TÉCNICO LISBOA

Supersonic and Transonic Adjoint-based Optimization of Airfoils

João Pedro Bernardes Lourenço

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. Fernando José Parracho Lau
Dr. Frederico José Prata Rente Reis Afonso

Examination Committee

Chairperson: Prof. Filipe Szolnoky Ramos Pinto Cunha
Supervisor: Prof. Fernando José Parracho Lau
Member of the Committee: Prof. José Manuel da Silva Chaves Ribeiro Pereira

November 2018

“Not everything that counts can be counted, and
not everything that can be counted counts.”
Albert Einstein

This page has been left intentionally blank.

Acknowledgments

I sincerely appreciated the guidance and helpful advice of my supervisor Prof. Fernando Lau and would like to thank him for making this work possible and for introducing me to the very interesting world of aerodynamic optimisation. I am also indebted to Dr. Frederico Afonso for his availability and all the guidance, explanations, knowledge and friendship. I also thank the past and present members of the aerospace office in Instituto Superior Tecnico, special to Prof. Hugo Policarpo e Simão Rodrigues for all the friendship, help and provided happy moments. A special acknowledgment must go to my parents and brother for all the love, for support me all the time and never letting me go down. To all my colleagues Artur Vasconcelos, Ricardo Xavier, Ricardo Couto, Diogo Neves, Roberto Valldosera, Rui Assis, Frederico Caldas, Vitor Lopes and so many others for their friendship and for all the good moments we shared. Finally, I would like to thank my girlfriend, Raquel Pacheco, for always supporting me through the difficult moments.

This page has been left intentionally blank.

Resumo

Uma metodologia de otimização de formas aerodinâmica de alta fidelidade, robusta e eficiente, para otimizar perfis aerodinâmicos foi desenvolvida, recorrendo a plataformas de código aberto para simulações computacionais de dinâmica de fluidos. Os códigos integrados foram: *SALOME* para geração de malha, SU^2 para otimização da forma aerodinâmica e *ParaView* para o pós-processamento. Foram considerados dois métodos de parametrização geométrica: o método *Hicks-Henne* e o *Free-Form Deformation* (FFD). Todos os casos de otimização foram investigados usando o método adjunto e um otimizador baseado em gradientes. O primeiro caso consiste na minimização da resistência aerodinâmica do perfil RAE 2822 em escoamento transónico viscoso sujeito aos constrangimentos de sustentação, momento de picada e área. Resultados semelhantes aos encontrados na literatura em termos de coeficiente de resistência aerodinâmica e custo computacional foram obtidos mostrando a aplicabilidade da metodologia. O segundo caso consiste em maximizar o coeficiente de planeio do perfil NACA 0012 a zero graus de ângulo de ataque em condições invíscidas para regimes transónico e supersónico. Utilizando o método FFD foi definida uma caixa adequada onde o número de pontos de controlo foi variado. No método *Hicks-Henne* foi analisado o parâmetro de controlo de largura das perturbações e o número de funções. Relativamente ao número de variáveis de projeto, o mesmo comportamento foi encontrado: aumento do coeficiente de planeio e custo computacional. Ambos os métodos foram capazes de aumentar a diferença de pressão para aumentar a sustentação, o que combinado com uma redução de área resultou num aumento coeficiente de planeio.

Palavras-chaves: Otimização de Perfis Alares, Métodos Paramétricos, Dinâmica de Fluidos Computacional, Método Adjunto

This page has been left intentionally blank.

Abstract

A robust and efficient high-fidelity aerodynamic shape optimization (ASO) methodology for airfoil shape optimization resorting to open-source platforms for computational fluid dynamics simulations was developed. This framework integrates open source codes: SALOME for the mesh generation; SU2 software for the aerodynamic shape optimization; and ParaView for post-processing. Two geometry parametrization methods, Hicks-Henne and Free-Form Deformation (FFD) methods are considered. All optimizations were performed using the adjoint method to compute sensitivities. The first case-study is a benchmark case which consists in drag minimization of the RAE 2822 airfoil in transonic viscous flow subject to lift, pitching moment and area constraints. Similar results to those found in the literature in terms of drag coefficient and computational cost were obtained showing the capability of the implemented ASO framework. In the second case-study, the aim was to maximize the lift to drag ratio starting with a NACA 0012 airfoil at zero degrees of angle of attack in inviscid conditions for two flow regimes (transonic and supersonic). For the FFD method a suitable definition of the box was set and the number of control points was allowed to change, while for the Hicks-Henne method both width bump control parameter and number of bump functions were analyzed. Regarding the number of design variables, the same behavior was found: increase of the lift to drag ratio and computational cost. In overall, they were able to increase pressure difference to increase lift which combined with an area reduction resulted in lift to drag ratio increase.

Keywords: Airfoil Shape Optimization, Parameterization Methods, Computation Fluid Dynamics, Adjoint Method

This page has been left intentionally blank.

Contents

List of Figures

This page has been left intentionally blank.

List of Tables

This page has been left intentionally blank.

List of Acronyms

1-D	One-Dimensional
2-D	Two-Dimensional
3-D	Three-Dimensional
AD	Algorithmic Differentiation
ADL	Aerospace Design Lab
ADODG	Aerodynamic Design Optimization Discussion Group
ASO	Aerodynamics Shape Optimization
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Levy
CST	Class-Shape Transformation
EFFD	Extended Free-Form Deformation
FFD	Free Form Deformation
IT	Information Technology
JST	Jameson-Schmidt-Turkel
KKT	Karush-Kuhn-Tucker
MPI	Message Passing Interface
NACA	National Advisory Committee for Aeronautics
NASA	National Aeronautics and Space Administration
N-S	Navier-Stokes
NURBS	Non-Uniform Rational Basis Spline
OS	Open-Source
PDEs	Partial Differential Equations
RANS	Reynolds-Averaged Navier-Stokes

SA	Spalart-Allmaras
SLSQP	Sequential Least SQuares Programming
SMEs	Small and Medium-sized Enterprises
SU^2	Stanford University Unstructured

Nomenclature

Latin symbols

a_i	= i th coefficients
A	= leading edge point
b	= degree of B-spline basis function
B	= trailing edge point
B_i^n, B_j^l	= i th or j th Bernstein basis polynomial functions of n or l degree
$\boldsymbol{B}$	= Bézier curve function
c	= airfoil chord
c_p	= specific heat at constant pressure
c_υ	= specific heat at constant volume
C_d	= drag coefficient
C_l	= lift coefficient
C_m	= pitch moment coefficient
C_p	= pressure coefficient
$C_{b_1}, C_{b_2}, C_{t_3}, C_{t_4}, C_{v_1}, C_{w_1}, C_{w_2}, C_{w_3}$	= constants used in Spalart-Allmaras turbulence model
C^*	= Sutherland's constant
$\mathbf{C}$	= vector of characteristic variables
$(\mathbf{C})_+$	= vector of positive characteristic variables
d_{ij}	= artificial dissipation term between nodes i and j
d_s	= distance to nearest surface
D	= drag force

$\mathbf{D}$ = vector of design/independent/input variables

e = specific internal energy

E = total energy *per* unit of mass

f = objective/interest function

f_{i} = i th Hicks-Henne bump functions

$f_{v_1}, f_{v_2}, f_w, f_{t_2}$ = functions used in the turbulence model

F_i = force applied to the fluid

$\tilde{F}_{ij}^{c}$ = numerical approximations of the convective flux between nodes i and j

$\tilde{F}_{ij}^{v}$ = numerical approximations of the viscous fluxes between nodes i and j

$\mathbf{F^C}$ = convective contribution to fluxes

$\mathbf{F^v}$ = viscous contribution to fluxes

g_{j} = inequality constraint functions

h = specific enthalpy

h_{i} = maximum location of the Hicks–Henne bump function

h_{k} = equality constraint functions

H = Ferguson spline function

$\mathrm{i}, \mathrm{j}, \mathrm{k}$ = counters

$\boldsymbol{I}$ = vector function of objective/interest/output

k = turbulent kinetic energy

l, n = degree of Bernstein polynomial

L = lift force

m = number of control points or design variables

M = Mach number

M = pitching moment

$\boldsymbol{n}$ = unit normal vector

n_s = direction normal to the surface

n_{D} = number of design/independent/input variables

n_I = number of objective/interest/output variables

$\boldsymbol{n_{ij}}$ = outward normal of the face between nodes i and j

$N(i)$ = set of neighbouring nodes to node i

N_{CFL} = Courant-Friedrichs-Levy number

$\mathcal{N}_{\mathrm{i}}^{b}$ = i th B-spline basis functions of b degree

$\mathcal{O}$ = number of surface mesh nodes

p = static pressure

p_i = pressure at node i

P_r = laminar Prandtl number

$(P_r)_t$ = turbulence Prandtl number

$\mathbf{P}_{\mathrm{i}}$, $\mathbf{P}_{\mathrm{ij}}$ = i th or i, j th control points

q_j = heat flux vector

$\dot{q}_{wall}$ = instantaneous heat flux

u_i = velocity component

$\boldsymbol{u}$ = flow velocity vector

U_i = state variables components at node i

$\mathbf{U}$ = vector of state variables

$\boldsymbol{r}$ = residuals values vector of the governing equations

R = perfect gas constant

R_i = governing equation residual at node i

s = normal displacement with respect to each mesh node on the geometry surface

s_2, s_4 = stretching parameters

S = solid wall boundary

$S_{airfoil}$ = airfoil area

S_E = energy source term

$\mathbf{S}_{\emptyset}$ = vector of generic source terms

t = time

t_{i} = thickness of Hicks–Henne bump function

T = parameter of the time-averaged turbulence term

T = temperature

T_A, T_B = tangent vectors to point A and B

T_{wall} = surface temperature

v = scalar parameterization variable

x_i = cartesian direction

x, y = cartesian coordinates

X = two-dimensional airfoil (x, y)

y_{xx} = curvature at crest location of the airfoil

y^+ = Non-dimensional distance to the wall

W = B-spline function

Greek symbols

α = angle of attack

α_b = boat tail angle

α_c = camber angle

α_{TE} = trailing edge direction angle

β_{TE} = trailing edge wedge angle

Γ_∞ = far-field domain boundary

γ = ratio of gas specific heats, equal to 1.4 for air

ΔS_{ij} = interface area between nodes i and j

ΔY_{TE} = trailing edge thickness

δ_{ij} = Kronecker's delta

$\varepsilon_{ij}^{(2)}, \varepsilon_{ij}^{(4)}$ = pressure switches

η_{i} = i th Hermite polynomial function

Θ_{i} = i th design variables

κ = thermal conductivity

$\kappa^{(2)}, \kappa^{(4)}$ = adjustable parameters

λ_{ij} = local spectral radius nodes i and j

$\lambda_{\mathrm{i}}^{\mathrm{conv}}$ = convective spectral radius at node i

$\lambda_{\mathrm{i}}^{\mathrm{visc}}$ = viscous spectral radius at node i

μ = laminar/dynamic viscosity

μ_{t} = turbulent viscosity

μ_{total} = total viscosity as a sum of laminar and turbulent components

μ^{*}_{total} = effective thermal conductivity

ν = kinematic viscosity

ν_t = turbulence kinematic viscosity

$\tilde{\nu}$ = dependent variable for turbulence model

ρ = fluid density

τ_{ij} = shear stress terms

τ_{ij}^{turb} = Reynolds stress tensor terms nodes i and j

ϕ = dependent variable of Reynolds averaging

φ_{ij} = scaling parameter nodes i and j

$\boldsymbol{\Psi}$ = adjoint vector

ω_{ij} = i, j th weight parameter

$\boldsymbol{\omega}$ = magnitude of vorticity

Ω = flow domain

Ω_i = control volume surrounding node i

$\nabla^2 U_i$ = undivided Laplacian node i

Other symbols

$\mathfrak{R}$ = residuals functions vector of governing equations

Superscripts

$initial$ = initial aerofoil to be deformed

$lower$ = lower airfoil surface

$upper$ = upper airfoil surface

^ = dimensional quantity

′ = fluctuating part in terms of Reynolds averaging

′′ = fluctuating part in terms of Favre averaging

$-$ = Reynolds time-averaged value

$\sim$ = Favre averaged value

$\circ$ = degree

Subscript

$baseline$	= baseline airfoil shape to be deformed
min/max	= minimum or maximum
LE	= leading edge
TE	= trailing edge
∞	= free-stream or far-field quantities

Chapter 1

Introduction

The aeronautical industry has been recognized as being one of the industry of high technological advancement and innovation, generating useful technologies for other sectors and promoting knowledge and innovation for the creation of new products. The sector is an ever-changing sector where the need for advanced technology and high know-how are fundamental for success. From an early age, computers prove to be highly important and useful tools for the industry [1]. Year after year, their evolution allowed to obtain a greater processing speed and data storage. New operating systems have begun to emerge, and new software have begun to be developed. With the growth of the information technology (IT) industry, companies have started to patent their software, protecting their intellectuality. For most commercial companies, their software source code is a trade secret, and access to this code by third parties would necessarily mean to sign a non-disclosure agreement. Several companies in industry are using it nowadays, however, investing in the required software, and consequently, hardware and commercial licenses are still an obstacle hurdle for small and medium-sized enterprises (SMEs). Commercial software in most cases are seen as black boxes without any knowledge of their internal working process and without the possibility of modifying their programming, whereas, open-source (OS) software have brought the possibility to view, change, enhance, and re-distribute any part of a code [2]. On the one hand, they provide a cheaper approach to simulations when they are compared to commercial software. On the other hand, one limitation of OS software is their dependence on a more knowledgeable user than the commercial code, as more freedom is provided to the user and documentation can be limited [3].

1.1. Topic Overview

One key focus of technological advancement in aeronautics is improving aerodynamic performance through different objectives such as drag minimization, maximization of lift to drag ratio or reducing pitching moment, while several and different constraints can be applied at the same time. With a computational fluid dynamics (CFD) solver it is possible to analyse a flow for specific design. When the solver is coupled with an optimization algorithm, geometry parameterization technique and control tools, it will be possible to perform aerodynamic shape optimization (ASO) [4]–[6]. Therefore, given an

aerodynamic shape, like an airfoil, wing or an entire aircraft, the solver will redesign and reanalyse, until the optimal shape is achieved. The information provided from a sensitivity analysis will allow a gradient-based optimization algorithm to arrive to the optimal shape, becoming crucial for an ASO. There are other types of optimization algorithms that do not depend on gradients, such as heuristic methods (genetic algorithms), that are not addressed in this work.

The sensitivity analysis is known as the study of how the changes in the inputs of a desirable model, numerical or otherwise, can change the outputs of the model [7]. An important field of sensitivity analysis is the computation of derivatives of one or more state variables (i.e. outputs) as a function of the design variables (inputs) to obtain reliable results in a lower computational time cost. A common classification for sensitivity analysis distinguishes between local and global sensitivity. The global sensitivity analysis allows to evaluate and quantify the response with respect to inputs, adapting more for models with large uncertainties. Local sensitivity analysis allows to evaluate and quantify the response for a fixed set of inputs and, contrarily to global sensitivity, is better suited for models where the uncertainties tend to be lower [8]. Sensitivity information has several and distinguish applications, being one of them the use of that information in gradient-based optimization methods. Since computing the gradients is a costly step in the optimization cycle, employing an efficient method that accurately calculate sensitivities is an important choice. Gradient information can be used to reduce the cost associated with uncertainty quantification and sensitivity analysis. The optimization process can be performed through gradient-based or gradient-free algorithms. Despite gradient-based optimization methods are usually more difficult to use than gradient-free optimization methods, it can be the best decision when an efficient gradient evaluation is available and the problems deal with a to large set of design variables [9]. In gradient-based optimization, accuracy of the derivative that affect the converge behaviour of the algorithm is a crucial step to ensure robust and efficient convergence, especially in problems with several constraints. The precision of the gradients will limit the optimal solution, and its inaccuracy can cause the optimizer to end or to go in a direction that will increase the iteration number until it reaches the optimum.

Most of gradient-based optimizers use the finite-differences method for sensitivity analysis. Using the appropriate finite-differences scheme, each input of interest is perturbed and the output is revaluated to determine its new value. The gradient is estimated through the difference between the output and the unperturbed value, divided by the perturbation value [8]. The method is not particularly accurate or efficient, however its simplicity and easy implementation make it one of the commonly used for computing the gradient. Inaccuracies in the gradients calculations, due to subtractive cancellation, are usually the reason for gradient-based optimizers not converging. In addition, the cost in computing the gradients is proportional to the number of design variables, and if this number is large it will be an obstacle to the development of the optimization cycle [10]. The substantial number of design variables involved can lead to slow convergence rates and extremely expensive finite difference gradient calculations [11]. With its benefits and limitations, the finite-differences method is suitable for certain applications but, when more accuracy and efficiency are required, for instance when a complex set of governing equations are being used for the flow analysis and a large number of design variable is essential, more computationally efficient methods are necessary.

Contrary to the finite-difference method, which is not recommended for ASO when dealing with several design variables, the adjoint method has brought the possibility to compute the gradients with greater accuracy and efficiency, thus reducing computational costs [12]. Many developments in ASO had occurred in the early 90's, after the adjoint method made it possible to efficiently compute shape gradients [12]–[14]. Jameson [13] was one of the first to propose an ASO based on systems governed by partial differential equations (PDEs). The adjoint method was extended together with the inviscid Euler equations in compressible flows and it was revealed to be appropriate for the design of transonic airfoils. Reuther *et al.* [15], extended the methodology to perform an ASO of complex aircraft configurations. Since then, adjoint implementations have increased and their applications were extended by several researchers. The aerodynamic design of transonic wings requires a model capable of capturing all physical problems, like shock-wave boundary layer interaction, due to the strong nonlinear connection between airfoil shape, wave drag and viscous effects [16]. Therefore, the adjoint method has been extended to the compressible Navier-Stokes (N-S) equations with turbulence models. In Anderson and Venkatakrishnan [17], the continuous adjoint approach for obtaining sensitivity derivatives and boundary conditions were derived for the incompressible N-S equations without inclusion of turbulence effects. This inclusion was performed in Elliot *et al.* [18], along with the discrete adjoint implementation for the compressible N-S equations. In Jameson *et al.*[19], the adjoint method is presented for optimal aerodynamic design considering flows governed by the compressible N-S equations, which was successfully applied to the design of a wing in transonic flow. Anderson and Bonhaus [20] performed a two-dimensional (2-D) design optimization methodology using the discrete adjoint approach and including the Spalart-Allmaras (SA) turbulence model for viscous flows. Continuing with the same approach, Nielsen and Anderson [21] achieved a drag reduction of the ONERA M6 wing using a three-dimensional (3-D) Reynolds-Averaged Navier-Stokes (RANS) equations coupled with a turbulence model SA. Numerical simulations methods to model the flow are used routinely and, with the increase in computer power, their integration into the optimization process produced ASO frameworks. Over the last decades, shape optimization has been successfully applied to 2-D [20], [22] and 3-D configurations [16], [21], [23]. Numerical simulation and optimization of complex and large-scale aircraft have become possible to be solved and new shapes began to emerge. ASO of transonic wing is another example of a complex design problem solved with respect to hundreds of design variables [23].

The use of high-fidelity tools has brought more confidence in the aerodynamic shape design. The success of ASO depends on: efficiency and accuracy of the CFD; the optimization algorithm; and also on the technique used to control the geometry shape along the optimization process. Geometry and control techniques will explore which new shapes can be generated during the optimization. The method must be flexible to handle with local changes and global shape changes, ensuring that the true optimal solution is achieved. One of the challenging topics in optimization is the selection of the mathematical representation of the airfoil design variable that provides a wide variety of possible airfoil shapes. The choice of the shape parameterization technique will have a huge impact on the formulation, implementation and solution of the optimization problem. Several methods have been used for airfoil parameterization and several researches and developments have been made in ASO field.

Efforts have been carried out to achieve high performance analyses in ASO frameworks, combining differences CFD solvers, adjoint formulations, optimization algorithms and geometric parameterizations techniques. Open source CFD software have their strengths and weakness, providing a variety of solvers to the customers. Some are well tested and their results were already validated with case studies and wind tunnel testing results, but others are still at an early stage. The Stanford University Unstructured (SU^2) software, an OS collection of software tools for performing CFD analysis and design, is one of the several software developed by individuals and organized teams around the world. What started as a small project in January of 2012 with the first release, rely nowadays on the contributions of thousands of users and hundreds of developers, for the official release SU^2 v6.0. The SU^2 software provide the combination of governing equations of fluid flows, adjoint formulations and several parameterization methods which with an optimization algorithm allow to achieve an ASO framework and made the SU^2 a powerful suite.

1.2. Motivation and Goals

Technological advances, and consequently the growth of the aeronautical sector, have led several companies to invest in commercial software or in creating their own codes in order to carry out their studies and analyses. However, not all companies, such as SMEs, have the opportunity of obtaining commercial software given their excessive cost and developing their own code takes time, validation and resources. Improvements in high performance computing have increased the development of high-fidelity numerical models and optimization techniques, allowing to create OS CFD software capable of analysing and optimizing 2-D, or even 3-D geometries. CFD tools and optimization algorithms have been employed to shorten design cycle times and to analyse design spaces in an efficiently way. A high fidelity ASO can result in high computational costs when traditional methods are applied to compute the gradient, such as the finite difference method. The adjoint methods, however, have brought the possibility to improve the computational efficiency of ASO.

CFD simulations requires expertise knowledge since it involves multiple areas, such as computational modelling, aerodynamics and programming. In the current days, high-fidelity ASO models are usually used by commercial software or by software developed by the companies themselves. Therefore, a robust and efficient high-fidelity ASO methodology with an adjoint method, that employs only open source tools will be developed. The main goal of this research is the development of a high fidelity computational framework able to perform an aerodynamic shape optimization of an airfoil using CFD, where one can use different parameterization techniques, turbulence models, meshes and optimization settings resourcing to python scripts and open-source software for pre and post-processing. In addition, the impact of the parameterization methods on an airfoil surface will be investigated. The methodology will be applied to a first case-study to achieve the drag minimization of the RAE 2822 airfoil in transonic viscous flow, an airfoil optimization case-study defined by the AIAA Aerodynamic Design Optimization Discussion Group (ADODG). A second case-study will be performed to maximize the lift over drag ratio of the modified NACA 0012 airfoil for inviscid flow conditions, examining the impact of varying the Mach number, M, in aerodynamic shape optimization.

1.3. Thesis Outline

The numerical model of the aerodynamic flow and the adjoint formulations are presented in chapter 2. The thesis was conducted as a combined literature study of the capabilities of the different parameterization methods. An overview of several parameterization techniques, with their strengths and weakness, is done in chapter 3. A description of the required software, methods and algorithms employed, and some limitations are indicated in chapter 4. Then, in chapter 5 the two-case studies and the optimization results are discussed. Finally, conclusions, recommendations and future work is discussed in chapter 6.

This page has been left intentionally blank.

Chapter 2

Theoretical background

Optimizing the shape of an airfoil means improving its aerodynamic properties by changing its shape. Some aerodynamic properties like lift, drag or the pressure distribution can be used to analyse the performance of an airfoil and, when a shape optimization is carried out, these aerodynamic properties can be mathematically seen as a cost function or constraints to the design. Numerical aerodynamic optimization involves the minimization of a chosen cost or objective function, such as drag, through the manipulation of a set of design variables. The design variables can also be mathematically defined to parameterize the shape of an airfoil. Gradient-based optimization methods have the aim of minimizing the given objective function with respect to a set of design variables, using the gradients (sensibility analysis) to guide the design towards an optimum design. On one iterative process, for each new step a new set of design variables is produced, causing a change in the objective function. A general constrained optimization problem can be defined according to:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{D}) \\ \text{w.r.t} \quad & \mathbf{D}, \\ \text{subject to} \quad & g_{\mathrm{j}}(\mathbf{D}) \leq 0, \mathrm{j} \in \{1, \dots, \mathrm{m}\} \\ & h_{\mathrm{k}}(\mathbf{D}) = 0, \mathrm{k} \in \{1, \dots, l\} \end{aligned} \tag{2.1}$$

where $\mathbf{D}$ is the vector of n_D design variables, f is the objective function to be minimised, or maximize, g_{j} are the inequality constraint functions and h_{k} are the equality constraint functions, to be satisfied. A common optimization problem could be to maximize the airfoil lift, or minimize its drag, while the base area of the airfoil is kept unchanged. With CFD tools, it is possible to solve the so-called N-S equations, obtaining the flow solution and achieving the aerodynamic properties which will be used to form the objective function.

This chapter will start with the mathematical formulation to solve the RANS equations. Subsequently, it is shown how to evaluate the gradients through the adjoint methods. Whenever necessary, reference will be made to SU^2 software and their mathematical implementation in the source code.

2.1. Governing equations of fluid dynamics

The motion of the particles in a continuously medium is represented by equations that describe mathematically the conservation of a quantity such as mass, momentum or energy. These conservation equations, also known as transport equations, are generically represented in their differential form (2.2), defined on a domain $\Omega \subset \mathbb{R}^3$, as:

$$\frac{\partial(\mathbf{U})}{\partial t} + \nabla \cdot \mathbf{F}^C = \nabla \cdot \mathbf{F}^\upsilon + \mathbf{S}_\emptyset \text{ in } \Omega, t > 0. \tag{2.2}$$

The PDEs system resulting from physical modelling of the problem should have the above structure with the appropriate boundary and temporal conditions that will be problem-dependent. In this general framework, also recognized as a 'conservative form' of transport equations, $\mathbf{U}$ represents the vector of state variables, $\mathbf{F}^C(\mathrm{U})$ are the convective fluxes, $\mathbf{F}^\upsilon(\mathrm{U})$ are the viscous fluxes and $\mathbf{S}_\emptyset(\mathrm{U})$ is the source term. The vector of conservative variables is described as $\mathbf{U} = (\rho, \rho u_1, \rho u_2, \rho u_3, \rho E)^\mathbf{T}$, where ρ is the fluid density, $\boldsymbol{u} = (u_1, u_2, u_3)$ is the velocity of the flow in a Cartesian coordinate system and E is the total energy *per* unit mass. Based on conservation laws, the fundamental governing equations of fluid dynamics are the mass conservation, momentum conservation and energy conservation. They are the mathematical statements of the physical principles upon which all fluid dynamics are based. The equation of mass conservation, also known as continuity equation (2.3), together with the momentum equation (2.4) and energy equation (2.5), form a set of five equations. In the compressible Newtonian fluid case, the governing equations in conservation form for the unsteady state are given by [1]:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i} = 0; \tag{2.3}$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial\left(\rho u_j u_i\right)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho F_i; \tag{2.4}$$

$$\frac{\partial}{\partial t}\left[\rho\left(e + \frac{1}{2}u_i u_i\right)\right] + \frac{\partial}{\partial x_j}\left[\rho u_j\left(h + \frac{1}{2}u_i u_i\right)\right] = \frac{\partial}{\partial x_j}\left(u_j \tau_{ij}\right) - \frac{\partial q_j}{\partial x_j} + S_E; \tag{2.5}$$

where u_i is the velocity component in direction x_i, e is the specific internal energy, τ_{ij} is the viscous stress tensor, q_j is the heat flux vector and h is the specific enthalpy, given by $h = e + p/\rho$, p is the static pressure, F_i is an force applied on the fluid and S_E is an energy source term. The latin subscripts i, j denote 3-D cartesian coordinates with repeated subscripts implying summation. In aerodynamics it is reasonable to assume the gas as a perfect gas, thus adding the state equation for a classic ideal gas to the governing equations:

$$p = \rho R T = (\gamma - 1)\rho e, \tag{2.6}$$

where γ is the ratio of gas specific heats. The convective heat flux q_j is defined as:

$$q_j = -\kappa \frac{\partial T}{\partial x_j}, \tag{2.7}$$

where κ is the thermal conductivity. Additionally, the specific internal energy and specific enthalpy are:

$$e = c_{\upsilon} T, \qquad h = c_p T, \tag{2.8}$$

where c_p and c_{υ} are the specific-heat coefficients. Note that $\gamma = c_p / c_{\upsilon}$ and $R = c_p - c_{\upsilon}$. Furthermore:

$$q_j == -\frac{\kappa}{c_p}\frac{\partial h}{\partial x_j} = -\frac{\mu}{P_r}\frac{\partial h}{\partial x_j}, \tag{2.9}$$

where P_r is the Prandtl number defined by:

$$P_r = \frac{c_p \mu}{\kappa}. \tag{2.10}$$

The laminar viscosity, or dynamic, μ, is determined through the Sutherland's law [20]:

$$\mu = \frac{\hat{\mu}}{\hat{\mu}_{\infty}} = \frac{(1 + \mathbf{C}^*)}{\left(\hat{T}/\hat{T}_{\infty} + \mathbf{C}^*\right)}\left(\frac{\hat{T}}{\hat{T}_{\infty}}\right)^{3/2}, \tag{2.11}$$

where $\mathbf{C}^*$ is the Sutherland's constant divided by a free-stream T_{∞}. The governing equations contain as further unknowns the viscous stress tensor components τ_{ij}:

$$\tau_{ij} = \mu\left[\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) - \frac{2}{3}\left(\frac{\partial u_k}{\partial x_k}\right)\delta_{ij}\right]. \tag{2.12}$$

When the equation (2.12) of the shear stress tensor for a Newtonian viscous fluid is taking into account in the equation (2.4), yields the so-called N-S equations of motion. When applied to a viscous flow, the set of equations are called as the N-S equations, while they are known as the Euler equations when applied to inviscid flow. Inviscid flow is a flow where the dissipative, transport phenomena of viscosity, mass diffusion, and thermal conductivity are not considered [1]. By neglecting all shear stresses and heat conducting terms in the N-S equations, the Euler equations are achieved. These equations describe non-viscous, non-heat conducting flows and are especially valid for flows at high Reynolds numbers outside of viscous regions that develop close to the solid surfaces. If the term $\mathbf{F}^{\upsilon}(\mathrm{U})$ present in equation (2.2) is not considered, the resulting equations for unsteady, compressible inviscid flow are the Euler equations described from:

$$\frac{\partial(\mathbf{U})}{\partial t} + \nabla \cdot \mathbf{F}^{C} = \mathbf{S}_{\emptyset} \text{ in } \Omega, t > 0. \tag{2.13}$$

To accurately model the complex flow field surrounding an airfoil, it is required to solve the complete, compressible N-S equations to capture phenomena such as separated flow regions, induced shock waves, and shock boundary-layer interactions. All the governing equations mentioned above are detailed in [1].

