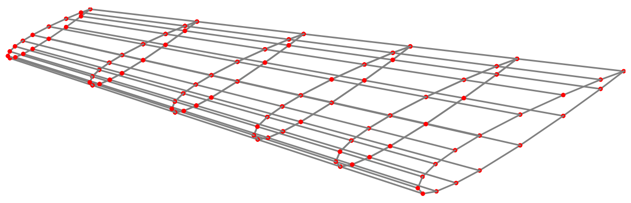


Efficient Aerodynamic Optimization of Aircraft Wings

Pedro Miguel Veríssimo Rodrigues

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor: Prof. André Calado Marta

Examination Committee

Chairperson: Prof. Filipe Szolnoky Ramos Pinto Cunha
Supervisor: Prof. André Calado Marta
Member of the Committee: Prof. João Orlando Marques Gameiro Folgado

April 2018

Dedicated to my family.

Acknowledgments

Firstly, I would like to express my profound gratitude to my supervisor, Professor André Calado Marta, for introducing me to the fascinating subject of aerodynamic optimization and for giving me the opportunity to developed this work. Also, I must thank him for his support and wise guidance since without it, the conclusion of this work with success would be impossible.

Secondly, I would like to thank my parents and my brother for all their support during my stay in Instituto Superior Técnico, specially to my mother who always believed in me.

Finally, I also express my gratitude to my friends for being a huge part of my life and for all their support during the development of this work.

Resumo

A integração disciplinar é um dos fatores-chave mais importantes para design eficiente. Multidisciplinary Design and Optimization é uma técnica promissora para o efeito, uma vez que combina análises multidisciplinares com otimização baseada em métodos de gradiente. Assim, esta técnica requer a avaliação das derivadas das funções de interesse em relação às variáveis de projeto, tarefa essa a mais pesada computacionalmente durante o processo de otimização. Tradicionalmente, o cálculo destas é impreciso e pouco eficiente, uma vez que se recorrem a métodos aproximados. Desta forma, o objetivo deste trabalho é o desenvolvimento de uma ferramenta de otimização eficiente com o propósito de resolver problemas de design aerodinâmico com recurso a informação do gradiente exata. Primeiro, é feito um levantamento dos vários métodos de análise de sensibilidade e assim entender as suas características. De seguida, um modelo aerodinâmico baseado no método do painel é adaptado em cinco módulos, na qual os respetivos módulos de análise de sensibilidade são construídos recorrendo a: diferenciação automática, diferenciação simbólica e método adjunto. Tanto o modelo como a respetiva análise de sensibilidade são verificados com uma ferramenta de design de asas e com o método das diferenças finitas, respetivamente. Um estudo paramétrico é também conduzido para uma asa de referência, analisando assim o impacto das variáveis de projeto nos coeficientes aerodinâmicos. Por último, problemas de design aerodinâmico são resolvidos com sucesso recorrendo à nova ferramenta pois, quando comparado ao uso do método das diferenças finitas, o tempo de otimização poderá ser reduzido em 90%.

Palavras-chave: métodos de gradiente, design aerodinâmico, análise de sensibilidade, método do painel, diferenciação automática, método adjunto.

Abstract

One of the most important keys to the successful design of complex systems is disciplinary integration. Multidisciplinary Design and Optimization is now a promising methodology for the efficient design of such systems, since it combines multidisciplinary analysis with gradient-based optimization techniques. Therefore, this methodology requires the derivatives evaluation of the functions of interest with respect to the design variables, which is the most demanding computational task in the optimization process. Traditionally, those derivatives are calculated inefficiently and inaccurately using approximate methods. Therefore, the objective of this work is to develop an efficient optimization framework to solve aerodynamic design problems using exact gradient information. Firstly, a survey on sensitivity analysis methods is conducted to identify which tools are available and understand their respective merits. Secondly, an aerodynamic model based on the panel method is reformulated into five smaller modules, in which the respective sensitivity analysis blocks are constructed using exact gradient estimation methods: automatic differentiation, symbolic differentiation and the adjoint method. Both the aerodynamic tool and respective sensitivity analysis are validated using a wing design tool and the finite-differences method, respectively. Subsequently, a parametric study is also presented for a baseline wing configuration to survey the impacts of changing the wing's design variables on the aerodynamic coefficients and therefore, to understand the wing's aerodynamic behavior. Finally, aerodynamic optimization problems are solved using the new tool with remarkable success since, when compared to the finite-differences method, the optimization time can be reduced by 90%.

Keywords: gradient-based optimization, aerodynamic design, sensitivity analysis, panel method, automatic differentiation, adjoint method.

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α	Angle of attack; Step size.
β	Generic intensive property.
Γ	Dihedral angle.
δ	Kronecker delta.
δ_r	Root twist angle.
δ_t	Tip twist angle.
Λ	Sweep angle.
λ	Lagrange multiplier for equality constraints; Taper ratio.
μ	Lagrange multiplier for inequality constraints; Dynamic viscosity; Doublet intensity.
$\boldsymbol{\xi}$	Vorticity vector.
ρ	Density.
σ	Stress tensor; Source intensity.
ϕ	Velocity potential.
ψ	Adjoint matrix.
Ω	Feasible region.

Roman symbols

B	Hessian approximation in SQP; Generic extensive property.
b	Wing span.
C_D	Coefficient of drag.
C_L	Coefficient of lift.
C_M	Coefficient of moment.
$\mathbf{CP}$	Concatenation vector of collocation points.
C_p	Coefficient of pressure.
c_r	Root chord.

c_t	Tip chord.
DS	Concatenation vector of panel areas.
$\mathbf{d}$	Search direction vector.
f	Objective function; Interest function.
g	Inequality constraint.
H	Hessian matrix.
h	Equality constraint.
$\mathbf{LPP}$	Concatenation vector of panel's corner points written in the panel's frame of reference.
$\mathbf{LV}$	Concatenation vector of panel's basis vectors.
$\mathbf{l}$	Concatenation vector of panel's 1^{st} basis vectors.
M	Chordwise number of panels.
MAC	Mean Aerodynamic Chord.
$\mathbf{m}$	Concatenation vector of panel's 2^{nd} basis vectors.
N	Semi spanwise number of panels.
$\mathbf{n}$	Concatenation vector of panel's 3^{rd} basis vectors; Normal vector.
$\mathbf{PP}$	Concatenation vector of panel's corner points.
p	Static pressure.
$\mathbf{R}$	Residual equations vector.
S	Wing area.
$\mathbf{V}$	Velocity vector.
$\mathbf{WP}$	Concatenation vector of input points to panel's corner points.
$\mathbf{WP_1}$	Concatenation vector of input points to $\mathbf{X_1}$.
$\mathbf{WP_2}$	Concatenation vector of input points to $\mathbf{X_2}$.
$\mathbf{WP_3}$	Concatenation vector of input points to $\mathbf{X_3}$.
$\mathbf{WP_4}$	Concatenation vector of input points to $\mathbf{X_4}$.
$\mathbf{X_1}$	Concatenation vector of panel's 1^{st} corner points.
$\mathbf{X_2}$	Concatenation vector of panel's 2^{nd} corner points.
$\mathbf{X_3}$	Concatenation vector of panel's 3^{rd} corner points.

$\mathbf{X_4}$	Concatenation vector of panel's 4^{th} corner points.
$\mathbf{x}$	Design vector; Bound vector.
$\mathbf{y}$	State vector.

Subscripts

0	Baseline value.
∞	Free-stream condition.
$airfoil$	Variable related with the airfoil shape.
DV	Indicates design variables.
eq	Equality.
geo	Variable related with the exterior wing shape.
i, j, m, n, k, h	Computational indexes.
k	Iteration number.
L	Stands for lower, in Kutta-condition.
l	Component in the 1^{st} panel's basis vector direction.
m	Component in the 2^{nd} panel's basis vector direction.
n	Normal component.
opt	Variable at optimum value.
U	Stands for upper, in Kutta-condition.
W	Stands for the wake.
x, y, z	Cartesian components.

Superscripts

1	Stands for the panel indicated by the assigned indexes.
2	Stands for the panel's image, referenced by the indexes of the original panel.
L	Stands for lower (bounds).
T	Transpose.
U	Stands for upper (bounds).
'	Means that variable is written in the panel's frame of reference.
*	Variables at their optimum value.

Glossary

ADiMat	Hybrid Automatic Differentiation tool for MATLAB® programs.
AD	Automatic Differentiation is a tool to compute derivatives in computer programs automatically, according to the chain-rule of differential calculus.
BFGS	Broyden-Fletcher-Goldfarb-Shanno formula is an update suited for approximate line search procedures to the Hessian matrix using only gradient information. The later is always positive definite.
BFP	Davidon–Fletcher–Powell formula is an update to the Hessian matrix using only gradient information, keeping the later positive definite.
CAD	Computer Assisted Design uses computer software to aid in the creation or modification of designs.
CFD	Computational Fluid Dynamics is a branch of fluid mechanics that uses numerical methods and algorithms to solve problems that involve fluid flows.
CSD	Complex-Step Derivative is a formula to calculate the derivative of a function accurately.
CSM	Computational Structural Mechanics is a branch of structure mechanics that uses numerical methods and algorithms to perform the analysis of structures and its components.
FD	Finite-Differences schemes are numerical techniques to estimate the derivative of a function.
FFD	Forward Finite-Differences is a first order numerical scheme to estimate the derivative of a function.
GDP	Gross Domestic Product is a monetary measure of the market value of all final goods and services produced in a period of time.

