{n}) dS + \int_{V_f} (Q_r + Q_q) dV \end{aligned} \tag{B.4}$$

The final expression using the divergence theorem and the conservation of energy in a stationary system ($\frac{d}{dt}\int_{V_f}\rho(e+\frac{v^2}{2})dV$=0) turns out:

$$\int_{V_f}(\frac{\partial\rho(e+\frac{v^2}{2})}{\partial t}+div\rho\overline{v}(e+\frac{v^2}{2})-div(\overline{\overline{\tau}}\cdot\overline{v})-\rho\overline{f_m}\overline{v}+ \\ +div\overline{q}-(Q_r+Q_q))dV=0 \tag{B.5}$$

Applying the mass conservation equation and rearranging we get to the differential form of the energy equation:

$$\rho\frac{De}{Dt}=div(\overline{\overline{\tau}}\cdot\overline{v})-\overline{v}\cdot div\overline{\overline{\tau}}-div\overline{q}+Q_r+Q_q \tag{B.6}$$

If we try to rewrite Eq. B.6 avoiding the tensor expression[1] of $\overline{\overline{\tau}}$ we get:

$$div(\overline{\overline{\tau}}\cdot\overline{v})=-p\frac{\partial v_j}{\partial x_j}+\tau'_{ij}\frac{\partial v_i}{\partial x_j}=-pdiv\overline{v}+\phi_v \tag{B.7}$$

where ϕ_v stands for those powers generated by the viscous effects[2].

With this definition we rewrite Eq. B.6 as follows:

$$\rho\frac{De}{Dt}=-pdiv\overline{v}+\phi_v-div\overline{q}+Q_r+Q_q \tag{B.8}$$

Now we seek to get the entropy equation in a differential form. We consider the first principle of thermodynamics and the entropy definition:

$$dq=Tds=de+pd(\frac{1}{\rho})=de-\frac{p}{\rho^2}d\rho \tag{B.9}$$

Therefore we can write the substantial derivative of the entropy as follows:

$$T\frac{Ds}{Dt}=\frac{De}{Dt}-\frac{p}{\rho^2}\frac{D\rho}{Dt} \tag{B.10}$$

Combining Eq. B.10 and Eq. B.8 we finally get:

$$\boxed{\rho T\frac{Ds}{Dt}=\phi_v-div\overline{q}+Q_r+Q_q} \tag{B.11}$$

[1]The stress tensor ($\overline{f_s}=\overline{n}\overline{\overline{\tau}}$) can be divided in the pressure and viscous terms: $\overline{\overline{\tau}}=-p\cdot\underline{\underline{I}}+\overline{\overline{\tau'}}$. The viscous term can be expressed following the Navier-Poisson law: $\tau'_{ij}=\mu(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i})+\lambda div\overline{v}\delta_{ij}$

[2]Rayleigh viscous stresses

In turbomachinery, and more specifically for turbines, we can define a manometric or hydraulic efficiency[3] of the turbine[4] as follows:

$$\eta_h^T = \frac{H_u}{H_n} \tag{B.12}$$

where H_u and H_n are the useful and net head respectively.

If we write $H_u = H_n - H_{i.l.}$, then the internal losses head $H_{i.l.}$ turns out:

$$H_{i.l.} = \frac{\widehat{\phi_v}}{gG} \tag{B.13}$$

And the final expression for the efficiency:

$$\boxed{\eta_h^T = 1 - \frac{\widehat{\phi_v}}{W + \widehat{\phi_v}}} \tag{B.14}$$

One could actually compute the viscous losses in terms of the entropy by integrating the former over the streamline. However, this process is complex and the results are not better than those that quantify the entropy production and the pressure losses.

[3]This efficiency does not take into account organic (mechanic) nor volumetric losses

[4]We extrapolate the definition for only this stage (stator) of the machine

Bibliography

[1] D. Pasquale, G. Persico, and S. Rebay, "Optimization of turbomachinery flow surfaces applying a cfd-based throughflow method," *Journal of Turbomachinery*, vol. 136, no. 3, p. 031013, 2014.

[2] M. Pini, D. Pasquale, G. Persico, and S. Rebay, "Adjoint method for shape optimization in real-gas flow applications," *Journal of Engineering for Gas Turbines and Power*, vol. 137, p. 032604, 2014.

[3] C. R. Reeves and J. E. Rowe, *Genetic Algorithms: Principles and Perspectives: A Guide to GA Theory*. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[4] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 5th ed., 2002.

[5] C. de Boor, "On calculating with b-splines," *Journal of Approximation Theory*, vol. 6, no. 1, pp. 50 – 62, 1972.

[6] L. Ramshaw, "Blossoming: a connect-the- dots approach to splines," 1987.

[7] E. Lee, "Choosing nodes in parametric curve interpolation," *Computer-Aided Design*, vol. 21, no. 6, pp. 363 – 370, 1989.

[8] J. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*. Springer My Copy UK, 1996.

[9] *ANSYS CFX, Solver Theory Guide*, November 2009.

[10] *ANSYS CFX, Solver User Manual*, November 2009.

[11] A. Martínez, *Mecánica de fluidos*. Ediciones Paraninfo. S.A., 2006.

[12] J. Eddy and K. Lewis, "Effective generation of pareto sets using genetic programming," 2001.

[13] T. Simpson, J. Poplinski, P. N. Koch, and J. Allen, "Metamodels for computer-based engineering design: Survey and recommendations," *Engineering with Computers*, vol. 17, no. 2, pp. 129–150, 2001.

[14] A. Lhhali, "Surrogate based optimization using kriging based approximation," *VU BA paper*.

[15] J. D. Martin and T. W. Simpson, "Use of kriging models to approximate deterministic computer models," *Aiaa Journal*, vol. 43, pp. 853–863, 2005.