2.1.1. Reynolds-Averaged Navier-Stokes equations

The field properties become random functions of space and time when turbulence flows are operating. The turbulent average process is introduced in order to avoid the laws of motion for the 'mean', time-averaged, turbulent quantities. The turbulent equations are computed, in time, over the entire spectrum of turbulent fluctuations. This results in the well-known RANS equations which require, in addition,

empirical or at least semi-empirical information about the turbulence structure and its relation with the mean flow. The averaged time needs to be defined in order to remove the influence of the turbulent fluctuations, without damage to the time dependence connected with other time-dependent phenomena with time scales distinct from those of turbulence [24]. Reynolds averaging refers to the process of averaging a variable or an equation in time. To model the RANS equations, all dependent variables that varies in time, ϕ, are split into a time-averaged part, $\bar{\phi}$, and a fluctuating part, ϕ', as:

$$\phi = \bar{\phi} + \phi', \tag{2.14}$$

introducing the time-averaged turbulence part with:

$$\bar{\phi} = \frac{1}{\mathrm{T}} \int_{\mathrm{T}} \phi(x,t) dt, \tag{2.15}$$

where T must be large enough compared to the time scale of the turbulence but small compared to the time scales of all other unsteady phenomena. More about this parameter is discussed in [24]. Favre-averaging is sometimes used in compressible flow to separate turbulent fluctuations from the mean flow. The average process leads to products of fluctuations between density and other properties like velocity and internal energy. Here, the dependent variable can be split into a mean part, $\tilde{\vartheta}$, and a fluctuating part, ϑ'', using the density weight average in the next equation:

$$\vartheta = \tilde{\vartheta} + \vartheta'', \tag{2.16}$$

with the term including the turbulent fluctuations and density fluctuations as:

$$\tilde{\vartheta} = \frac{\overline{\rho\vartheta}}{\bar{\rho}}. \tag{2.17}$$

In many cases it is not required to use Favre-averaging since turbulent fluctuations, in the majority of the cases, do not lead to any significant fluctuations in density. In that case, the Reynolds averaging is recommended to be used [25]. After the Favre decomposition for the u, e and h and performing a standard time-averaging decomposition for p and ρ, the properties are inserted into the governing equations (2.3) to (2.5) and considering the time-averaged part for the governing equations the following equations are achieved [26]. The equations are expressed as a system of conservation laws that relate the time rate of change mass, momentum and energy equations in a control volume:

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial (\bar{\rho}\tilde{u}_i)}{\partial x_i} = 0; \tag{2.18}$$

$$\frac{\partial \bar{\rho}\tilde{u}_i}{\partial t} + \frac{\partial \left(\bar{\rho}\tilde{u}_j\tilde{u}_i\right)}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial \bar{\tau}_{ij}}{\partial x_j} - \frac{\partial \left(\overline{\rho u_j'' u_i''}\right)}{\partial x_j} + \bar{\rho}F_i; \tag{2.19}$$

$$\begin{aligned} &\frac{\partial}{\partial t}\left[\bar{\rho}\left(\tilde{e} + \frac{1}{2}\tilde{u}_\iota\tilde{u}_\iota\right) + \frac{1}{2}\overline{\rho u_\iota'' u_\iota''}\right] + \frac{\partial}{\partial x_j}\left[\bar{\rho}\tilde{u}_j\left(\tilde{h} + \frac{1}{2}\tilde{u}_\iota\tilde{u}_\iota\right) + \tilde{u}_j\frac{\overline{\rho u_\iota'' u_\iota''}}{2}\right] \\ &\qquad = \frac{\partial}{\partial x_j}\left(\tilde{u}_\iota\overline{\tau_{\iota j}} - \tilde{u}_\iota\left(\overline{\rho u_\iota'' u_j''}\right) - \overline{\rho u_j'' h''} + \overline{\tau_{\iota j} u_\iota''} - \overline{\rho u_j'' \frac{1}{2} u_\iota'' u_\iota''}\right) - \frac{\partial \bar{q}_j}{\partial x_j} + S_E, \end{aligned} \tag{2.20}$$

with

$$\bar{p} = (\gamma - 1)\bar{\rho}\tilde{e}. \tag{2.21}$$

The term $\overline{\rho u_i'' u_j''}$ can be modelled using an eddy-viscosity assumption for the Reynolds-stress tensor:

$$\tau_{ij}^{turb} = \overline{-\rho u_i'' u_j''} = 2\mu_t \left(\frac{1}{2}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right) - \frac{1}{3}\frac{\partial \tilde{u}_k}{\partial x_k}\delta_{ij} \right) - \frac{2}{3}\bar{\rho}k\delta_{ij}, \tag{2.22}$$

where μ_t is the turbulent dynamic, or eddy viscosity, which is determined with recourse to a turbulence model, and k is the turbulent kinetic energy. All the equations, coefficients and assumptions details are available in [27].

2.1.2. Turbulence model

To model the governing equations with RANS equations, it is required to model the turbulent scales. According with the standard approach to turbulence modelling based upon the Boussinesq hypothesis [28], the effect of turbulence can be represented as an increased viscosity, with the total viscosity separated into a laminar, μ, and a turbulent, μ_t, component. The dynamic viscosity can be taken as function of temperature only, $\mu = \mu(T)$, while the turbulent viscosity μ_t can be given by an appropriate turbulence model involving the flow state, and a series of new variables [29], and:

$$\mu_{total} = \mu + \mu_t, \qquad \mu_{total}^{*} = \frac{\mu}{P_r} + \frac{\mu_t}{(P_r)_t}, \tag{2.23}$$

where $(P_r)_t$ is the turbulent Prandtl number. Different models deal with different approach to model turbulence, being selected for the current studies the one-equation SA turbulence model [30]. The SA turbulence model is a popular model used in aeronautical applications [6], [16], [20], [31] with the main advantage being its robustness and fast implementation; it is also quite stable and does not require intensive memory resource [32]. The model is a “cheap way” of computing the boundary layers in aerodynamics. This is an one-equation turbulence model, where the turbulent viscosity is computed as follows:

$$\mu_t = \rho\nu_t = \rho\tilde{\nu} f_{\nu_1}, \tag{2.24}$$

where

$$f_{\nu_1} = \frac{\chi^3}{\chi^3 + C_{\nu_1}^3}, \tag{2.25}$$

$$\chi = \tilde{\nu}/\nu, \tag{2.26}$$

$$\nu = \mu/\rho. \tag{2.27}$$

The eddy viscosity parameter $\tilde{\nu}$ is calculated by solving a transport equation as:

$$\begin{aligned} \frac{\partial \rho\tilde{\nu}}{\partial t} + u_j\frac{\partial \rho\tilde{\nu}}{\partial x_j} = \frac{1}{\sigma}\left[\frac{\partial}{\partial x_j}\left((\mu + \rho\tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_j} \right) + C_{b_2}\rho\frac{\partial \tilde{\nu}}{\partial x_i}\frac{\partial \tilde{\nu}}{\partial x_i} \right] + C_{b_1}(1 - f_{t_2})\tilde{S}\rho\tilde{\nu} \\ - \left(C_{w_1} f_w - \frac{C_{b_1}}{C_k^2} f_{t_2} \right)\left(\frac{\tilde{\nu}}{d_S} \right)^2 \rho, \end{aligned} \tag{2.28}$$

where $f_{t_2} = C_{t_3} exp\left(-C_{t_4}\chi^2\right)$, $\tilde{S} = \boldsymbol{\omega} + (\tilde{\nu}/(C_k^2 d_S^2))f_{v_2}$ and $\boldsymbol{\omega}$ is the magnitude of the vorticity. The turbulence length scale comes from $k \cdot d_S$, where d_S is the distance from the field point to the nearest wall. The wall functions are dependent on the parameters f_{v_2} and f_w given by:

$$f_{v_2} = 1 - \frac{\chi}{\left(1 + \chi f_{v_1}\right)}, \tag{2.29}$$

$$f_w = \mathrm{g}\left(\frac{1 + C_{w_3}^6}{\mathrm{g}^6 + C_{w_3}}\right)^{1/6}, \tag{2.30}$$

where $\mathrm{g} = r + C_{w_2}(r^6 - r)$ and $r = \tilde{\nu}/\left(\tilde{S} C_k^2 d_S^2\right)$. The set of constants of the model: σ, C_{b_1}, C_{b_2}, C_k, C_{w_1}, C_{w_2}, C_{w_3}, C_{v_1}, C_{t_3} and C_{t_4} can be found in [33]. After the transport equation is solved for $\tilde{\nu}$, the eddy viscosity is computed from (2.24). According with [34], the physical meaning of the far-field boundary condition for the turbulent viscosity is the imposition of some fraction of the laminar viscosity at the far-field. The $\tilde{\nu}$ is brought to zero on the viscous wall which corresponds to the absence of turbulent eddies very close to the wall. All the model details are available in [30].

2.1.3. Boundary Conditions

An important aspect of numerical solution are the boundary conditions. Without the physically proper implementation of the boundary conditions and their numerically proper representation, it will be hard or even impossible to obtain a proper numerical solution for the flow problem. Applying the appropriate boundary conditions to a compressible RANS solver is needed since the boundary conditions will dictate the solution to be obtained from the governing equations. Consider the governing equations in a domain $\Omega \subset \mathbb{R}^3$, with a disconnected boundary divided in far-field, Γ_∞, and an adiabatic wall boundary, S, as shown in Figure 1.

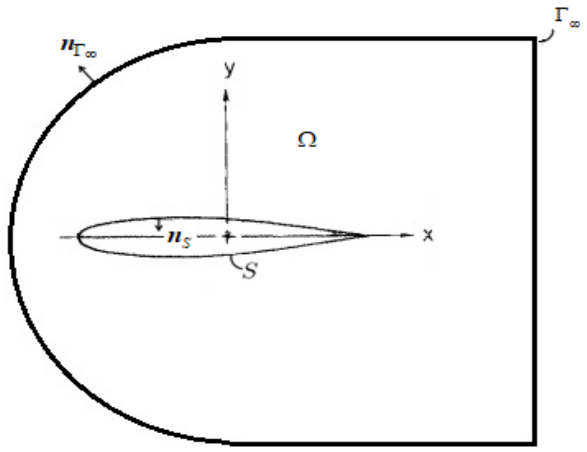


Figure 1 - Identification of the flow domain, boundaries and normal surface in a generic airfoil.

The surface S is the outer surface line of the airfoil and it is considered continuously differentiable [35]. To get the proper physical boundary conditions for a viscous flow, the boundary condition on the surface is given by zero relative velocity between the surface S and the flow immediately at the surface. In the

2-D space it leads to both tangential and normal velocities must be zero. This condition is called the no-slip condition at the solid wall [1]. If the surface is stationary, with the flow moving through it, then:

$$\boldsymbol{u} = 0 \text{ (at the surface for a viscous flow).} \quad \textbf{(2.31)}$$

In addition, a boundary condition is used to couple the temperature in the fluid and the surface with an analogous “no-slip” condition associated. Considering the surface temperature define by T_{wall}, then the temperature of the fluid layer directly in contact with the surface will also be T_{wall}. If T_{wall} is known, then the proper boundary condition on the flow temperature T will be:

$$T = T_{wall} \text{ (at the surface).} \quad \textbf{(2.32)}$$

Otherwise, if the surface temperature is unknown but is changing as a function of time due to aerodynamic heat transfer to or from the surface, the Fourier law of heat conduction gives the boundary condition at the surface:

$$\dot{q}_{wall} = -\left(\kappa \frac{\partial T}{\partial n_s}\right)_{wall}, \quad \textbf{(2.33)}$$

where $\dot{q}_{wall}$ is the instantaneous heat flux to the surface and n_s the direction normal to the surface. When the surface temperature becomes such that there is no heat transfer, this surface temperature can be called the adiabatic wall temperature. Hence, for a adiabatic wall, the boundary condition is [1]:

$$\left(\frac{\partial T}{\partial n_s}\right)_{wall} = 0. \quad \textbf{(2.34)}$$

The no-slip condition is used normally to model viscous flows, however can be neglected in favour of the no-penetration condition. In the cases of inviscid flows, there is no friction to promote its sticking to the surface. For a nonporous wall, there can be no mass flow into or out of the wall. This means that the flow velocity vector, $\boldsymbol{u}$, immediately adjacent to the wall must be tangent to the wall. Therefore, the non-penetrability condition is used, forcing the flow to be tangent to a solid surface, the upper and lower walls of the airfoil. Here the effect of boundary layers is neglected. If $\boldsymbol{n}$ is a unit normal vector at a point on the airfoil surface, the wall boundary condition can be formulated as:

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0 \quad \textbf{(2.35)}$$

In the domain boundaries, the far-field, Γ_∞, a characteristic-based boundary condition is applied [35]:

$$(\mathbf{C})_+ = \mathbf{C}^\infty, \quad \textbf{(2.36)}$$

with $\mathbf{C}$ representing the characteristic variables. The idea behind the far-field boundary condition is to attribute free-stream values to all the flow properties at the external boundary. More developments about the far-field boundary condition are available in the literature [36] and [37].

2.1.4. Finite Volume Discretization

Numerical solution of conservation laws resorting to the finite volume method (FVM) has become quite common in CFD in general due to its simplicity and ease of implementation for arbitrary grids, structured or unstructured. The principle of the method is based on the discretization of the integral form of the

conservation law. The FVM discretizes the governing equations by first dividing the physical space into several control volumes. The accuracy of the spatial discretization depends on the scheme where the fluxes will be evaluated. Different ways allow to define the shape and position of the control volumes with respect to the grid, being the two basic approaches distinguished as cell-centered and cell-vertex as shown in Figure 2 [38].

(a) (b)

Figure 2 - Cell-centered (a) and cell-vertex (b) structured finite volume mesh [38].

In FVM, the evolution of cell-averaged values is considered, which distinguishes it from the finite difference and finite element methods, where the numerical quantities are local function values at the mesh points. Once the grid is generated, the method will associate a control volume to each mesh point and apply the integral conservation law in that control volume [24]. On an arbitrary grid, the discretized space is formed by a set of small cells, being each cell associated to one mesh point as it is shown in Figure 2. The FVM has the advantage on handling with any type of mesh, structured or unstructured, where a large number of options are open for defining the control volumes on which the conservation laws are expressed. Depending on the type of mesh, the control volume for every cell node is created differently.

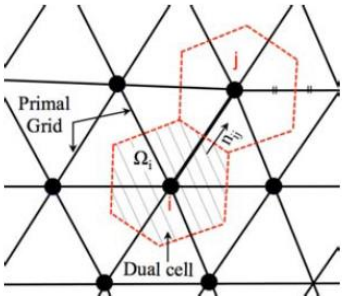


Figure 3 - Primal mesh scheme and control volume of a dual mesh [29].

A description of the spatial and time discretization scheme is then carried out, taking into account the implementation performed in SU^2 software given in [29]. The governing equations are discretized separately in space and time to allow for the selection of different types of numerical schemes for the spatial and temporal integration. Usually, spatial discretization is performed using the FVM, whereas integration in time is achieved through one of the available explicit or implicit methods [24].

In SU^2, the PDEs are discretized resorting to the FVM with a standard edge-based data structure applied in a dual grid where the control volumes are constructed based on the centroids of each primal grid cells connected with the mid-points of the surrounding edges using a median-dual, vertex-based scheme, as shown in Figure 3. The control volumes are formed by connecting the centroids, face and

edge midpoints of all cells sharing a particular node. The governing equations are integrated over a control volume and with the application of the Gauss's theorem, the semi-discretized integral form of a typical PDEs is given by [29]:

$$\int_{\Omega_i} \frac{\partial \mathbf{U}}{\partial t} d\Omega + \sum_{j \epsilon N(i)} \left(\tilde{F}_{ij}^{c} + \tilde{F}_{ij}^{v}\right) \Delta S_{ij} - \mathbf{S}_{\emptyset} |\Omega_i| = \int_{\Omega_i} \frac{\partial \mathbf{U}}{\partial t} d\Omega + R_i(\mathbf{U}) = 0, \tag{2.37}$$

where $\tilde{F}_{ij}^{c}$ and $\tilde{F}_{ij}^{v}$ are the numerical approximations of the convective and viscous fluxes, respectively, projected into the local normal direction, ΔS_{ij} is the area of the face associated with the edge, Ω_i is the volume of the control volume, $R_i(\mathbf{U})$ is the numerical residual of all spatial terms at node i and $N(i)$ are the set of neighbouring nodes to node i. The evaluation of the convective and viscous fluxes is made at the midpoint of an edge.

2.1.5. Spatial discretization

In the present work, the Jameson-Schmidt-Turkel (JST) scheme [39] for the numerical convective fluxes is used, which contains a blend of two types of artificial dissipation. One of them are computed using the undivided Laplacians (higher-order) of the connecting nodes and the other are the difference in the conserved variables (lower-order) on the connecting nodes. JST is a high-resolution scheme for steady state simulations that involves shock waves. The method tries to retain a second-order spatial accuracy in the smooth flow regions and allows to employ some kind of limiter in the shock region. There are several versions of the JST scheme, some of them which can be reformulated as the total variation diminishing (TVD) scheme [40]. In contrast to the Roe's scheme [41], where the dissipative term is an implicit consequence of how the Riemann problem is solved at the cell interface [42], the JST scheme is based on the discretization of the space using central finite differences with an explicit modelled dissipation term written as follows:

$$\tilde{F}_{ij}^{c} = \tilde{F}\left(\mathrm{U}_i, \mathrm{U}_j\right) = \tilde{F}^{c}\left(\frac{\mathrm{U}_i + \mathrm{U}_j}{2}\right) \boldsymbol{n}_{ij} - d_{ij}, \tag{2.38}$$

where $\boldsymbol{n}_{ij}$ is the outward normal of the face between nodes i and j, U_i is the vector of the conserved variables at point i, $\tilde{F}^{c}$ is the convective flux and d_{ij} is the artificial dissipation term between nodes i and j. On structured grids this term (artificial dissipation) is a blending of second and fourth differences that correspond to low and high order dissipation scaled by the maximum eigenvalue of the inviscid flux Jacobian. Note that the second term damps oscillations in the vicinity of the shock sensed by a pressure-based sensor. The fourth term avoids the decoupling and provides the background dissipation in the smooth regions of the flow. The generalization on unstructured grids is a combination of an undivided Laplacian (higher-order) of connecting nodes and a biharmonic operator. The two levels of dissipation are blended using a pressure switch to stimulate the low order dissipation in the vicinity of shock waves. The artificial dissipation can be expressed along the edge connecting nodes i and j as:

$$d_{ij} = \left(\varepsilon_{ij}^{(2)}\left(\mathrm{U}_j - \mathrm{U}_i\right) - \varepsilon_{ij}^{(4)}\left(\nabla^2 \mathrm{U}_j - \nabla^2 \mathrm{U}_i\right)\right) \varphi_{ij} \lambda_{ij}, \tag{2.39}$$

where λ_{ij} is local spectral radius, φ_{ij} is the scaling parameter, $\varepsilon_{ij}^{(2)}$ and $\varepsilon_{ij}^{(4)}$ are the pressure switches and $\nabla^2 \mathrm{U}_i$ is the undivided Laplacian which is defined as:

$$\nabla^2 \mathrm{U}_i = \sum_{k \epsilon N(i)} (\mathrm{U}_k - \mathrm{U}_i). \tag{2.40}$$

The local spectral radius of the inviscid flux Jacobian can be expressed as:

$$\lambda_{ij} = \left(\left|\boldsymbol{u}_{ij} \cdot \boldsymbol{n}_{ij}\right| + c_{ij}\right)\Delta S_{ij}, \tag{2.41}$$

with the

$$\boldsymbol{u}_{ij} = \frac{\boldsymbol{u}_i + \boldsymbol{u}_j}{2}, \qquad c_{ij} = \sqrt{\frac{\gamma R\left(T_i - T_j\right)}{2}}. \tag{2.42}$$

The scaling parameter, in the implemented version, is expressed as:

$$\varphi_{ij} = 4\frac{\varphi_i \varphi_j}{\varphi_i + \varphi_j}, \qquad \varphi_i = \left(\frac{\sum_{k \epsilon N(i)} \lambda_{ik}}{4\lambda_{ij}}\right)^p. \tag{2.43}$$

Important terms, with strong influence, are the switching parameters $\varepsilon_{ij}^{(2)}$ and $\varepsilon_{ij}^{(4)}$. The function of the parameter $\varepsilon_{ij}^{(2)}$ is to achieve less low-order dissipation in the smooth regions to prevent oscillations in the vicinity of shock waves. Therefore:

$$\varepsilon_{ij}^{(2)} = \kappa^{(2)} s_2 \left(\left|\sum_{k \epsilon N(i)} (p_k - p_i)\right| \Big/ \sum_{k \epsilon N(i)} (p_k - p_i)\right). \tag{2.44}$$

The fourth order dissipation, which is mixed by $\varepsilon_{ij}^{(4)}$, damps high frequencies which helps to reach steady state quickly. The existence of overshoots near shock waves needs to be switched off using:

$$\varepsilon_{ij}^{(4)} = s_4 max\left(0, \kappa^{(4)} - \varepsilon_{ij}^{(2)}\right) \tag{2.45}$$

In the above equations (2.44) and (2.45) some term, or parameters, need to be defined: p_i is the pressure at node i, s_2 and s_4 are stretching parameters, $\kappa^{(2)}$ and $\kappa^{(4)}$ are adjustable parameters. Note that if the parameter $\kappa^{(4)}$ is not properly chosen, there may still be the occurrence of overshoots at shocks waves. This parameter should be large enough to guarantee that the higher order terms are turned off when the lower order dissipation is active [40].

Relatively to the viscous fluxes using an FVM approach, the flow quantities and their first derivatives are required at the faces of the control volumes. The values of the flow variables, including the velocity components, dynamic viscosity and heat conduction coefficient k, are averaged at the cell faces in SU². To compute the gradients of the flow variables the Green-Gauss method, which is based upon a contour integral around a control volume, has been widely employed. The method is not exact even for linear functions and its accuracy is highly dependent on cells shapes. Methods based in a least squares procedure are usually better and produce more accurate approximations to the solution gradients on distorted meshes [43]. The least-squares gradient construction is a technique which is unrelated to the mesh topology. Since SU² has a weighted least-squares method already implemented, it will be used in

this work to calculate the gradients of the flow variables at all grid nodes, that are then averaged to obtain the gradients at the cell faces [44].

2.1.6. Time discretization

Since one of the goals of this thesis is to carry out relatively fast aerodynamic simulations capturing the main flow phenomena for airfoil shape optimization, only steady simulations are considered. Equation (2.37) needs to be valid over the whole time interval. To evaluate the residual, $R_i(\mathrm{U})$, implicit methods, t^{n+1} are chosen [34]. Using implicit integration, the easiest way to discretize the system is through the implicit Euler scheme:

$$\frac{|\Omega_i^n|}{\Delta t_i^n}\Delta \mathrm{U}_i^n = -R_i(\mathrm{U}^{n+1}), \qquad \Delta \mathrm{U}_i^n = \mathrm{U}_i^{n+1} - \mathrm{U}_i^n. \tag{2.46}$$

Since the residual at time $n+1$ is unknown, linearization about, t^n is needed:

$$R_i(\mathrm{U}^{n+1}) = R_i(\mathrm{U}^n) + \frac{\partial R_i(\mathrm{U}^n)}{\partial t}\Delta t_i^n + \mathcal{O}(\Delta t^2) = R_i(\mathrm{U}^n) + \sum_{j \epsilon N(i)} \frac{\partial R_i(\mathrm{U}^n)}{\partial \mathrm{U}_j}\,\Delta \mathrm{U}_j^n + \mathcal{O}(\Delta t^2). \tag{2.47}$$

To handle the right-hand side of the above equation, the chain rule and the implicit Euler discretization for the time derivative was used. The following linear system should be solved to find the solution update $(\Delta \mathrm{U}_i^n)$:

$$\left(\frac{|\Omega_i^n|}{\Delta t_i^n}\delta_{ij} + \frac{\partial R_i(\mathrm{U}^n)}{\partial \mathrm{U}_j}\right)\cdot \Delta \mathrm{U}_j^n == -R_i(\mathrm{U}^n). \tag{2.48}$$

Although implicit schemes are unconditionally stable in theory, a specific value of Δt_i^n is still needed to relax the problem due to its high nonlinearity. A local-time-stepping technique, implemented in SU², is used to accelerate convergence to a steady state that allows each cell in the mesh to advance at a different local time step. The calculation is based on the estimation of the eigenvalues of the inviscid and viscous and first-order approximations to the Jacobian at every node i:

$$\Delta t_i^n = N_{CFL} min\left(\frac{|\Omega_i|}{\lambda_i^{conv}}, \frac{|\Omega_i|}{\lambda_i^{visc}}\right), \tag{2.49}$$

where N_{CFL} is the so-called Courant-Friedrichs-Levy (CFL) number, λ_i^{conv} is the convective spectral radius and λ_i^{visc} is the viscous spectral radius which are described in [29]. More details about numerical methods to discretize space and time can be found in [29].

2.2. Aerodynamics coefficients

In external flows, an important task is the prediction of the forces acting on the surface of the airfoil. Airfoils are 2-D representations of lifting devices, like a cross-section of an aircraft wing, which, due to its curved shape, impose curvature on the streamlines of a fluid flowing around the airfoil. In Figure 4, it is shown an airfoil immersed in a uniform flow at a velocity with a certain angle of attack, α, and its technical terminology. The importance of studying 2-D flows around airfoils comes from the fact that such flows are equivalent to 3-D flows around wings of infinite span. In practice, the wing span is always

finite, but a simplified 2-D models approach can provide a useful information about 3-D flows around a wing, especially for long-spanned wings [45].

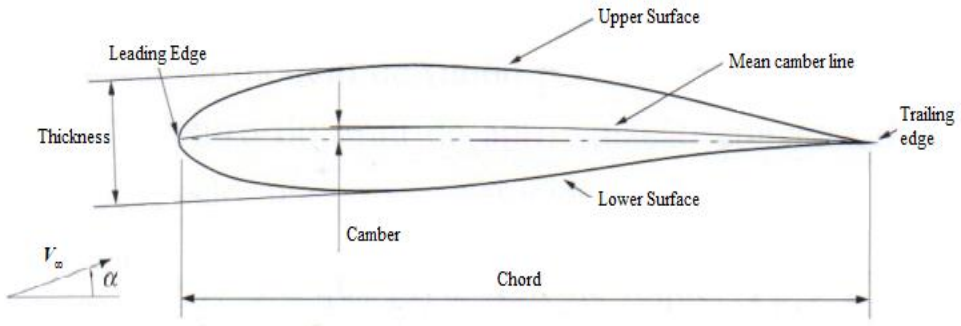


Figure 4 - Diagram of airfoil's geometry [46]**.**

When an airfoil shaped body is immersed in a fluid domain, its movement through the fluid produces an aerodynamic force. The airfoil curvature causes a pressure differential between the upper surface and the lower surface and, when the static pressure on the lower surface is higher than on the upper surface a force acting perpendicular to the direction of the motion is generated, designated by lift force. The motion between the airfoil and the fluid also leads to the appearance of a force component in the streamwise direction: the drag force. Drag is generated by the difference in velocity between the airfoil and the fluid, acting opposite to the motion of the airfoil. In normal operating conditions of the airfoil the drag component is much lower than the lift component [47]. Airfoils shapes are designed to provide high lift values at low drag for given flight conditions. Pressure and shear stress distributions on the airfoil surface, and their integrations over the airfoil area, allows to evaluate the aerodynamic forces. The pressure p acts in the normal direction to the airfoil and the shear-stress τ acts tangentially. The effect of p and τ distributions integrated over the airfoil surface, $S_{airfoil}$, is a resultant aerodynamic force, and moment, M. The resultant force can be split into the lift and drag components, L and D, respectively. The aerodynamics forces applied to an airfoil are usually presented in its non-dimensional form, i.e. the lift and drag coefficients, C_l and C_d, respectively, given according with:

$$C_l = \frac{L}{\frac{1}{2}\rho_\infty u_\infty^2 c}, \tag{2.50}$$

$$C_d = \frac{D}{\frac{1}{2}\rho_\infty u_\infty^2 c}, \tag{2.51}$$

where $\boldsymbol{u}_\infty$ is the free-stream velocity and c is the chord of the airfoil. The moment coefficient, or pitching moment, C_m, relates the twisting moment to a point, usually located at one quarter of the mean chord from the leading edge, and given by:

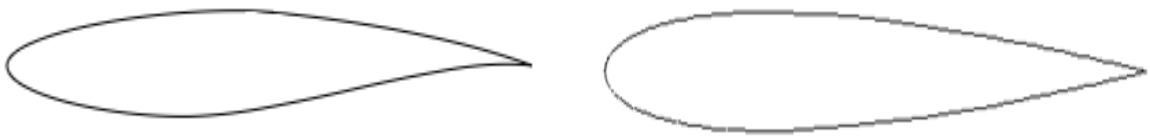
$$C_m = \frac{M}{\frac{1}{2}\rho_\infty u_\infty^2 S_{airfoil} c}. \tag{2.52}$$

Furthermore, the pressure distribution on the airfoil surface can be represented in terms of the pressure coefficient, C_p, defined as:

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty u_\infty^2}, \tag{2.53}$$

where p_∞ is the static pressure in the free-stream flow.

The two airfoils analysed in this thesis are the RAE 2822 airfoil and a modified NACA 0012 airfoil. The RAE 2822 airfoil, also known as the RAE 2822 transonic airfoil, is an airfoil which has become a widely used airfoil for turbulence modelling and is composed of a maximum camber of 1.3%, a camber position of 75.7% of the chord and a maximum thickness/chord ratio of 12.1%, according with [48]. It is often used in CFD in order to model shockwaves and other phenomena in 2-D flow. The NACA 0012 airfoil, one of the most widely used airfoils in verification and validation exercises, is a symmetrical airfoil, without camber, with a maximum thickness of 12% of the chord, designed by the National Advisory Committee for Aeronautics (NACA), currently known as National Aeronautics and Space Administration (NASA). The definition of the modified NACA 0012 is slightly altered from the original definition to allows the upper and lower surface of the airfoil close at a chord equal to one with a sharp trailing edge [49]. The two airfoil are shown in the Figure 5.



(a) (b)

Figure 5 - Geometry of the (a) RAE 2822 and (b) modified NACA 0012 airfoils

2.3. Adjoint methods

Adjoint methods have become a popular method, due, among other factors, to its ability for computing the derivatives at a cost comparable to solving the state PDEs [50]. In these methods, the objective function is augmented with the flow equations enforced as constraints through the use of Lagrange multipliers (adjoint variables). The adjoint methods are appropriate for aerodynamic design optimization when the number of design variables is large in comparison with the number of aerodynamic constraints. The main reason is related with the calculation of the derivatives with respect to all design variables for each objective function and aerodynamic constraint, which can be obtained with a computational effort equivalent to that needed for a single solution of the flow equations. Adjoint method are divided, in a general form, into discrete and continuous adjoint methods. On one hand in the discrete approach [20], [31], [51]–[53], the augmented cost function is discretized before variations are considered, while on the other hand for the continuous approach [17] variations are performed first and then followed by discretization. One advantage of the discrete adjoint method is the fact that, due to the equations being associated to the flow equations, the derivatives obtained are consistent with the finite-difference gradients independent of the size of the mesh. However, this methodology requires the transpose of the matrix that represents the linearization of the residual with respect to the flow variables. In this section focus is given on how to compute analytical sensitivities with adjoint methods.