KKT	Karusch-Kuhn-Tucker conditions are the necessary conditions for optimality in a constrained optimization problem.
LP	Linear programming is a type of optimization problems where all functions (objective and constraints) are linear.
MDO	Multi-Disciplinary Optimization is an engineering technique that uses optimization methods to solve design problems incorporating two or more disciplines.
MILP	Mixed-Integer Linear Programming is a type of optimization problems with linear objective and constraints, where some components of the independent variables are discrete.
NACA	National Advisory Committee for Aeronautics under which airfoils were developed.
NLP	Nonlinear Programming is a type of optimization problems where the objective and constraints are nonlinear functions.
QP	Quadratic Programming is a type of optimization problems where the objective function is quadratic and the constraints are linear functions.
RANS	Reynold Averaged Navier-Stokes equations are time-averaged equations of motion for fluid flows.
RPK	Revenue Passenger Kilometer is a transportation industry metric that shows the number of kilometers traveled by paying passengers.
SD	Symbolic Differentiation is a tool to differentiate functions analytically using computer software.
SQP	Sequential Quadratic Programming is a powerful algorithm to solve nonlinear constrained optimization problems.

Chapter 1

Introduction

1.1 Motivation

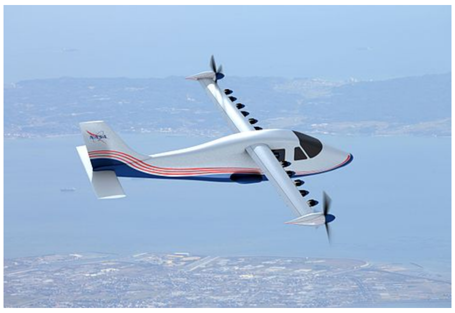
Men always desired to fly. Leonardo Da Vinci, a great artist and inventor of the 15th century, had produced sketches of rudimentary fixed-wing gliders, man powered ornithopters, among other inventions. Despite several attempts to fly were tried through the centuries, the first controlled, powered, heavier than air, manned flight was carried by the Wright brothers, in 1903. Since then, new aircrafts were quickly developed, in part motivated by two incoming world wars and increased civilian demand. In that sense, an example of two remarkable aircrafts are the Douglas DC-3, represented in Figure 1.1 (a), and the Supermarine Spitfire, in Figure 1.1 (b). The first was a fixed-wing propeller driven airliner that revolutionized civil aviation in the 1930s and 1940s. Its design was inspired in the older DC-2 version. At the time, it was able of good range, being capable of transatlantic flights. It was also reliable, comfortable, easy to maintain, fast and completely made of metal, providing competition to the airliner Boeing 247. The second was a single-seat fighter interceptor mainly used by the Royal Air Force during the Second World War. It was, by far, the most produced British aircraft. The elliptical wings were designed to be aerodynamic efficient and to carry an expandable number of weapons. Moreover, the fuselage was prepared to allocate different engine versions allowing to extend the aircraft's power. The aircraft was all built in metal, able to achieve higher speeds than its competitors and it played an important role on the allies victory [1].

(a) Douglas DC-3 [2]

(b) Supermarine Spitfire [3]

Figure 1.1: Remarkable aircrafts during Second World War

Nowadays, new concepts and technology are still being created and explored. Great advances have been achieved specially on propulsion and lighter materials. For example, NASA's researchers are developing alternatives to the traditional carbon-based fueled engines, substituting those by electric propulsion systems. The ongoing project NASA X-57, Figure 1.2 (a), is an electric propeller aircraft, with 14 motors distributed along the wing span and it is expected to achieve a greener, more efficient and silent propulsion system when compared to the traditional propeller systems available [4]. Conversely, aircraft manufacturers are also investing in incorporating new materials, specially composites. An example of this is the new Boeing 787, Figure 1.2 (b), which is a wide-body passenger aircraft, characterized by about 50% of its main structure made of composites. With a more efficient aircraft, Boeing claims to use 20% less fuel than similar aircrafts with the same mission profile, adding value to the company and to its costumers [5].


(a) NASA X-57 [6]

(b) Boeing 787 [7]

Figure 1.2: New aircraft concepts

Through an economical point of view, the aircraft industry have been experiencing a stable and resilient growth in the past two decades. According to Boeing's market outlook of 2014 [8], the global economy is expected to grow, since the gross domestic product (GDP) index is expected to grow at a 3% rate annually, for approximately the next twenty years. Associated with this forecast, passenger traffic is expected to grow by 4.9 percent annually, during the same period. Thus, Boeing's company is expecting to sell 38,850 airplanes, evaluated in more than 5.6 trillion US dollars, in the next twenty years. Figure 1.3 shows that the market is becoming more diverse since it is expected a significant increase in demand from Asia and middle-east countries, being those markets comparable with the US and European markets, here measured by the revenue passenger kilometers (RPK) parameter. In 2014, this increase in demand, associated with the low oil price, represented profits of 20 billion US dollars to the airlines.

Considering the previous arguments, aircraft industry is no different from others, being highly driven by demand and competitive advantage. These may include the fulfillment of new mission requirements or simply newer and more efficient technology became available. Associated with highly expensive developing programs, new radical aircraft configurations capable to fulfill those requirements are not likely to be tried. Instead, new designs may be based on existing concepts with slight modifications. Due to the strong market competitiveness, companies are enforced to have accurate answers in the early stages of design. The massive development of Computer Sciences in the late past century allowed engineers to design and predict accurately the system's behavior, using tools such as Computer-Aided Design (CAD), Computational Fluid Dynamics (CFD) and Computational Structural Mechanics (CSM). As the system's

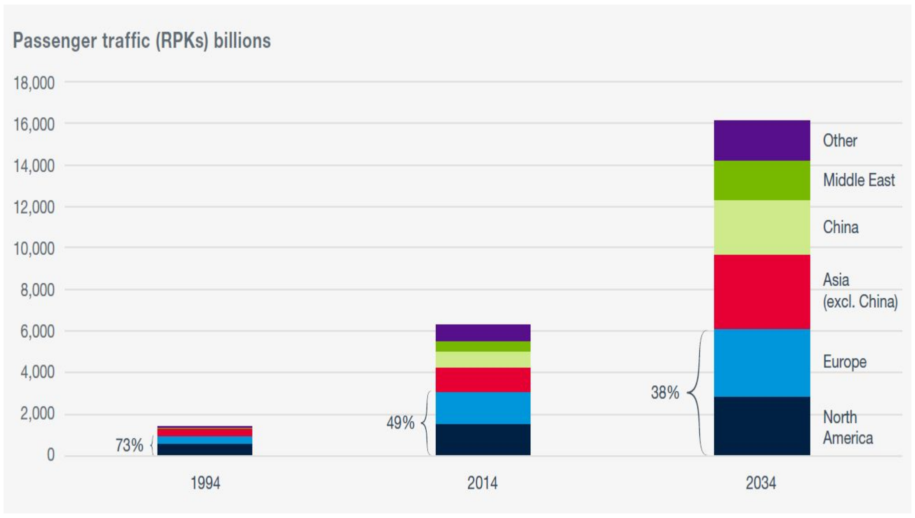


Figure 1.3: Aircraft market share for different regions [8]

complexity increases, a key factor to efficient design is disciplinary integration. Multidisciplinary Design Optimization (MDO) attempts to solve this problem as it will be shown next.

1.2 Multidisciplinary Design and Optimization

Sobieski and Haftka [9] define Multidisciplinary Design and Optimization (MDO) as a methodology to design complex systems in which a strong interaction between disciplines must be considered. In other words, Multidisciplinary Design and Optimization is an Engineering field that applies optimization techniques to design systems where, at least, two disciplines are present and interconnected, both at the analysis and optimization level. In that sense, a structural optimization where the designer, for example, is trying to optimize the wing's structure to avoid flutter is not MDO since the structure/aerodynamics interaction is present only at the analysis level.

Breaking down the name on its basic concepts, Multidisciplinary Design is present when the design team tries to create the system considering all or part of the governing disciplines. An example when that does not happen is well patented in Figure 1.4. As it can be observed, incompatible designs would be generated if no communication is made. Certainly, the most aerodynamic design is not the most resistant, or the cheapest design is not the fastest. Therefore, a design is always based on a trade-off, in some sense.