[16] B. Adams, M. Ebeida, M. Eldred, J. Jakeman, L. Swiler, J. Stephens, D. Vigil, T. Wildey, W. Bohnhoff, K. Dalbey, J. Eddy, K. Hu, L. Bauman, and P. Hough, "Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.0 user's manual," in *Sandia Technical Report*, July 2014.

[17] A. Giunta, L. Swiler, S. Brown, M. Eldred, M. Richards, and E. Cyr, "The surfpack software library for surrogate modeling of sparse irregularly spaced multidimensional data," *AIAA-2006-7049*.

[18] J. Denton, "Loss mechanisms in turbomachines," *Journal of Turbomachinery*, vol. 115, no. 4, pp. 621 – 656, 1993.

[19] D. Pasquale, A. Ghidoni, and S. Rebay, "Shape optimization of an organic rankine cycle radial turbine nozzle," *Journal of Engineering for Gas Turbines and Power*, vol. 135, no. 4, 2013.

[20] P. Colonna and S. Rebay, "Numerical simulation of dense gas flows on unstructured grids with an implicit high resolution upwind euler solver," *International Journal for Numerical Methods in Fluids*, vol. 46, no. 7, pp. 735 – 765, 2004.

[21] M. Eldred, A. Giunta, S. Collis, N. Alexandrov, and R. Lewis, "Second-order corrections for surrogate-based optimization with model hierarchies," in *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY,, Aug*, 2004.

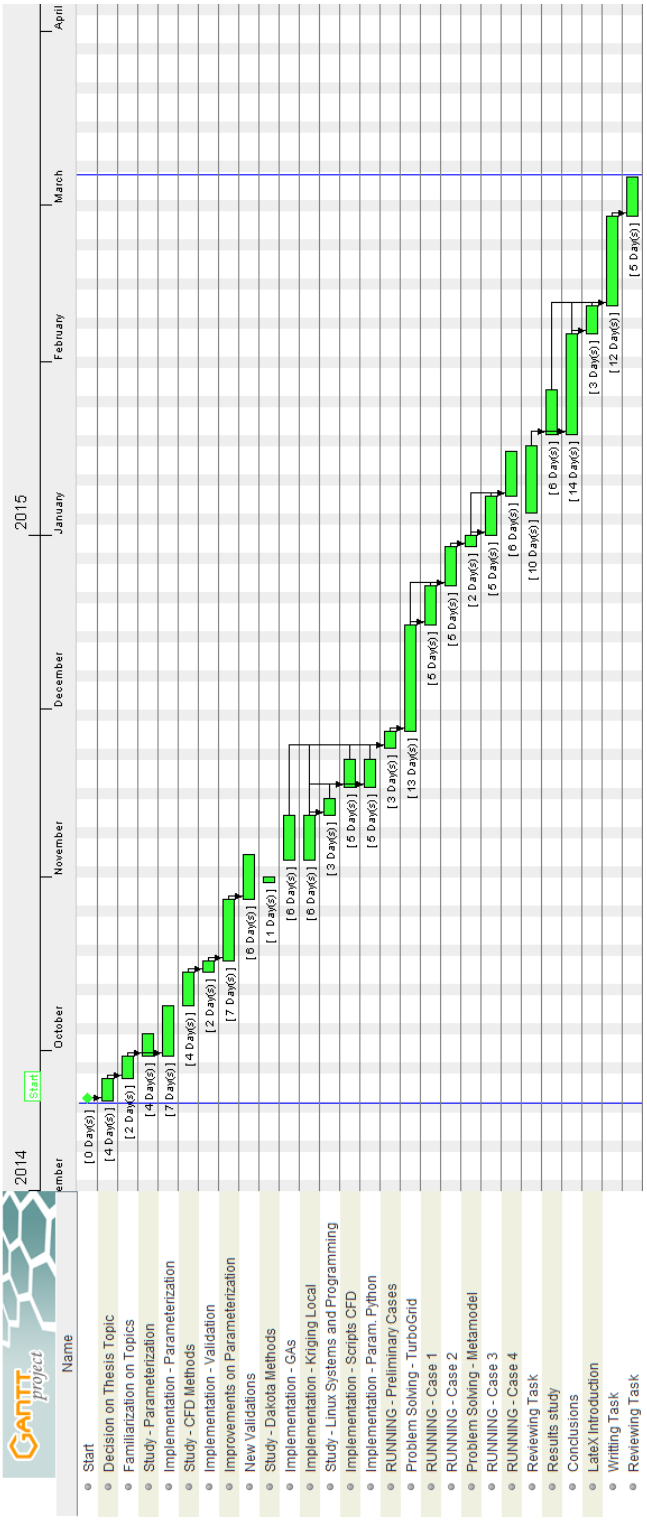
Información PFC

Presupuesto

Este proyecto ha sido financiado con una beca Erasmus.es del Ministerio de Educación, Cultura y Deporte, además de una significante ayuda del programa Atlantis, de la Unión Europea. Los recursos computacionales y de mantenimiento empleados en Politecnico di Milano han sido proporcionados tanto por la universidad como por el grupo de investigación de turbomáquinas. En la siguiente tabla se especifican los costes totales del proyecto, teniendo en cuenta una estancia de 6 meses.

Concepto	Precio en EUR
Licencia académica ANSYS	3,000
Mantenimiento Cluster	3,000
Recursos Humanos	11,000
TOTAL	**17,000**

Diagrama de Gantt



Escuela de Acogida para PFC

Este proyecto ha sido realizado con la colaboración del grupo "Fluidodinamica delle turbomacchine", perteneciente al departamento de energía del Politecnico di Milano. Esta prestigiosa universidad italiana ubicada en Milan, región de Lombardia, fue etablecida en 1863 y actualmente se encuentra en la posición 28 del ranking mundial de universidades de ingeniería y tecnología.