Solving an optimization problem using a gradient-based algorithm requires the derivatives of the objective function with respect to each design variable as well as the derivatives of all the constraints with respect to the design variables. Consider a vector-valued function $\boldsymbol{I}$ with respect to a vector of independent variables $\boldsymbol{D}$. Computing the derivatives of that vector-valued function yields a Jacobian matrix:

$$\frac{\partial \boldsymbol{I}}{\partial \mathbf{D}} = \begin{bmatrix} \frac{\partial I_1}{\partial D_1} & \cdots & \frac{\partial I_1}{\partial D_{n_D}} \\ \vdots & \ddots & \vdots \\ \frac{\partial I_{n_I}}{\partial D_1} & \cdots & \frac{\partial I_{n_I}}{\partial D_{n_D}} \end{bmatrix}, \tag{2.54}$$

which is a $n_I \times n_D$ matrix, where n_I is the number of output variables and n_D is the number of input variables. Analytical methods are the most accurate and efficient methods to evaluate the sensitivities, requiring knowledge of the governing equations and the algorithm that is used to solve those equations. They are applicable when a quantity of interest $\boldsymbol{I}$ depends on an independent variable of interest $\mathbf{D}$, also known as design variables, as [8]:

$$\boldsymbol{I} = \boldsymbol{I}\big(\mathbf{D}, \mathbf{U}(\mathbf{D})\big), \tag{2.55}$$

where $\mathbf{U}$ denotes the state variables vector that depend on the independent variables through the solution of the governing equations that can be written as residuals. Consider the residuals of the governing equations of a given system:

$$\boldsymbol{r} = \mathfrak{R}\big(\mathbf{D}, \mathbf{U}(\mathbf{D})\big) = 0. \tag{2.56}$$

Residuals need to converge to zero, which in practice corresponds to a small tolerance. The number of equations must be equal to the number of unknowns, the state variables. Disturbances in the variables of this system of equations must result in no variation of the residuals if the governing equations are satisfied. So, since the governing equations must be satisfied, the total derivative of the residuals with respect to the design variables must also be zero [54]. The sensitivity of the numerical residual $\boldsymbol{r}$ to a parameter $\mathbf{D}$ can be written, using the chain rule, as:

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\mathbf{D}} = \frac{\partial \mathfrak{R}}{\partial \mathbf{D}} + \frac{\partial \mathfrak{R}}{\partial \mathbf{U}} \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}\mathbf{D}} = 0. \tag{2.57}$$

The main goal is to obtain the sensitivity of the function of interest that can be an objective function and/or a set of constraint. The function $\boldsymbol{I}$ depends implicitly on the independent and state variables, and the total variation of $\boldsymbol{I}$ using the chain rule. In a similar way, the sensitivity of the objective function $\boldsymbol{I}$ to the parameter $\mathbf{D}$ can be written as:

$$\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\mathbf{D}} = \frac{\partial \boldsymbol{I}}{\partial \mathbf{D}} + \frac{\partial \boldsymbol{I}}{\partial \mathbf{U}} \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}\mathbf{D}}, \tag{2.58}$$

where the result is a $n_I \times n_D$ Jacobian matrix. In equation (2.58) it is important to analyse the equation by distinguishing the total and partial derivatives definition. The total derivative matrix $\mathrm{d}\mathbf{U}/\mathrm{d}\mathbf{D}$ takes into account the change in $\mathbf{U}$ that is required to keep residual equations (2.56) equal to zero. The partial derivatives represent the variation of $\boldsymbol{I} = \boldsymbol{I}(\mathbf{D})$ with respect to changes $\mathbf{D}$ for a fixed $\mathbf{U}$. The second term

on the right-hand-side means the variation of the function due to the change of the state variables when the governing equations are solved. The partial derivatives can be computed using the finite differences methods. However, computing the total derivative matrix in equations (2.57) and (2.58) has a much higher computational cost than any of the partial derivatives due to the need for the solution of the residual equations. The linearized residual equations (2.57) can be rearranged to provide the means for computing the total derivative matrix by solving [11]:

$$\frac{\partial \mathfrak{R}}{\partial \mathbf{U}} \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}\mathbf{D}} = -\frac{\partial \mathfrak{R}}{\partial \mathbf{D}}. \tag{2.59}$$

Recalling the equation (2.58) and substituting the above equation yields:

$$\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\mathbf{D}} = \frac{\partial \boldsymbol{I}}{\partial \mathbf{D}} - \frac{\partial \boldsymbol{I}}{\partial \mathbf{U}} \left[\frac{\partial \mathfrak{R}}{\partial \mathbf{U}}\right]^{-1} \frac{\partial \mathfrak{R}}{\partial \mathbf{D}}, \tag{2.60}$$

with

$$\boldsymbol{\Psi} = -\frac{\partial \boldsymbol{I}}{\partial \mathbf{U}} \left[\frac{\partial \mathfrak{R}}{\partial \mathbf{U}}\right]^{-1} \text{ and } -\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}\mathbf{D}} = \left[\frac{\partial \mathfrak{R}}{\partial \mathbf{U}}\right]^{-1} \frac{\partial \mathfrak{R}}{\partial \mathbf{D}}, \tag{2.61}$$

where $\boldsymbol{\Psi}$ is the adjoint variables matrix. The adjoint matrix is expressed as a vector in the literature [54] but, to be consistent with the formulation used so far, here it is presented as a $n_I \times n_D$ matrix. The adjoint methods allows to compute the sensitives $\mathrm{d}\mathbf{U}/\mathrm{d}\mathbf{D}$ to be used in a gradient-based optimization. The adjoint matrix is in the total derivative equation (2.60). After some mathematic manipulating [55] it is possible to achieve an alternative option for computing the total derivative. Considering the linear system involving the square Jacobian matrix $\partial \mathfrak{R}/\partial \mathbf{U}$ and solving with $\partial \boldsymbol{I}/\partial \mathbf{U}$ as the right-hand side, the following linear system (the adjoint equations) is achieved:

$$\left[\frac{\partial \mathfrak{R}}{\partial \mathbf{U}}\right]^{T} \boldsymbol{\Psi} = -\left[\frac{\partial \boldsymbol{I}}{\partial \mathbf{U}}\right]^{T}. \tag{2.62}$$

This linear system needs to be solved for each column of $[\partial \boldsymbol{I}/\partial \mathbf{U}]^T$, and thus the computational cost is proportional to the number of functions of interest. Substituting the adjoint vector in the above equation, the variation of the objective function is expressed by the total derivative as:

$$\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\mathbf{D}} = \frac{\partial \boldsymbol{I}}{\partial \mathbf{D}} + \boldsymbol{\Psi}^{T} \frac{\partial \mathfrak{R}}{\partial \mathbf{D}}. \tag{2.63}$$

The cost to calculate the Jacobian $\partial \boldsymbol{I}/\partial \mathbf{D}$ using the adjoint method will be independent of the number of design variables and proportional to the number of functions of interest. The adjoint method requires the computation of a different adjoint vector $\boldsymbol{\Psi}$ for each function of interest.

In the literature it is possible to find several and different cases that explore the adjoint methods. Anderson and Venkatakrishnan [17] developed an aerodynamic optimization on unstructured grids via a continuous adjoint approach and analysed it for inviscid and viscous flows. They also detailed the discretization of the adjoint equations. In Palacios *et al.* [29], the continuous adjoint approach is applied to the N-S equations. A framework for ASO problem involves minimizing a desirable cost function with respect to changes in the shape of the boundary S, whose value depends on the flow variables $\mathbf{U}$ obtained from the solution of the fluid flow equations. The continuous adjoint approach is also applied to the RANS equations, by Palacios *et al.* [35].

This page has been left intentionally blank.

Chapter 3

Parameterization Methods

Over the past several decades, different parametric techniques have been successfully applied to represent two-dimensional and three-dimensional geometric configurations, such as an airfoil shape or a wing. Parameterization has acquired a significant role in airfoil design and optimization. The main goal consists in exploring as many design solutions as possible, while keeping the number of design parameters to a minimum. Possible object shapes and shape changes are defined by a set of parameters which are used as design variables during the optimization process. Therefore, the choice of design variables is a crucial step in the optimization procedure. The success of the optimization process is dependent on the parametric description of the object geometry which influences the computational time cost of the optimization as well as the quality of its results. Using an appropriate parameterization technique is crucial for each optimization task [56]. According to Samareh [57], the parameterization process must: (1) be automated; (2) provide consistent geometry changes across all disciplines; (3) provide sensitivity derivatives (preferably analytical); (4) fit into the product development cycle times; (5) have a direct connection to the CAD system used for design; and (6) produce a compact and effective set of design variables for the solution time to be feasible.

Common types of parameterization methods are currently used for ASO. In Masters *et al.*, the various airfoil parameterization methods are categorized as constructive or deformative based on how the design variables/parameters influence the airfoil shape [58]. In constructive methods, the shape parameterization will give optimal shape from scratch representing airfoil shape based on the values of the parameters specified. Examples include polynomials and splines[59], partial differential equation methods [60], and class–shape transformations (CSTs) [61]. Parameterization of sharp edges, creases and other un-smooth profiles could cause difficulties. Deformative methods take the existing geometry shape and deform it to create the new shape like a reference airfoil or wing, such as discrete, basis vector, [57], analytical, like Hick-Henne “bump” function [62], and free-form deformation (FFD) [6], [63], [64] methods. Parameterization over sharp edges, creases and other un-smooth profiles remains smooth [56], [58].

The list of appropriate parameterization methods for ASO is long. Samareh classified the parameterization techniques into eight categories: basis vector, domain element, partial differential

equation, discrete, polynomial and spline, CAD-based, analytical, and free-form deformation [57]. In Sripawadkul *et al.* study, five airfoil parameterization techniques were considered: Ferguson's curves, Hicks-Henne bump functions, B-Splines, PARSEC and Class/Shape function Transformation. These techniques are ranked according to five metrics: Parsimony, Completeness, Intuitiveness, Orthogonality, and Flawlessness [65]. Their purpose is to help the designer in choosing airfoil parameterization methods to be used for multidisciplinary design optimization right from an early design stage [65].

Some methods were specially developed for parameterization of airfoils, like PARSEC [46], but they can be used for wing parameterizations too. Individual cross-sections of the wing are parameterized and interpolation of the geometry between them is used. However, this will limit possible local shape modifications and, for more complex 3-D objects more sophisticated parameterization methods can be chosen [56].

3.1. Polynomial and spline approach

By using polynomial and spline representations for shape parameterization one can reduce the number of design variables in an optimization problem. A polynomial function allows to describe a curve in a very compact form requiring a small set of design variables comparing with a discrete method [57]. A discrete approach will use grid-point coordinates as design variables, Figure 6(a), requiring a larger number of design variables when compared with Bézier and B-spline curves that use a set of control points to assist the airfoil parameterization, Figure 6(b). The Bézier and B-spline curves are well adjusted for shape optimization [57]. Although the polynomial and spline approach considerably reduce the number of parameters required it is not recommended for the leading-edge representation, for example, since it, requires a high-degree polynomial, to properly capture the required shape, which greatly increases the number of parameters required.

(a) **(b)**

Figure 6 - Model of airfoil by grid-point coordinates (a) and control points (b) [57]**.**

3.1.1. Bézier Curve

One of the most popular parameterization techniques to represent an airfoil is through the help of Bézier curves. The Bézier Curves were independently developed around the 60s by two French mathematicians who worked in the French automobile industry. In 1959, the French car company Citroën hired a young mathematician, Paul de Faget de Casteljau, to solve some theoretical problems that arose from the blueprint-to-computer challenge. He began to develop a system which primarily aimed at the design of curves and surfaces adopting the use of Bernstein polynomials and, what is nowadays known as the de Casteljau algorithm, for the curves and surfaces definitions [66]. His work was kept in secret by Citroën for a long time because of its competitor Renault. Pierre Bézier was

another developer of Bézier curves. He headed the design department of Renault and also realized the need for computer representations of mechanical parts and that Bézier curves could be expressed in terms of Bernstein polynomials [67], [68]. The secrets between companies did not allow Casteljau to publish his work. The name of Bézier curves is due to Bézier who got an authorization from Renault and published his work. The curves can be defined as a set of points, by using the Casteljau algorithm or explicitly as a function of Bernstein's polynomials [68].

The Bézier curves are polynomial parametric curves that can be represented in the plane or space. The Bézier curve is a weighted sum of control points and their influence is determined by their weighting $B_{\mathrm{i}}^{n}(\mathrm{v})$. The curve passes through initial and final control points, but it is not mandatory to pass through each intermediate control point. The closer the curve passes to the control points the higher should be this influence [69]. Before presenting the general form for a degree n Bézier curve, consider two distinct points $\mathbf{P}_0$ and $\mathbf{P}_1$ as shown in Figure 7(a). In this simplest case, a first order Bézier curve, the curve is a straight line between those two control points.

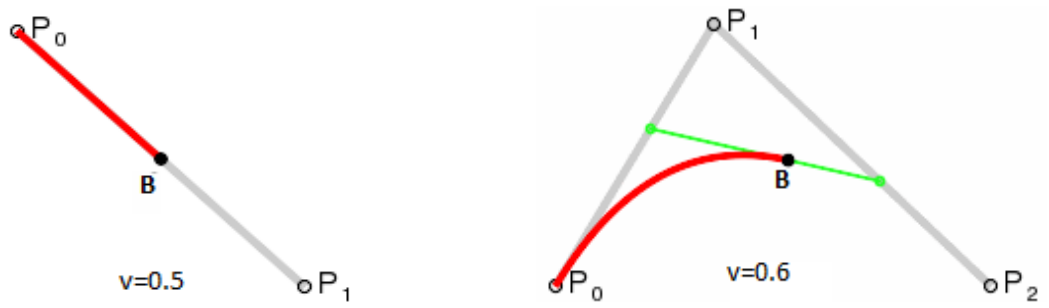


Figure 7 – First order (linear) (a) and second order (quadratic) (b) Bézier curve with control points.

The parametric representation of this linear Bézier curve has the form:

$$\mathbf{B}(\mathrm{v}) = (1-\mathrm{v})\mathbf{P}_0 + \mathrm{v}\mathbf{P_1}, \qquad 0 \le \mathrm{v} \le 1 \tag{3.1}$$

where v is the parameter coordinate along the curve. As v varies from 0 to 1, $\mathbf{B}(\mathrm{v})$ describes a straight line from $\mathbf{P}_0$ to $\mathbf{P}_1$. For instants, when $\mathrm{v} = 0.5$, $\mathbf{B}(\mathrm{v})$ is one half of the way from point $\mathbf{P}_0$ to $\mathbf{P}_1$. For a quadratic curve, second order Bézier curve, the path traced by the function $\mathbf{B}(\mathrm{v})$, given the control points $\mathbf{P}_0$, $\mathbf{P}_1$ and $\mathbf{P}_2$ will describes a segment of a parabola, Figure 7(b), according with:

$$\mathbf{B}(\mathrm{v}) = (1-\mathrm{v})^2\mathbf{P}_0 + 2\mathrm{v}(1-\mathrm{v})\mathbf{P}_1 + \mathrm{v}^2\mathbf{P}_2, \qquad 0 \le \mathrm{v} \le 1 \tag{3.2}$$

A Bézier curve is defined by a set of control points $\mathbf{P}_0$ through $\mathbf{P}_n$, where n is the order. The curve order is equals to the number of control points minus one. In general, a Bézier curve of degree n is defined by $n+1$ control points as described by described by:

$$\mathbf{B}(\mathrm{v}) = \sum_{\mathrm{i}=0}^{n} B_{\mathrm{i}}^{n}(\mathrm{v}) \cdot \mathbf{P}_{\mathrm{i}}, \qquad \mathrm{i} = 0,1,\dots,n, \tag{3.3}$$

where $\mathbf{P}_{\mathrm{i}}$ are the control points, $\mathrm{v} \in [0,1]$ is the parameter coordinate along the curve, B_{i}^{n} are the Bernstein basis polynomials of degree n given by the following general form:

$$B_{\mathrm{i}}^{n}(\mathrm{v}) = \binom{n}{\mathrm{i}} \mathrm{v}^{\mathrm{i}}(1-\mathrm{v})^{n-\mathrm{i}}. \tag{3.4}$$

The binomial coefficients are defined as:

$$\binom{n}{\mathrm{i}} = \frac{n!}{\mathrm{i}!\,(n-\mathrm{i})!}. \tag{3.5}$$

- **Advantages and Disadvantages**

The advantages of Bezier curves are related to the ease of computation, its stability for lower degrees curves and the possibility to allowing rotation and translation by performing the operations on the control points. Although they become unstable at higher degrees, Bézier curves can be split at any point into sub curves of lower degrees to represent any shape.

One disadvantage of the method is related with the control points. The control points can be used to define an airfoil shape. However, the number of points and their locations are not known *a priori*. The method does not provide a full control over the slope of the fitted curve and in some cases may lead to undesirable shapes. The locations of the control points may also influence the optimum shape. Even with the correct selection of control points, a large number of design variables to provide sufficient flexibility for transonic airfoil design can be required thus increasing the complexity of the optimization process and the computation resources [70]. Bézier curves are accurate when used to represent simple curves. The degree of the polynomials must increase as the curve complexity also increases. Higher order curves will require more intermediate points and more computationally expensive to evaluate. In this way, the error also increases. To reduce the error it is more efficient to separate the curve into segments and use a set of low order Bézier curves [65].

3.1.2. B-Splines

As was previously said, complex curves can be represented in a high-degree Bézier form using a set of low-degree Bézier segments to represent the curve. The resulting composite curve is referred as a spline or, more preciously, a B-spline [57]. The B-splines curves, where the “B” means ‘basis’, were proposed by Isaac Jacob Schoenberg in 1946 as a solution for curve approximation problems [67] and are the generalization of the Bézier curve. Used for producing multidimensional smooth curves, B-splines are defined based on the product of a set of basic functions and a set of control points [58]. Multi-segmented B-spline curves are parameterized by a scalar v and can be defined by:

$$W(\mathrm{v}) = \sum_{\mathrm{i}=0}^{m} \mathbf{P}_{\mathrm{i}} \cdot \mathcal{N}_{\mathrm{i}}^{b}(\mathrm{v}), \quad \mathrm{v} \in [0,1], \tag{3.6}$$

where $\mathbf{P}_{\mathrm{i}}$ are the control points, b is the degree and $\mathcal{N}_{\mathrm{i}}^{b}(\mathrm{v})$ is the i th B-splines basis function of degree b and m the number of control points. Just as the Bézier curves are defined as a function of the Bernstein base polynomials, B-splines are defined by basis functions.

B-spines can be used to represent airfoils in a variety of ways. Giammichele *et al.* [71] presented a framework that allows performing multiresolution manipulation of piecewise polynomial B-spline curves with linear constraints on the curve: thickness and camber. They investigated a technique with the

purpose of airfoil design as a novel tool for the generation and optimization of 2-D wing profile. Masters *et al.* [58] adopted a configuration for B-spline method in order to satisfy the constraints imposed by other parameterization methods: each airfoil is represented by two B-splines, one for the upper surface and the other for the lower surface. Two distributions of the control points in the chordwise direction were investigated: a uniform and a cosine distribution. Comparing the results, they found that both configurations provided very similar performance. The dependence of the uniform distribution with higher order polynomials is not desirable because the computation time for initializing the B-spline is related to the degree of the polynomial. Thus, they considered only the cosine distributed configuration. A detailed description about this method can be found in [58].

- **Advantages and Disadvantages**

B-splines have some key characteristics that may make them preferable to other methods. One of them is associated with the control points. Local control of the spline is possible in such a way that moving or changing one control point does not affect the entire spline, but only affects the spline in the vicinity of that point whose extent of this vicinity is determined by the degree of the B-spline [46]. Another useful property is that basis degree *p* controls the location of the influence of the control points. Thus, for a low-order curve, the influence of any change in the control point position will be more localized when compared to a high-order curve [58]. It is important to note that any global control of the curve is disabled preventing easy manipulation. Other disadvantage is the lack of intuitive control and the difficulty when trying to generate an airfoil form.

3.2. Ferguson's Spline

In the late 1950s, parametric surfaces were studied by several companies. Introduced by James Ferguson at the Boeing Company in 1964, this is a formulation with simple math and intuitive behaviour. He first introduced the curve definition in Computer Aided Design (CAD) [67]. A Ferguson spline is a curve $H = (\mathrm{v})$, connecting two points $H(0) = A$ and $H(1) = B$ [46]. Two tangents are imposed to this curve at its end points: $\dot{H}(0) = T_A$ and $\dot{H}(1) = T_B$, where $\dot{H}$ denotes derivative with respect to v, as shown in Figure 8:

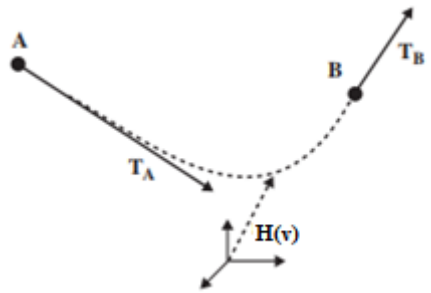


Figure 8 - Ferguson spline and the corresponding boundary conditions (adapted from [46]**).**

The curve is defined as the cubic polynomial:

$$H(\mathrm{v}) = \sum_{\mathrm{i}=0}^{3} \boldsymbol{a}_{\mathrm{i}} \mathrm{v}^{\mathrm{i}} = a_0 + a_1 \mathrm{v} + a_2 \mathrm{v}^2 + a_3 \mathrm{v}^3, \tag{3.7}$$

Where $\mathrm{v} \in [0,1]$, the coefficients, $\boldsymbol{a}_{\mathrm{i}}$, are expressed in terms of A, B, T_A and T_B:

$$\begin{gathered} A = a_0; \\ B = a_0 + a_1 + a_2 + a_3; \\ T_A = a_1; \\ T_B = a_1 + 2a_2 + 3a_3. \end{gathered} \tag{3.8}$$

Solving the equations for the coefficients in terms of the geometric end conditions of the curve:

$$\begin{gathered} a_0 = A; \\ a_1 = T_A; \\ a_2 = 3[B - A] - 2T_A - T_B; \\ a_3 = 2[A - B] + T_A + T_B, \end{gathered} \tag{3.9}$$

and by substituting the coefficients into the Ferguson representation, equation (3.9), we can rewrite the equation as follows:

$$H(\mathrm{v}) = A\eta_0(\mathrm{v}) + B\eta_1(\mathrm{v}) + T_A\eta_2(\mathrm{v}) + T_B\eta_3(\mathrm{v}), \tag{3.10}$$

where $\eta_{\mathrm{i}}(\mathrm{v})$ are known as Hermite polynomials or blending functions given by:

$$\begin{gathered} \eta_0(\mathrm{v}) = (1 - 3\mathrm{v}^2 + 2\mathrm{v}^3); \\ \eta_1(\mathrm{v}) = (3\mathrm{v}^2 - 2\mathrm{v}^3); \\ \eta_2(\mathrm{v}) = (\mathrm{v} - 2\mathrm{v}^2 + \mathrm{v}^3); \\ \eta_3(\mathrm{v}) = (-\mathrm{v}^2 + \mathrm{v}^3). \end{gathered} \tag{3.11}$$

The above equation is essentially a Hermitian interpolation and the bracketed factors can be viewed as its basis functions, see Figure 9.

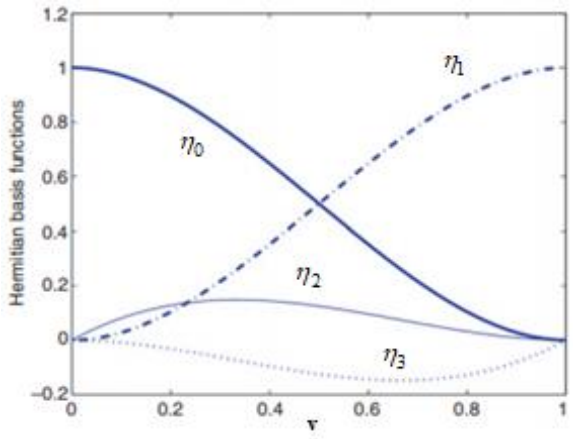


Figure 9 - The four basic functions with respective multipliers (adapted from [46]**).**

The Ferguson spline and a Bézier curve have some similarities. While the end tangents of a Bézier curve are controlled by changing the control points' position, the control points of a Ferguson spline are controlled by modifying the end tangents directly. Both methods are a weighted sum of basis functions and the end points are the only contributor to the curve at its ends [46]. However, the Ferguson spline allows defining explicitly the end tangents unlike the Bézier method. A tangent vector, with direction and

magnitude, has an important meaning for the control of the curve. The greater the magnitude, the less flexible will be the curve in the neighbourhood of the corresponding start/end point.

Based on a pair of Ferguson splines it is possible to generate a parametric airfoil section geometry, one for each surface of the airfoil connected at each end. The concept is simple. Six parameters are required to build a complete airfoil. The curves start from leading edge, A, to trailing edge, B, given also their corresponding tangent vectors T_A and T_B. The tangent of the upper surface $H^{upper}(\mathrm{v})$ in A correspond to T_A^{upper} and its tangent in B to T_B^{upper} (same nomenclature applies for the lower surface). The leading edge-point tangents, T_A^{upper} and T_A^{lower}, must be vertical on both surfaces to ensure first-order continuity. Their magnitudes and orientations can be used to define the shape of the nose of the airfoil. The magnitude defines the tension in the spline. The tangents at the trailing edge, T_B^{upper} and T_B^{lower} are controlled by the boat tail angle, α_b, and the camber angle, α_c, as well as the curvatures of the rear sections of the two surfaces. The camber angle defines the orientation of T_B^{lower} and the boat tail angle defines the orientation of T_B^{upper}. Their magnitudes determine the shape of the middle section of the airfoil [46], [65], [72]. The six design variables are the four tangent magnitudes and the two angles as shown in Figure 10.

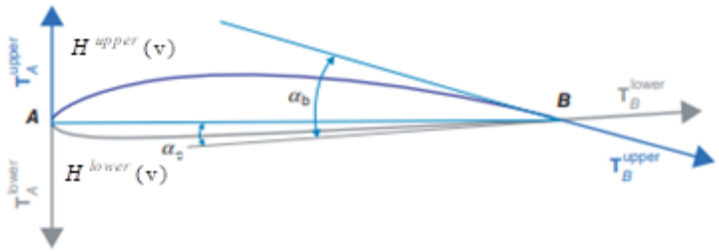


Figure 10 - Two Ferguson splines to represent an airfoil (adapted from [46]**).**

- **Advantages and Disadvantages**

The Ferguson formulation gives a significant amount of shape control of the curve without having to introduce interpolation points between the end points, adjusting the tensions and the directions of the endpoints tangents; it allows the representation of a broad range of airfoils suited to low-speed applications, using only six design variables for a given trailing edge thickness [46], [72]. However, this technique has some limitations. It is unable to reproduce airfoils with multiple inflections or relatively flat portions on either surface like supercritical airfoils (suitable for transonic flow regime) [46].

3.3. Parametric Section (Parsec)

The parameterized sections (PARSEC) method was developed in 1998 by Sobieczky [73] as a system to create airfoils based on meaningful properties. For the purposes of optimization, a simpler description of an airfoil shape is usually desirable. This method is considered to be appropriate for design optimization problems, since the geometric constraints on the airfoil shape can be described by some simple linear constraints. It uses twelve geometric characteristics as design variables to represent an airfoil, manipulating directly the shape, where each design variable is connected to a specific feature of

the airfoil: upper and lower leading edge radius (r_{LE}), upper crest location (x^{upper}, y^{upper}), lower crest location (x^{lower}, y^{lower}), upper and lower curvature at the crests ($y_{xx}^{upper}, y_{xx}^{lower}$), trailing edge ordinate (y_{TE}) and direction (α_{TE}) trailing edge thickness (Δy_{TE}) and wedge angle (β_{TE}) [65], [74]. Figure 11 shows the twelve geometric parameters introduced above and required to define the airfoil.

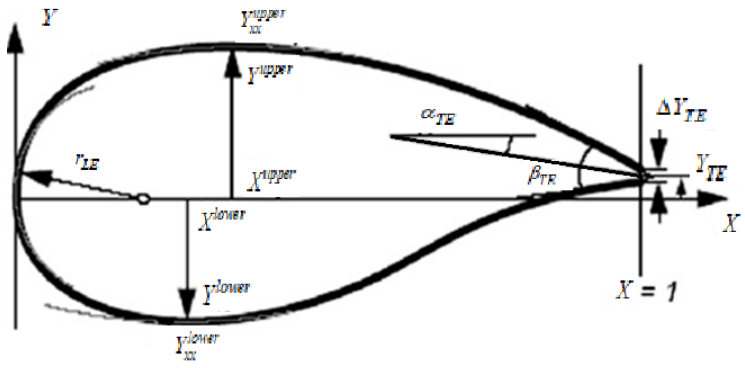


Figure 11 - PARSEC method for airfoil parameterization with eleven design variables (adapted from [70]).

In this method, the airfoil shape is described by a linear combination of shape functions:

$$y^{upper}(x) = \sum_{\mathrm{i}=1}^{6} \boldsymbol{a}_{\mathrm{i}}^{upper} (x^{\mathrm{i}-0.5})^{upper}. \tag{3.12}$$

The unknown coefficients $\boldsymbol{a}_{\mathrm{i}}$ are determined with the help of defined geometric parameters. The airfoil is divided in two surfaces, upper and lower, and the coefficients are determined using information of each section. The system of equations for the upper profile can be found in [65]. The subscript $lower$ is used for the lower surface.

- **Advantages and Disadvantages**

PARSEC method allows a strong control over curvature, prescribing leading edge radius and upper and lower crest curvature. Controlling the curvature on upper and lower surfaces and their location will provide a higher control in the occurrence of a shock wave and its corresponding strength. A disadvantage of this method is related to the trailing edge. It adjusts a smooth curve between the point of maximum thickness and the trailing edge which does not allow to make the necessary changes in the curvature near the trailing edge. Since this method does not provide sufficient control over the trailing edge shape, important flow phenomena can occur without being accounted for [70], [74].

Ferguson's spline and PARSEC method adopt a fixed minimum number of design variables imposed by their formulation, while the other considered methods can use more or fewer design variables depending on the application [65].

3.4. Hicks-Henne "bump" functions

In 1978, Hicks and Henne proposed an alternative shape parameterization technique [62]. Like the PARSEC method, the Hicks-Henne "bump" function was specifically developed to generate airfoil profiles. Their compact formulation has been useful for an aerodynamic purpose such as changing the base geometry in airfoil optimization or to model uncertainties on the geometries of an airfoil. The

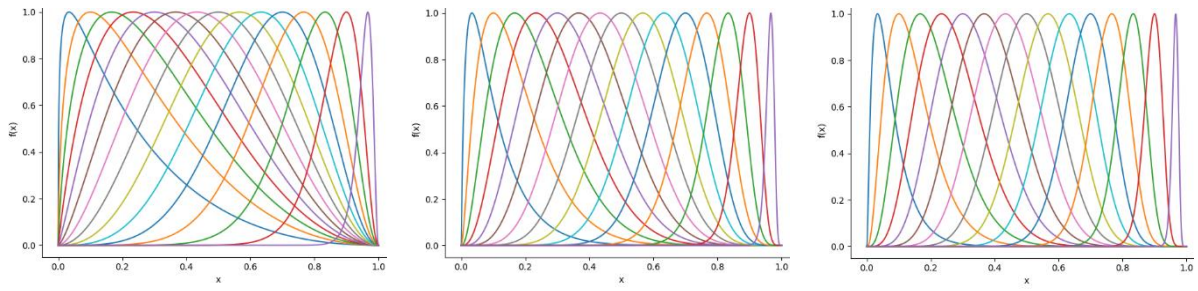
formulation consists in using a base airfoil to which is added a linear combination of a set of m shape (or "bump") functions to perturb the initial geometry. The perturbation is a linear superposition of analytical shape functions, defined through sine functions, on baseline airfoil shape. Using this weighted sum of Hicks-Henne sine "bump" functions it is possible to achieve the desirable parameterized geometry [65]. The baseline airfoil shape, $y_{baseline}$, is deformed to yield a new airfoil shape, y. Each upper and lower surface is deformed to create a new airfoil shape. The new airfoil shape can be written as:

$$y = y_{baseline} + \sum_{\mathrm{i}=1}^{m} \Theta_{\mathrm{i}} f_{\mathrm{i}}(x)\,, \qquad 0 \le x \le 1 \tag{3.13}$$

where m is the number of bump functions, the coefficients Θ_{i} for $\mathrm{i} = 1, \dots, m$, employed to control the magnitude of the shape functions and acts as the weighting coefficients, are the design variables that multiply the various Hicks-Henne functions, $f_{\mathrm{i}}(x)$, defined as:

$$f_{\mathrm{i}}(x) = \left[\sin\left(\pi x^{\frac{\ln(0.5)}{\ln(h_{\mathrm{i}})}}\right)\right]^{t_{\mathrm{i}}}, \tag{3.14}$$

where h_{i} is the location of the maximum point of the bump function and t_{i} controls the width of the bump. The contribution of each parameter is determined by the value of the coefficients Θ_{i} associated with that function. If all the coefficients Θ_{i} are set to zero, the first computation gives the baseline geometry. By analyzing equation (3.14), each bump function is defined by three variables, each of which can be optimized or fixed. To ensure that the optimization process is only a function of the design variables, Θ_{i}, the bump width control parameter t and the bump maximum positions h_{i} are kept fix during the optimization. For the bump width control parameter, a default value of three is set in the SU² code. Thanks to its open source capabilities, the SU² code allows to modify this parameter. Figure 12 illustrate three sets of the Hicks-Henne bump functions for different values of t. As the width control parameter t increases, a narrowing of the bump width is observed. Relatively smaller values of t can provide more global shape control, while a higher value of t will generate more local shape control.