On the other hand, Optimization is present when designers are concerned with improving a certain design. The optimization process is carried by the information supply from the different disciplines to an optimizer. The latter changes some design characteristics, according to some objective function and constraints. Figure 1.5 shows multidisciplinary optimization in the MDO perspective against sequential disciplinary optimization. Figure 1.5 (a) represents the latter case, where each disciplinary optimization is performed sequentially. This situation corresponds, as suggested by Figure 1.5 (b), to travel in perpendicular directions, changing the design variables associated with the ”active” discipline in each optimization's iteration. Although it may be tempting and easier, this would lead to an improved design

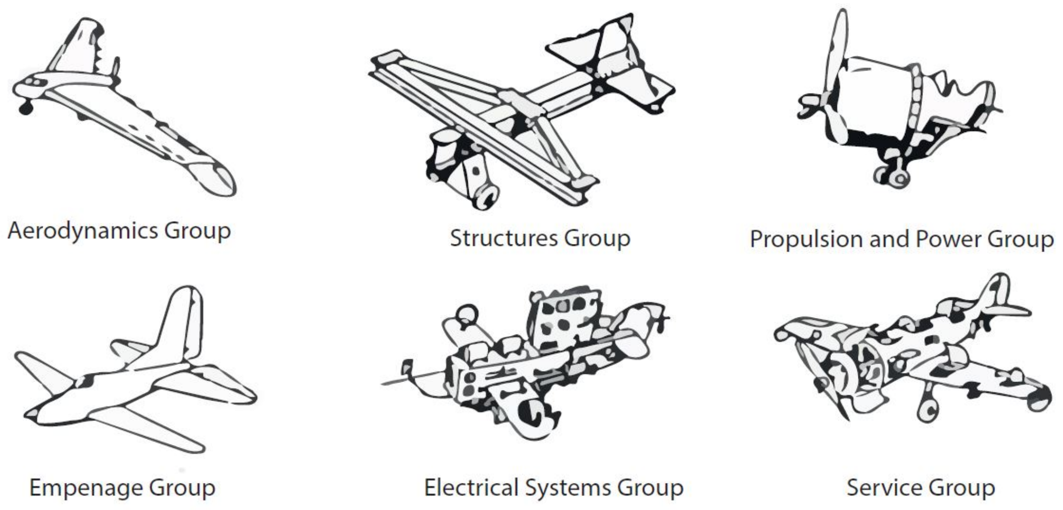
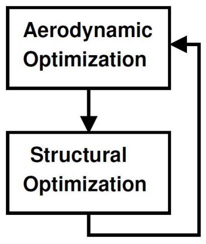
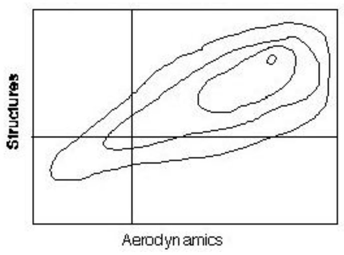


Figure 1.4: Best design according to each discipline, (adapted from [10])

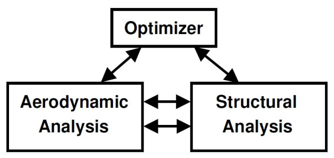
but not the best one, even in the local sense. MDO operates differently and accordingly to Figure 1.5 (c). The travel direction in Figure 1.5 (b) is such that it produces changes in all the design variables at the same time. The result is a optimal design, where a trade-off between disciplines is assured, according to some objective function, while satisfying the design constraints.



(a) Sequential aero-structural optimization



(b) Objective function curves for aero-structural optimization



(c) Simultaneous aero-structural optimization

Figure 1.5: Sequential vs simultaneous multidisciplinary optimization, (adapted from [10])

Due to its strongly tightly coupled multidisciplinary nature, MDO was pioneered by aircraft design. One of the first published works found in literature was presented by Haftka [11]. He developed a procedure to optimize wing's structures subject to drag, stress and strain constraints using Newton's Method.

Nowadays, aircraft designs are parametrized using hundreds or thousands design variables. When dealing with such a large number of parameters, gradient-based optimization strategies are the most efficient algorithms to be used, due to faster convergence rates comparing with heuristic and gradient-free methods. A common feature between gradient-based algorithms is precisely the requirement to evaluate the gradient of the objective function. The adjoint method proved to be accurate and the most efficient for calculating those gradients, or usually called sensitivities, being those calculated exactly and independently of the number of design variables. Several researchers took advantage of this method, both for single discipline optimization and MDO. For example:

- Jameson [12] applied the control theory to optimize the surface of airfoils in transonic regime. His goal was to achieve an optimal configuration for a desired velocity distribution, which is called inverse design. A conformal mapping to a circle was used to parametrize the geometry. The framework consists in a potential flow equation solver, an adjoint solver and an optimization procedure. Several numerical examples were tested with success proving the feasibility of his framework;

- Kennedy [13] developed a new parametrization to be used in composites and he proposed a new beam theory. Due to anisotropic properties of such materials, millions degrees of freedom were required. To deal with those numbers, a parallel direct Schur factorization method was implemented. The structural tool was incorporated in a MDO framework in order to obtain a trade-off between wing weight and induced drag, using gradient-based optimization. A significant reduction in the induced drag was obtained with minimal penalization on the wing's weight;

- Martins [14] developed an aero-structural framework for the optimization of a complete supersonic aircraft. The aerodynamic framework has two modes: an Euler and a RANS equations solver. The structural framework consists on a linear finite-element model with two types of elements. Also, an adjoint solver was presented for the coupled sensitivity analysis, benchmarked with the complex-step derivative and finite-differences. Figure 1.6 shows the normalized runtime (aero-structural optimization runtime divided by the aero-structural analysis runtime) plotted against the number of design variables using the adjoint method, finite-differences and the complex-step derivative. The adjoint method presents a clear advantage comparing with the other methods, being the normalized computational runtime almost independent on the number of design variables, as stated previously.

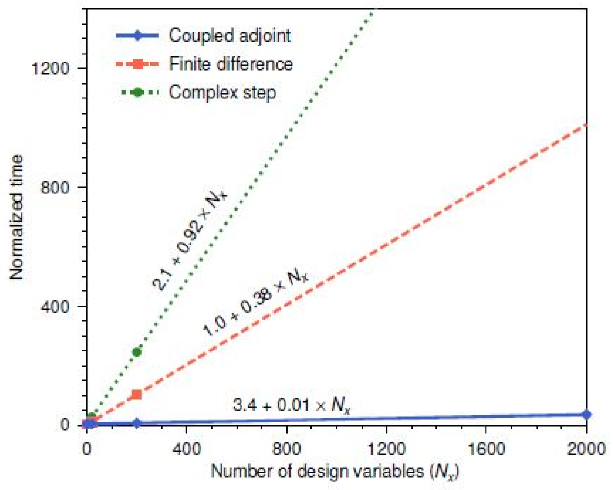


Figure 1.6: Sensitivity analysis methods - Computational cost as a function of the number of design variables, Martins [14]

Nevertheless, MDO still faces many challenges. The two major obstacles are the high computational cost and high organizational complexity. Typically, the individual analysis and optimization times increase at superlinear rates, thus the MDO cost is much higher than the sum of costs of each constituent discipline. Nowadays, a lot of research is being conducted in order to solve these problems. One of these research topics is on MDO architectures, look e.g. Martins and Lambe [15]. An MDO architecture defines not only how the different analysis communicates with each other but also specifies how the overall optimization problem is carried. According to the authors, much research is still needed to be done, not only new and innovative architectures are required but also the existing ones should be benchmarked using a relevant predefined set of problems.

1.3 Framework and Objectives

Almeida [16] developed a dynamic structural model which was integrated in a framework with both dynamic aeroelastic and static aero-structural capabilities to study the behavior of aircraft wings. The wing's structure was modeled using a 3D finite-element model, applied to the wing's neutral axis. To calculate the nodal forces, the aerodynamic loads were calculated using an existing panel method code, developed by Cardeira [17]. In order to couple both the fluid and structural frameworks, staggered (or loosely-coupled) algorithms were implemented, including both volume-continuous and volume-discontinuous methods.

Almeida took advantage of the framework's static aero-structural capabilities to perform gradient-based optimization, using forward finite-differences to estimate the gradient of the interest functions with respect to the design variables. His objective was to minimize the wing's mass, being the root chord, maximum stress at the wing's root and tip deflection constraints to the optimization problem. The lift coefficient was also imposed to be constant. The structural design variables were chosen to be the relative spars location and the spar and skin thicknesses. In the aerodynamic side, the chosen design variables were the angle of attack, taper ratio, sweep, dihedral and twist angles at the wing tip and root. Although an improved design was obtained, the optimization process proved to be quite inefficient due to inaccurate gradient estimation and extensive computational effort, inherent to the finite-difference approach.

In order to obtain an efficient structural optimization, Freire [18] developed a sensitivity analysis framework using tools such as automatic differentiation and the adjoint method. Freire compared his framework performance to the finite-differences approach to estimate the gradient and it was observed that the computational runtime was roughly reduced in half, proving the benefits of efficient gradient estimation.