El grupo de Fluidomecánica de Turbomáquinas tiene su propio laboratorio y es uno de los más notables en el area de la ingeniería energética. Desarrolla su actividad en el campo de la fluidodinámica numérica y experimental, con atención particular a los aspectos inherentes de las turbomáquinas generadoras y motoras, tanto con fluidos compresibles como incompresibles. Concretamente, el grupo de fluidodinámica numérica ha desarrollado sus propios códigos de cálculo para la descrición del campo fluido y la evaluación de las prestaciones de los diferentes componentes de la turbomáquina. Los numerosos contratos de investigación llevados a cabo tanto para la industria italiana como industrias internacionales son testimonio de la gran competencia de sus proyectos.

POLITECNICO
DI MILANO

Escuela de Acogida para Realización de Estudios

Durante el curso académico 2013-2014 recibí la beca EAGLES, perteneciente al programa Atlantis, de intercambio académico de doble título entre diferentes instituciones europeas y americanas: Universidad Politécnica de Madrid (Madrid, España), Politecnico di Milano (Milán, Italia), Drexel University (Philadelphia, EEUU) y University of Connecticut (Connecticut, EEUU). Específicamente en mi caso, he realizado un año de estudios de Máster en Drexel University y una estancia de investigación en Politecnico di Milano.

La beca EAGLES (Engineers As Global Leaders for Energy Sustainability) es un prestigioso programa transatlántico de doble título en el área de sostenibilidad energética. El objetivo del consorcio EAGLES es la creación de ingenieros capaces de enfrentarse a los problemas globales actuales de sostenibilidad energética. A través de este programa, los estudiantes adquieren amplias habilidades en análisis científico, computación y modelado matemático, además de un profundo conocimiento de los complejos desafios en la producción energética actual. Además, el contenido académico se complementa con un desarrollo personal que prepara a los futuros ingenieros como líderes globales en el campo de la ingeniería. Para lograr los objetivos anteriores, el proyecto es financiado por la Unión Europea y el Departamento de Educación de los Estados Unidos, a través del programa Atlantis.

EAGLES ofrece:

1. Una experiencia internacional significativa en varias instituciones europeas y americanas.

2. Una oportunidad única para internacionalizar las credenciales académicas.

3. Financiación para estudiar en Europa y Estados Unidos durante un año, consiguiendo un doble título en sostenibilidad energética.

4. Una oportunidad para aprender y mejorar un idioma adicional (además del inglés).

Drexel University es una institución americana fundada en 1891 y ubicada en Philadelphia, en el estado de Pennsylvania, en la costa este de los Estados Unidos. Drexel es conocida por sus programas de "Educación Cooperativa" de universidad-empresa y sus actividades de investigación, con numerosos proyectos y financiación pública y privada. Durante mi estancia en Drexel University (de Septiembre 2013 a Junio 2014), me especialicé en métodos matemáticos, fluidomecánica, transferencia de calor

avanzada y física de plasmas. Estos cursos pertenecen al departamento de ingeniería mecánica, del cual recibí finalmente el título de Master of Science. Al mismo tiempo, para complementar la educación recibida en clase, comencé un proyecto de investigación en computación y análisis de descargas de plasma en fluidos supercríticos, lo que me permitió poner en práctica los conocimientos aprendidos en clase y preparame para el presente trabajo.

Resumen del Proyecto

El proyecto aquí descrito pertenece al programa EAGLES, de energía y sostenibilidad, enmarcado en el convenio Atlantis de intercambio académico entre instituciones europeas y americanas. Como aspecto importante se encuentra la internacionalización de las habilidades académicas y de investigación. Como consecuencia, se ha realizado enteramente en inglés, y a continuación se proporciona un resumen explicativo en español de la metodología que se ha llevado a cabo, así como de los resultados y conclusiones más generales. Para información precisa sobre los procedimientos de optimización y métodos empleados, por favor consúltese el documento en la lengua original.

Introducción

Hoy en día, los métodos numéricos y computacionales son una rama muy importante en cualquier campo de la ingeniería. Sobre todo en aquellos casos donde las soluciones análiticas sean inaccesibles, ya sea por complejidad del problema matemático o la imposibilidad de realización de experimentos.

De manera específica, el amplio campo de las turbomáquinas está en constante desarrollo. Desde el diseño preliminar de la turbomáquina hasta la definición de la forma exacta de los álabes en cada etapa. Muchos estudios se han llevado a cabo para mejorar métodos de optimización y ayudar en el diseño de turbomáquinas. Por otro lado los métodos CFD son ampliamente utilizados. Sin embargo, una herramienta de optimización automática de las geometrías de los álabes con un amplio campo de aplicación aún no ha sido desarrollada.

En este estudio se propone la creación de un algoritmo de optimización para álabes de turbomáquinas que de manera automática, mediante la estrategia evolutiva y metamodelos, ayuda en el diseño aerodinámico de los álabes.

De manera específica para este trabajo, se propone la optimización de la geometría del estátor de la primera etapa de una turbina transónica de un ciclo orgánico de Rankine (ORC). Como resultado se demuestra que la optimización de este tipo de turbinas mediante la estrategia evolutiva es muy efectiva. Además, un análisis de diferentes metamodelos permite evaluar su eficacia, y así ayudar al diseño de álabes de turbinas de la manera más automática posible.