(a) $t = 2$ **(b) $t = 6$** **(c) $t = 10$**

Figure 12 – Sets of Hicks-Henne bump functions with different settings of t for $m = 15, \Theta_{\mathrm{i}} = 1$ and $h_{\mathrm{i}}\, \epsilon\, [0.03,\ 0.97]$

Different authors have carried out various combinations of fixing and optimizing these variables. Zhang *et al.* [75] chose to set the value of the width of the bump at $t_{\mathrm{i}} = 4$ and the bump maximum positions h_{i}

equally-distributed over $[0.5/m, 1-0.5/m]$. Wu *et al.* [76] decided to fix the width of the bump in the same value and let the bump max points h_{i} according to the "cosine" distribution equation:

$$h_{\mathrm{i}} = \frac{1}{2}\left[1 - \cos\left(\frac{\mathrm{i}\pi}{m+1}\right)\right], \qquad \mathrm{i} = 1, \ldots, m \tag{3.15}$$

while Khurana *et al.* [77] opted to vary the three variables. In Masters's *et al.* study [58] it was decided to vary Θ_{i} while keeping h_{i} with the same cosine function as Wu *et al.* and the thickness parameter t_{i} as $2((m-\mathrm{i})/(m-1))^3 + 1$. Figure 13 illustrates a set of sixteen Hicks-Henne bump functions with parameter $t_{\mathrm{i}} = 10$ as defined in [78]. The bump functions reach maximum for the following values of $h_{\mathrm{i}} = 0.05, 0.10, \ldots, 0.95$.

The results obtained in [65] shows that the Hicks-Henne bump functions, which are described with the highest number of design variables can capture a greater number of airfoils, while B-splines capture less, followed by PARSEC and CST; the Ferguson's curves are the ones that capture the lowest number of airfoils, since it describes airfoil geometry with the lowest number of design variables. Depending on the airfoil geometry, the number of design variables necessary for the convergence varies. Studies performed in [58] revealed that the number of design variables required to converge lift and drag coefficients vary between 26 to 82 and 30 to 68, respectively. In some cases, the Hicks-Henne method represents a worse aerodynamic performance than other parameterization methods.

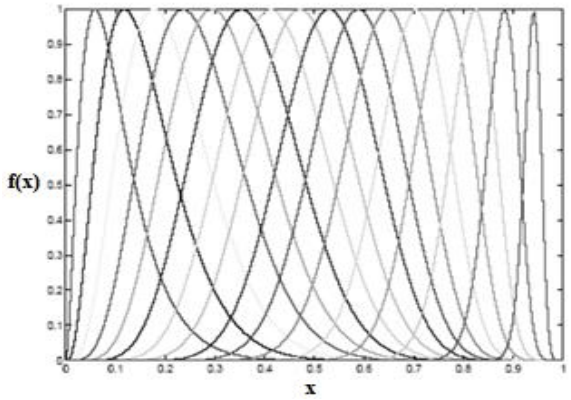


Figure 13 – The Hicks-Henne shape functions [78].

- **Advantages and Disadvantages**

The Hicks-Henne functions can be chosen according to the region that we want to refine by keeping the remaining object to be optimized virtually undisturbed. The result will allow for fewer design variables to provide an appropriate design space. The method has revealed to be able to achieve good approximation to the aerodynamic profiles. According to [57] this method is very effective for wing parameterization. One disadvantage of this parametric method is related to the non-orthogonality. Hicks-Henne functions are not orthogonal which unable them from representing the complete set of continuous functions that vanish at the endpoints $x = 0$ and $x = 1$. Thus, it is not guaranteed that the solution will be attained, which is the case for example in an inverse problem with a certain target pressure distribution. The sine base functions used are non-orthogonal, and thus the solutions are not guaranteed to be attained [78].

3.5. Free-Form Deformation

In the 80's, Sederberg and Parry presented a classic technique for deforming solid geometric models in a free-form manner, that can deform surfaces of any type or degree such as planes or conic sections [63]. The technique was based on previously studies developed by Alan Barr [79] and, four years after Sederberg and Parry's publication, Coquillart [64] has proposed an improvement. Developed for computer graphics, the method can deform almost any type of geometrical model, making it an ideal parameterization method for objects of high geometry complexity. This technique has become a powerful modelling tool for 3-D deformable objects, referred as a free-form deformation (FFD). Coquillart extended the technique that she called Extended Free Form Deformations (EFFD). The intention was to remove one of the limitations of the FFD original algorithm and thus to obtain a bigger variety of deformations [56], [64]. FFD is one of the parameterization methods that deform existing shapes, contrary to Hicks-Henne bump functions and PARSEC, where both are specifically developed to generate airfoil profiles.

The idea of FFD is to embed a flexible object, which will be deformed, into a parallelepiped lattice of control points that define the parametric space. Then, by modifying the lattice, a deformation will be transmitted to the included object in a similar way, creating a new shape [63]. The FFD control volume (or FFD box) has a topology of a parallelepiped when deforming 3-D objects or a rectangular plane for 2-D objects, so that it can be parameterised. FFD parameterization can smoothly deform anything that is embedded within a lattice of control points. Depending on the desired deformation, it is possible with FFD to deform part of the domain while keeping the rest undeformed. This type of deformation is known as local deformation and the borders between the two parts are smoothed. If the whole object is included in the domain, it is designated by global deformation. This parametric method allows to deform a solid model in such a way that its volume is perverted. A good analogy for FFD is to consider the parallelepiped as made of rubber that can be compressed, tapered, stretched, twisted or bent preserving its topology. Mathematically, the parametric space is represented by a volume of 3-D splines and the parallelepiped is the lattice of control points, oriented along the coordinate system, which can be moved to generate deformations. FFD is usually linked with polynomial and spline parameterization techniques. The parametric space can be characterized by Bézier curves, B-splines or non-uniform rational basis spline (NURBS) [56]. Their choice will influence the FFD abilities. The construction of parametric space can be represented by a one dimension (1-D), 2-D or 3-D lattice around, or in the object, that will be deformed. This defines the parametric coordinate system.

The mathematical formulation of the classic FFD method is fully described for 3-D cases in Sederberg and Parry [22]. Here, it is presented the 2-D formulation that is discussed in detail in [80]. The Bézier surface FFD technique, for 2-D cases, deforms a 2-D shape by embedding it with a Bézier surface constrained to a plane which is controlled by the associated control points. The deformation of an initial airfoil $X^{initial} = (x^{initial}, y^{initial})$ with respect to a set of $n \times l$ Bézier surface control points $\mathbf{P}_{ij}$ is given by:

$$X = \sum_{\mathrm{j}=0}^{l} \sum_{\mathrm{i}=0}^{n} B_{\mathrm{i}}^{n} \left(\frac{x^{initial} - x_{\min}}{x_{\max} - x_{\min}} \right) B_{\mathrm{j}}^{l} \left(\frac{y^{initial} - y_{\min}}{y_{\max} - y_{\min}} \right) \mathbf{P}_{\mathrm{ij}}, \tag{3.16}$$

where B_{i}^{n} and B_{j}^{l} are the Bernstein polynomials of degree n and l, respectively. Note that $(x_{\min}, y_{\min})$ and $(x_{\max}, y_{\max})$ are the corners of a deformation region. The number of control points is given by $n+1$ vertical columns and $l+1$ horizontal rows. The control points of the FFD box are defined as design variables, a number which depends on the degree of the chosen Bernstein polynomials. $\mathbf{P}_{\mathrm{ij}}$ are the homogeneous coordinates of the displaced control point i, j defined by $\mathbf{P}_{\mathrm{ij}} = \omega_{\mathrm{ij}}(x_{\mathrm{ij}}, y_{\mathrm{ij}}, 1)$, where ω_{ij} is the weight parameter. Being all the weight $\omega_{\mathrm{ij}} = 1$ and the lattice of control points rectangular, the initial control point positions are given by:

$$\mathbf{P}_{\mathrm{ij}}^{initial} = \left(x_{\min} + \frac{\mathrm{i}}{n}(x_{\max} - x_{\min}), y_{\min} + \frac{\mathrm{j}}{l}(y_{\max} - y_{\min}) \right). \tag{3.17}$$

Regarding the SU² code, the FFD method is numerically executed in three steps. In order to embed the object, a mapping is performed from the physical domain to the parametric domain of the FFD box. This is performed only once before the optimization. The parametric coordinates of each node of the surface mesh are determined and persist unchanged during the design optimization cycle. In a second step, the flexible structure of the FFD control points is perturbed, which leads to the deformation of the FFD box together with the embedded object. Finally, when the FFD box has been deformed, the new cartesian coordinates of the object in the physical domain are computed and a new shape can be studied.

Although there are several versions of FFD in the literature, there is a standard procedure mutual to all of them. In Sarakinos's *et al.* [81] research, it is described the procedure for the most general case, the 3-D case. Amoiralis [82] went further and, in addition to describe the 3-D procedure, presented the differences between the 2-D and the 3-D cases. According to [81], the four main steps for the FFD technique are: construction of the parametric lattice; embedding the object within the lattice; deformation of the parametric volume and evaluating the effects of the deformation. Figure 14 shows an undisplaced FFD lattice of control points and their influence. All the points belonging to the geometric model that are outside the lattice will be not affected by the movement of the lattice.

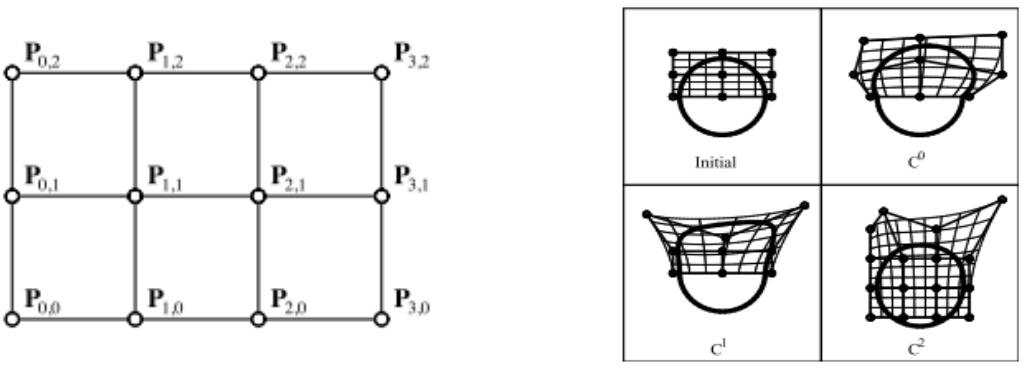


Figure 14 - FFD lattice of control points: (a) without displacement and (b) influence and surface continuity [80].

Currently, there are different variations and conjunctions of FFD technique developed and exploited to increase its abilities by several authors [56], [64], [83], [84]. FFD is a remarkably powerful and versatile tool. Due to its multiple advantages, the method is widely used in the field of geometric modelling and computer graphics [63], [64], [79], [80], [85], [86] , aerodynamic shape optimization of 2-D [81], [82] and 3-D [6], [9], [31], [53] configurations.

- **Advantages and Disadvantages**

The FFD parameterization method has several powerful advantages being one of them the ability to deform almost any type of geometrical model. This ability is due to its formulation being independent of the object's mesh topology. The method can be applied locally, while maintaining derivative continuity with adjacent, undeformed parts of the model, or globally. The FFD technique can control surface continuity as well as volume preservation [56]. It also allows to deform arbitrary shapes with the minimal number of control variables. Beyond solid modelling, this parameterization technique can be applied to surfaces or polygonal models [63]. FFD is very easy to use and a number of FFDs can be used hierarchically to handle specific parameterization tasks, to reach local and global deformations [56]. In shape optimization, FFD provides a simple and intuitive method for both 2-D and 3-D shape parameterization that produces a minimal set of design variables. It facilitates the manipulation of geometric models composed of complex free-from surfaces. Due to FFD being independent of the mathematical definition of the model, it allows to predict the modification made in the grid and the associated CAD model [87].

Although powerful, there are several limitations to this technique and Sederberg and Parry [63] were able to identify three of them. One is related with the impossibility to perform general filleting and blending. However, FFD can be used with any fillet and blending formulation. The second one is when local deformation is applied forming a planar boundary between the deformed and undeformed portion of the object. To form an arbitrary boundary curve, it would be necessary to start with an FFD that is already in a deformed orientation, and then deform a little more; However, this would be costly. The last of the three limitations identified by Sederberg and Parry [63] are operations on trivariate Bernstein polynomials, like subdivision, which are costlier than operations on bivariates. In Richard's book [87] others limitations are mentioned: when a Bézier volume is used to represent the parametric space, computational cost for deformation increases as control points are added; another undesirable limitation is related with the movement of each control point, which produce a global deformation of the entire object, and thus it is not easy to predict the final shape. FFD can only perform a parallelepiped volume which limits its use in adequately handling complex topologies.

This page has been left intentionally blank.

Chapter 4

Aerodynamic shape optimization methodology

The principal aim of the following chapter is to give a brief overview of the open-source framework used to perform an ASO. For this propose, the three open-source software used will be presented and some of their limitations are given. A mesh generator SALOME software will be used and for pre- and post-processing SU² and ParaView will be considered, respectively. In addition, scripts programmed in python were run in order to make an interaction between the different open-source software.

As the first step in both case-study of the RAE 2822 and the NACA 0012 airfoil, an initial baseline geometry is required to design the desirable airfoil. A python script was developed to allow the transformation of an airfoil coordinates' data file to a configuration file which the mesh generator is able to read. The script was programmed such that for a given airfoil a design is generated. Once the mesh has been created around the desired airfoil all the information about the elements and points, together with the type of elements and boundaries, will be exported to a file in a native SU² format. The mesh generator used in this thesis does not allow for a direct exportation of the mesh into a SU² native format file. With a python script, which was programmed for that purpose, it had become possible to link this mesh generator with the SU² solver.

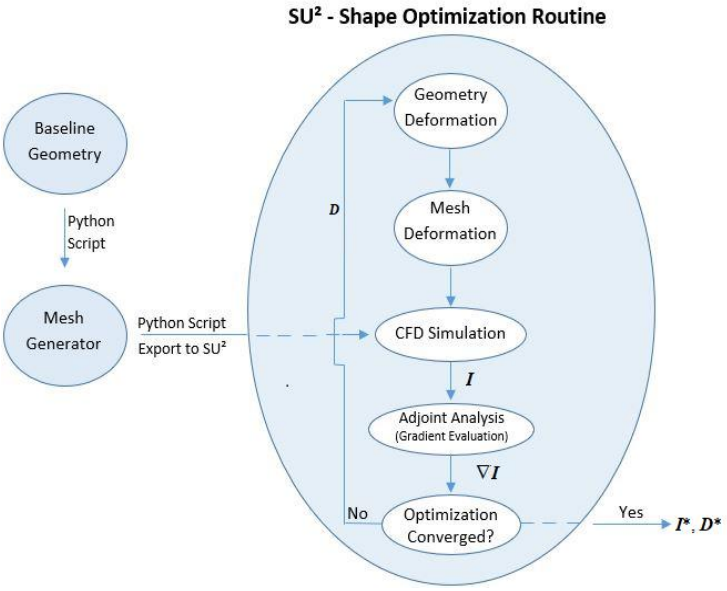


Figure 15 - Open-source framework: optimization procedure.

The mesh file in the SU² native format, will then be used as an input to the shape optimization cycle together with the selected objective function ($\boldsymbol{I}$), constraints and a vector of design variables ($\boldsymbol{D}$) (based on the selected parameterization method). A gradient-based optimizer using the adjoint method is then

enabled to drive the design cycle, and the iterative design loop will proceed until one of the optimization convergence criterions is fulfilled. The procedure is illustrated in Figure 15.

4.1. SALOME Platform

To perform an analysis on the desired mesh it is first necessary to generate it. SU^2 has a native grid format, designed for readability and ease of use, allowing to create simple scripts to generate the appropriate meshes. However, for more complex configurations, grid generation software is more appropriate and should have the capability to export to SU^2 native formats file.

SALOME platform is an open-source software, released under the GNU Lesser General Public License, which provides a generic platform for pre- and post-processing of numerical simulations. Based on an open and flexible architecture made of reusable components, it was initially developed for Linux and later extended to Windows. During the evolution of SALOME, the list of officially supported platforms is constantly changed. For this thesis the SALOME 8.2.0 package developed for the Linux Ubuntu 64-bits distribution [88] was used.

From the point of view of the architecture, Salome provides several integrated modules. Geometry module allows the users to develop their own geometry using a wide range of CAD functions, to modify CAD models as well as importing or exporting in different formats. Mesh module generates meshes on the created geometrical models, or imported CAD models, and allows to import or export a mesh in different formats. SALOME also provides a user interface and a Python console through which all the functionalities of Salome are accessible [89]. SALOME allows to dump a study, with the dump option, which automatically generates a set of Python scripts from data created using Salome GUI. All functions used during the study will be saved in a Python file and the program can be launched in batch mode and operated just with the scripts. The Python scripts can be stored and loaded to recreate the original study [89].

4.1.1. Grid generation

Grid generation, also known as mesh generation, is the art of dividing the physical domain into a number of sub-domains called cells. A well-constructed grid can improve the quality of the solution, being the techniques for creating the cells one of the basis of grid generation. In general, the finer the cells the more accurate the results are, although at a given computational cost. Thus, a proper balance should be defined between the grid size, accuracy of the results and computational time.

The accuracy of the analysis results depends on the selected mesh. There exist three different mesh types for generating the cells using the finite volume method, as shown in Figure 16. On one hand structured meshes are the ones that allow for more accuracy on the results, however are limited to simple cells geometry. On the other hand, unstructured meshes limit the accuracy but are extensible to more complex cells. Hybrid meshes are designed to take advantage of positive aspects of both structured and unstructured meshes by adding local structured meshes in complex flow/geometry regions [90].

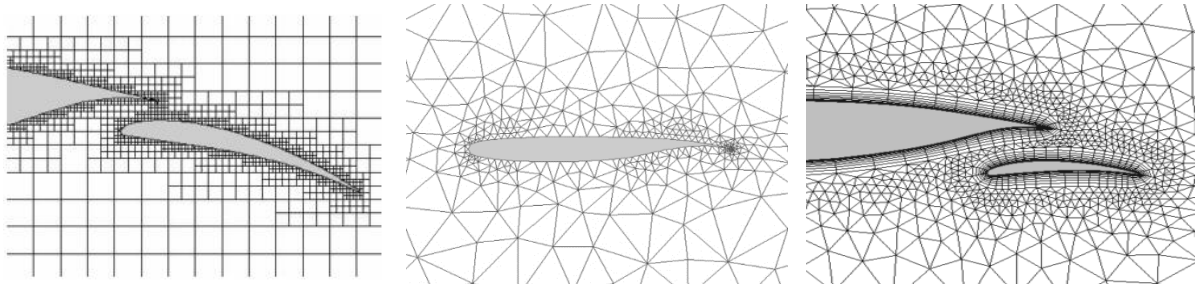


Figure 16 - Grids configuration: a) Structured mesh, b) Unstructured mesh and c) Hybrid mesh [90]**.**

Each type of grid configuration uses one or various basic geometric shapes as cells. In the case of two-dimensional there are two types of cell shapes that are most commonly used: triangles and quadrilaterals. Beyond the surface elements created by triangular or quadrilateral element, there are another type of element used in CFD simulation to represent the volume such as: tetrahedral element (which has four triangular faces and may be used in surface mesh); prism element (with two triangular faces and three quadrilateral faces); pyramid element (with four triangular faces and one quadrilateral face); and hexahedral element (with six quadrilateral faces).

The structured meshes are mostly used due to their simplicity and efficiency. In this type of mesh generation, the domain is typically divided into a structured assembly of quadrilateral cells. A structured mesh is one in which all interior vertices are topologically alike, and each interior node is surrounded by the same number of mesh cells (or elements). This makes possible to easily identify the nearest neighbours of any node on the mesh. They allow to have a better control of the interior node locations and sizes as interior node and sizes. Structured grids are typically aligned in the flow direction which leads to more accurate results and a better convergence for CFD solvers. However, this will lead to orthogonality restrictions. One disadvantage of this grid type is its difficulty or inability to wrap around complex geometry. In general, structured meshes offer simplicity and easy data access, while unstructured meshes offer more convenient mesh adaptivity and are better for complicated domains.

The principle of hybrid mesh is to add structured quadrilateral meshes in the regions where the flow gradients are significant such as vortices, wakes, mixing regions or boundary layer, to better capture the physics of these regions. Using the unstructured grids in hybrid mesh for filling the far-field will allow for a reduction of the number of cells when compared with a completely structured mesh. However, such as for the generation of structure meshes, hybrid meshes requires a priori knowledge of the flow to adequately place the structured grids [90]. Figure 12 depicts a generic hybrid mesh around an airfoil generating with resource to SALOME. This mesh configuration will be used to perform the flow analysis, solving the RANS equations, and the optimizations around the RAE 2822 airfoil case as presented in the next chapter. The mesh was generated considering the ratio between the elements and the skewness. A mesh convergence study was carried out, increasing the level of refinement, in order to achieve more accurate results.

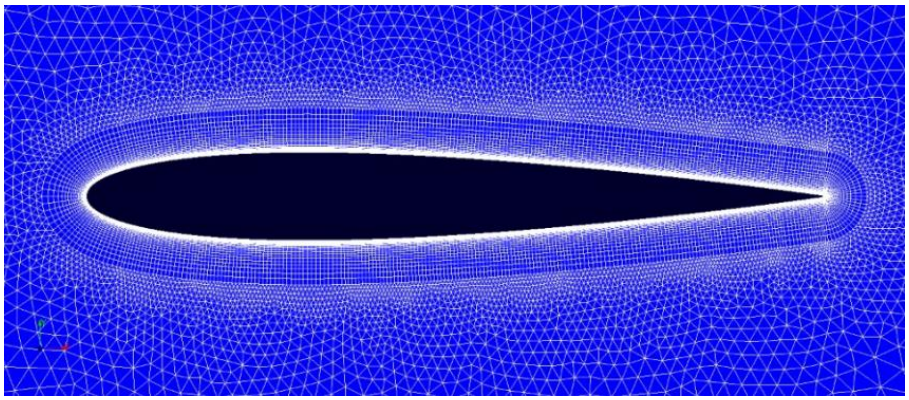


Figure 17 - Exempla of a hybrid mesh around an airfoil with viscous layer.

An issue was observed when increasing the refinement level around the airfoil surface. With the increase of the number of points along the leading edge, a difference between the size elements near the boundary of the viscous layer and the far-field domain is observed. Concerning the external viscous layer shape, this mean the "offset" around the whole airfoil, the edges created perpendicular to the airfoil in the leading edge section do not follow a radius distribution. The mesh quality in that region is very poor with elements with high skewness which can cause problems in the flow analysis and affects all the design optimization process. Without the possibility of increasing the refinement in this grid refinement study, for the RAE 2882 test case performed, the spacing of the mesh points at the leading edge was refined until a configuration similar to Figure 18 does not occur. A configuration with an equidistant spacing points is maintained for the convergence mesh study performed, as illustrated in Figure 19. It must be taken into account that the mesh quality is only considered reasonable due to non-orthogonality errors at the leading edge. This reduce accuracy of flow prediction at the interface.

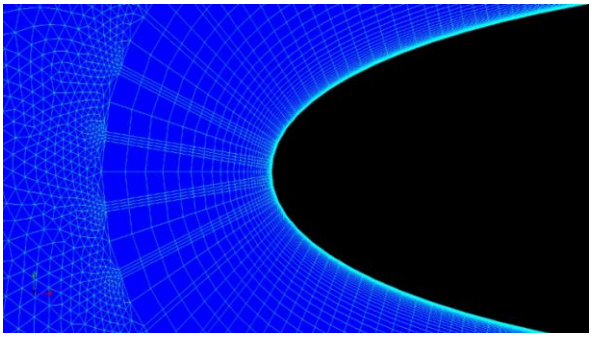


Figure 18 - Increasing of the mesh refinement near to the leading edge cause a poor element quality.

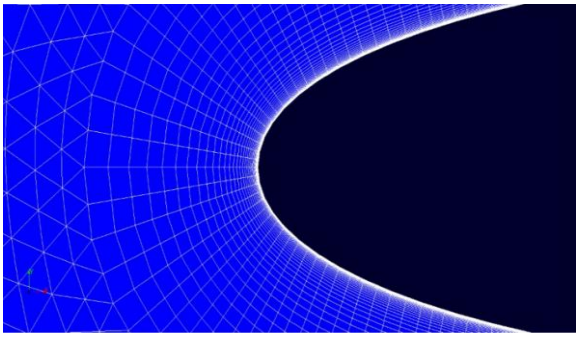


Figure 19 - Leading edge refinement.

4.2. Stanford University Unstructured - SU²

The SU² software is an OS program composed of a set of C++ and Python based software modules for multi-physics simulation and design on three different types of mesh above mentioned. It is under constant development by the Aerospace Design Lab (ADL) of the Department of Aeronautics and Astronautics at Stanford University in collaboration with several universities located all over the world. It has been developed for solving PDEs analyses and PDEs-constrained optimization problems, including

optimal shape design. Primarily developed for CFD analysis and aerodynamic shape optimization, nowadays it is extensible to other physical governing equations (*e.g.* wave equation and plasma equations). SU^2 has a broad community of developers that contribute to the advance of this tool in difference areas.

The code has been divided into several primary modules, where each software module is designed for a specific function. The C++ based software modules aim at performing a wide range of CFD analysis and optimization tasks with PDE constraints. Although some modules can be executed individually, the coupling of the modules with the optimization framework allows the software to perform much higher complex activities like design shape optimization. To couple the various software modules of SU^2 for carrying out complex analysis and design tasks, the developers apply Python scripts that share the same C++ class structure. The shape optimization Python script is available with all the C++ modules needed to orchestrate a shape optimization. These characteristics make the SU^2 an ideal platform for new numerical methods, discretization schemes, governing equation sets, mesh perturbation algorithms, adaptive mesh refinement schemes, and parallelization schemes, among others. More details can be found in [29] and [34].

The execution of the SU^2 occurs via command line arguments in a native terminal application. Each one of the C++ modules can be executed directly from the command line without requiring Python scripts. However, the coupling of C++ modules is only possible via the execution of Python scripts. The most relevant modules called by the optimiser for this study are: SU^2_CFD (main module to perform the flow analysis by solving governing PDEs); SU^2_CFD_AD (module to perform the adjoint flow simulations using the AD); SU^2_DOT_AD (gradient projection module to compute the gradient of the objective function with respect to the design variables, from a suitable geometry parameterization method, by projecting the adjoint surface sensitivity into the design space through a dot product operation); SU^2_DEF (mesh deformation module to compute the geometry surface deformations and deforms the surrounding volume mesh given a set of design variables during the optimization process) and SU^2_GEO (geometry definition and constraints module to evaluates the airfoil thickness and its gradient with respect to each design variable) [34].

SU^2 has been designed with ease of installation and use in mind. To perform shape design optimization the shape optimization Python script is available with all the C++ modules needed. Python scripts have additional dependencies that are extremely important. The NumPy and SciPy packages are open-source Python libraries fundamental for scientific computing and technical computing. SciPy library provides efficient numerical routines for numerical integration and optimization. The SciPy optimization package provides several commonly used optimization algorithms. One of them is the sequential least squares programming (SLSQP) optimization algorithm available in the SU^2 shape optimization script [34]. This script coordinates and synchronizes the steps to run a shape optimization problem using the specified design variables and objective function. The SciPy and NumPy packages are freely available and more details can be found, respectively, in [91] and [92].

SU^2 includes integrated support for algorithmic differentiation (AD) based on operator overloading to compute arbitrary derivatives, such as, the discrete adjoint solver. For many applications, such as optimization, it is important to obtain gradients of the responses computed by SU^2 to variations of a

potentially very large number of design parameters. From the point of view of design optimization, SU² includes continuous and discrete adjoint solver implementations for efficiently computing shape design gradients [29].

The capabilities of SU² can be extended using externally-provided software. SU² allows parallel computation, which requires an installed implementation of Message Passing Interface (MPI) for data communication. A variety of implementations of the MPI standard is freely available. The OpenMPI is an open source MPI library project implemented in the Linux distribution that combine expertise, technologies and resources from all across the high-performance computing community. The ParMETIS software package available on SU² source code is used to perform the partitioning. ParMetis is an MPI-based parallel library that implements a variety of algorithms for partitioning unstructured graphs, meshes, and for computing fill-reducing orderings of sparse matrices [93].

4.2.1. Fluid dynamic solver

The CFD module implemented in SU² is able to solve different flow models such as Euler and RANS. In this work, the compressible Euler and RANS equations are used to describe the flow around the modified NACA 0012 airfoil and the RAE 2822 airfoil, respectively. The SU² solver supports several boundary condition types, such as: Euler (flow tangency) and symmetry wall, no-slip wall (adiabatic and isothermal), far-field and near-field, inlet boundaries based in characteristics (stagnation, mass flow or supersonic conditions prescribed), outlet boundaries based in characteristics (back pressure prescribed), Riemann and Dirichlet, among others [35]. The list of available boundaries conditions makes the SU² a suitable solver for external and internal flows. In the current work, only the boundaries conditions applied to external outflow, presented in chapter two, are considered. For steady problems, SU² use the free-stream fluid state as initial condition for the mean flow. The RANS equations with the boundary conditions applied to the domain and the airfoil can be described as [34]:

$$\frac{\partial(\mathbf{U})}{\partial t} + \nabla \cdot \mathbf{F}^{C} - \nabla \cdot \mathbf{F}^{\upsilon} - \mathbf{S}_{\emptyset} = \mathfrak{R}(\mathbf{U}) = 0 \text{ in } \Omega, t > 0. \tag{4.1}$$

$$\boldsymbol{u} = 0 \text{ on } S, \tag{4.2}$$

$$\left(\frac{\partial T}{\partial n_s}\right) = 0 \text{ on } S, \tag{4.3}$$

$$(\mathbf{C})_{+} = \mathbf{C}_{\infty} \text{ on } \Gamma_{\infty}. \tag{4.4}$$

The governing equations are solved using the FVM. The convective fluxes can be discretized using central or upwind methods in SU². Several numerical schemes have been implemented, such as JST, Roe, Lax-Friedrich and HLLC among others; and due to code architecture, a rapid implementation of new schemes is possible. The classical JST scheme is used to discretise the equations in space and the solution is marched forward in time using the implicit Euler scheme until a steady state is achieved. To deal with the viscous terms, the one-equation SA turbulence model, described in chapter two, is used.

4.2.2. Adjoint solver and gradient evaluation

One of the key features of SU² is its capability to perform optimal shape design based on the discrete and continuous adjoint approach. For the adjoint solver, SU² is able to solve the adjoint Euler and RANS equations. Both adjoint approach have been implement but only the discrete approach is used to solve the adjoint Euler and RANS equations in this work. SU² uses the adjoint formulation to compute the sensitives of a wide range of objective functions, commonly required in optimization problems, such as drag minimization, lift maximization, pitching moment or aerodynamic efficiency (which corresponds in SU² to lift-to-drag ratio) maximization. Also, it allows several constraints to be applied: fixed non-dimensional flow parameters (minimum lift, maximum drag and pitch moment, among others) and geometrical estimations (maximum or minimum required area, thickness and curvature, among others) [29]. Once the adjoint script has called the SU2_CFD module to generate a flow solution, the SU2_CFD_AD module is then called to run an adjoint computation based on the objective function specified in the configure file. Next, the SU2_DOT_AD module is called to map the surface sensitivities onto the specified design variables. When the gradient of the objective function is computed, the gradient-based optimization algorithm is called to guide the search for the optimal solution.