The objective of this work is to improve the already developed aero-structural tool. By providing a sensitivity analysis framework to accurate and efficient gradient estimation of the interest functions with respect to the design variables of the aerodynamic discipline, it is expected to solve aerodynamic optimization problems more efficiently. The already developed tools and the proposed solution are presented in Figure 1.7. The solid boxes correspond to the work previously done and the dashed boxes to the new implemented features. The optimization procedure will rely on a gradient-based algorithm since it provides the best solution known at the time for efficient aerodynamic optimization.

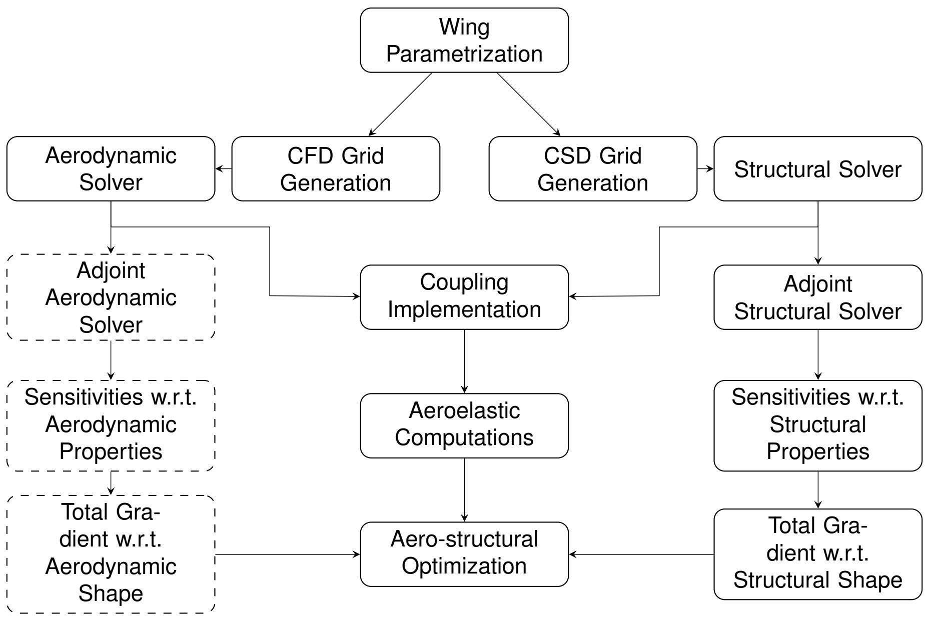

Wing Parametrization
Aerodynamic Solver
CFD Grid Generation
CSD Grid Generation
Structural Solver
Adjoint Aerodynamic Solver
Coupling Implementation
Adjoint Structural Solver
Sensitivities w.r.t. Aerodynamic Properties
Aeroelastic Computations
Sensitivities w.r.t. Structural Properties
Total Gradient w.r.t. Aerodynamic Shape
Aero-structural Optimization
Total Gradient w.r.t. Structural Shape

Figure 1.7: Flowchart illustrating the implementation structure of the aeroelastic framework

1.4 Thesis Outline

This dissertation is structured as follows:

Chapter 2 starts by addressing basic definitions found in Optimization literature. Secondly, a survey on optimization methods is conducted. The concepts of gradient-free and gradient-based optimization methods are explored, highlighting their respective merits and providing some examples.

Chapter 3 addresses the subject of sensitivity analysis, providing several available methods for gradient evaluation, including the discrete-adjoint method, highlighting their main advantages and disadvantages. An example is provided for clarification.

Chapter 4 provides both the necessary theoretical background to support the choice of the aerodynamic model and its implementation. Firstly, the equations of fluid flows are presented but, a special attention is given to incompressible potential flows. Secondly, a numerical technique to calculate the aerodynamic loads on bodies with arbitrary shape is presented, namely the panel method. Subsequently, the implementation of the model is presented in detail since some modifications were made to improve the aerodynamic framework. Finally, the modified program is benchmarked with a similar tool to verify the results.

Chapter 5 presents the new developed sensitivity analysis framework. First, an overview is provided showing how the new tool is organized. Then, each framework's component is presented in detail and the justifications for accuracy and computational efficiency are given as well. At the end of the chapter, the tool is benchmarked with finite-differences, highlighting its main advantages.

Chapter 6 presents a parametric study in order to understand the wing's aerodynamic response to changes in the design variables. First, a baseline wing configuration is chosen. Next, a mesh refinement study is conducted to find a suitable mesh to present the results. After, a series of studies are performed, namely, the impact on the aerodynamic coefficients by changing the taper ratio, wing twist, sweep and dihedral angles, airfoil thickness and camber.

Chapter 7 illustrates the benefits of gradient-based optimization with efficient gradient estimation when compared to the traditional approach of finite-differences. In that sense, three representative optimization problems are solved using both approaches.

Chapter 8 overviews the contents presented in this work highlighting the achievements and provides some ideas for future work.

Chapter 2

Optimization Methods

2.1 Definitions

In this chapter, a survey on optimization methods will be presented to provide enough information to make a conscious choice between the algorithms available when facing aerodynamic optimization problems. Special attention is given to unconstrained and constrained gradient-based methods since they are the most used to this effect. Before proceeding further, some basic definitions found in optimization literature, specially in the MDO context are now addressed:

Design variables, represented here by $\mathbf{x}$, are a given set of parameters that characterizes the system, which are always under the explicit control of the optimizer. They should be as independent as possible. During the optimization process, design variables will change their value until an optimal solution is obtained. They may be continuous or discrete.

State variables, represented here by $\mathbf{y}$, are the result of disciplinary analysis. They may or may not be under control of the optimizer. Usually, they depend implicitly on the design variables through the solution of disciplinary governing equations of the type $\mathbf{R}(\mathbf{x}, \mathbf{y}) = 0$.

Objective function, represented here by f, is a quantitative measure that allows the comparison between two designs. It may be a linear or nonlinear function given implicitly or explicitly with respect to design variables. For example, could be the drag coefficient of an aircraft, its structural weight or even a combination of both.

Constraints are a set of equality or inequality mathematical statements that restricts the values $\mathbf{x}$ might take. Constraints on design variables are called bounds. The subset of design variables that satisfies the constraints are called the design space or feasible region.

2.2 Classification

Several different categories are possible when dealing with optimization methods. A common division found in the literature is into deterministic or heuristic (stochastic) methods [19]. Deterministic methods provides theoretical guarantee that, at least, a local optimum will be found. Nevertheless, these type of methods rely on a strong set of assumptions about the problem. Many times, those assumptions cannot be made and the only way the optimization is possible is through heuristics. Deterministic methods are of zeroth order or gradient-free if only objective function evaluations are performed. Conversely, they are first order or gradient-based if the derivatives of the objective function with respect to design variables are also computed [20].

Most of the times, heuristic methods are used when deterministic fails. Typically, these are used when the objective function is noisy, or design variables are discrete. Unlike deterministic methods, these methods do not require as many assumptions about the optimization problem, in fact, for most algorithms, only objective function evaluations are required [21]. But, on the other hand, running an heuristic does not guarantee that an optimal solution will be reached. Moreover, higher computational power is necessary to continually evaluate the objective function since heuristics performance depends highly on the problem's dimension. Figure 2.1 shows a schematic overview of the different methods found in the literature.

Choosing the best optimization method is highly problem dependent, nevertheless, a good method is such that provides a reliable solution with the least computational effort possible [10]. Gradient-based methods are usually preferred when the objective and constraints functions are smooth and gradients can be computed cheaply. Gradient-free methods are used whether the objective functions are non-differentiable or design variables are discrete [20]. The latter main advantage is the ability to find the global optimum, if it exists, regardless solution's first guess. However, gradient-based algorithms provide a clear stopping criteria and they converge much faster than the gradient-free methods since those usually require less function evaluations to converge and less computational power.