Metodología

El algoritmo se caracteriza por la implementación conjunta de cuatro módulos:

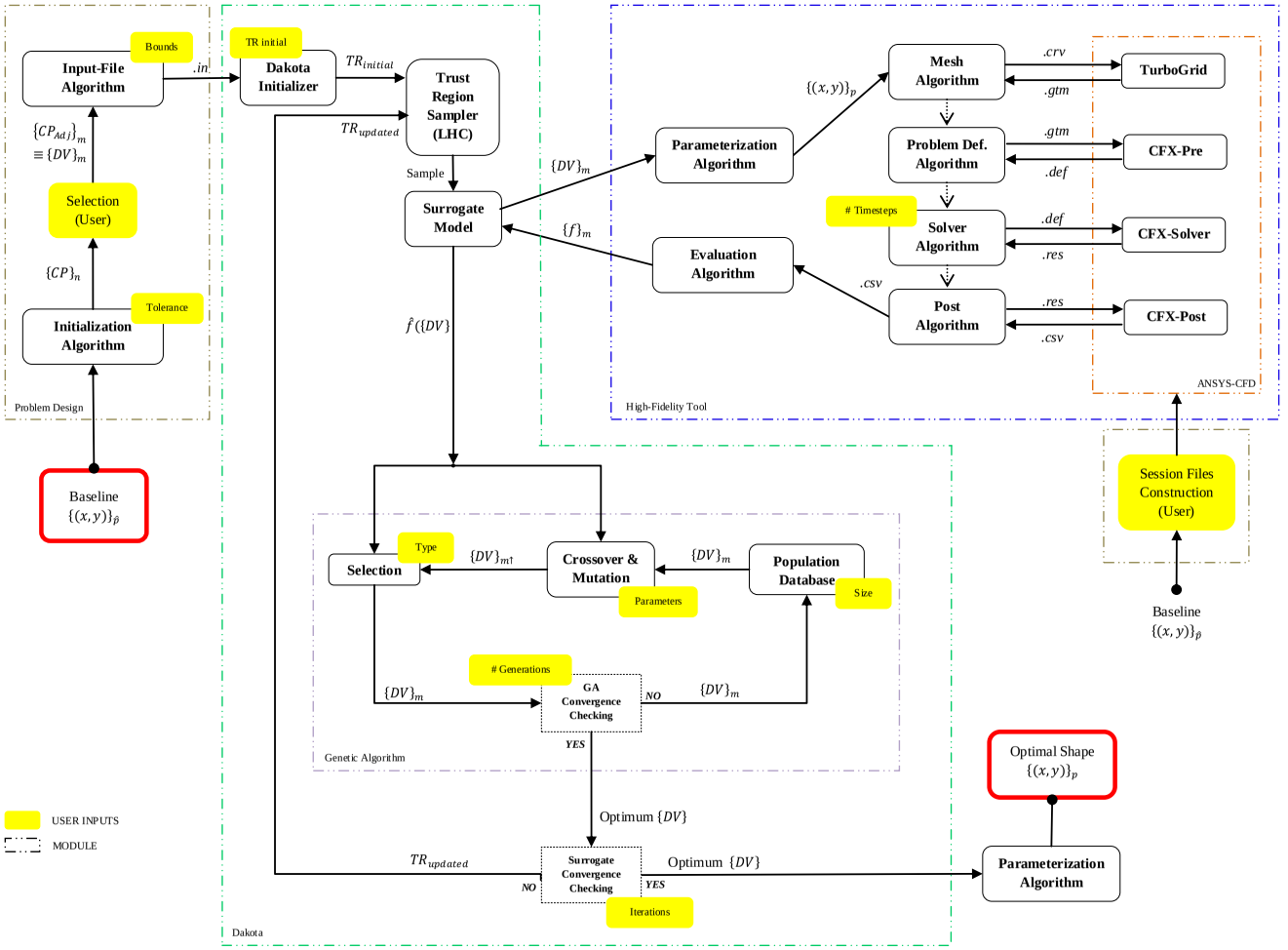
1. Algoritmos de Parameterización
2. Códigos de CFD
3. Metamodelos
4. Algoritmos Genéticos.

El método es de naturaleza sencilla, aunque su implementación conlleva cierta complejidad. La finalidad de la herramienta es la optimización de las características físicas del álabe. Para ello, una técnica de optimización debe ser seleccionada. En este trabajo se ha demostrado que los algoritmos genéticos, de naturaleza heurística, ofrecen una potente alternativa a los métodos del gradiente. Estos algoritmos, como se verá a continuación permiten crear una herramienta de carácter no intrusivo, empleando la mínima información física del problema. Sólo será necesario el valor de la función objetivo, obtenida mediante el uso de un código de simulación computacional (CFD) de alta fidelidad. El acople de algoritmos genéticos y códigos CFD, realizado de forma directa, conlleva un alto coste computacional, por lo tanto se hace necesario el empleo de metamodelos que simplifiquen la evaluación de la función objetivo durante el proceso de optimización. Por último y no menos importante, las técnicas de parameterización caracterizan la herramienta, pues simbolizarán el enlace entre el fenómeno físico (geometría que conllevará un cierto flujo fluido) y la optimización matemática (algoritmo genético). A continuación se explicarán cada uno de los módulos individuales y su implementación. Por favor, para una información más detallada y completa, véase el documento en la lengua original.

En la siguiente página se puede ver un diagrama completo de la herramienta optimización.

Shape-Optimization Scheme

Pablo Rodríguez Fernández



Algoritmos de Parameterización

Por su naturaleza automática y computacional, como entrada al código se proporcionan los puntos del álabe original. Dado que el algoritmo de optimización tiene necesariamente que trabajar con un número limitado de variables de diseño, tanto un método de parameterización como de interpolación debe ser desarrollado.

En este estudio, se trabajará con B-Splines, un tipo especial de curvas que tienen en cuenta efectos locales y globales de la geometría a través de un cierto número de puntos, llamados puntos de control. Estos serán considerados variables de diseño, o entrada, para el algoritmo genético de optimización.