In gradient-based optimization, the optimizer needs to establish a new search direction and, for this propose, it is necessary for the gradient to be evaluated in each design step. In the SU² software, the objective and the constraint functions gradients are computed through the chain rule by relating the geometric and surface sensitivities [94]. The sensitivity of the function of interest, which can be the objective or constraint functions with respect to the design variables, is computed following the equation (4.5):

$$\begin{bmatrix} \frac{\partial I}{\partial D_1} \\ \vdots \\ \frac{\partial I}{\partial D_{n_D}} \end{bmatrix} = \begin{bmatrix} \frac{\partial s_1}{\partial D_1} & \cdots & \frac{\partial s_\sigma}{\partial D_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial s_1}{\partial D_{n_D}} & \cdots & \frac{\partial s_\sigma}{\partial D_{n_D}} \end{bmatrix} \begin{bmatrix} \frac{\partial I}{\partial s_1} \\ \vdots \\ \frac{\partial I}{\partial s_\sigma} \end{bmatrix} \tag{4.5}$$

where σ is the number of surface mesh nodes and s_i $(i = 1, \dots, \sigma)$ are the local surface normal displacement for each discrete mesh node on the geometry surface. The term $\partial \boldsymbol{I}/\partial s$, referred as the surface sensitivities, represents the variation of the function of interest with respect to infinitesimal perturbations of the geometry shape in local surface normal direction. To find the sensitivities of the function of interest for the surface based deformations, it is necessary to consider a small perturbation of the boundary S which, without loss of generality, can be parameterized by an infinitesimal deformation of the size δS along the normal direction to the surface S. The variation of $\boldsymbol{I}$ due to the infinitesimal deformation can be evaluated with the solution for the adjoint equations, assuming a regular flow solution and a smooth boundary S [95]. At each node of the mesh, the surface sensitivities are calculated by solving the adjoint equations only once, which offers a reduction in computational cost.

The geometric sensitivities, given by the Jacobian $\partial s/\partial \boldsymbol{D}$, measure the influence that the change of design variables has on the position of each grid point on the surface mesh. For a given parameterization method of the surface deformation through a set of desire variables $\boldsymbol{D}$, the sensitivity of the surface with respect to the design variables, $\partial s/\partial \boldsymbol{D}$, is computed using finite difference method. Thus, the

computational cost is negligible in comparison to the cost of the flow solution since it does not require the solution of governing PDE equations [94]. The gradients $\partial \boldsymbol{I}/\partial \boldsymbol{D}$ are then calculated by the gradient projection module through a dot product operation between the geometric and surface sensitivities, equation (4.5).

4.2.3. Optimization framework

The optimization algorithm allows for the minimization/maximization of the objective function of one or more design variables. The choice for the solver is related to the available options linked with SU^2 which limits the choice. The optimization method used in this work is the Sequential Least SQuare Programming (SLSQP) algorithm. SciPy have other types of optimization solvers available [91]. The solver options allow for the selection of the objection function and constraints, the boundaries of the design variables, the tolerance value of the objective function in the stopping criterion, the step size used for numerical approximation of the Jacobian and the maximum number of iterations. The convergence of the optimization process is achieved when the Karush-Kuhn-Tucker (KKT) optimality condition is within the tolerance imposed (10^{-10} by default). However, if this parameter is not satisfied the optimization process will stop by the maximum number of iterations exceeds (100 by default).

4.3. ParaView

The post process of the solution files requires a data visualization tool. In order to maintain an open-source philosophy and due to the fact that SU^2 software supports .vtk output formats, the ParaView application was chosen. ParaView is an open-source, multi-platform data analysis and visualization application developed to analyse extremely large datasets using distributed memory computing resources. All the data visualization was performed using ParaView 5.4.1 version for Linux Ubuntu 64-bits distribution [96].

Chapter 5

Aerodynamic shape optimization results

Two optimization cases were employed to evaluate the impact of the selected parameterization methods (FFD and Hicks-Henne bump functions). The first one consists in minimizing the drag coefficient of RAE 2822 airfoil, which is a test case of the AIAA Aerodynamic Design Optimization Discussion Group (ADODG). In the second case, the aim is to maximize the lift to drag ratio for two Mach conditions, starting from a NACA 0012 airfoil.

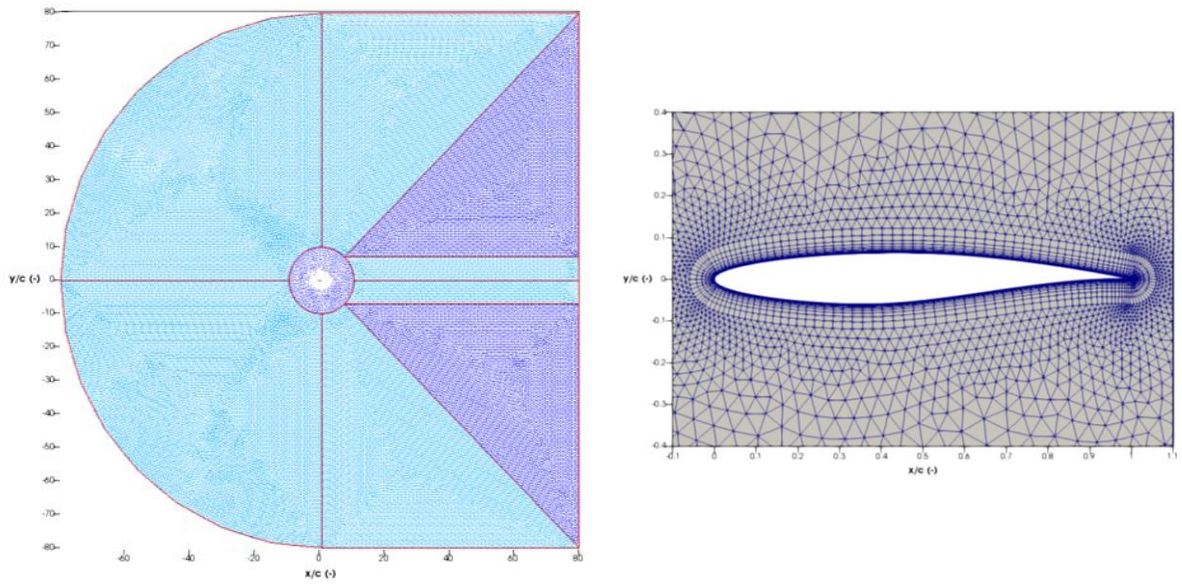
5.1. Case 1: Drag minimization of the RAE 2822 airfoil

The first optimization problem is the drag minimization of the RAE 2822 airfoil in viscous conditions, submitted to a transonic, turbulence flow, which requires solving the RANS equations. The RAE 2822 airfoil is a supercritical airfoil commonly used for validation of turbulence models and aerodynamic shape optimization. The ADODG proposal [22] is considered to minimize the drag coefficient of the RAE 2822 for the conditions of case nine published in the AGARD report [48]. The free-stream Mach number of 0.734, the angle of attack of 2.79 degrees that corresponds to a wind tunnel correction for a lift coefficient of 0.824, the Reynolds number of 6.5 million considering a unitary chord, and the free-stream temperature of 288.15K are assumed for the optimization. Furthermore, a ratio of specific heats of 1.4 and a Prandtl number of 0.71 are employed as suggested in the ADODG proposal [22]. The lift coefficient is constrained to 0.824 and the pitch moment evaluated at quarter-chord must not be less than -0.092. The airfoil area must be greater than or equal to the baseline area during the optimization design process. The optimization problem can be summarized in a mathematical formulation with the objective function and the associated constraints as follows:

$$\begin{aligned} \text{minimize} \quad & C_d \\ \text{subject to} \quad & C_l = 0.824 \\ & C_m \geq -0.092 \\ & S_{airfoil} \geq \left(S_{airfoil}\right)_{baseline} \end{aligned}$$

5.1.1. Grid convergence study

A C-topology multiblock grid is used in this case study, which is displayed in Figure 20(a). The airfoil coordinates are based on the design coordinates present in the AGARD report from 1979 by Cook *et al.* [48]. The far-field is positioned approximately at 80 chord lengths away from the airfoil to minimize any artificial reflections from the outer boundary conditions and to establish the whole field. Different refinement areas were created to capture the physics of the problem and reduce computational cost, by giving more detail in the critical areas. The aerodynamic body is enclosed by one family grid lines, which also form the wake region in the viscid case. The flow is characterized by a normal shock on the upper surface interacting with the boundary layer, which thickness increases until the end of the trailing edge. To account for the shock-wave boundary layer interaction on the suction side and benefit from the flexibility of unstructured grids, a hybrid mesh formed by quadrilateral and triangular finite volumes was selected as previous mention. The mixed-element grid features a region of quadrilaterals wrapped around the near-body region of the airfoil, similar to a C-grid topology for a structured mesh. A non-uniform distribution of points is set for the upper and lower part of the airfoil [35], [97]. Small grid points spacing is set at leading and trailing edges of the airfoil (see Figure 20(b)).



(a) **(b)**

Figure 20 - Case 1: Computational domain (a) and medium mesh (b) for the RAE 2822 airfoil.

Following a similar procedure as in [98] to establish grid convergence, the coarse grid was repeatedly refined in each direction by a refinement factor of two for the medium grid and a factor of three for the fine one. Three levels of mesh are generated with the same topology. The first grid point off the airfoil surface is set at a distance which ensures a y^+ value below one (since the SA turbulence model was chosen) on the airfoil, *i.e.* $y^+ \leq 1.0$. A characteristic-based far-field boundary condition is applied on the far-field boundary, and a no-slip, adiabatic condition is applied on the airfoil. The grid study developed is summarized in Table 1. The compressible RANS simulations were then conducted on each grid level by fixing the angle of attack at 2.79 degrees. The aerodynamic coefficients obtained at convergence are

reported in Table 2. These results were computed considering that the residual of the density equation is less than 10^{-6} in respect to the initial value.

Table 1 - Case 1: Mesh convergence study: Gird parameters for the RAE 2822 airfoil.

Mesh Refinement	Number of grid points	Number of elements	$N_{Surf.Airf.}$	Off-wall Spacing
Coarse	16247	28442	132	$2.7 \cdot 10^{-6}$
Medium	47837	90126	156	$9.9 \cdot 10^{-7}$
Fine	176836	341130	327	$3.6 \cdot 10^{-7}$

Table 2 - Case 1: Results of the mesh study for the RAE 2822 airfoil: Comparison of global coefficients.

Mesh Refinement	y^+	C_l	C_d	C_m
Coarse	1.05	0.7933	200.48	-0.0912
Medium	0.42	0.7916	191.57	-0.0904
Fine	0.15	0.8014	192.44	-0.0922

In the mesh convergence study, all meshes capture the shock position and present similar behaviour between each other in the lower and upper surface. Despite the penalty in the accuracy of the results, the medium mesh is chosen since it allows for faster calculations with a reasonable accuracy (relative errors to the fine grid of 0.92% and 1.43% for C_l and C_d, respectively). To evaluate the accuracy of the medium mesh, numerical results were obtained and compared to the experimental results for RAE 2822 case nine from AGARD [48]. The pressure distribution for the medium mesh can be observed in Figure 21 and Figure 22.

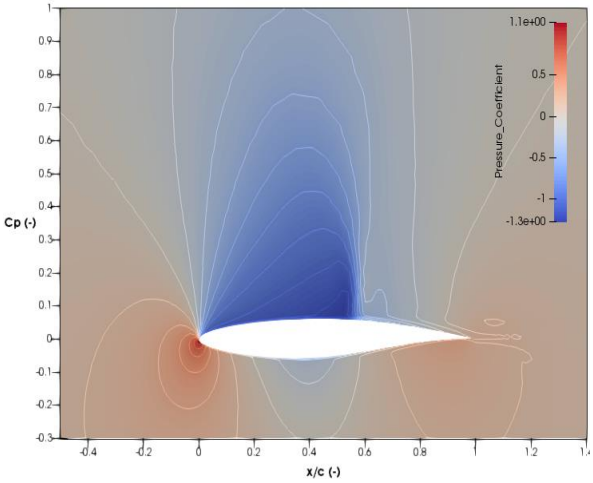


Figure 21 - Case 1: Pressure contours around the RAE 2822 airfoil for the medium mesh.

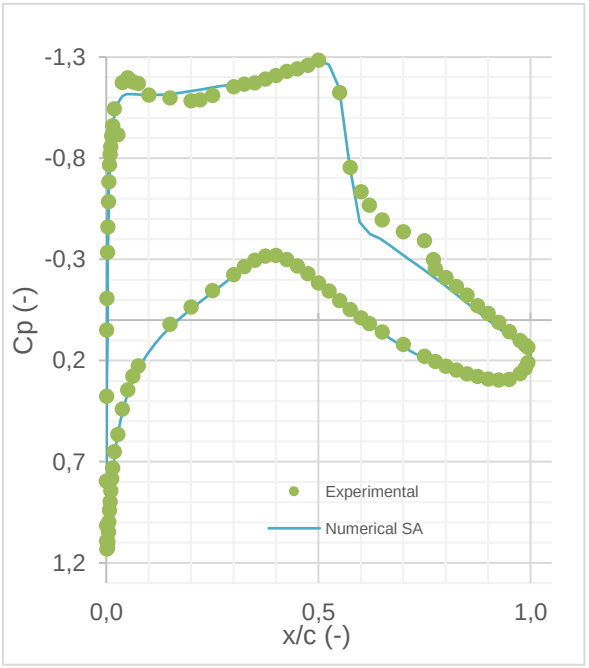


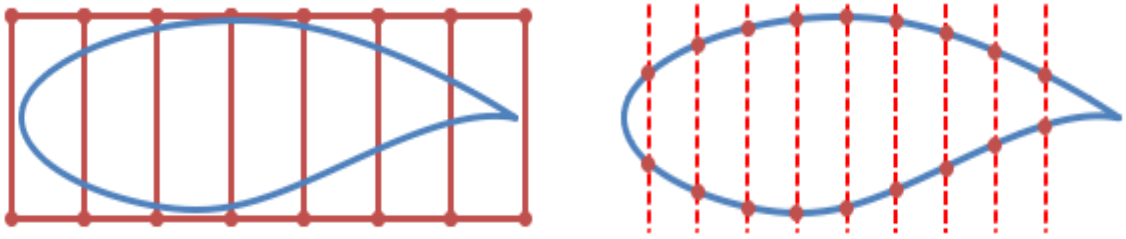
Figure 22 - Case 1: Comparison of experimental and numerical pressure coefficient on medium mesh level.

According to the experimental case nine setup, the wind tunnel free-stream conditions correspond to a Mach number of 0.730, an incidence angle of 3.19º and Reynolds number of 6.5 million based on a unitary chord. Unfortunately, the experimental data suffers from wind tunnel influences, which cannot be quantified *a posteriori*. The AGARD flow conditions were employed with the wind tunnel corrected angle of attack of 2.79º proposed by ADODG [22]. Case 9 is characterized by a moderate shock that does not induce separation (see Figure 21). The agreement between the experimental results and the ones obtained numerically for the medium mesh with the SA turbulence model is revealed in Figure 22 through the pressure coefficients distribution. As one can see, the numerical results shows good agreement with the experimental ones in the lower surface of the airfoil; and in the upper surface the occurrence of the shock is captured with its position located slightly before the experimental one, due to the selected turbulence model. According to Franke *et al.* study [32], other turbulence models were shown to be able to better capture the shock position when compared to SA for this specific case. However, only the SA turbulence model was employed for the current analyses. The C_d of the experimental test is 0.0168 while the numerical one, considering the medium mesh, is 0.0170, which corresponds to a relative difference of 1.19%. Furthermore, the difference between numerical and experimental pitch moment coefficient and normal force coefficient is 12.12% and 2.12%, respectively.

5.1.2. Geometric parameterization impact and Sensitivity analysis

To examine the effect of design space dimensionality: the optimizations with the FFD control point approach were conducted varying the number of design variables (starting with a total of 6 until reaching 30) defined along the top and bottom parts of the FFD control volume (or FFD box); and the optimizations with the Hicks-Henne bump functions were conducted with an increase number of chordwise design variables (from 10 to 30) distributed on the upper and lower surfaces of the RAE 2822 airfoil. The basic FFD concept, as described in chapter 3, consists of embedding the airfoil inside a flexible volume and deforming both simultaneously by perturbing the lattice of control points in the FFD box. The FFD box has a topology of a rectangular plane for 2-D objects. The Bézier curves are available in SU^2 as FFD blending functions and they are used in this study. Figure 23(a) illustrates the FFD box encapsulating the RAE 2822 airfoil, where a lattice of control points is uniformly spaced on the surface of the FFD box. The control points of the FFD box are defined as the design variables, the number of which depends on the degree of the chosen Bernstein polynomials. Performing an FFD approach can be done in three steps.

First of all, the construction of the parametric lattice to embed the airfoil is carried out, performing a mapping of the physical space to the parametric space of the FFD box. The parametric coordinates of each surface grid node are determined, and they will remain unchanged during the optimization. The mapping is evaluated only once in each design cycle. Secondly, the FFD control points are perturbed to deform the FFD box as well as the embedded airfoil. Lastly, after the FFD lattice of control points has been deformed and the effects of the deformation have been evaluated, the new Cartesian coordinates of the embedded airfoil in physical space are algebraically calculated using the equation (3.16).



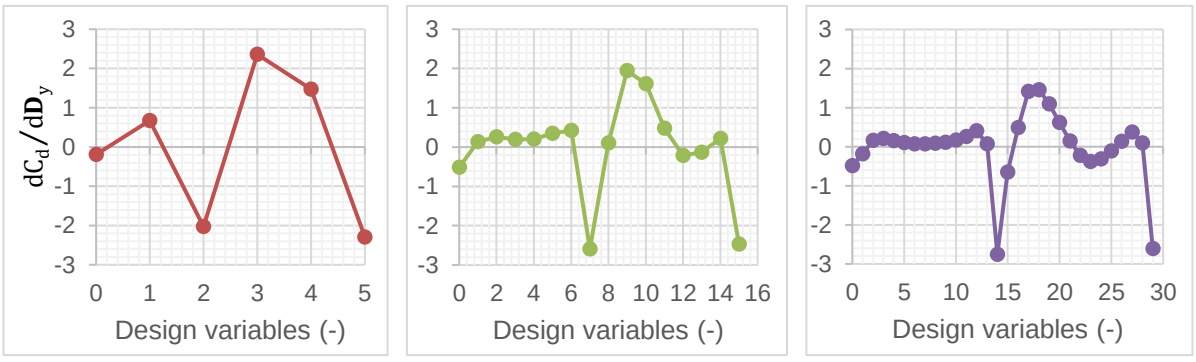
(a) Initial FFD optimization setup.

(b) Initial Hicks-Henne optimization setup.

Figure 23 - Case 1: RAE 2822 airfoil with an initial of 16 FFD design variables between the top and bottom of the FFD box (a) and initial 18 chordwise geometric design variables per surface for the Hicks-Henne bump functions (b).

In the case of the Hicks-Henne bump functions approach, the initial geometry of the RAE 2822 airfoil is considered and a linear combination of bump functions for the upper and lower surface of the airfoil are used to create a new shape. As previous said in chapter 3, each bump function is defined by three parameters, each of which can be fixed or varied during the optimization. For the bump width parameter, t_i, a constant value is considered as specified in the SU² code which, by default, is set to three. Relative to the bump maximum positions, h_i, a uniform distribution over the range of $[0.5/m, 1 - 0.5/m]$ is taken into account. Thus, one can ensure that the parameterization is a linear function of the design variables, since only the bump magnitude coefficients θ_i can vary and therefore treated as design variables, keeping the other parameters fixed. Figure 23(b) shows the initial Hicks-Henne optimization setup for 18 design variables on the RAE 2822 airfoil. The red dashed lines specify where the bump function is applied in the upper and lower surface.

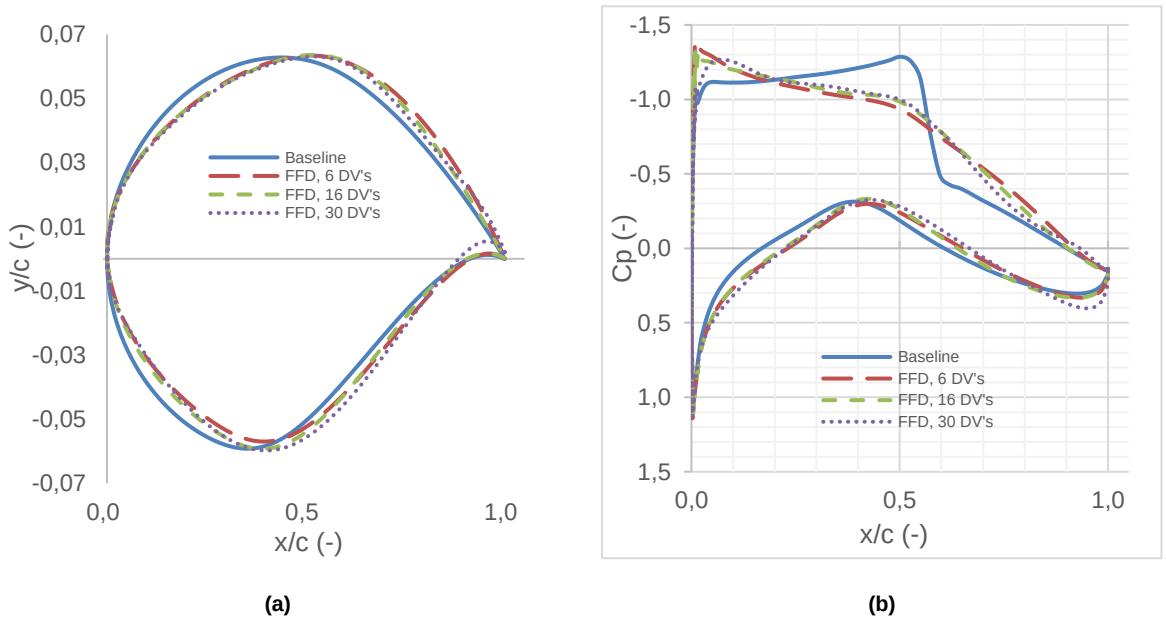
After solving the flow analysis, the adjoint analysis is performed which offers an efficient approach for calculating the gradient of the drag function and the constraints with respect to the design variables. This will be used directly for the gradient-based optimization framework. Figure 24 shows the computed drag sensitivities with respect to vertical motion for the cases with 6, 16 and 30 FFD design variables. All of them show the same trend, showing the curve more details with the increase of the number of design variables. The same behaviour was also noted by Economon *et al.* [99] for the RAE 2822 airfoil in viscous conditions, submitted to a transonic flow for the same Reynolds number but with a different Mach number of 0.729 and angle of attack of 2.31º.



(a) (b) (c)

Figure 24 - Case 1: RAE 2822 drag sensitivities results with respect to vertical movement (y-axis) for: (a) 6 FFD design variables, (b) 16 FFD design variables and (c) 30 FFD design variables.

The effect of design variable dimensionality is now investigated for both parametric methods. The optimization results obtained using the FFD parameterization method for 6, 16 and 30 design variables are plotted in Figure 25, from which a similar behaviour is found between considered sets of design variables. Starting in the leading edge until 40% of the chordwise, the curvature in the upper surface of the baseline airfoil is reduced, which causes a flow deceleration and hence eliminating the shock (as one can observe in Figure 25(b) from the pressure distribution). However, on the upper and lower surface, between 50% of the chord until the trailing edge, the increase of curvature is primarily to satisfy the area constraint. For the case with 6, 16 and 30 design variables, Figure 25(b) compares the optimization results for the pressure distribution, obtained with the FFD method, against the baseline airfoil. Similar pattern exists for the surface perturbation as well as C_p distribution for all FFD cases. The pressure discontinuity, from the initial shock, is replaced by a smooth pressure recovery, thus eliminating the shock wave. The constraints applied were all respected with an increase of 0.4% of the airfoil baseline area.



(a) **(b)**

Figure 25 - Case 1: Influence of design variables dimensionality on the optimization results using the FFD parameterization method: (a) Airfoil shapes and (b) pressure coefficient distribution.

The optimization results obtained for the Hicks-Henne parameterization method for 10, 20 and 30 design variables are plotted in Figure 26. Such as in FFD approach, a similar behaviour is found between considered sets of design variables. However, using this parameterization method, the achieved drag reduction does not respect the imposed constraints. A reduction of the airfoil curvature in the upper and lower surface is notable from Figure 26(a), which causes a reduction of the initial airfoil area. Thus, according with equation (2.50), a reduction in the airfoil area will reduce the lift coefficient. The lift coefficient and baseline airfoil area are below the values imposed to the optimizer, which corresponds to a relative difference of 0.3% and 14.4%, respectively. Only the pitch moment constraint is satisfied with a relative difference of approximately 6.3%. The pressure coefficient distribution, depicted in Figure 26(b), evidences an optimized value of C_p higher than the C_p optimized value obtained with the FFD approach in the stagnation point. While the airfoil geometric are nearly identical, the C_p distributions,

however, show different behaviour between 40% and 60% of the chord with variations of the pressure due to the airfoil optimizer shape.

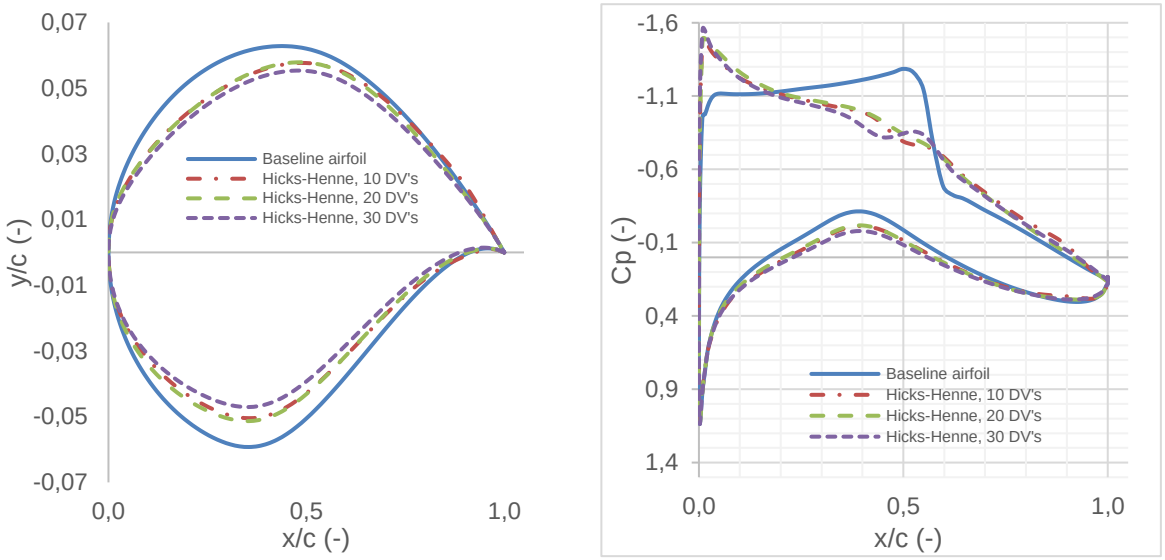


Figure 26 – Case 1: Influence of design variables dimensionality on the optimization results using the Hicks-Henne bump functions method: (a) Airfoil shapes and (b) pressure coefficient distribution.

5.1.3. Optimization results analysis

For the RAE 2822 airfoil case study is desirable to minimize the drag coefficient (objective function) by changing the surface profile shape. The effect of the number of design variable was previously investigated for both parametric methods and now the final drag results (measured in drag counts) are plot in Figure 27. It is worth to note that the optimization performance does not improve when the number of design variables increase for both methods in the present case. Few design variables are enough to cover the design space. One can also notice from Figure 27 that, with the FFD approach, an efficient drag reduction of approximately 34% is achieved for this optimization problem. In the case of the Hicks-Henne bump functions, from it is evident Figure 27 that the same optimization performance trend is achieved using different number of design variables, which implies that this optimization case is insensitive to the distribution of the bump functions: a low number of design variables allows to achieve the same result for the drag coefficient as a larger number of design variables.

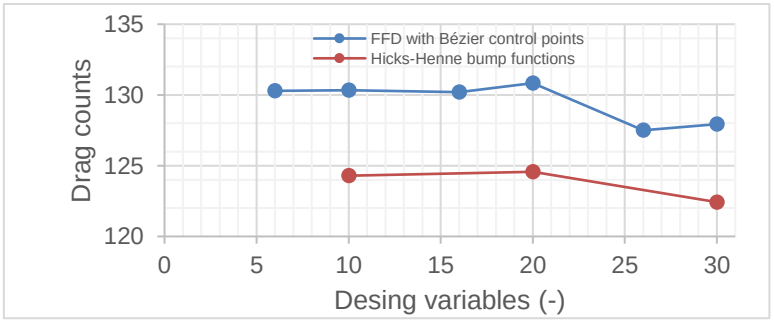


Figure 27 - Case 1: Influence of the FFD and the Hicks-Henne design variables on drag coefficient.

Lee *et al.* [51] and Bisson *et al.* [100] opted for an approach different to the one used in this work. Firstly, they had fixed a lift coefficient of 0.824 and updated the angle of attack iteratively during the flow analysis to meet the target lift coefficient. Then, the angle of attack was also set as a design variable during the optimization procedure. Yang and Ronch [101] followed the same approach using the Hicks-Henne bump functions to parameterize the RAE 2822 airfoil. Approximately 38% of drag reduction is achieved with a set of few design variables to cover the design space too. Although the lift and the pitch moment constraint are respected, the area constraint in the final design is violated in 0.01%.

In the present work, a different approach was tried to reduce the drag coefficient by fixing the angle of attack in agreement with the wind tunnel corrected angle of attack of 2.79º proposed by ADODG. However, without fixing the lift coefficient to satisfy the lift constraint and without setting the angle of attack as a design variable, it was not possible with the Hicks-Henne approach to achieve feasible results for the drag reduction given the set of selected design variables. All optimizations were performed until they met one of the stop criteria imposed on the configuration file. To determine the convergence of the optimization, the KKT conditions needs to be satisfied with a tolerance set as $1 \cdot 10^{-6}$ for this optimization case. However, due to the computation capacity available, if the KKT conditions are not satisfied, the optimizer will stop when 30 optimization design cycles are achieved, in order to save computational time. In all optimization cases, the convergence criteria reached was the tolerance.

5.1.3.1. Comparison of benchmark results

Table 3 summarizes and compares some results obtained by other researchers for the same case study with the present results. All the researchers used gradient-based approaches taking advantage of the adjoint formulation to evaluate the objective function and constraints. Different parameterization techniques were used for the RAE 2822 airfoil by the different researchers. A certain reduction in the drag coefficient, C_d , is observed for both the baseline and optimized airfoils. It can be concluded that a significant reduction of C_d (in regard to the baseline one) in the optimized value is achieved in the range of 42-49%. From the comparison of the achieved values for the minimization of C_d, one can notice that the results of the present method are well within the bounds achieved by a large number of researchers. By comparing the present result for the FFD approach with 26 design variables with respect to the similar gradient-based approaches, both the performance and the computational effort (measured here in total number of functions evaluated) levels are in agreement.

Table 3 – Case 1: RAE 2822 optimization airfoil: comparison with other researchers results.