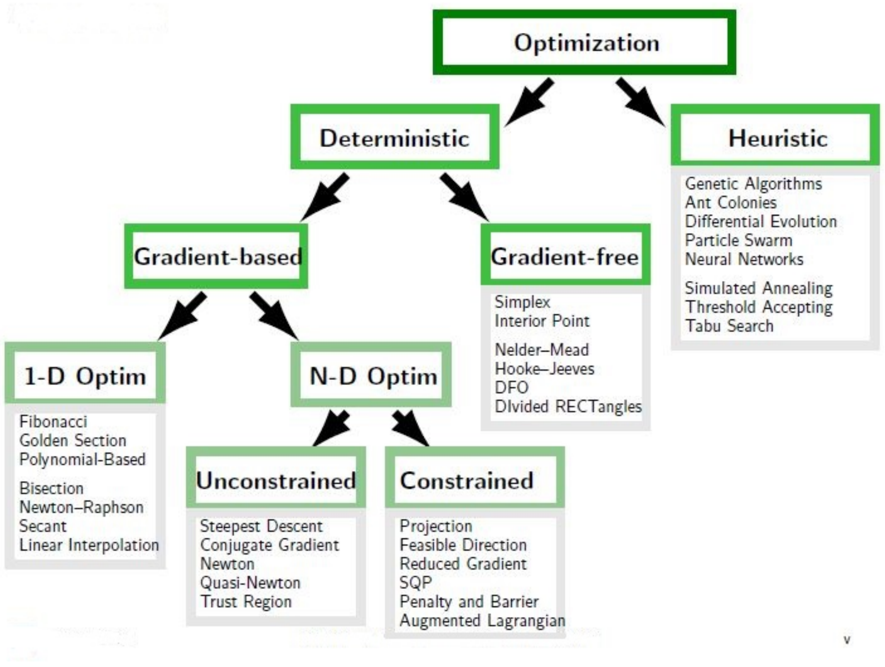


Figure 2.1: Classification of optimization methods [10]

Following a deterministic approach, Belegundu and Chandrupatla [20] define optimization as the process of minimizing a given objective function while satisfying a given set of constraints. A typical engineering optimization problem may be generally expressed as nonlinear programming (NLP) stated as

$$\begin{aligned}
&\text{minimize} && f(\mathbf{x}) && \\
&\text{with respect to} && \mathbf{x} \in \mathbb{R}^n && \\
&\text{subject to} && g_i(\mathbf{x}) \leq 0 && \text{for } i = 1, ..., m \\
& && h_j(\mathbf{x}) = 0 && \text{for } j = 1, ..., \ell \\
& && \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U &&
\end{aligned} \tag{2.1}$$

where $\mathbf{x}$ is the design vector, or the independent variables, $\mathbf{x}^L$ and $\mathbf{x}^U$ are the lower and upper bounds. The objective function is f, h_j and g_i are the equality and inequality constraints, respectively. A deterministic optimization problem may additionally be classified about:

Linearity: An optimization problem is said to be linear programming if both the constraints and objective function are linear, quadratic programming if the objective and constraint functions are quadratic and linear, respectively. The problem is said to be nonlinear programming if both objective and constraint functions are nonlinear.

Constraints: A problem is said to be constrained if the feasible region Ω, is given by: $\Omega = \left\{\mathbf{x} : \mathbf{g}(\mathbf{x}) \leq 0,\, \mathbf{h}(\mathbf{x}) = 0,\, \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U\right\}$ and unconstrained if $\Omega = \mathbb{R}^n$

Convexity: An optimization problem is said to be convex if a convex objective function is minimized under a convex set of design variables, and non-convex otherwise.

Design Variables: The problem is said unidimensional if only one independent variable is present and multidimensional otherwise. Also, those may be continuous or discrete.

2.3 Gradient Based Methods

2.3.1 Unconstrained Gradient Based Methods

Unconstrained optimization problems arise when no restriction on the design variables are imposed, or those are accounted by using penalty functions [20]. The unconstrained optimization problem may be expressed as

$$\begin{aligned}
&\text{minimize} && f(\mathbf{x}) \\
&\text{w.r.t.} && \mathbf{x} \in \mathbb{R}^n
\end{aligned} \tag{2.2}$$

where $\mathbf{x}$ is the design vector and f is the objective function.

Belegundu and Chandrupatla [20] state that the solution of a given unconstrained optimization problem is divided in two main parts:

1. Finding a search direction based on the gradient information;

2. minimize f along that direction (line search).

Unconstrained optimization methods differ mainly on step 1, and they will be addressed in the next sections. Line search corresponds to find the step size α, at iteration k, such that a substantial reduction is obtained without spending to much computational effort,

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \leq f(\mathbf{x}_k) \tag{2.3}$$

where $\mathbf{d}_k$ is the search direction. An accepted step is obtained if the strong Wolfe conditions are satisfied:

$$\begin{aligned} f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) &\leq c_1 f(\mathbf{x}_k) \\ |\nabla_k f(\mathbf{x}_k + \alpha_k \mathbf{d}_k).\mathbf{d}_k| &\leq c_2 |\nabla_k f.\mathbf{d}_k| \end{aligned} \tag{2.4}$$

where c_1 and c_2 are constants. If $c_2 = 0$, then the line search is exact.

Also, a stopping criterion is required which corresponds to the necessary and sufficient conditions to optimality. The necessary condition is

$$\nabla f(\mathbf{x}^*) = 0 \tag{2.5}$$

where $\mathbf{x}^*$ is the optimal solution. Equation (2.5) states that $\mathbf{x}^*$ is a stationary point and it can be a minimum, maximum or a saddle point. Another condition must then be imposed to guarantee that $\mathbf{x}^*$ is a minimum:

$$H(\mathbf{x}^*) = \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial^2 x_n} \end{bmatrix}_{\mathbf{x}=\mathbf{x}^*} \quad \text{is positive definite} \tag{2.6}$$

where H is the Hessian matrix. Therefore, Equations (2.5) and (2.6) are the sufficient conditions to optimality.

Steepest Descent Method

The steepest descent method is one of the oldest methods available for unconstrained optimization problems, firstly introduced by Cauchy [22]. This method uses the symmetrical of the gradient vector at each iteration k for the search direction,

$$\mathbf{d}_k = -\nabla f(\mathbf{x}_k) \tag{2.7}$$

This method has a linear convergence rate and exhibits a unique feature if exact line search is performed. In that particular case, the gradient at iteration point $\mathbf{x}_k$ is perpendicular to the gradient at the next iteration point $\mathbf{x}_{k+1}$. The algorithm is presented in Algorithm 1.

Newton's Method

Despite of Newton's method by itself is not very robust to be used as an optimization method, the concepts behind it are very powerful and they are used on other algorithms [20]. Its main idea consists on approximate the objective function quadratically as

$$f(\mathbf{x}_{k+1}) \approx f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{d}_k + \frac{1}{2}\mathbf{d}_k^T \nabla^2 f(\mathbf{x}_k) \mathbf{d}_k \tag{2.8}$$

```
Input: function $f$, starting point $x_0$ and convergence parameters $\epsilon_r$, $\epsilon_a$ and $\epsilon_g$
Output: $x^*$,minimum of $f$
begin
    repeat
        compute $g(\mathbf{x}_k) = \nabla f(\mathbf{x}_k)$
        if $|\mathbf{g}_k| \leq \epsilon_g$ then
            converged
        else
            compute normalized search direction $\mathbf{d}_k = -\frac{g(\mathbf{x}_k)}{||g(\mathbf{x}_k)||}$
        end
        perform line search to find the step size $\alpha_k$ in the direction of $\mathbf{d}_k$
        update the current point: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
        evaluate $f(\mathbf{x}_{k+1})$
        if $|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| \leq \epsilon_a + \epsilon_r |f(\mathbf{x}_k)|$ $satisfied\ for\ 2\ consecutive\ iterations$
         then
            converged
        else
            set $k = k + 1$
            set $\mathbf{x}_{k+1} = \mathbf{x}_k$
        end
    until converged;
end
```

Algorithm 1: Steepest Descent Method

where $\nabla^2 f(\mathbf{x}_k)$ is the Hessian matrix, $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$, where $\mathbf{d}_k$ is the step. Next, minimization is performed at each iteration k by setting the condition $\frac{\mathrm{d}f}{\mathrm{d}\mathbf{d}_k} = 0$, which is equivalent to

$$\nabla^2 f(\mathbf{x}_k)\mathbf{d}_k = -\nabla f(\mathbf{x}_k) \tag{2.9}$$

When successful, the minimization process is carried by solving Equation (2.9) which gives a series of points that converge to the optimal solution. Despite the quadratical convergence rate for general nonlinear functions, the method may not converge when the initial guess is to far away from the optimum. Another disadvantage is related with the requirement to calculate second-order derivatives in addition to the gradient.

Other methods

Other unconstrained gradient-based methods are available such as the Conjugate Gradient Method, Quasi-Newton's Method and Trust Region methods. The Conjugate Gradient Method [23] is an improvement of the Steepest Descent Method. It introduces the notion of conjugate directions set, which allied with gradient information, the search direction at each step is chosen. It can find a solution of a quadratic function of n variables in n iterations since it converges quadratically. The Quasi-Newton's method [20] is an improvement of the Newton's method. The method's concept is the same as in Newton's method but, an upgrade is made by including a step parameter, estimated by line search. Compared to Newton's method, performing line search avoids the optimizer to reach an higher function value during the iterative process, which can happen for highly nonlinear functions. Another upgrade is that the Hessian matrix does not need to be calculated explicitly but can be estimated by some updating formula such as the BFP and BFGS formulas. Those keep the Hessian approximation positive definite, guaranteeing that the search direction is a descent direction in every iteration. The Trust Region method [24] is an alternative to the pure Newton's method since it solves its lack of robustness. The objective function is approximated by a suited model, typically, a quadratic approximation. This model is then minimized inside a trusted region, a confined space where the model is a good approximation of the function. In each iteration, the calculated minimum is benchmarked with the actual function value to make a decision about moving to the next point and enlarge the trust region or stay in the same point diminishing the latter. The process repeats until the trust region is small enough, according to user specifications.