Para nuestra aplicación, una B-Spline puede escribirse de la forma:

$$x(u) = \sum_{j=0}^{L} d_j N_j^n(u) \tag{5.15}$$

donde d_j son los puntos de control y las funciones $N_j^n(u)$ se denominan bases de B-Spline. Estas bases pueden hallarse de manera recursiva siguiendo la fórmula:

$$N_j^k(u) = \frac{u - u_{j-1}}{u_{j+k-1} - u_{j-1}} N_j^{k-1}(u) + \frac{u_{j+k} - u}{u_{j+k} - u_j} N_{j+1}^{k-1}(u); \tag{5.16}$$

$$N_j^0(u) = \begin{cases} 0 & if \quad u_{j-1} \leq u < u_i, \\ 1 & if \quad else \end{cases} \tag{5.17}$$

Las B-Splines aquí descritas de forma simplificada se caracterizan por una alta controlabilidad. Además de los puntos de control, las bases de B-Spline necesitan como parámetro una sequencia de nudos que caracterizarán el soporte local. Estas curvas pueden ser de diferentes grados. En este estudio trabajaremos con curvas de tercer grado, pues se ha demostrado que estas curvas pueden definir perfectamente la forma de los álabes que se estudiarán.

Un ejemplo de B-Spline generada a partir de unos puntos de control se muestra a continuación:

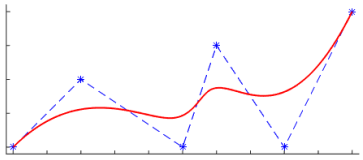


Figure 5.1: B-Spline (n=3).

En la siguiente figura se puede observar cómo es posible reproducir la geometría aerodinámica de un álabe mediante el uso de B-Splines de tercer grado. En este ejemplo se observa únicamente la parte de succión:

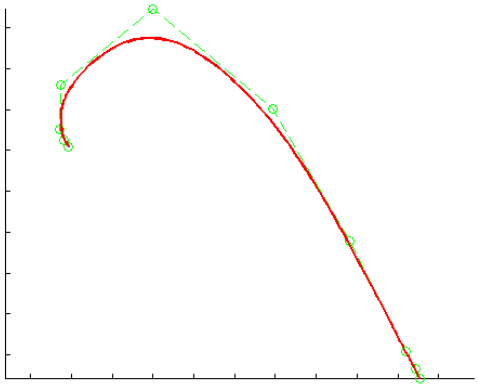


Figure 5.2: Ejemplo de parametrización.

Además del algoritmo de parameterización, el cual genera la curva B-Spline a partir de unos determinados puntos de control, necesitamos desarrollar un algoritmo de interpolación que lleve a cabo el proceso contrario: a partir de los puntos de la curva ya especificados, generar unos puntos de control que regeneren la forma original dentro de unas tolerancias.

Para la interpolación llevaremos a cabo un ajuste de mínimos cuadrados, que puede definirse como la minimización de la siguiente función:

$$f(x) = \sum_{i=0}^{P} \|p_i - x(w_i)\| \tag{5.18}$$

Teniendo en cuenta la definición de B-Spline en Eq. 5.15, podemos escribir esta última expresión como:

$$f(\{d_j\}_0^L) = \sum_{i=0}^{P} \|p_i - \sum_{j=0}^{L} d_j N_j^n(w_i)\| \tag{5.19}$$

De esta expresión podemos extraer un sistema de minimización por mínimos cuadrados dado por:

$$\sum_{j=0}^{L} d_j \sum_{i=0}^{P} N_j^n(w_i) N_k^n(w_i) = \sum_{i=0}^{P} p_i N_k^n(w_i); \tag{5.20}$$

Este sistema lineal puede interpretarse como un sistema matricial, resoluble mediante descomposición de Cholesky, adecuada para nuestro caso:

$$\begin{gathered} A \cdot x = B \\ L \cdot L^* \cdot x = B \\ L \cdot y = B \Rightarrow y \\ L^* \cdot x = y \Rightarrow x \end{gathered} \tag{5.21}$$

La implementación del algoritmo de parameterización y del algoritmo de interpolación permitirá la reproducción exacta de la geometría a partir de un número limitado de puntos, que serán las variables de diseño de la herramienta de optimización.

En el siguiente gráfico se representa el algoritmo de interpolación mediante el cual se inicializa la herramienta de optimización. A partir de los puntos del álabe original se generan los puntos de control adecuados para una cierta tolerancia (para información detallada sobre cada uno de los bloques, por favor véase la versión en lengua original):