Reference Articles	Initial C_d	Optimized C_d	Reduction C_d (%)	Parameterization Method	Optimizer Type	DV Number	Function Evaluated
Carrier *et al.* [102]	0.0202	0.0111	45.0	Bézier and B-Spline	Gradient-based (SLSQP)	10	25
Poole *et al.* [4]	0.0174	0.0090	48.3	RBF with SVD for selection of DV	Gradient-based (FSQP)	7	40
Lee *et al.* [51]	0.0234	0.0132	43.6	B-Splines and FFD	Gradient-based (SNOPT)	34	250
Bisson *et al.* [100]	0.0178	0.01071	42.7	Third-order B-Spline	Gradient-based (SNOPT)	16	25
Yang and Ronch [101]	0.0241	0.01514	38.0	Hicks-Henne bump functions	Gradient-based (SNOPT)	30	-
Present	0.0196	0.01275	34.9	FFD	Gradient-based (SLSQP)	26	47

5.2. Case 2: Maximization of lift to drag ratio of NACA 0012

The second optimization problem is the maximization of the lift to drag ratio of a modified NACA 0012 airfoil, considering inviscid flow, for two different cases of free-stream Mach numbers in order to analyze the impact of the parametric methods on the initial geometry. The Mach numbers analyzed cover the transonic (0.85) and supersonic (2.0) regimes for an angle of attack fixed at zero degrees. The governing equation is the 2-D Euler equation with a constant ratio of specific heats of 1.4. The free-stream temperature of 288.15K is assumed for the optimizations. A minimum thickness constraint greater than or equal to one percent of the chord, along the entire chord, is imposed in order to maintain a realistic airfoil design and prevent control points to cross-over. The constraint is applied to 25%, 33%, 50%, 67% and 75% of the airfoil chord. The optimization problem can be mathematically formulated as:

$$\text{Maximize:} \quad {}^{L}/_{D}$$

$$\text{subject to:} \quad y \geq 0.01 \cdot \left(y_{(i)_{x/c}}\right), \mathrm{i} \in \left\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}\right\}, \forall\, x \in [0,1]$$

where x is the x-coordinate of a point on the airfoil, y is the y-coordinate of a point on the optimized airfoil, and $y(i)_{x/c}$ are the y-coordinate of the corresponding point on the airfoil were the thickness constrains are imposed.

5.2.1. Grid convergence study

As the first step to perform CFD simulations, the influence of the mesh on the solution results is investigated. A more accurate numerical solution is obtained as more nodes are used, however, using additional nodes will require more computer memory and computational time. The determination of the appropriate number of nodes can be done by increasing the number of nodes until the mesh is satisfactorily fine, so that the additional refinement does not change the results.

For the purposes of this optimization, the definition of the NACA 0012 airfoil is slightly altered from the original definition so that the airfoil closes in the trailing edge with a sharp edge. As opposed to the airfoil used by Vassberg *et al.* [103], a modified NACA 0012 airfoil with zero-thickness trailing edge is defined according with the formula [49]:

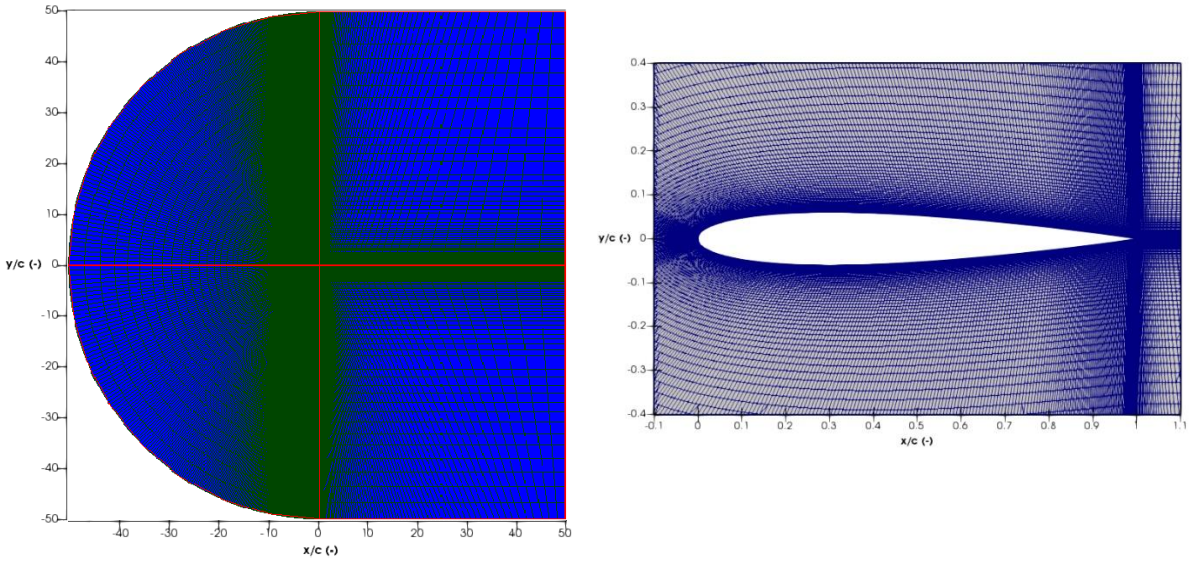
$$y(x) = \pm 0.6\left(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1036x^4\right) \qquad \textbf{(5.1)}$$

where $x \in [0,1]$. The zero-thickness trailing edge is achieved through a modification of the x^4 coefficient. The two-dimensional discretized volume consists on a structured C-topology grid around the NACA 0012 airfoil generated using the Salome software. The domain boundaries are placed at a distance of 50 chord lengths away from the airfoil surface to avoid the effects of the far-field boundaries on the surface solution. A grid convergence study has been performed to evaluate the evolution of the drag coefficient. To establish grid convergence, a family of three C-grid multi-block computational meshes has been defined with increasing level of refinement. The refined meshes are obtained by doubling the number of points in both chord-wise and normal directions starting from the coarse mesh. Table 4 shows the grids characteristics in terms of number of points in the x-coordinate and y-coordinate directions (N_x

and N_y, respectively), the number of points along the airfoil surface ($N_{Surf.Airf.}$) and the grid spacing along the leading edge and the trailing edge. Small grid spacing is set at both the leading edge and trailing edge in order to ensure that the airfoil geometry is accurately represented and important flow features are captured in all the analysis (see Figure 28(b)). The resolution and density of the mesh is greater in regions where higher accuracy is needed, such as the near wall region of the airfoil. The computational domain and the medium mesh (the one selected for the optimizations) are displayed in Figure 28.

Table 4 - Case 2: Mesh convergence study and gird parameters for the NACA 0012 airfoil.

Mesh Refinement	N_x	N_y	$N_{Surf.Airf.}$	Number of grid points	LE/TE Spacing
Coarse	65	39	257	22202	$8 \cdot 10^{-4}$
Medium	129	77	513	86892	$4 \cdot 10^{-4}$
Fine	257	153	1026	343760	$2 \cdot 10^{-4}$



(a) (b)

Figure 28 - Case 2: Computational domain (a) and medium mesh (b) for the NACA 0012 airfoil.

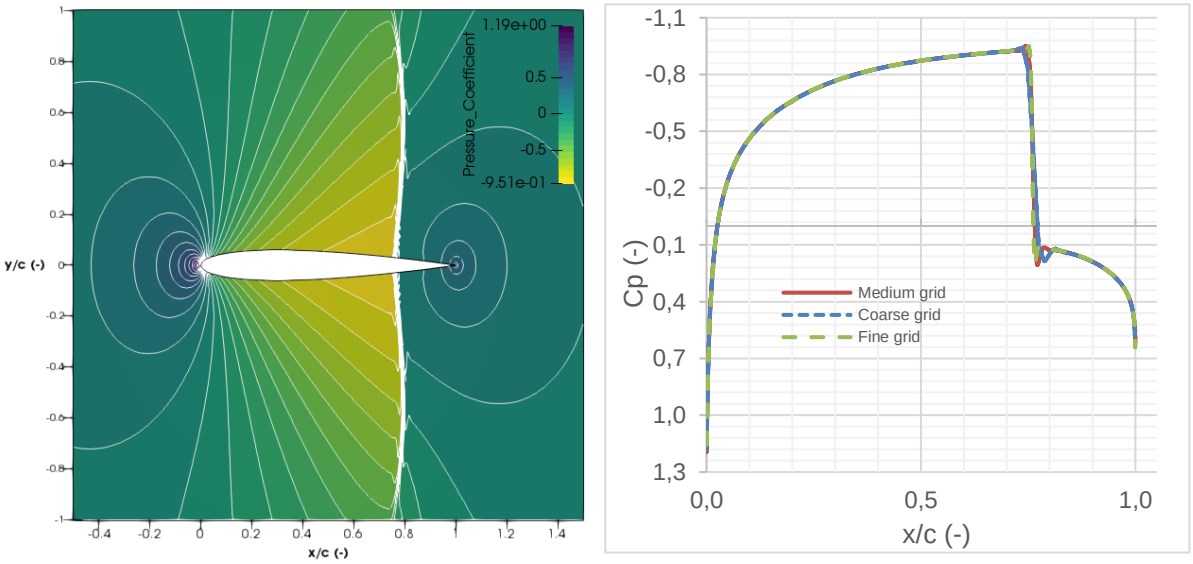
Firstly, the inviscid compressible flow analysis over the modified NACA 0012 airfoil was performed for the transonic flow, M = 0.85 with α = 0º, and the aerodynamic coefficients obtained for the baseline airfoil at convergence (6 order of magnitude drop of the residual of density equation) are reported in Table 5. Note that the C_d values are for the complete airfoil geometry. The Euler wall boundary condition are applied to force the flow to be tangent to the solid surface, this means at the upper and lower walls of the airfoil. The far-field boundary conditions are applied to the outer domain boundaries. The idea is to attribute free-stream values to all the flow properties at the external boundary. Between the level of refinement of the medium and fine meshes, the resolution of 0.119 drag counts (1 drag count is equal to a C_d of $1 \cdot 10^{-4}$) is achieved. It is noticed that the medium mesh can produce drag results with enough level of accuracy. Thus, the medium level of mesh refinement is again used for the optimization.

In Figure 29, the pressure coefficient distribution for the mesh convergence study is shown. The computed SU² solutions for the mesh study are in good agreement with the ones obtained in [4], [75],

[98] and [102] . The computed values match the numerical pressure coefficient through the suction peak, the pressure recovery region, and the trailing edge for the select angle of attack. All the three solutions place the shock approximately at three-quarters of the chord, Figure 29(a).

Table 5 - Case 2: Drag results for the NACA 0012 airfoil grid convergence study (M = 0.85, α = 0º)

Mesh Refinement	C_l/C_d	C_d (counts)
Coarse	approx. 0	470.861
Medium	Approx. 0	470.564
Fine	Approx. 0	470.445



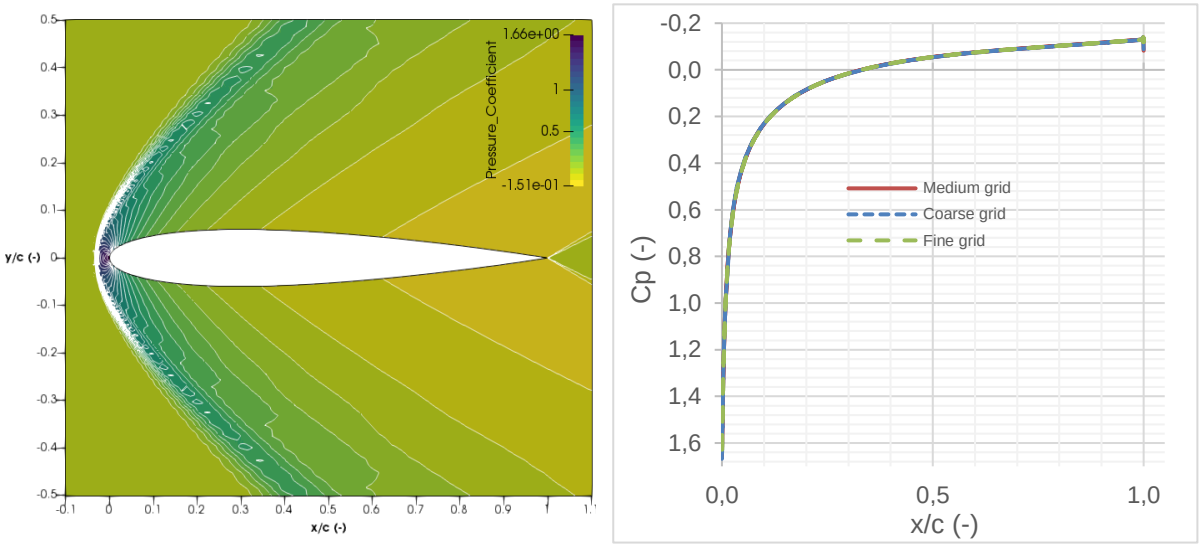
(a) (b)

Figure 29 - Case 2: Pressure coefficient contours for the medium mesh (a) and numerical comparison between the three levels refinement mesh (b) around the NACA 0012 (M = 0.85, α = 0º).

The supersonic flow case over the modified NACA 0012, M = 2.0 with α = 0º, was simulated using the same grids of the mesh refinement study of the transonic case. For each Mach number, a mesh suitable for the flow conditions should be performed and a mesh refinement study should be carried out. However, this would lead to an exhaustive number of meshes and flow analyses that, without the adequate computational resources, become difficult to perform. The drag coefficient results for the baseline airfoil are listed in Table 6. Despite the small penalty in the accuracy of the results, the medium mesh is chosen for all the analyses performed with the supersonic Mach number. The relative error between the drag results (in counts) for the medium mesh and the fine mesh is less than one percent, which is considered acceptable. The Euler wall boundary condition and the far-field boundary conditions are applied to the upper and lower walls of the airfoil and to the outer domain boundaries, respectively. The location of the start of the shock is -0.033 measuring from the leading edge of the airfoil in the current simulation (see Figure 30(a)). The C_p distribution found using the three different grids is in very good agreement, as shows Figure 30(b).

Table 6 - Case 2: Drag results for the NACA 0012 airfoil grid convergence study (M = 2.0, α = 0º)

Mesh Refinement	C_l/C_d	C_d (counts)
Coarse	Approx. 0	914.345
Medium	Approx. 0	919.307
Fine	Approx. 0	922.349



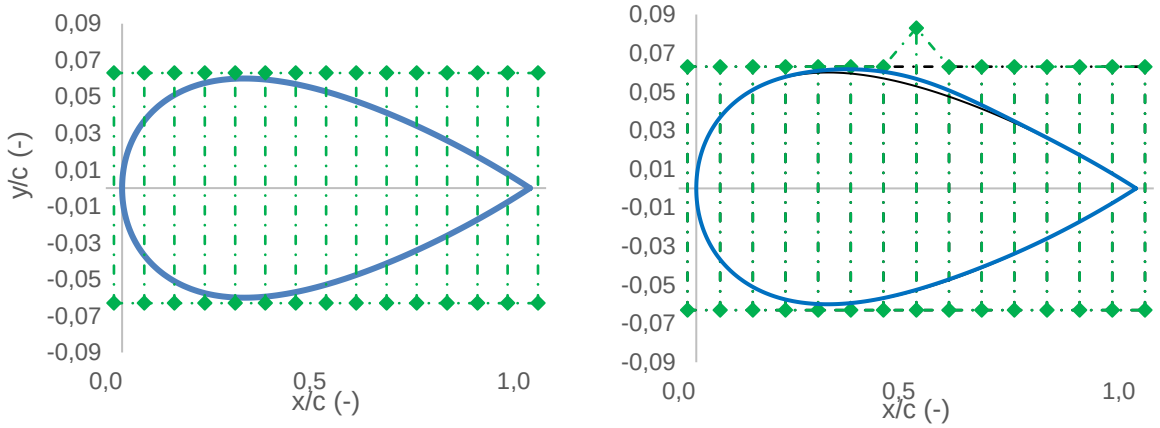
(a) (b)

Figure 30 - Case 2: Pressure coefficient contours for the medium mesh (a) and numerical comparison between the three levels refinement mesh (b) around the NACA 0012 (M = 2.0, α = 0º).

5.2.2. Geometric parameterization impact

The FFD method and the Hicks-Henne bump functions are employed as the parameterization techniques for the NACA 0012 airfoil. For both parameterization methods, several parameters of each method must be determined prior to the optimization to analyze their impact. Different settings of these parameters may generate different optimization results and, consequently, different airfoil shapes. Therefore, in order to find a set of parameters that allows to produce better optimization results, the impact of these parameters on the optimization performance is first analyzed.

Firstly, the FFD control point parameterization method is investigated. An example of this approach is demonstrated in Figure 31(a), where the control points contained on the upper and lower surface of the FFD box are set as design variables. Only the FFD control point approach for 2-D airfoil embedded in SU² framework is employed in this study. The thickness at specific chordwise position of the airfoil can be modified by manipulating the control points (see Figure 31(b)) which can move with different amplitudes in the same or opposite directions. Total freedom in the y direction is given to the optimizer to allow the modification of the baseline airfoil. Note that, the complete airfoil geometry and the corresponding mesh are generated for the FFD control points approach using the SU²_DEF module.



(a) **(b) Black lines indicate undeformed shape.**

Figure 31 – Case 2: View of NACA 0012 airfoil embedded in an FFD box with the 15 control points: undeformed FFD box (a) and FFD box deformation influence (b). Green points indicate the control point positions.

The FFD box applied for this case has a rectangular shape being defined by four boundaries located not too far from the airfoil: top, bottom, right and left boundaries. The NACA 0012 airfoil is embedded inside the FFD box and, for the success of the optimization design, the boundaries cannot intersect with the embedded geometry. Nonetheless, it was found in this study that the lift to drag ratio optimization results have a dependence on different settings of the FFD box position. Although the control points can be placed along the four boundaries, they are only placed for the present case at the upper and lower boundaries. The position of the four boundaries that can achieve the best optimization performance for the present case are listed in Table 7. These FFD box settings are used hereafter for the optimization of the NACA 0012 when subjected to transonic and supersonic flow conditions. All the control points are allowed to move only in the y direction keeping the x fixed. This way, the optimizer only can reduce or increase the airfoil thickness without changing the baseline chord of the airfoil. Despite the freedom given to y movement of control points, the thickness constraint greater than or equal to one percent of the chord is satisfied by the optimizer during the optimization design process.

Table 7 – Case 2: FFD box positions settings for the modified NACA 0012 airfoil optimization case for both Mach numbers (M = 0.85, α = 0º) and (M = 2.0, α = 0º).

	Top	Bottom	Right	Left
FFD control point approach	0.0630	-0.0630	1.0200	-0.0200

The second parameterization method corresponds to the Hicks-Henne approach with special interest in the three parameters, as presented in chapter 3, which are firstly investigated. For the Hicks-Henne bump functions, the bump amplitude coefficients θ_i are defined as design variables. In terms of the location of the maximum point of the bump function h_i, the optimizations for the transonic and supersonic Mach numbers were conducted using an even distribution approach over the range $[0.5/m, 1 - 0.5/m]$ of design variables m. The distribution of the bump maximum positions for fifteen bump functions along the NACA 0012 airfoil are shown in Figure 32, valid for both Mach numbers.

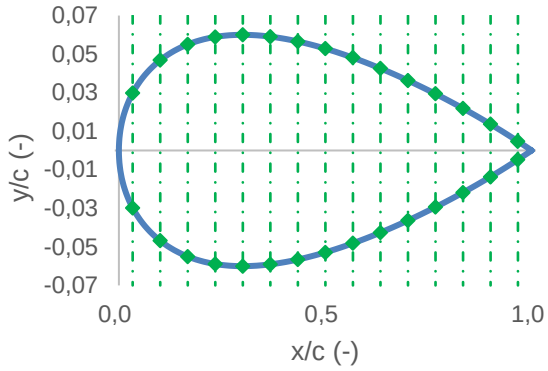
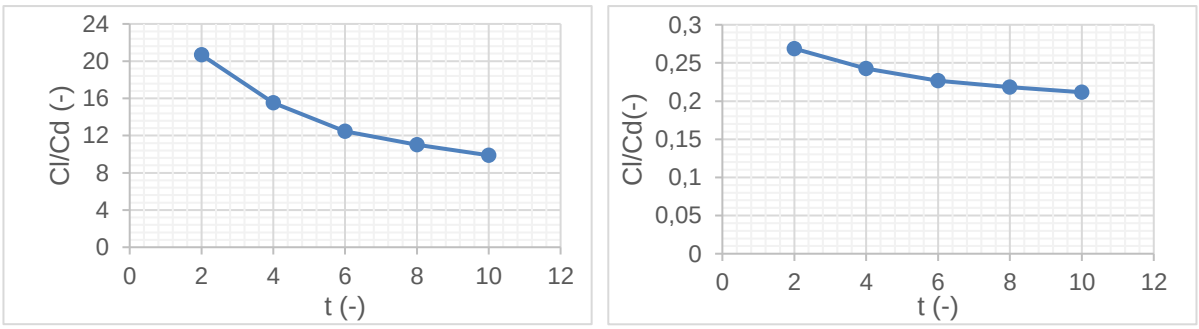


Figure 32 - Case 2: Even distribution of the Hicks-Henne bump functions over the NACA 0012 airfoil for $m = 15$. Green points over the airfoil indicates bump maximum positions.

Regarding the width of the bump t_i, a series of t values from 2 to 10 are set for optimization and its impact on the results is investigated for both Mach numbers. The final lift to drag ratio results for the case M = 0.85 and M = 2.0 with fifteen design variables are plotted in Figure 33. It is clearly observed that there is a gradual decrease of the lift to drag ratio results when t increase. To find out the explanition for this fact, three representative optimization results for transonic and supersonic Mach numbers, with respect to fifteen design variables, are shown in Figure 34 and Figure 35, respectively. Although in both cases the value of the objective function decreases with higher values of t, the effect of this parameter is higher in the transonic case (with a reduction of 52% between the ends of the analyzed spectrum) than the supersonic case (with a reduction of 21% between the ends of the evaluated spectrum).

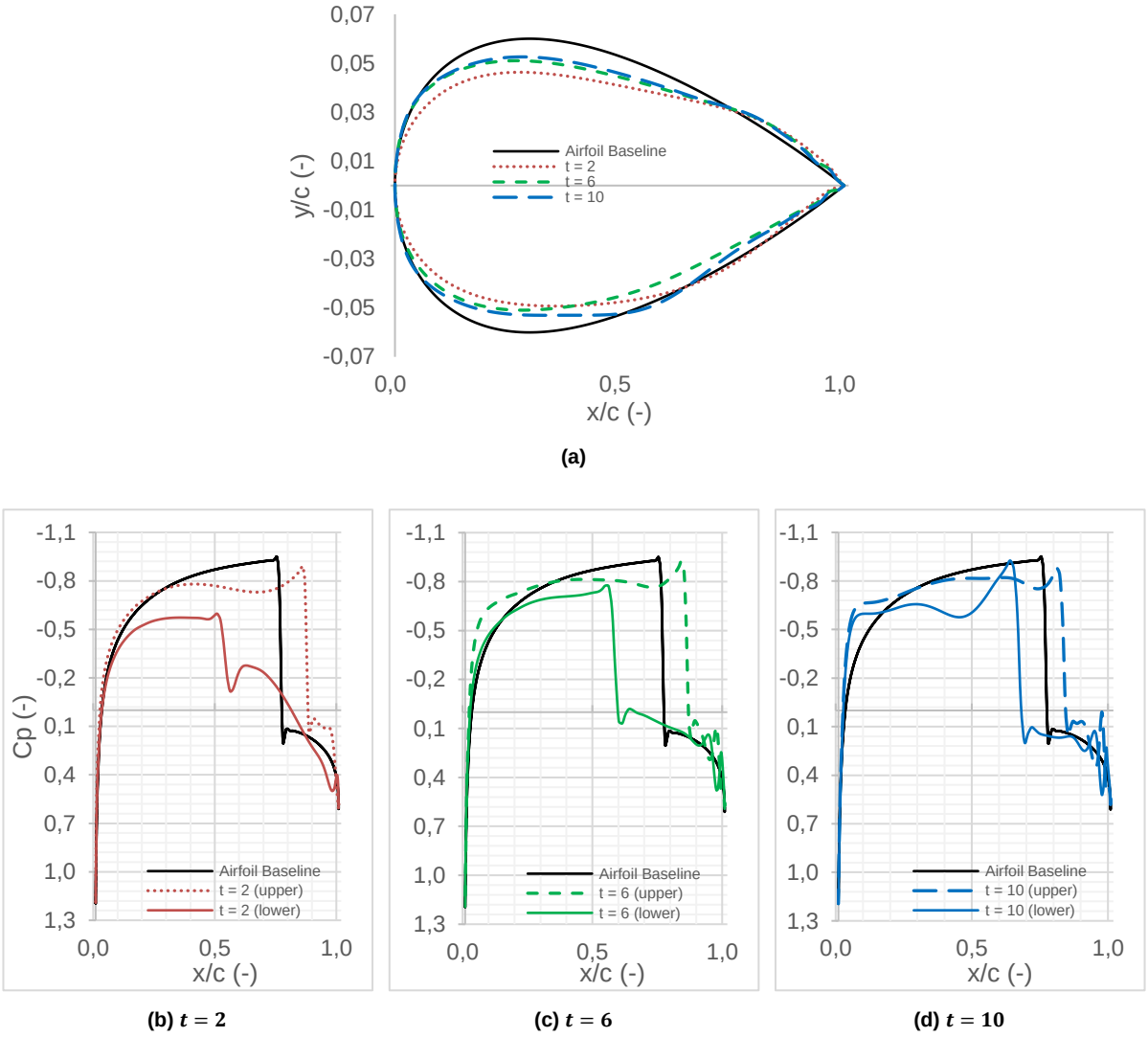


(a) M = 0.85 and α = 0º. **(b) M = 2.0 and α = 0º.**

Figure 33 - Case 2: Influence if Hicks-Henne bump width control parameter on the lift to drag ratio for the transonic (a) and supersonic (b) flow conditions.

Concerning the optimized airfoil shape for the transonic Mach number, Figure 34(a), a primary difference is observed relatively to the airfoil area, being more pronounced for small values of t. For $t = 6$ and $t = 10$, a significantly perturbation in the trailing edge region is generated, which differs from that obtained for $t = 2$. In the leading edge, the method softens the curvature of the initial airfoil for small values of t. As previous mentioned in chapter 4, more local shape control can be achieved with a relatively larger value of t, which is consistent with the results obtained. Specially, setting $t = 6$ and $t = 10$ allows for the deformation of the airfoil in a region near the training edge. Thus, in turn, it is generated C_p distributions (see Figure 34) in that region with some perturbations which are of lower intensity for $t = 2$. In the leading edge, the C_p value tends to be higher for larger values of t, whereas the case $t = 2$

generates a C_p distribution which is nearly unchanged from the baseline airfoil near this region. The three cases exhibit a similarly strong shock for the upper surface in the trailing edge direction. However, on the lower surface, the shock is moving upstream the trailing edge, losing its intensity as the values of t decrease. Based on the C_p distribution, the case with a lowest value of t offers a higher lift to drag ratio. Since different values of t give different optimization results, to save computational time and to have a t parameter that allows a higher local shape control, $t = 6$ is selected for the setting of the bump width control parameter from now on for the NACA 0012 airfoil optimization case.



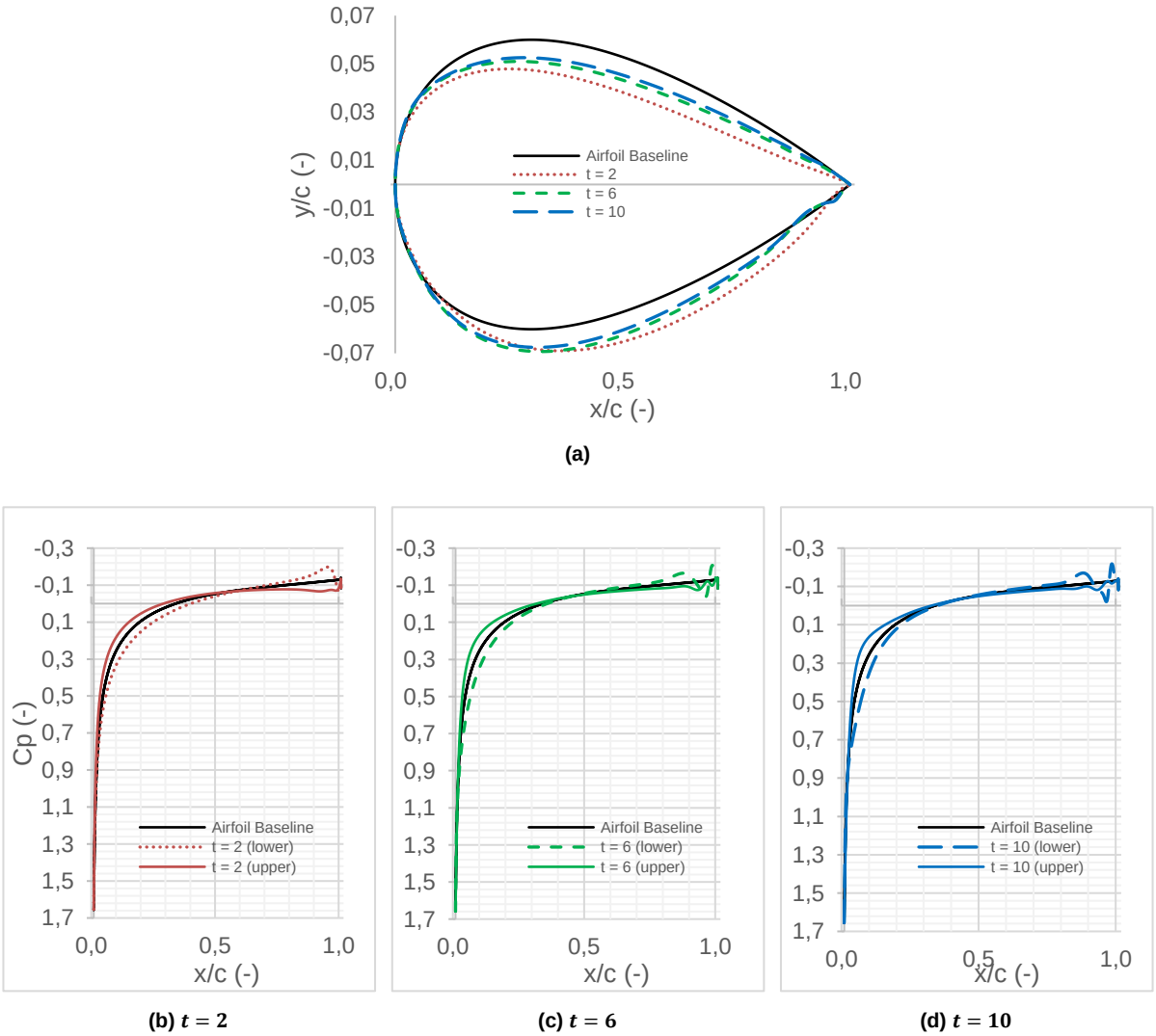
(a)

(b) $t = 2$ **(c) $t = 6$** **(d) $t = 10$**

Figure 34 - Case 2: Influence of Hicks-Henne bump width control variable: (a) Airfoil shapes and (b), (c) and (d) pressure coefficient distributions for the optimization results (M = 0.85, α = 0°).

Analyzing the effect of the width bump control parameter in the supersonic case (see Figure 35(a)), a primary difference observed is related to the reduction of the airfoil area, being more pronounce for small values of t, in agreement with the transonic case. Once again, more local shape control can be achieved with a relatively larger value of t, which is consistent with the results obtained previous, specially, for $t = 6$ and $t = 10$ that allows for a greater modification in the training edge region. Increasing the value of t increases the perturbations in this region. For higher values of t is also notice an increasing of the

lower surface curvature. In terms of C_p distributions (see Figure 35) there is a slight increase in pressure in the lower surface, when compared to the upper surface, which is more evidence for higher t values.