2.3.2 Constrained Gradient Based Methods

Constrained optimization problems are the most commons in engineering applications. It may be, for example, a structural design problem, subject to displacement and stress constraints. Consider a general nonlinear programming subjected to equality and inequality constraints, expressed as in Equation (2.1).

Assuming well-behaved smooth functions, it can be proven that for $\mathbf{x}^*$ to be a minimum, the first order Karusch-Kuhn-Tucker (KKT) conditions [20] must be necessarily satisfied:

$$\begin{array}{lll}
\text{Optimality:} & \dfrac{\partial \mathcal{L}(\mathbf{x}^*)}{\partial x_k} = \dfrac{\partial f(\mathbf{x}^*)}{\partial x_k} + \displaystyle\sum_{i=1}^{m} \mu_i \dfrac{\partial g_i(\mathbf{x}^*)}{\partial x_k} + \sum_{j=1}^{\ell} \lambda_j \dfrac{\partial h_j(\mathbf{x}^*)}{\partial x_k} = 0 & \text{for } k = 1, ..., n \\
\text{non-negativity:} & \mu_i \geq 0 & \text{for } i = 1, ..., m \\
\text{Complementarity:} & \mu_i g_i(\mathbf{x}^*) = 0 & \text{for } i = 1, ..., m \\
\text{Feasibility:} & g_i(\mathbf{x}^*) \leq 0 & \text{for } i = 1, ..., m \\
 & \dfrac{\partial \mathcal{L}(\mathbf{x}^*)}{\partial \lambda_j} = h_j = 0 & \text{for } j = 1, ..., \ell
\end{array} \tag{2.10}$$

where, $\mathcal{L} = f + \sum_{i=1}^{m} \mu_i g_i + \sum_{j=1}^{\ell} \lambda_j h_j$ is the Lagrangian function, μ_i and λ_j are Lagrange multipliers. Although Equation (2.10) must be verified, it does not guarantee that $\mathbf{x}^*$ is a minimum. It may be a minimum, a maximum or a saddle point. Thus, another condition must be imposed in order to guarantee that $\mathbf{x}^*$ is a minimum:

$$\nabla^2 \mathcal{L}(x^*) = \nabla^2 f(x^*) + \sum_{i=1}^{m} \mu_i \nabla^2 g_i(x^*) + \sum_{j=1}^{\ell} \lambda_j \nabla^2 h_j(x^*) \quad \text{is positive definite} \tag{2.11}$$

where $\nabla^2 \mathcal{L}$ is the Hessian matrix of the Lagrangian function. Equation (2.10) and (2.11) are sufficient conditions for optimality of the constrained optimization problem, stated in Equation (2.1).

Method of Feasible Directions

This method was first presented by Zoutendijk [25] and it is considered one of the most robust available for constrained optimization problems. The method of feasible directions is able to solve nonlinear problems with inequality constraints casted as

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{w.r.t.} && \mathbf{x} \\ &\text{subject to} && g_i(x) \leq 0 \quad \text{for} \quad i = 1, ..., m \end{aligned} \tag{2.12}$$

The algorithm is divided in three main sub-problems. At each iteration:

1. find a feasible point x_k and define the active set of constraints, I;
2. introduce $\alpha = \max\left\{\nabla f^T \mathbf{d}, \nabla g_i^T \mathbf{d}, \text{ for each } i \in I\right\}$ and minimize α with respect to $\mathbf{d}$ to find a descent-feasible direction;
3. perform constrained line search along $\mathbf{d}$ to find the step size, using the lower and upper bounds, $\mathbf{x}^L$ and $\mathbf{x}^U$.

```
   Input: initial feasible point $x_0$, constraints tolerance $\epsilon$
   Output: $x^*$,minimum of $f$
 1 begin
 2     repeat
 3         Determine active set $I = \{i : g_i(x_k) + \epsilon \geq 0, \ \ i = 1, ..., m\}$
 4         Solve sub-problem 2 to find $\mathbf{d}$
 5         if $\alpha = 0$ then
 6             $x_k$ satisfies the KKT conditions (Equation (2.10))
 7         else
 8             Solve sub-problem 3
 9         end
10     until $x_k$ satisfies the optimality conditions;
11 end
```

Algorithm 2: Method of Feasible Directions

The main disadvantage is related with the inability to handle equality constraints. Those may be eventually treated using suitable penalty functions. The detailed methodology is presented in Algorithm 2.

Reduced Gradient Method

The Reduced Gradient method was firstly introduced by Wolfe [26]. This method deals with nonlinear equality constraints although inequalities can be handled introducing a slack variable to the problem. The mathematical statement is

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{w.r.t.} && \mathbf{x} \in \mathbb{R}^n \\ &\text{subject to} && h_j(\mathbf{x}) = 0 \qquad \text{for } j = 1, ..., \ell \\ &\text{and} && \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \end{aligned} \tag{2.13}$$

First, a partition of the design variables are performed using pivoted Gauss elimination,

$$\mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} \tag{2.14}$$

being $\mathbf{y}$ the dependent variables with dimension ℓ, and $\mathbf{z}$ are the independent variables with dimension $n - \ell$. The gradient of the constraints with respect to $\mathbf{y}$ and $\mathbf{z}$ are then calculated as

$$[\nabla h]^T = [B, C] = \left[\frac{\partial h_j}{\partial y_i}, \frac{\partial h_j}{\partial z_k}\right] \tag{2.15}$$

The dependent variable $\mathbf{y}$ is chosen such that B in a non-singular matrix. Consequently, the implicit function theorem guarantees that $\mathbf{y} = \mathbf{y}(\mathbf{z})$ is differentiable with $\mathbf{h}(\mathbf{y}(\mathbf{z}), \mathbf{z}) = 0$ and thus, f can be treated as an implicit function of $\mathbf{z}$, $f(\mathbf{y}(\mathbf{z}), \mathbf{z})$. Differentiating the constraints and the objective function with respect to the independent variables, the reduced gradient,$\frac{df}{dz}$, can be expressed in terms of B and C. The latter is used to compute the search direction, $\mathbf{d}$. Subsequently, line search is performed along $\mathbf{d}$ to determine how far to go on the design space using a bisection strategy. When dealing with nonlinear constraints, it is possible that the new point is not feasible. To guarantee feasibility, the Newton's method is used to find a new feasible point solving $\mathbf{h}(\mathbf{y}, \mathbf{z}_{k+1}) = 0$, where $\mathbf{z}$ is held fixed. The algorithm finishes when $\mathbf{d} = 0$. The detailed algorithm may be found in Belegundu and Chandrupatla [20].

Sequential Quadratic Programming

Nowadays, the Sequential Quadratic Programing method (SQP) is one of the most powerful algorithms available to solve nonlinear constraint optimization problems. This method is now a standard tool to solve complex optimization problems in both academia and industry, which has already proven the ability to solve practical problems efficiently [27, 28]. This method became particular advantageous in several aspects [20, 28]: Neither initial or subsequent points have to be feasible, higher rate of convergence comparing to similar methods, it can handle both inequality and equality constraints and only the active set of constraints are required at each iteration step. This method is characterized by solving a suitable quadratic programming problem that catches the nonlinearities from the initial problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{d}_k^T B_k \mathbf{d}_k + \nabla f(\mathbf{x}_k)^T \mathbf{d}_k \\ \text{w.r.t.} \quad & \mathbf{d}_k :\ \mathbf{d}_k \in \mathbb{R}^n \\ \text{subject to} \quad & \nabla h(\mathbf{x}_k)^T \mathbf{d}_k + h(\mathbf{x}_k) = 0 \\ \text{and} \quad & \nabla g(\mathbf{x}_k)^T \mathbf{d}_k + g(\mathbf{x}_k) \geq 0 \end{aligned} \tag{2.16}$$

whose solution is equivalent to solve the KKT optimality conditions using the Newton's method, if B_k is the Hessian matrix and $\mathbf{d}_k$ the search direction vector. One of the method's features is related with the approximation of the Hessian matrix. Normally, the latter is approximated by the damped BFGS updating formula. After the subproblem is solved, a line search stabilization is performed along $\mathbf{d}_k$ [27], where merit functions are employed to control the step size. A detailed review may be found in Boggs and Tolle [28].