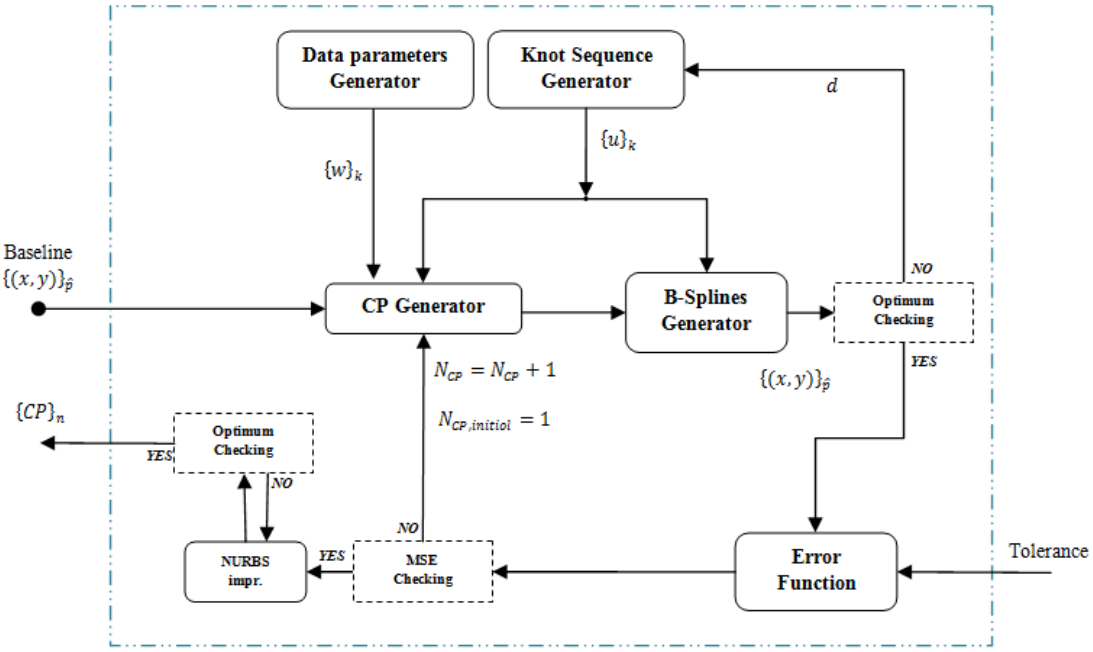


Figure 5.3: Algoritmo de Inicialización.

Códigos CFD

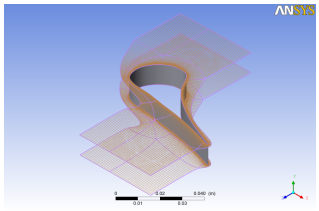
Las ecuaciones de la fluidomecánica tienen solución para un limitado número de aplicaciones, normalmente con fluidos y consideraciones muy simplificados. En ingeniería y diseño estos casos son muy excasos y tradicionalmente se recurría a datos experimentales. Según evoluciona la ingeniería, los casos son más y más complejos, lo que llevó a la invención de códigos computacionales capaces de simular los casos reales, y en cierta medida sustituir la necesidad de costosos y complejos experimentos. La Mecánica de Fluidos Computacional (CFD por sus siglas en inglés) es hoy en día un campo muy amplio y que es extensamente usado en ingeniería de diseõ, análisis e investigación.

Los métodos CFD son, sin embargo, aproximaciones a la realidad y cuyos resultados deben ser considerados con cautela. Una solución computacional a un problema ingenieril contiene aproximaciones en el modelo empleado, aproximaciones en el método de resolución y aproximaciones en el proceso iterativo de convergencia. Para minimizar estos errores, los códigos CFD más potentes cuentan con técnicas de discretización y solución muy avanzadas, dejando al usuario la tarea de minimizar los errores por modelado y convergencia.

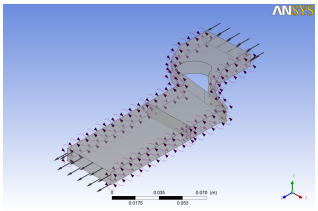
En este estudio se empleará un código de volúmenes finitos desarrollado

por ANSYS, una empresa de ingeniería a nivel mundial líder en software de simulación. Se hará uso de una herramienta específica para tubomáquinas, permitiendo definir de manera automática la malla de discretización espacial, específicamente creada para álabes de turbinas.

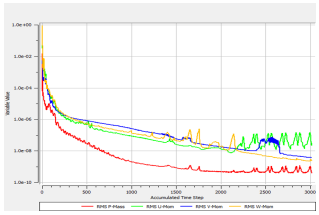
El proceso de simulación conlleva 4 etapas. En primer lugar, el diseño de la malla es un proceso crucial, ya que se tratará de mallas muy densas para asegurar la fiabilidad. Además, dada la naturaleza automática de la herramienta, se empleará un generador de creación de mallas optimizadas diseñado por ANSYS (ATM Meshing and Topology). En segundo lugar, el pre-procesamiento del problema consiste en la asignación de las condiciones de contorno y de fluido. Posteriormente los datos de la malla junto con las condiciones son llevadas al Solver, el cual itera numéricamente y obtiene una solución, cuyo post-proceso provee de los resultados requeridos para el problema.



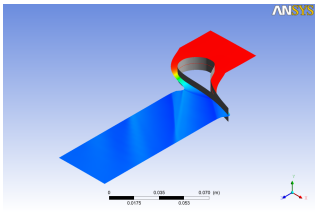
(a) TurboGrid



(b) Pre-Processor



(c) Solver (Residuos)



(d) Post-Processor

Figure 5.4: Ejemplo de simulación CFD.

Algoritmos genéticos

Para el proceso de optimización se empleará una estrategia evolutiva. Los algoritmos evolutivos (EAs por sus siglas en inglés) son una serie de métodos heurísticos desarrollados en las últimas décadas y de amplio uso en aquellos

casos en los que la complejidad del problema no permita la obtención de datos de gradiente o donde las funciones a optimizar no sean regulares. Los algoritmos genéticos (GAs) pertenecen a esta categoría y su funcionamiento está basado en la teoría de la evolución y selección natural neo-Darwiniana.

Los algoritmos genéticos permiten trabajar con funciones objetivo oscilantes e irregulares, así como con problemas multiobjectivo. Éstos permiten identificar de manera sencilla un gran número de soluciones optimizadas y pueden ser adaptados a problemas muy diferentes con una mínima intrusión en el código. Es por ello que se emplearán en la herramienta automática de optimización, lo que nos permitirá aplicarla a problemas muy dispares sin necesidad de adentrarnos en el código.