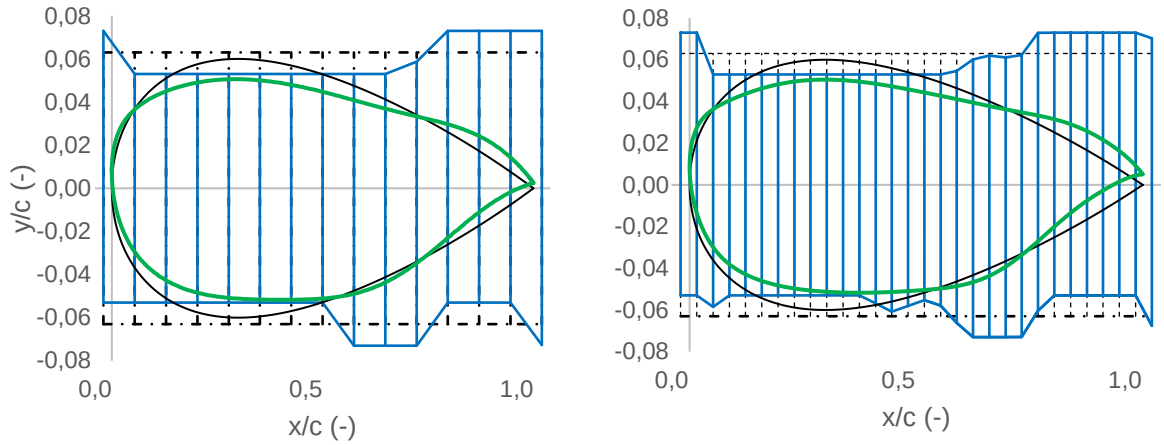


(a)

(b) $t = 2$ **(c) $t = 6$** **(d) $t = 10$**

Figure 35 - Case 2: Influence of Hicks-Henne bump width control variable: (a) Airfoil shapes and (b), (c) and (d) pressure coefficient distributions for the optimization results (M = 2.0, α = 0º).

To examine the effect of design space dimensionality for the current objective function, the design optimization was performed for different numbers of design variables. The optimizations were conducted using: the Hicks-Henne bump functions with a total of 5, 15, 30 and 45 chordwise design variables distributed on the upper and lower surfaces; and the FFD control point approach varying the same number of design variables but defined between the top and bottom parts of the FFD box. A transonic Mach number of 0.85 was firstly selected and investigated. Figure 36 displays the final deformed FFD boxes and the optimized airfoil geometries using fifteen and thirty design variables in the FFD control points parameterization method for the transonic case. By increasing the number of control points on the surface of the FFD box, more freedom is given to the optimizer to explore the design space.



(a) FFD, 15 DV's

(b) FFD, 30 DV's

Figure 36 - Case 2: FFD box deformation and optimized airfoil geometry for dimensionality study using FFD control point parameterization method (M = 0.85, α = 0º).

Figure 37 displays the final optimized airfoil geometries and the C_p distributions for three sets of design variables using the FFD control points approach. With more design variables, the leading edge tends to increase its curvature and the aft section of the airfoil gets thicker, special in the upper surface. It is also worth mentioning the formation of the geometric incidence angle. A considerable reduction of the initial airfoil area is remarkable, being more pronounced in the lower surface. In the airfoil aft section, it is clear an increase of the curvature at the lower surface. This geometric shape is typical of supercritical airfoil (normally selected for transonic conditions) which is a good indicator of the achieved result given the flow conditions. Regarding the change in the leading edge region, a suction peak is formed in the C_p distribution and becomes steeper as more design variables become available. The shock moves further downstream towards the trailing edge in the upper surface and trends to dissipate in the lower surface moving towards the leading edge (increasing the negative pressure zone on the upper surface and reducing it on the lower surface, causing an increase in lift force). The difference in pressure distributions between the upper and lower surface leads to an increasing in lift force.

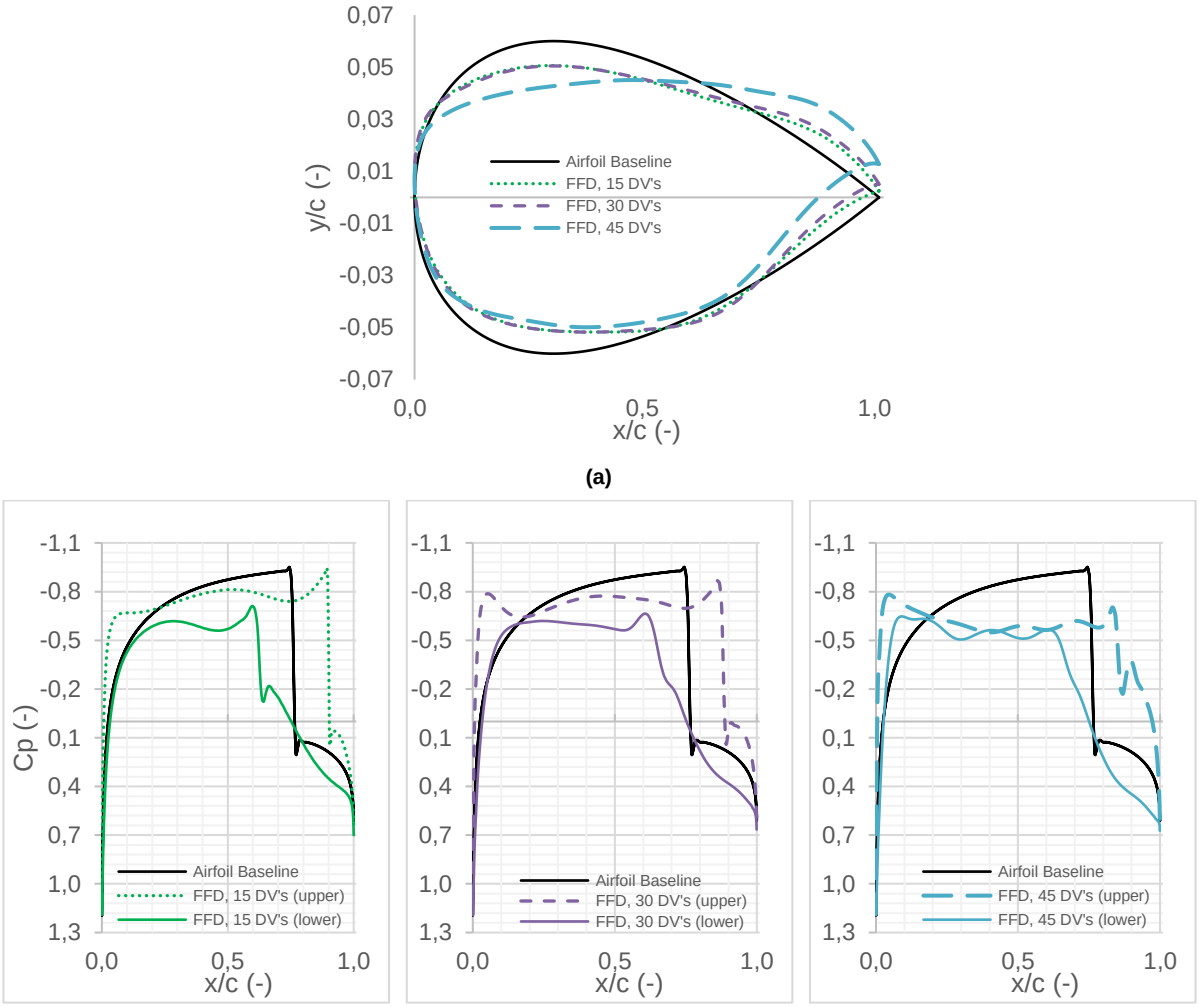


Figure 37 - Case 2: Influence of design variables dimensionality on the optimization results using the FFD control point method: (a) airfoil shapes and (b), (c) and (d) pressure coefficient distributions (M = 0.85, α = 0º).

Figure 38 plots the final geometrical shape of the optimized airfoil and the C_p distributions using the Hicks-Henne bump functions for the transonic case. With the increase of the design variables number, a similar behaviour to the FFD control point approach is observed. A considerable area reduction is reached, being more pronounced in the lower surface. Due to the formulation of the Hicks-Henne method, the start and end points of the airfoil (x=0 and x=1) are not allowed to be treated as design variables. Contrary to the FFD control point approach, these specific points are fixed without any movement during the optimization. The change in the aft region of the airfoil leads to the smoothing of the initial shock which becomes smoother as more design variables become available. For the case of 45 design variables the optimized shape presents a C_p distribution with some behaviour similar to the FFD approach with the same number of design variables. In the leading edge region it is clear the formation of a suction peak and the initial shock position tends to dissipate in the lower surface moving towards the leading edge and downstream towards the trailing edge in the upper surface. The differences between the C_p distributions in the upper and lower surface evidence the gain of positive lift force.

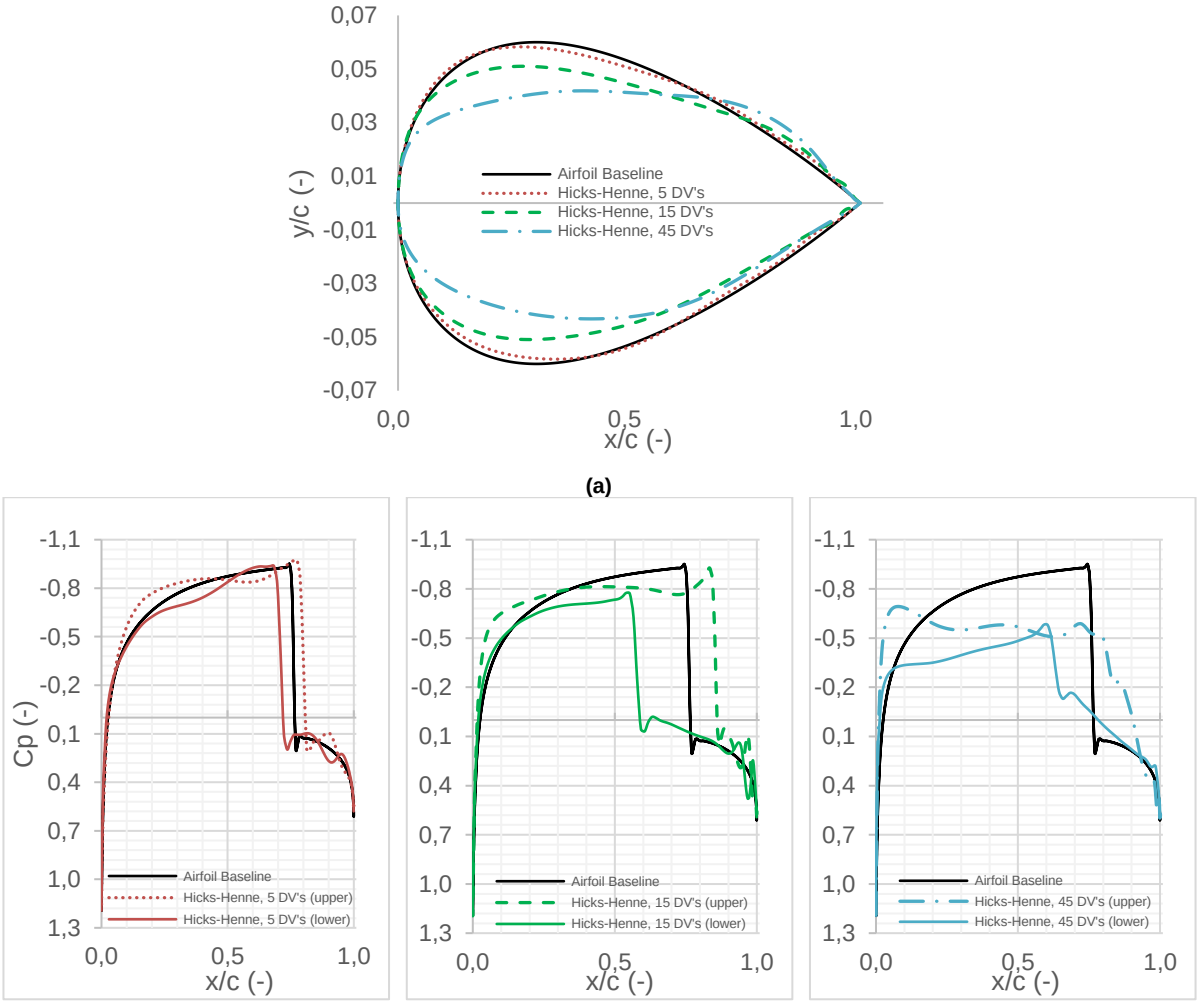
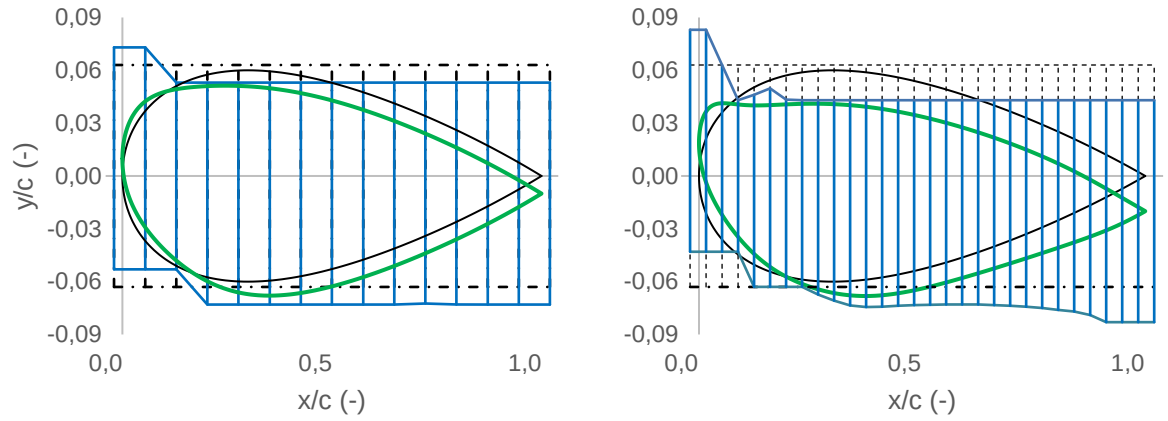


Figure 38 - Case 2: Influence of design variables dimensionality on the optimization results using the Hicks-Henne bump functions method: (a) airfoil shapes and (b), (c) and (d) pressure coefficient distributions (M = 0.85, α = 0º).

To analysis the effect of design space dimensionality for the supersonic case, the design optimization is performed using the same parameterization methods and the same design variables approach. Starting with the FFD control point approach, Figure 39 shows the final deformed FFD boxes and the optimized airfoil geometries using fifteen and thirty design variables. Again, by increasing the number of control points on the surface of the FFD box, more freedom is given to the optimizer to explore the design space. With thirty design variables, the optimizer can push the trailing edge more in the negative y direction and is allowed to modify more the leading edge region. For both numbers of design variables, a geometric incidence angle appears.



(a) FFD, 15 DV's **(b) FFD, 30 DV's**

Figure 39 - Case 2: FFD box deformation and optimized airfoil geometry for dimensionality study using FFD control point parameterization method (M = 2.0, α = 0º).

Figure 40 plots the final optimized airfoil shape and the C_p distributions for the FFD control point method. After the optimization design process, the leading edge and the trailing edge points move in the y positive direction and the y negative direction, respectively, increasing their movement with more design variables. The geometric incidence angle tends to increase its value as well as the curvature of the airfoil in the lower surface with the increase of the design variables. Such behaviour leads to an increase of the pressure difference between the two surfaces of the airfoil, generating a positive lift force. The leading edge region generates a suction peak for thirty design variables and decrease its intensity for 45 designs.

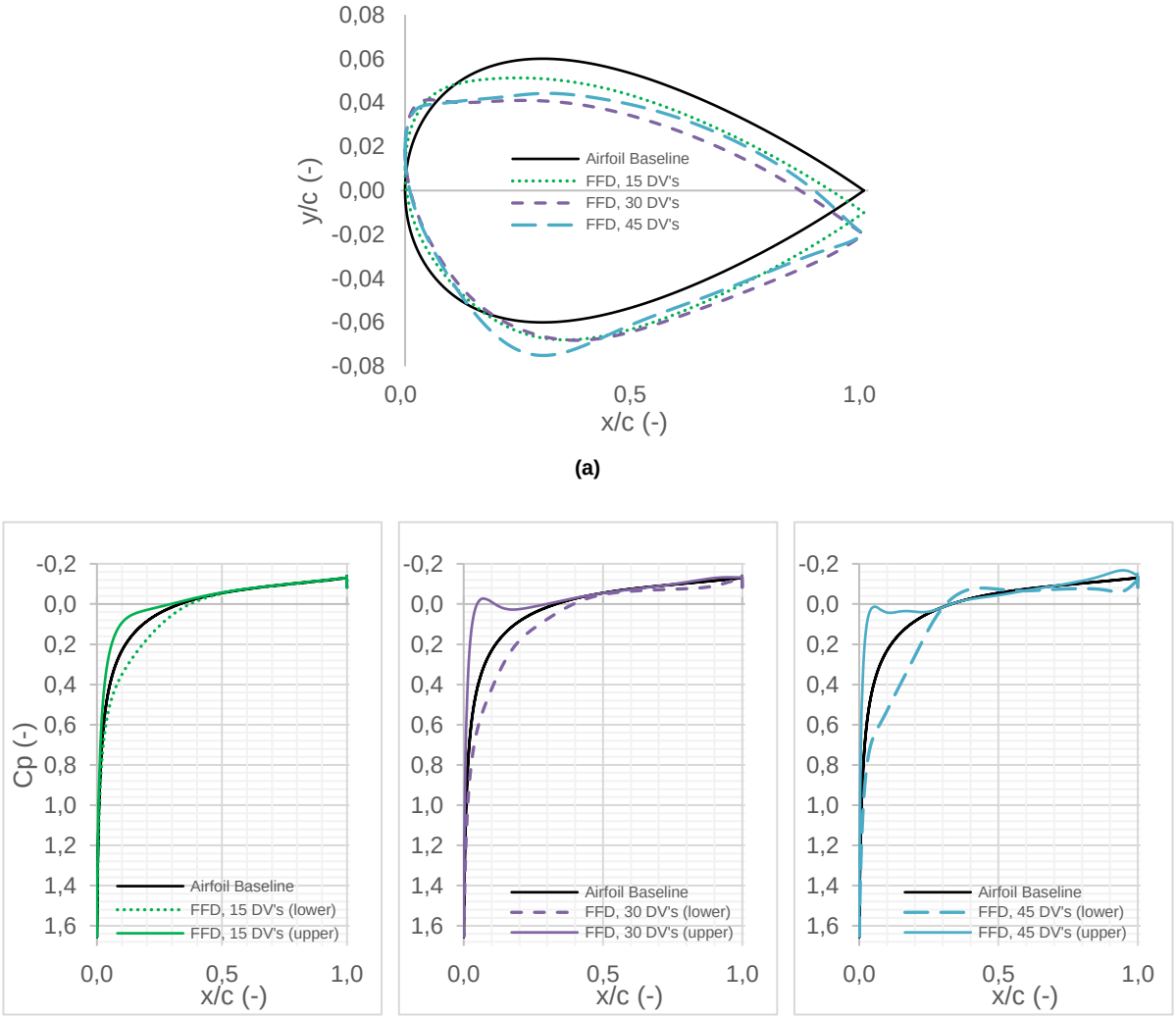
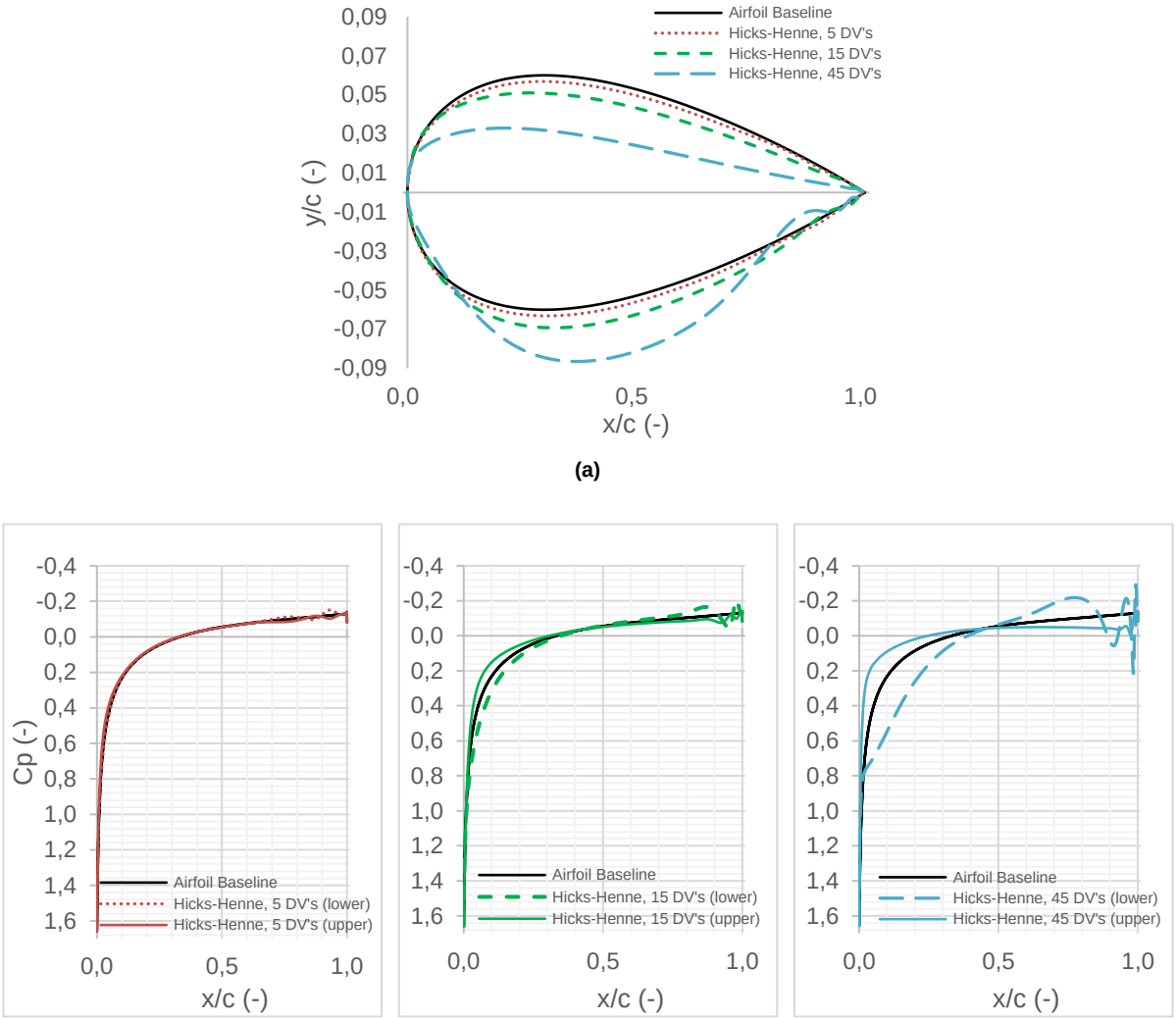


Figure 40 - Case 2: Influence of design variables dimensionality on the optimization results using the FFD control point method: (a) airfoil shapes and (b), (c) and (d) pressure coefficient distributions (M = 2.0, α = 0º).

The upper airfoil surface tends to decrease its curvature by approaching a flat plane. With more design variables available a suction peak is generated moving to the leading edge. The same pressure distribution behaviour is verified for the Hick-Henne method.



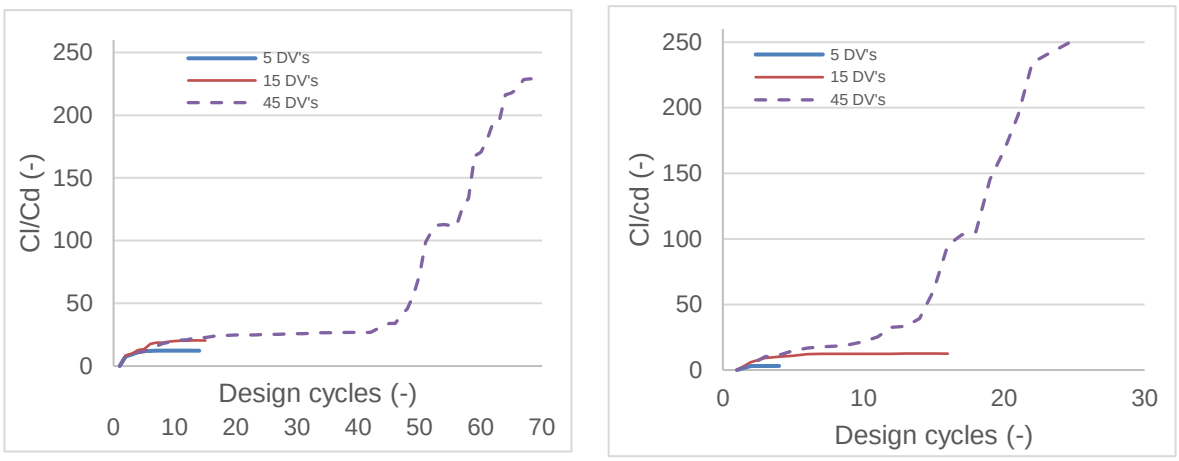
(a)

(b) Hicks-Henne, 5 DV's **(c) Hicks-Henne, 15 DV's** **(d) Hicks-Henne, 45 DV's**

Figure 41 - Case 2: Influence of design variables dimensionality on the optimization results using the Hicks-Henne bump functions method: (a) airfoil shapes and (b), (c) and (d) pressure coefficient distributions (M = 2.0, α = 0º).

5.2.3. Optimization results analysis

For both parameterization methods, the convergence histories of the objective function for the transonic study are shown in Figure 42. When more design variables are established, more design cycles are needed for the optimizer to achieve the local minimum. In terms of convergence performance of the optimization, the Hicks-Henne bump functions approach converges performs better than the FFD control point approach as it requires less design iterations to meet the KKT condition (the tolerance is set as $1 \cdot 10^{-7}$ for this optimization case), especially for the cases with more design variables.



(a) FFD control point **(b) Hicks-Henne bump functions**

Figure 42 - Case 2: Convergence histories of the design optimization of the modified NACA 0012 airfoil using the FFD control points (a) and Hicks-Henne bump functions (b) as parameterization methods (M = 0.85, α = 0º).

The final lift to drag ratio results of the dimensionality study are plotted in Figure 43 for both methods. For the FFD control point method, the objective function presents a higher value for five and fifteen design variables, whereas for the Hicks-Henne bump function method, with more design variables it is possible to achieve higher values of the lift to drag ratio. This fact is possibly caused by the difference of the design variable distribution associated to each parameterization method. The bump functions are distributed following an even distribution actuating on the airfoil surface. By contrast, the Bézier control points are placed on the surface of the FFD box, which means that more design variables are needed for the optimizer to explore all the design space. The other important factor that influences the results is related to the freedom given to the airfoil geometry. In the FFD approach the airfoil is enclosed inside to the FFD box. Total freedom is given to the movement of the y-coordinate of all design variables, allowing local deformation and displacement of the leading edge and the trailing edge points. However, using the Hicks-Henne bump functions approach, the movement of these two points is not allowed, being fixed due to the formulation of the method.

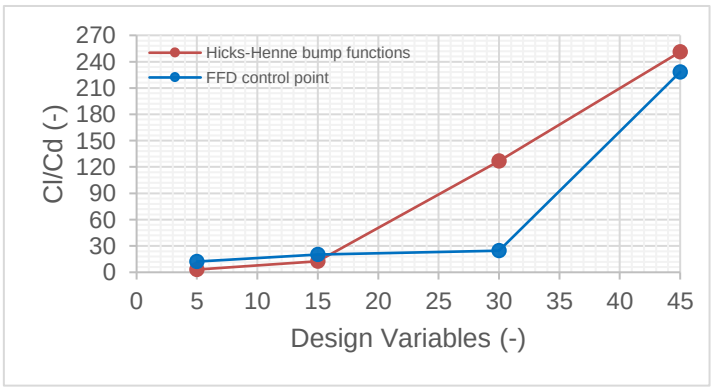
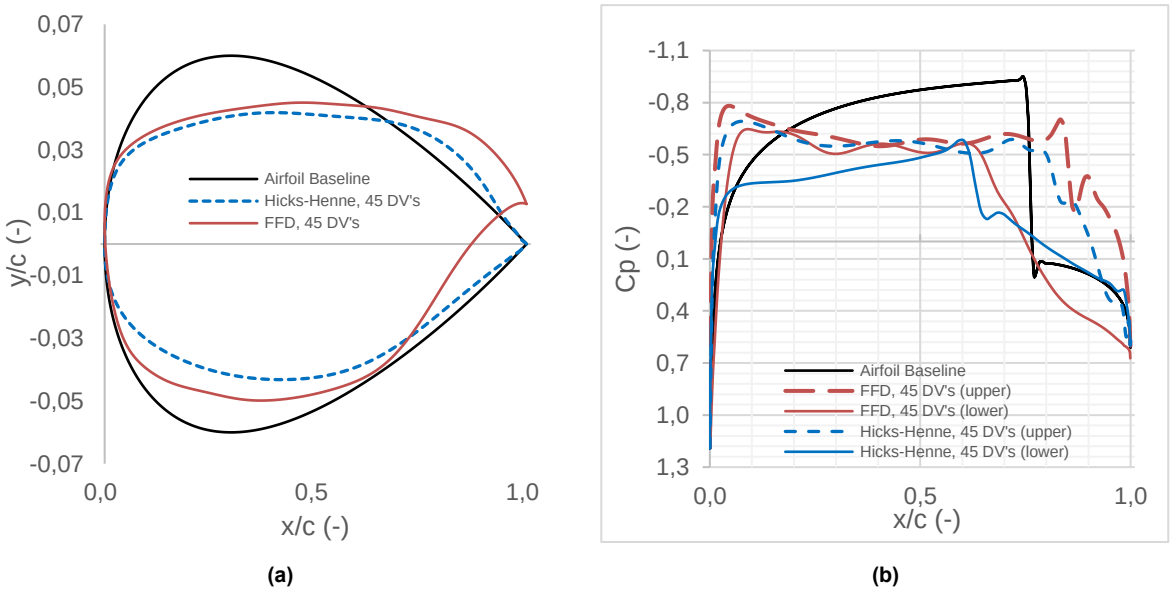


Figure 43 - Case 2: Lift over drag ratio results obtained from the dimensionality study using the both parameterization methods (M = 0.85, α = 0º).

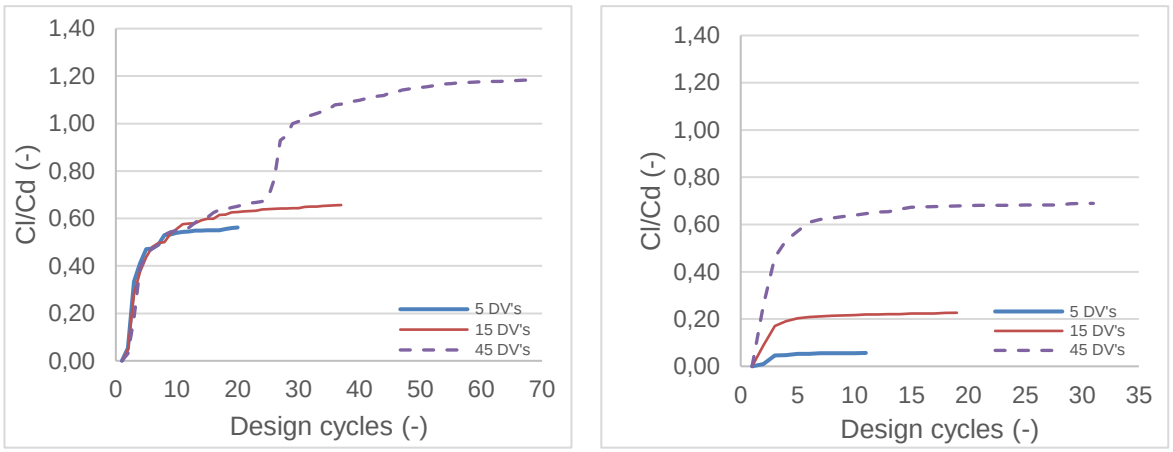
Relatively to the optimization performance, a higher lift to drag ratio is achieved with the best result for each parameterization method. Specifically for the case with 45 design variables, the Hicks-Henne bump function method produces a higher lift to drag ratio, which corresponds to $C_l/C_d = 251.15$ against the 228.28 obtained with the FFD control point method. The optimized airfoil shapes and the C_p distribution are then shown and compared in Figure 44. Similarities are observed among the optimization results, which implies that both methods are equally effective. In the rear section of the upper surface, the Hicks-Henne method trends to increase the airfoil thickness and the curvature, similar to the FFD method. An area reduction is also found, being more accentuated on the lower surface for the Hicks-Henne method. About 70% of the airfoil chord, and moving toward the trailing edge, it is evident an increasing of the airfoil curvature for the FFD approach (more than the Hicks-Henne approach). The optimized shape presents a C_p distribution with a suction peak in the leading edge stronger for the FFD method and a more intense shock in the lower surface for the Hicks-Henne method.