2.4 Heuristic and Gradient Free Methods

Recalling the discussion in the beginning of the present chapter, there are situations where gradient-based optimization processes fail. Such cases are found to be noisy or flat objective functions, the design space is discrete or semi-discrete, objective or constraint functions are discontinuous and thus, derivatives may not exist. Therefore, a solution may only be obtained using deterministic gradient-free or heuristics methods. Common gradient-free methods are the deterministic sampling methods, such as the Hooke-Jeeves pattern search method [29], the simplex method of Nelder-Mead [30] and the DIRECT algorithm [31]. Hooke-Jeeves algorithm is a simple example of a sampling method. It requires two main steps: local minimum search and pattern establishment. First, the current base point $\mathbf{x}_B$ is symmetrically perturbed in each of the coordinate axis directions, according to some step length, and the objective function is evaluated. The best point $\mathbf{x}_P$ in the neighborhood is then calculated, such that $f(\mathbf{x}_P) < f(\mathbf{x}_B)$. Then, a new base point $\mathbf{x}_B$ is obtained such that: $\mathbf{x}_{B'} = \mathbf{x}_B$, $\mathbf{x}_B = \mathbf{x}_P$ and a pattern is constructed with the aid of $\mathbf{x}_E$, given by $\mathbf{x}_E = 2\mathbf{x}_B - \mathbf{x}_{B'}$. If no improved point was found, the step size is reduced and new probing is performed about $\mathbf{x}_B$. Otherwise, probing in the vicinity of the auxiliary variable $\mathbf{x}_E$ is performed to find another improved point $\mathbf{x}_P$, as previously. Suppose that a new improved point $\mathbf{x}_P$ was found from the last operation. If $f(\mathbf{x}_P) < f(\mathbf{x}_B)$, then the new base point is again updated: $\mathbf{x}_{B'} = \mathbf{x}_B$ and $\mathbf{x}_B = \mathbf{x}_P$. Otherwise, the exploration is performed around point $\mathbf{x}_B$ and the step size is reduced. The process is repeated until the step size is reduced according to some stopping criteria.

Another approach is the heuristic optimization. It is common to relate this approach with processes that simulates some behavior found in nature. Although in an econometric perspective, a comprehensive overview on heuristic optimization may be found on the work of Gilli and Winker [21]. Heuristics may be population-based methods such as the Ant Colonies [32], Genetic Algorithms [33], Differential Optimization [34] and Particle Swarm Optimization [35]. Conversely, they may be trajectory-based such as the Simulated Annealing [36] and the Threshold Accepting method [37, 38]. An example of a population-based optimization method is the Ant Colonies, firstly introduced in 1992, by Marco Doringo [39], to solve discrete optimization problems. The goal of this method is to find the optimal path in a graph until the best solution is reached. First, the ant search randomly in the surrounding environment for food. If food is found, pheromones are left in the way back to the colony with an intensity proportional to the abundance, quality and shortness to the nest. The stronger the pheromones, the more ants will follow that path towards their objective. At the same time, other ants are exploring other paths. When the food runs out at that location, the pheromones trails evaporates and other paths are explored. In computational terms, the food is identified as the objective function, the surrounding environment is the search space and the pheromones are modeled through adaptive memory.

An example of a trajectory-based algorithm is the Simulated Annealing method. The former results on the application of statistical mechanic principles since it applies a parallelism between the annealing process of metals to combinatorial optimization. The physical state of the system is taken to be the possible solutions. The objective function is the system's energy and the optimal solution is the minimum energy state. The system's temperature is a control parameter that will be diminished at each outer iteration. For each outer loop, a determined number of inner loops are specified by the user. The algorithm runs until a stopping criteria is verified such as a target temperature. In the inner loop, an initial feasible state $\mathbf{x}_i$ is admitted. Then, a random search in the neighborhood of $\mathbf{x}_i$ is performed to obtain $\mathbf{x}_j$. If $f(\mathbf{x}_j) < f(\mathbf{x}_i)$, $\mathbf{x}_j$ replaces $\mathbf{x}_i$, otherwise $\mathbf{x}_j$ is accepted with a probability given by the Metropolis distribution criterion [40], which depends on the current temperature. This acceptance criterion enforces that for lower temperatures and positive values of $f(\mathbf{x}_j) - f(\mathbf{x}_i)$, the probability of accepting a worse state is diminished.

2.5 Method Selection

According to the information provided previously, it seems a good decision to chose a gradient-based deterministic method over any other to solve aerodynamic optimization problems since the aerodynamic model is built by a composition of regular functions where derivatives exist. These methods provide the advantage of converging much faster since they do not require as many function evaluations and computational power. On the other hand, algorithms that accept constraints will be chosen since the optimization problems addressed in this work have project restrictions that must be accounted. The SQP method is perhaps the most attractive to solve aerodynamic design problems: it is a state-of-the-art solver which can handle both equality, inequality and bound constraints; it presents a quadratic convergence rate; and it is implemented in MATLAB®. Nevertheless, the performance of gradient-based algorithms such as the SQP is highly dependent on how efficiently the required gradients are calculated, thus it is necessary to survey efficient ways of sensitivity analysis which is addressed in the next chapter.

Chapter 3

Sensitivity Analysis Methods

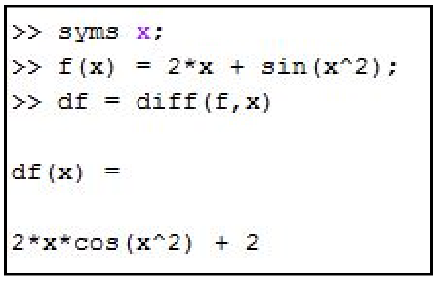
In the previous chapter, it was shown the advantages of using gradient-based optimization techniques when comparing to other available methods to solve nonlinear optimization problems. A common feature between those methods is related with the requirement of providing the derivatives of both the objective function and constraints to the optimization algorithm.

Martins and Hwang [41] define sensitivity analysis as the study of how changes in the input variables will affect the outputs of a given physical system. It has great importance not only in gradient-based optimization but also on uncertainly qualification, error analysis, model development and computational model-assisted decision making. According to Adelman and Haftka [42], the development of automated structural optimization processes led to the use of gradient-based optimization techniques where the sensitivities were used to find a search direction in the design space, in the early 1960s.

When implementing a sensitivity analysis framework, one is concerned with two main aspects: accuracy and computational cost [43]. Since there is not an optimal method for every case, several sensitivity analysis methods will be provided next, and their main characteristics highlighted.

3.1 Symbolic Differentiation

Symbolic differentiation means to apply the well known rules of differentiation, such as the ones applied to the sum, difference, product and quotient of functions, using computational software. This methodology is restricted to explicit functions and may be difficult to implement in large problems. Some libraries are available such as the *Symbolic Math Toolbox* to MATLAB® and *SymPy* written for Python™. An example using the toolbox from MATLAB® is provided in Figure 3.1. First, a symbolic variable type is declared and then, a symbolic function is constructed. In the present case $f(x) = 2x + sin(x^2)$. The differentiated function is readily obtained with the aid of the MATLAB® function `diff()`, as depicted.



```
>> syms x;
>> f(x) = 2*x + sin(x^2);
>> df = diff(f,x)

df(x) =

2*x*cos(x^2) + 2
```

Figure 3.1: Symbolic differentiation example in MATLAB®

3.2 Finite-Differences Method

Finite-differences (FD) is one of the oldest and simpler methods to estimate sensitivities. Consider the vector valued output function $\mathbf{F} = [F_1, ..., F_m]^T$ dependent on the input vector $\mathbf{x} = [x_1, ..., x_n]^T$. The partial derivative of F_j with respect to x_i may be obtained using Taylor-series expansions as

$$F_j(\mathbf{x}_0 + \mathbf{e}_i h) = F_j(\mathbf{x}_0) + h\frac{\partial F_j(\mathbf{x}_0)}{\partial x_i} + \frac{h^2}{2!}\frac{\partial^2 F_j(\mathbf{x}_0)}{\partial x_i^2} + \frac{h^3}{3!}\frac{\partial^3 F_j(\mathbf{x}_0)}{\partial x_i^3} + (...) \tag{3.1a}$$

$$F_j(\mathbf{x}_0 - \mathbf{e}_i h) = F_j(\mathbf{x}_0) - h\frac{\partial F_j(\mathbf{x}_0)}{\partial x_i} + \frac{h^2}{2!}\frac{\partial^2 F_j(\mathbf{x}_0)}{\partial x_i^2} - \frac{h^3}{3!}\frac{\partial^3 F_j(\mathbf{x}_0)}{\partial x_i^3} + (...) \tag{3.1b}$$

where $\mathbf{e}_i$ is the i^{th} basis vector of $\mathbb{R}^n$ and h is the step size. Solving Equation (3.1a) to $\frac{\partial F_j}{\partial x_i}$ one may get

$$\frac{\partial F_j(\mathbf{x}_0)}{\partial x_i} = \frac{F_j(\mathbf{x}_0 + \mathbf{e}_i h) - F_j(\mathbf{x}_0)}{h} + \mathcal{O}(h) \tag{3.2}$$

and solving now Equation (3.1b) to $\frac{\partial F_j}{\partial x_i}$ yields

$$\frac{\partial F_j(\mathbf{x}_0)}{\partial x_i} = \frac{F_j(\mathbf{x}_0) - F_j(\mathbf{x}_0 - \mathbf{e}_i h)}{h} + \mathcal{O}(h) \tag{3.3}$$

The formulas in Equation (3.2) and Equation (3.3) are the forward finite-difference (FFD) and backward finite-difference, respectively. These formulas are first order accurate because the truncation error is proportional to the step size h. If higher convergence rate is desired, a central finite-differences formula may be obtained, which is second order accurate. Subtracting Equation (3.1b) from Equation (3.1a), dividing by h and solving for $\frac{\partial F_j}{\partial x_i}$ yields

$$\frac{\partial F_j(\mathbf{x}_0)}{\partial \mathbf{x}_i} = \frac{F_j(\mathbf{x}_0 + \mathbf{e}_i h) - F_j(\mathbf{x}_0 - \mathbf{e}_i h)}{2h} + \mathcal{O}(h^2) \tag{3.4}$$

Since the truncation error depends on the step size, one may be tempted to set h smaller as possible. Nevertheless, if the step size is too small, subtractive cancellation will happen, leading the derivative value to zero. According to Martins et al. [44], there exists an optimum h that minimizes the overall error, although it may be impracticable to find it, if the cost of evaluating F is high.