En ese estudio se empleará la librería JEGA (Java Engine for Genetic Algorithms), implementada en el entorno de optimización Dakota. JEGA interpreta cada miembro de la población (conjunto de puntos de control que definen una geometría) como un entero binario de 32 bits, lo que permite interpretarlo como un conjunto de cromosomas que caracterizan al individuo. La librería tiene una amplia variabilidad en los parámetros de mutación, cruce, selección y poblaciones, por lo que definiremos los parámetros deseados dependiendo de la aplicación.

Metamodelos

El algoritmo genético de optimización basa su funcionamiento en la evaluación de la función objetivo (función a optimizar) para cada miembro de la población en cada una de las generaciones (iteraciones del algoritmo genético). Esto conlleva un gran número de evaluaciones CFD, lo que implica un gran coste computacional. Para evitarlo, en este trabajo se hace uso de los llamados metamodelos, modelos sustitutivos o superficies de respuesta, dependiendo del autor.

Estos métodos se asemejan a técnicas de regresión, en las que a partir de una muestra se aproxima una función de estimación. El metamodelo actuará de la misma forma, creando una muestra aleatoria en el espacio de diseño (mediante la técnica LHC) y generando una función matemática que intenta simular el comportamiento de la función objectivo real, o sea la simulación CFD. La muestra aletoria se genera en cada iteración del metamodelo y se hace cada vez más precisa según avanza el proceso. De esta forma, el coste computacional se reduce a la evaluación de la muestra, en lugar de cada miembro de la población de cada generación.

El modelo matemático, cuyos parámetros se estimarán, se conoce como modelo de Kriging o modelo Gaussiano y puede describirse como:

$$\tilde{f}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{\beta} + \boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{R}^{-1}(\boldsymbol{f} - \boldsymbol{G}\boldsymbol{\beta}) \tag{5.22}$$

donde $\boldsymbol{x}$ es el punto actual , compuesto por n variables de diseño; $\boldsymbol{g(x)}$ es el vector de evaluaciones de las bases de la función de tendencia; $\boldsymbol{\beta}$ en el vector de estimaciones de mínimos cuadrados de los coeficientes de la función de tendencia; $\boldsymbol{r(x)}$ en el vector de correlación; $\boldsymbol{R}$ es la matriz de correlación; $\boldsymbol{f}$ es el vector de respuestas; y $\boldsymbol{G}$ es la matriz de evaluaciones de la función de tendencia.

En este estudio se empleará un *proceso de optimización basado en un metamodelo local*. Éste puede describirse como sigue:

1. Generación de región de confianza en el espacio de diseño.

2. Evaluación CFD de la muestra.

3. Estimación de parámetros matemáticos de Kriging.

4. Optimización del modelo de Kriging y evaluación del óptimo mediante CFD.

5. Generación de nueva región de confianza en torno al óptimo encontrado.

Resultados y Conclusiones

Como aplicación de la herramienta de optimización, se llevará a cabo el rediseño del álabe del estátor de la primera etapa de una turbina supersónica de un ciclo orgánico de Rankine (ORC por sus siglas en inglés). Al pasar por la garganta del álabe, el flujo se vuelve supersónico y se generan ondas de choque que provocan una disminución en la eficiencia de la turbina en conjunto. El objetivo de la optimización consistirá en la reducción de la intensidad de las ondas de choque. Analizando el flujo supersónico aguas abajo del estátor tenemos:

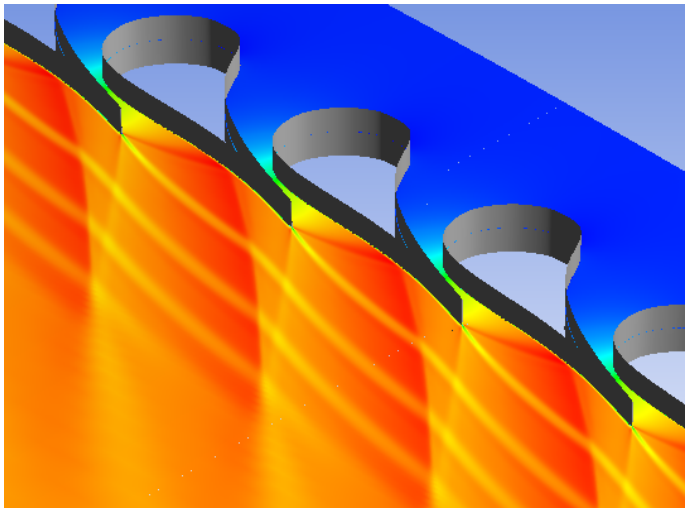


Figure 5.5: Distribución del Número de Mach tridimensional.

Para la herramienta de optimización consideraremos un espacio de diseño entorno a la geometría de partida. En primer lugar, deberemos realizar la interpolación de la geometría. Para ello, establecemos una tolerancia sobre los resultados original y obtenemos la siguiente distribución de puntos de control:

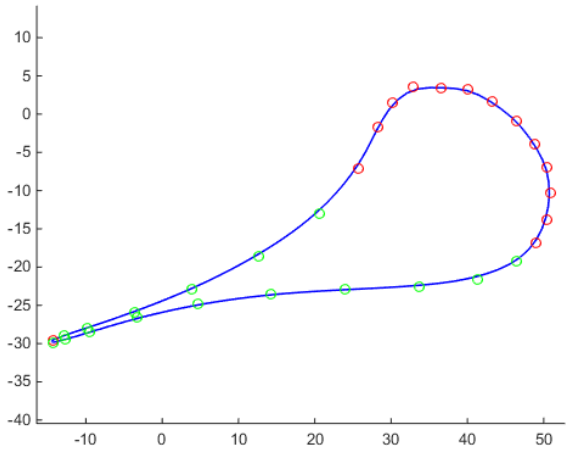


Figure 5.6: División en parte fija (rojo) y parte móvil (verde).