(a) **(b)**

Figure 44 - Case 2: Comparison of optimization results obtained from using the FFD control points and Hicks-Henne bump functions as parameterization methods: (a) airfoil shapes and (b) pressure coefficient distributions (M = 0.85, α = 0º and 45 DV's).

The convergence histories for the supersonic case are represented in follow Figure 45 for the FFD control point and the Hicks-Henne bump function parameterization method. Like in the transonic case, when more design variables are using, more design cycles are needed for the optimizer to achieve the local minimum. In terms of convergence performance of the optimization, the Hicks-Henne bump functions approach revealed a better performance, requiring less design iterations to meet the KKT condition (the tolerance is set as $1 \cdot 10^{-6}$ for this optimization case), especially for the cases with more design variables. Due the available computation capacity, if the KKT conditions are not satisfied, the optimizer will stop when 80 optimization design cycles are achieved, in order to save computational time. In all optimization cases, the convergence criteria reached was the tolerance.



(a) FFD control point **(b) Hicks-Henne bump functions**

Figure 45 - Case 2: Convergence histories of the design optimization of the modified NACA 0012 airfoil using the FFD control points (a) and Hicks-Henne bump functions (b) as parameterization methods (M = 2.0, α = 0º).

Despite the equal number of design variables for both parameterization methods, different values of lift to drag ratio and optimization performance are achieved. The FFD approach with 45 design variables reaches a maximum value of 1.18 against a value of 0.69 using the Hicks-Henne method. Once again both methods show a dependency on the number of design variable which allow the optimizer to explorer more the design space.

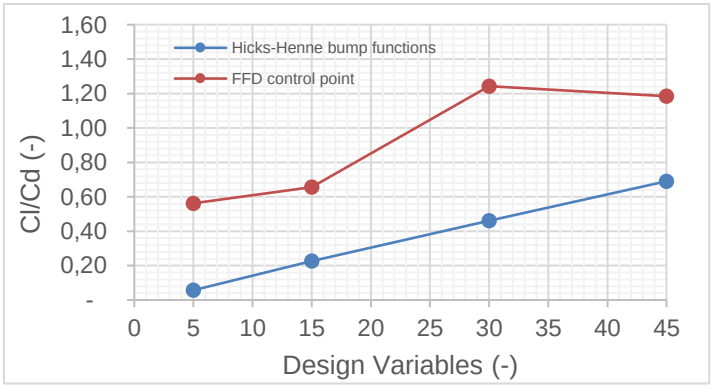
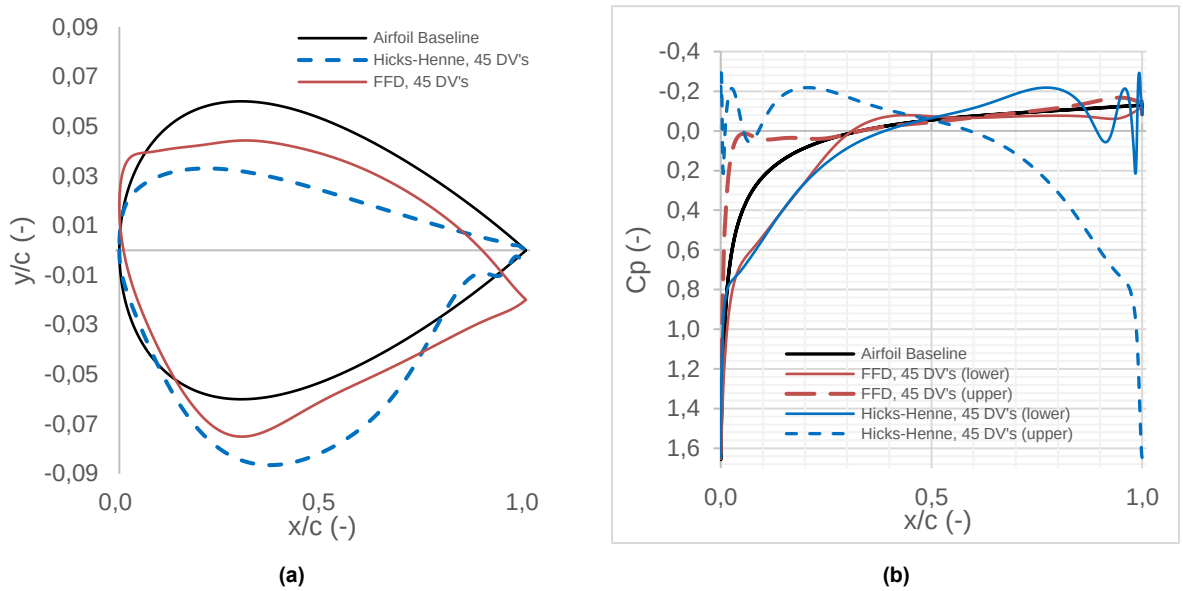


Figure 46 - Case 2: Lift over drag ratio results obtained from the dimensionality study using the both parameterization methods (M = 2.0, α = 0º).

Analyzing the Hicks-Henne method (see Figure 47), an increasing of the area in the lower airfoil surface, compared with the upper surface, allows a difference of the pressure distribution which increase the lift force and reaches a higher lift to drag ratio. Although the angle of attack is zero and the initial airfoil is symmetric, the optimized shape is no longer symmetric to allow for generating a positive lift and thus a positive lift to drag ratio. This behavior is found in the FFD method too. However, in the optimized shape, with the change of the trailing edge position in the y negative direction, a deflection is created, and the lift coefficient can increase his value. As in the transonic case, a geometric incidence angle is formed due the leading edge displacement. In term of C_p distribution, a suction peak appears at this point.



(a) **(b)**

Figure 47 - Case 2: Comparison of optimization results obtained from using the FFD control points and Hicks-Henne bump functions as parameterization methods: (a) airfoil shapes and (b) pressure coefficient distributions (M = 2.0, α = 0º and 45 DV's).

.

Chapter 6

Conclusions and Future Work

In this chapter, a summary of what was achieved during this work is presented, followed by some suggestions for the improvement of the aerodynamic shape optimization and parametric methods.

6.1. Concluding remarks

In the present work, a framework consisting of three open-source software was implemented to perform an ASO analysis. As mesh generator SALOME software was applied to create the mesh configuration around the two airfoils under investigation. For pre- and post-processing SU^2 and ParaView software are considered and scripts programmed in Python language were run in order to link the different open-source software. To validate the ASO framework, a benchmark optimization problem defined by the AIAA ADODG was performed and the results were compared with those obtained by other researchers. In order to further test the capabilities of the optimization framework, the NACA 0012 airfoil was selected and submitted to two different flow conditions, transonic (M=0.85) and supersonic (M=2.0) to maximize the lift to drag ratio at zero degrees of angle of attack.

Two case studies with different airfoil geometries are used to evaluate the procedure described in the preceding sections. The first is a viscid problem, applied to the RAE 2822 airfoil, where the flow behavior was predicted using the compressible RANS equations and the respective adjoint equations. The second case study, starting from a NACA 0012 airfoil, is an inviscid problem where compressible Euler equations were solved, followed by the corresponding adjoint equations. Both meshes used to start the optimization calculations were generated using SALOME software; and then with a python script transferred to the SU^2 software, where the flow analysis and the adjoint solutions can be obtained through the SU^2 code. In the first case study a hybrid mesh was used, composed of structured quadrilateral elements in the boundary layer region, and triangular elements in the far-field. For the second case of study, a structured mesh was built for both Mach numbers.

The first case study consists in benchmarking the implemented framework by solving the AIAA ADODG constrained shape optimization problem of the RAE 2822 airfoil, in viscous transonic flow, to minimize the drag coefficient. The optimized airfoil shows a smother curvature downstream the leading edge at the upper surface. Correspondingly, the flow acceleration is eased, and the shock wave is

eliminated, reducing the total drag counts from 191.6 to 127.5, and corresponding to a reduction of 34.9%. In terms of the two parameterization methods applied for this case study, only the FFD control point approach is effective. The parameterization method did not reveal dependence on the number of design variables. With a small number of design variables a minimization of the drag coefficient is achieved. A difference of 1.8% is obtained in relation to the minimum and maximum number of the design variables studied. The Hicks-Henne bump function method did not allow to achieve a drag reduction, keeping the area constraint higher or equal to the initial airfoil area. A reduction in the airfoil area reduced the lift coefficient too. With lift coefficient and airfoil area below the constrained values imposed to the optimizer (relative difference of 0.3% and 14.4%, respectively), an infeasible solution is achieved. With respect to other benchmark results given in the literature, that used a gradient-based optimizer for the FFD control point parameterization method, the achieved results show a good agreement with the range of optimized C_d values.

The NACA 0012 proved to be a very interesting case study. In fact, given the initial definition of the design space, the optimum airfoil shape was partially laying on design space borders and the optimization results showed high sensitivity the to problem dimensions. Starting from a NACA 0012 airfoil in transonic flow conditions, despite the optimized airfoil showed area reduction, a thicker section near the trailing edge compared with the baseline geometry is obtained. The strong shock is substantially weakened and pushed further downstream the trailing edge in the upper surface and further upstream in the lower surface, causing a pressure difference such that it allows for simultaneously reducing the drag coefficient and increasing the lift coefficient, thus resulting in a higher lift to drag ratio. Both parameterization methods are equivalently effective and show a strong dependence on several parameter settings. The optimization performance using the FFD approach revealed a final shape resembling a supercritical airfoil. By increasing the number of control points on the surface of the FFD box, more freedom is given to the optimizer to explore the design space. This was confirmed by the even more visible modifications over the initial design when the number of design variables is increased. Furthermore, it was found that the lift to drag ratio optimization results have a dependence on the different settings of the FFD box position. For the Hicks-Henne bump functions parameterization method three parameter were investigated. Regarding the bump maximum positions h_{i}, an even distribution is selected. For the width of the bump control parameter, t_{i}, a series of t values from 2 to 10 are set for optimization and its impact revealed to be strong on the results. Therefore, a reduction of 52% of the lift to drag ratio between the ends of the range evaluated t for the transonic case is achieved. Noteworthy, different airfoil shapes are created with a reduction of the initial airfoil area being more intense for smaller t and a greater impact on the geometric deformation of the trailing edge for higher t. More local shape control can be achieved with a relatively larger value of t contrary to lower values which gives more global shape control. The analysis of the number of design variables shows their influence in the optimized shape allowing to increase the lift to drag ratio from 3.1 until 251.2 between the minimum and the maximum value. As in the previous parametrization method, the increase of the design variables leads the optimizer to further explore the design space, increasing computational time. Similar behaviour is achieved with both parameterization methods, especially in the trailing edge region. However, in the Hickes-Henne bump function method the movement of the airfoil ends points

from the initial position is not allowed, contrary to the FFD method. In this parameterization method, the airfoil enclosed inside the FFD box can move in the y direction which allows the leading and trailing edge to change from the initial position to a new one in the optimal shape.

For the supersonic flow conditions, the optimized airfoil showed an increase of the lower surface area and a modification of the curvature in the upper surface that tends to soften. The stronger shock wave is substantially decreased from the initial one, being more attached to the optimized shape in the upper surface than on the lower surface. The difference in the c_p distributions between the upper and the lower airfoil surface allows to achieve a higher lift to drag ratio. In terms of the two parameterization methods used in this flow regime both proved to be efficient and showed a strong dependence on several parameter settings, as in the previous flow regime (transonic). A higher number of design variables in both methods allows for greater airfoil modifications. A major transformation is evidenced with the increase of the airfoil curvature at the lower surface and a smoother curvature in the upper surface. It should be noted that, for width of the bump control parameter of the Hicks-Henne bump funtion, t_{i}, a lower impact is revealed in the optimization results, contrary to the transonic case. A reduction of 21% of the lift to drag ratio between the ends of spetrum t is achieved.

6.2. Recommendations and Future Work

As future work is recommended the extension of the proposed framework to 3-D geometries to allow designing complex geometries such as wings (which will require a considerable increase of the computational cost). The application of different parametrization methods, aerodynamic meshes, turbulence models and optimization algorithms should be considered in the future. Also, the continuous adjoint method could be explored.

This page has been left intentionally blank.

References

[1] J. D. Anderson, “Computational Fluid Dynamics: The Basics with Applications,” *McGraw-Hill Inc*. p. 366, 1995.

[2] J. Jackson, “Open Source vs. Proprietary Software,” *PC World*, vol. 114, no. 18, pp. 26–31, 2011.

[3] D. Woods and G. Guliani, *Open Source for the Enterprise: Managing Risks, Reaping Rewards*. O’Reilly Media, 2005.

[4] D. J. Poole, C. B. Allen, and T. Rendall, “Control Point-Based Aerodynamic Shape Optimization Applied to AIAA ADODG Test Cases,” *53rd AIAA Aerosp. Sci. Meet.*, no. January, pp. 1–20, 2015.

[5] D. J. Poole, C. B. Allen, and T. C. S. Rendall, “High-fidelity aerodynamic shape optimization using efficient orthogonal modal design variables with a constrained global optimizer,” *Comput. Fluids*, vol. 143, pp. 1–15, 2017.

[6] C. Lee, D. Koo, and D. W. Zingg, “Comparison of B-Spline Surface and Free-Form Deformation Geometry Control for Aerodynamic Optimization,” *AIAA J.*, vol. 55, no. 1, pp. 228–240, 2017.

[7] A. Saltelli *et al.*, *Global Sensitivity Analysis. The Primer*. 2007.

[8] J. R. R. A. Martins and J. T. Hwang, “Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models,” *AIAA J.*, vol. 51, no. 11, pp. 2582–2599, 2013.

[9] Z. Lyu, Z. Xu, and J. R. R. A. Martins, “Benchmarking Optimization Algorithms for Wing Aerodynamic Design Optimization,” *Eighth Int. Conf. Comput. Fluid Dyn.*, p. 18, 2014.

[10] O. Baysal and K. Ghayour, “Continuous adjoint sensitivities for optimization with general cost functionals on unstructured meshes,” *AIAA J.*, vol. 39, no. 1, pp. 48–55, 2001.

[11] J. R. R. A. Martins, G. K. W. Kenway, and T. Brooks, “Multidisciplinary Design Optimization of Aircraft Configurations Part 2 : High-fidelity aerostructural optimization,” no. May, pp. 1–47, 2016.

[12] O. Baysal and M. E. Eleshaky, “Aerodynamic sensitivity analysis methods for the compressible {E}uler equations,” *J. Fluids Eng.*, vol. 113, no. 4, pp. 681–688, 1991.

[13] A. Jameson, “Aerodynamic design via control theory,” *J. Sci. Comput.*, vol. 3, no. 3, pp. 233–260, 1988.

[14] O. Baysal and M. E. Eleshaky, “Aerodynamic Design Optimization Using Sensitivity Analysis and Computational Fluid Dynamics,” *AIAA J.*, vol. 30, no. 3, pp. 718–725, 1992.

[15] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders, “Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation,” *34th Aerosp. Sci. Meet. Exhib.*, no. January, 1996.

[16] Z. Lyu, G. K. W. Kenway, and J. R. R. A. Martins, “Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark,” *AIAA J.*, vol. 53, no. 4, pp. 968–985, 2015.

[17] W. K. Anderson and V. Venkatakrishnan, “Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation,” *Comput. Fluids*, vol. 28, no. 4–5, pp. 443–480, 1999.

[18] J. Elliott and J. Peraire, “Aerodynamic optimization on unstructured meshes with viscous effects,” *AIAA Pap.*, vol. 97, p. 1849, 1997.

[19] A. Jameson, N. a. Pierce, and L. Martinelli, “Optimum aerodynamic design using the Navier-Stokes equations,” *35th Aerosp. Sci. Meet. Exhib. AIAA Pap. 97-0101*, no. January, 1997.

[20] W. K. Anderson and D. L. Bonhaus, “Airfoil Design on Unstructured Grids for Turbulent Flows,” *AIAA J.*, vol. 37(2), no. 2, pp. 185–191, 1999.

[21] E. J. Nielsen and W. K. Anderson, “Aerodynamic Design Optimization on Unstructured Meshes Using the Navier-Stokes Equations,” *AIAA J.*, vol. 37, no. 11, pp. 1411–1419, 1999.

[22] S. K. Nadarajah, “Aerodynamic Design Optimization: Drag Minimization of the NACA 0012 in Transonic Inviscid Flow,” pp. 1–4, 2013.

[23] G. B. Cosentino and T. L. Holst, “Numerical optimization design of advanced transonic wing configurations[J],” *J. Aircr.*, vol. 23, no. 3, pp. 192–199, 1986.

[24] C. Hirsch, *Numerical computation of internal and external flows, Volume 1: Fundamentals of Computational Fluid Dynamics*. 2007.

[25] W. Malalasekera and H. K. Versteeg, *An Introduction to Computational Fluid Dynamics - The Finite Volume Method*, vol. 44, no. 2. 2006.

[26] C. Nguyen, “Turbulence Modeling,” *Mit*, vol. 1, no. 8, pp. 1–6, 2005.

[27] B. Eisfeld, “Numerical simulation of aerodynamic problems with a Reynolds stress turbulence model,” *Notes Numer. Fluid Mech. Multidiscip. Des.*, vol. 92, pp. 413–421, 2006.

[28] D. C. Wilcox, *Turbulence Modelling for CFD*. .

[29] F. Palacios *et al.*, “Stanford University Unstructured (SU 2): An open-source integrated computational environment for multi-physics simulation and design .,” no. January, pp. 1–60, 2013.

[30] P. R. Spalart, S. R. Allmaras, and J. Reno, “A One-Equatlon Turbulence Model for Aerodynamic Flows Boeing Commercial Airplane Group 30th Aerospace Sciences,” *AIAA Pap. 1992-0439*, 1992.

[31] R. P. Liem, J. R. R. A. Martins, and G. K. W. Kenway, “Expected drag minimization for aerodynamic design optimization based on aircraft operational data,” *Aerosp. Sci. Technol.*, vol. 63, pp. 344–362, 2017.

[32] N. Kroll and J. Fassbender, *MEGAFLOW-Numerical flow simulation for aircraft design*. 2006.

[33] Christopher Rumsey, “Spalart-Allmaras Model.” [Online]. Available: https://turbmodels.larc.nasa.gov/spalart.html. [Accessed: 18-Apr-2018].

[34] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso, “SU2: An Open-Source Suite for Multiphysics Simulation and Design,” *AIAA J.*, vol. 54, no. 3, pp. 1–19, 2015.

[35] F. Palacios *et al.*, “Stanford University Unstructured (SU 2): Open-source Analysis and Design

Technology for Turbulent Flows," no. January, pp. 1–33, 2014.

[36] C. Hirsch, *Numerical computation of internal and external flows, Volume 2: Computational Methods for inviscid and Viscous flows*. 2007.

[37] E. L. I. Turkel, "Far Field Boundary Conditions for Compressible Flows *," vol. 199, pp. 182–199, 1982.

[38] J. Blazek, *Computational Fluid Dynamics: Principles and Applications*. Elsevier Science, 2015.

[39] A. Jameson, W. Schmidt, and E. Turkel, "Numerical Solution of the Euler Equations by Finite Volume Methods Schemes," *AIAA 14th Fluid Plasma Dyn. Conf.*, vol. M, pp. 1–19, 1981.

[40] A. Jameson, "The Origins and Further Development of the Jameson-Schmidt-Turkel (JST) Scheme," *AIAA Aviat.*, no. June, pp. 1–41, 2015.

[41] P. L. Roe, "Approximate Riemann solvers, parameter vectors, and difference schemes," *J. Comput. Phys.*, vol. 43, no. 2, pp. 357–372, 1981.

[42] P. Wesseling, *Principles of Computational Fluid Dynamics*. Springer Berlin Heidelberg, 2009.

[43] D. J. Mavriplis, "Dimitri J. Mavriplis," *Imr'02*, no. June, p. 2, 2002.

[44] V. Lattarulo, T. Kipouros, and G. T. Parks, "Application of the multi-objective Alliance algorithm to a benchmark aerodynamic optimization problem," *2013 IEEE Congr. Evol. Comput. CEC 2013*, pp. 3182–3189, 2013.

[45] I. H. Abbott and A. E. Von Doenhoff, "Theory of Wing Sections: Including a Summary of Airfoil data," *Press*, vol. 11, p. 693, 1959.

[46] A. Sóbester and A. I. J. Forrester, *Aircraft Aerodynamic Design: Geometry and Optimization*. Wiley, 2014.

[47] V. de Brederode, *Fundamentos de aerodinamica incompressível*. Lisboa : Edicao do autor, 1997.

[48] P. H. Cook, M. A. McDonald, and M. C. . Firmin., *AGARD Advisory Report No. 138 EXPERIMENTAL DATA BASE FOR COMPUTER PROGRAM ASSESSMENT*, no. 138. 1979.

[49] Christopher Rumsey, "2D NACA 0012 Airfoil Validation." [Online]. Available: https://turbmodels.larc.nasa.gov/naca0012_val.html. [Accessed: 22-Sep-2018].

[50] M. Giles and N. Pierce, "Adjoint equations in CFD: duality, boundary conditions and solution behaviour," *AIAA Pap.*, pp. 1–17, 1997.

[51] C. Lee, D. Koo, K. Telidetzki, H. Buckley, H. Gagnon, and D. W. Zingg, "Aerodynamic Shape Optimization of Benchmark Problems Using Jetstream," *53rd AIAA Aerosp. Sci. Meet.*, no. January, 2015.

[52] B. Nath, "Aerodynamic shape optimization via discrete adjoint formulation using Euler equations on," 1998.

[53] J. Brezillon and R. P. Dwight, "Aerodynamic Shape Optimization Using the Discrete Adjoint of the Navier-Stokes Equations: Applications towards Complex 3D Configurations," *KATnet II Conf. Key Aerodyn. Technol.*, no. 36, pp. 3–10, 2009.

[54] J. a C. Delaney and J. a D. Seeger, *Sensitivity Analysis*, vol. 12. 2012.

[55] A. Saltelli, K. Chan, and E. Scott, "Sensitivity Analysis," p. 494, 2009.

[56] F. Of and M. Engineering, "Institute of Aerospace Engineering Adaptive Parameterization for Aerodynamic Shape Optimization in Aeronautical Applications," 2015.

[57] J. A. Samareh, “Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization,” *AIAA J.*, vol. 39, no. 5, pp. 877–884, 2001.

[58] D. A. Masters, N. J. Taylor, T. C. S. Rendall, C. B. Allen, and D. J. Poole, “Geometric Comparison of Aerofoil Shape Parameterization Methods,” *AIAA J.*, no. Article in Advance, pp. 1–15, 2017.

[59] V. Braibant and C. Fleury, “Shape optimal design using B-splines,” *Comput. Methods Appl. Mech. Eng.*, vol. 44, no. 3, pp. 247–267, 1984.

[60] M. I. G. Bloor and M. J. Wilson, “Efficient parametrization of generic aircraft geometry,” *J. Aircr.*, vol. 32, no. 6, pp. 1269–1275, 1995.

[61] B. M. Kulfan, “‘ CST ’ Universal Parametric Geometry Representation Method With Applications to Supersonic Aircraft,” *4th Int. Conf. Flow Dyn.*, pp. 1–33, 2007.

[62] R. M. Hicks, “Wing Design by Numerical Optimization,” vol. 15, no. 7, pp. 407–412, 1978.

[63] T. W. Sederberg and S. R. Parry, “Free-form deformation of solid geometric models,” *Proc. 13th Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH ’86*, vol. 20, no. 4, pp. 151–160, 1986.

[64] S. Coquillart, “Extended free-form deformation: a sculpturing tool for 3D geometric modeling,” *ACM SIGGRAPH Comput. Graph.*, vol. 24, no. 4, pp. 187–196, 1990.

[65] V. Sripawadkul, M. Padulo, and M. Guenov, “A Comparison of Airfoil Shape Parameterization Techniques for Early Design Optimization,” *13th AIAA/ISSMO Multidiscip. Anal. Optim. Conf.*, no. September, pp. 1–9, 2010.

[66] “Chapter 1 – A History of Curves and Surfaces in CAGD,” in *Handbook of Computer Aided Geometric Design*, 2002, pp. 1–21.

[67] G. E. Farin, J. Hoschek, and M.-S. Kim, *Handbook of computer aided geometric design*. Elsevier, 2002.

[68] R. Simoni, “Teoria Local das Curvas Teoria Local das Curvas,” p. 81, 2005.

[69] A. Sóbester, “Geometry and Optimization.”

[70] A. Shahrokhi and A. Jahangirian, “Airfoil shape parameterization for optimum Navier-Stokes design with genetic algorithm,” *Aerosp. Sci. Technol.*, vol. 11, no. 6, pp. 443–450, 2007.

[71] N. Giammichele, J. Trépanier, and C. Tribes, “Airfoil Generation and Optimization using Multiresolution B-spline Control with Geometrical Constraints,” *48th AIAA/ASME/ASCE/AHS/ASC Struct. Struct. Dyn. Mater. Conf.*, no. April, 2007.

[72] A. Sóbester and T. Barrett, “The quest for a truly parsimonious airfoil parameterisation scheme,” *Design*, no. September, pp. 1–24, 2008.

[73] M. B. Zupiroli, T. M. L. da Silva, R. Pereira, and A. C. P. B. Junior, “Estudo sobre parametrização de aerofólios para otimização utilizando metodo rmrb,” 2006.

[74] N. P. Salunke, J. Ahamad R. A., and S. A. Channiwala, “Airfoil Parameterization Techniques: A Review,” *Am. J. Mech. Eng.*, vol. 2, no. 4, pp. 99–102, 2014.

[75] A. Aerospace, S. Meeting, A. Aerospace, and S. Meeting, “Transonic Airfoil and Wing Design Using Inverse and Direct Methods,” 2015.

[76] H. H.-Y. Wu, S. Yang, F. Liu, and H.-M. H. Tsai, “Comparison of three geometric representations of airfoils for aerodynamic optimization,” in *16th AIAA Computational Fluid Dynamics Conference, Orlando, Florida*, 2003, no. June.

[77] M. S. Khurana *et al.*, “Airfoil Geometry Parameterization Through Shape Optimizer and Computational Fluid Dynamics,” *Aerospace*, no. January, pp. 1–18, 2008.

[78] E. S. Tashnizi, A. A. Taheri, and M. H. Hekmat, “Investigation of the Adjoint Method in Aerodynamic Optimization Using Various Shape Parameterization Techniques,” *J. Brazilian Soc. Mech. Sci. Eng.*, vol. 32, no. 2, pp. 176–186, 2010.

[79] A. H. Barr, “Global and local deformations of solid primitives,” *ACM SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 21–30, 1984.

[80] T. W. Sederberg, “Computer Aided Geometric Design - Course Notes,” 2011.

[81] S. S. Sarakinos, E. Amoiralis, and I. K. Nikolos, “Exploring Freeform Deformation Capabilities in Aerodynamic Shape Parameterization,” *EUROCON 2005 - Int. Conf. Comput. as a Tool""*, vol. 1, pp. 49–52, 2005.

[82] E. I. Amoiralis and I. K. Nikolos, “Freeform Deformation Versus B-Spline Representation in Inverse Airfoil Design,” *J. Comput. Inf. Sci. Eng.*, vol. 8, no. 2, p. 024001, 2008.

[83] O. Amoignon, J. Hradil, and J. Navratil, “A numerical study of adaptive FFD in aerodynamic shape optimization,” *52nd Aerosp. Sci. Meet.*, no. January, pp. 1–11, 2014.

[84] H. Gagnon and D. W. Zingg, “Two-Level Free-Form Deformation for High-Fidelity Aerodynamic Shape Optimization,” no. September, pp. 1–14, 2012.

[85] S. Lee, K. Chwa, S. Y. Shin, and G. Wolberg, “Image Metamorphosis Using Snakes and Free-Form Deformations,” *Comput. Graph. (ACM).*, vol. 29, pp. 439–48, 1995.

[86] T. W. Sederberg and J. Zheng, “Chapter 15 – Algebraic Methods for Computer Aided Geometric Design,” in *Handbook of Computer Aided Geometric Design*, 2002, pp. 363–387.

[87] R. W. Johnson, *Handbook of Fluid Dynamics, Second Edition*. Taylor & Francis, 2016.

[88] Open Cascade EDF CEA, “SALOME.” [Online]. Available: http://www.salome-platform.org/.

[89] O. C. E. CEA, “Salome Platform Documentation,” *2017*. [Online]. Available: http://docs.salome-platform.org/latest/gui/GUI/salome_architecture_page.html.

[90] J.-E. Lombard, “Introduction to Structured Grid Generation for Aeronautics,” *Ec. Polytech. Fédérale Lausanne / Swiss Inst. Technol. Lausanne*, p. 29, 2011.

[91] SciPy Community, “SciPy Reference Guide 0.13.0,” p. 1229, 2013.

[92] N. Community, “NumPy User Guide,” p. 109, 2013.

[93] G. Karypis and K. Schloegel, “ParMETIS: Parallel Graph Partitioning and Sparce Matrix Ordering Library,” *Version 1.0, Dept. Comput. Sci. Univ. …*, pp. 1–32, 2013.

[94] P. Hewitt, “Aerofoil Optimisation Using CST Parameterisation in SU2 Aerofoil Optimisation Using CST Parameterisation in,” no. August, 2014.

[95] T. Lukaczyk, F. Palacios, J. J. Alonso, and S. Ca, “Managing Gradient Inaccuracies while Enhancing Optimal Shape Design Methods,” no. January, pp. 1–19, 2013.

[96] “ParaView.” [Online]. Available: https://www.paraview.org/.

[97] K. H. Kumar, C. H. K. Kumar, and T. N. Kumar, “Cfd Analysis of Rae 2822 Supercritical Airfoil At Transonic Mach Speeds,” pp. 256–262, 2015.

[98] E. Iuliano, “Global optimization of benchmark aerodynamic cases using physics-based surrogate models,” *Aerosp. Sci. Technol.*, vol. 67, pp. 273–286, 2017.

[99] J. J. Alonso, T. Albring, and N. R. Gauger, “Adjoint Formulation Investigations of Benchmark Aerodynamic Design Cases in SU2,” no. June, pp. 1–13, 2017.

[100] D. Shi-Dong, C.-H. Chen, and S. Nadarajah, “Adjoint-Based Aerodynamic Optimization of Benchmark Proble,” *35th AIAA Appl. Aerodyn. Conf.*, no. January, pp. 1–20, 2017.

[101] G. Yang and A. Da Ronch, “Aerodynamic Shape Optimisation of Benchmark Problems Using SU2,” *2018 AIAA/ASCE/AHS/ASC Struct. Struct. Dyn. Mater. Conf.*, no. January, pp. 1–28, 2018.

[102] G. Carrier *et al.*, “Gradient - Based Aerodynamic Optimization with the elsA Software,” *52nd Aerosp. Sci. Meet.*, no. January, pp. 1–31, 2014.

[103] J. Vassberg, N. Harrison, D. Roman, and A. Jameson, “A Systematic Study on the Impact of Dimensionality for a Two-Dimensional Aerodynamic Optimization Model Problem,” *29th AIAA Appl. Aerodyn. Conf.*, no. June, pp. 1–19, 2011.