El espacio de diseño se establece imponiendo los límites inferior y superior a los puntos de control móviles, resultado:

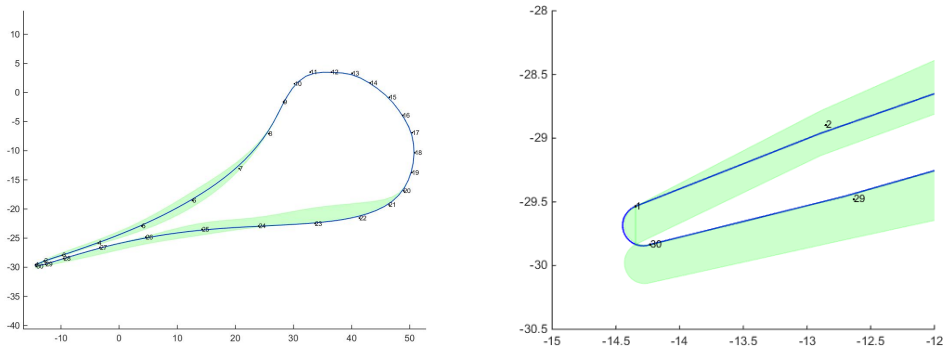


Figure 5.7: Espacio de Diseño.

Para el proceso de optimización, dado que se busca la reducción de las ondas de choque, se optará por maximizar la uniformidad del flujo aguas abajo. Una disminución de la desviación típica de la distribución circunferencial de la presión media cuerda axial aguas abajo del álabe tendrá el efecto deseado. A continuación se muestra el resultado final, en el cual "*Pini M. (2015)*" hace referencia a los resultados publicados con anterioridad sobre el mismo álabe, aplicando el método númerico del Adjunto, un método basado en el gradiente que ofreció resultados muy interesantes.

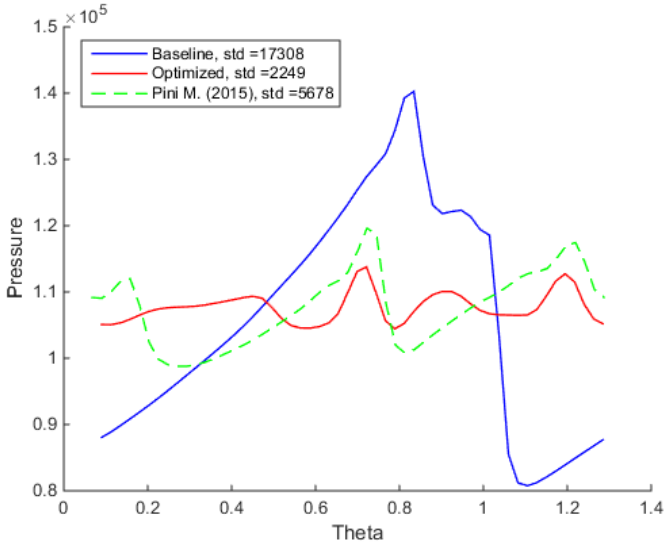
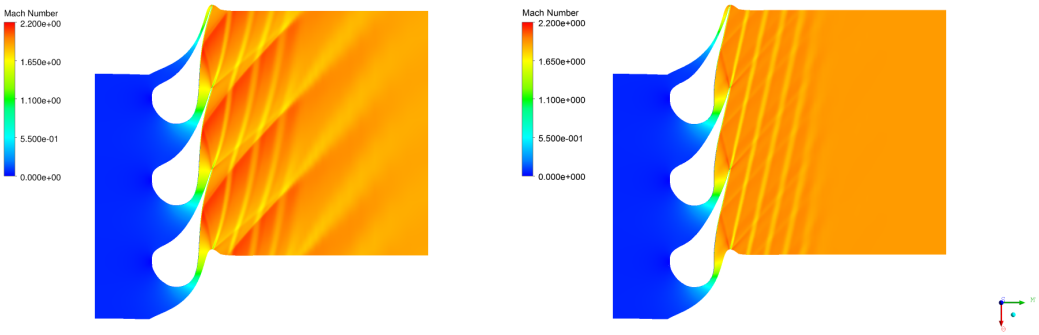


Figure 5.8: Distribución de la presión.

Como puede observarse, los resultados son muy prometedores, pues la

desviación típica se reduce considerablemente y mejora los resultados obtenidos por *Pini M. (2015)*. Para observar su efecto sobre la uniformidad del flujo se tiene:



(a) Original (b) Optimizado

Figure 5.9: Distribución del Número de Mach.

Se puede observar cómo el método ofrece una manera sencilla y muy eficaz de optimización, obteniendo resultados muy buenos con la mínima información física del problema y con un coste computacional moderado. Las ondas de choque producidas por el flujo transónico son muy reducidas en el nuevo caso, con una variación mínima en la geometría. Esto provoca una disminución en las pérdidas, las cuales pueden evaluarse mediante el coefficiente de pérdidas de presión (Fig. 5.10).

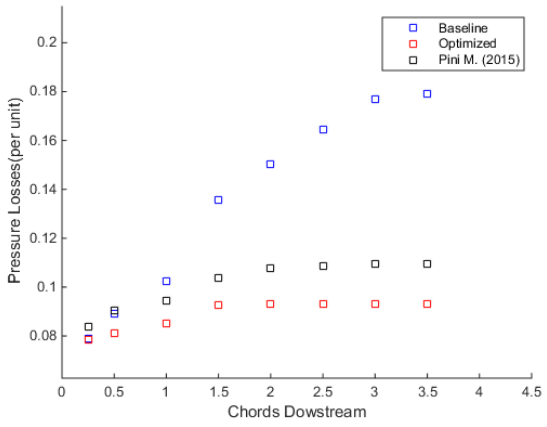


Figure 5.10: Coeficiente de Pérdidas de Presión.