Several gradient-based optimization codes use this method to calculate the sensitivities. Their main advantage is related with the easiness of implementation since very little is required to know about F. From a practitioner perspective, finite-differences may be a solution for gradient estimation if the model is unknown. Nevertheless, the cost of estimating the function's gradient using these formulas is directly proportional to the size of $\mathbf{x}$. More precisely, the number of function evaluations are $n + 1$ using forward or backward finite-differences, and $2n$ using central FD, for each component of $\mathbf{F}$. If $\mathbf{x}$ represents the design variables in some typical engineering optimization problem, both the cost of evaluating F and the number of design variables may turn the usage of this method impracticable.

3.3 Complex-Step Derivative

The complex-step derivative (CSD) allows to estimate sensitivities using notions of complex-variable calculus. The method was first presented in Lyness [45], and in Lyness and Moler [46]. Later, this theory was rediscovered by Squire and Trapp [47] who developed a formula to estimate the first derivative. Consider Taylor-series expansion of the real vector valued function $\mathbf{F} = [F_1, ..., F_m]^T$, dependent on the input vector $\mathbf{x} = [x_1, ..., x_n]^T$, in the imaginary axis direction,

$$F_j(\mathbf{x}_0 + ih\mathbf{e}_k) = F_j(\mathbf{x}_0) + ih\frac{\partial F_j(\mathbf{x}_0)}{\partial x_k} - \frac{h^2}{2!}\frac{\partial^2 F_j(\mathbf{x}_0)}{\partial x_k^2} - \frac{ih^3}{3!}\frac{\partial^3 F_j(\mathbf{x}_0)}{\partial x_k^3} + (...) \tag{3.5}$$

where ih is a pure imaginary step. Taking the imaginary part of both sides of Equation (3.5) and dividing by h yields

$$\frac{\partial F_j(\mathbf{x}_0)}{\partial x_k} = \frac{\mathrm{Im}[F_j(\mathbf{x}_0 + ih\mathbf{e}_k)]}{h} + \mathcal{O}(h^2) \tag{3.6}$$

The formula presented in Equation (3.6) is second order accurate, since the truncation error depends quadratically on the step size. A major advantage comparing to finite-differences is the non existence of subtractive cancellation since there is no subtraction operations. Thus, one can obtain sensitivities with as much precision as the machine allows. Although great advantages are encountered, this method is harder to implement than finite-differences. Unlike the latter, the complex-step derivative operates with complex algebra which is not supported by all programming languages [41]. Therefore, it may be required extra implementation effort in certain programming languages since both data types and intrinsic functions may have to be redefined. Moreover, the cost of estimating the partial derivatives of F_j with respect to all the components of $\mathbf{x}$ require n function evaluations. Nevertheless, the complex-step derivative is a powerful method, specially for benchmarking derivatives obtained with other methods.

In order to compare this method to the FD approach, a simple example is presented. Consider the analytical function given by

$$f(x) = \frac{sin(x) + sin(3x)}{(x - \pi - 1)^2} \tag{3.7}$$

whose the first derivative is

$$f'(x) = \frac{cos(x) + 3cos(3x)}{(x - \pi - 1)^2} + \frac{2(sin(x) + sin(3x))}{(x - \pi - 1)^3} \tag{3.8}$$

This study consists in benchmarking derivative approximations given by finite-differences formulas and the complex-step derivative with the exact value, given by Equation (3.8). Furthermore, the exact derivative value will be benchmarked with those formulas for several step sizes, at point $x = 1$. Figure 3.2 presents the result of this study, where the normalized relative error is plotted against the step size, for the different formulas. This error was calculated as

$$\epsilon = \left| \frac{f'_{num} - f'_{exact}}{f'_{exact}} \right| \tag{3.9}$$

where f'_{num} is the derivative estimated by FD or the complex-step derivative, and f'_{exact} is the exact value. Observing Figure 3.2, one can realize that the forward finite-difference approximation converges linearly to the exact value until an optimum step size of approximately 10^{-9}. The normalized error is observed to be approximately 10^{-10}, for that step size. As the step size is reduced further, subtractive cancellation effects become important, increasing the normalized relative error until it becomes equal to 1. Similar trends are found for the central finite-difference approximation. As the step size is reduced, the formula converges quadratically until the step reaches about 10^{-6}. Reducing the step size even further, the same trend of subtractive-cancellation is verified, as previously. The normalized error is approximately 10^{-12} for the optimum step size. Considering the accuracy of the previous formulas, central finite-differences would be preferable since a better approximation is achieved. Nevertheless, using this formula is roughly twice computationally more expensive than the forward finite-difference formula.

Considering the complex-step derivative approximation, the formula converges quadratically to the exact value as the step size is reduced until a step size of 10^{-8}. Reducing the step size even further has no impact on improving the accuracy since the normalized error is approximately equal to 10^{-16}. One can conclude that for a sufficient small step size, the accuracy of the formula is the same as the evaluation of the derivative function, being the round-off the only source of error. On the other hand, this formula proves to be several times more accurate than forward and central finite-differences. Table 3.1 summarizes the results, showing the value, whose digits in agreement with the exact value are colored in red, the optimal step size and the error relative to the exact value, for each formula.

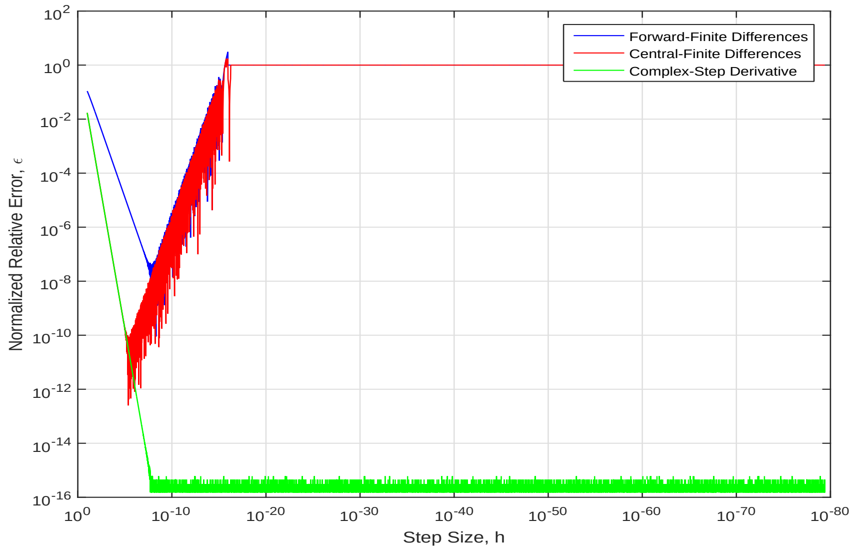


Figure 3.2: Error in the derivative estimation by finite-differences and the complex-step derivative as a function of the step size

3.4 Semi-Analytical Methods

According to Peter and Dwight [48], the adjoint method is the best option for efficient aerodynamic shape optimization. The first application of the adjoint method in aerodynamics goes back to Bristow and Hawk [49, 50], using the panel method. In this subsection, two methods with a common derivation

	Value	h_opt	Normalized Relative Error
Exact	-0.182797431022074	-	-
FFD	-0.182797431005627	4.836E-09	8.997E-11
CFD	-0.182797431022028	4.344E-06	2.513E-13
CSD	-0.182797431022074	$>$ 1.954E-08	3.037E-016

Table 3.1: Accuracy of the finite-difference and the complex-step derivative formulas to estimate f' at $x = 1$

are presented: the direct and the adjoint method. They are called semi-analytical methods because they are derived using concepts of differential calculus but, nothing is said about how the intermediate partial derivatives should be calculated, thus they may be computed using approximation methods such as finite-differences.

Consider again a vector valued output function $\mathbf{F} = [F_1, ..., F_m]^T$, the independent variables vector $\mathbf{x} = [x_1, ..., x_n]^T$ and the state vector $\mathbf{y} = [y_1, ..., y_k]^T$. The dependence of the outputs on the inputs through $\mathbf{F}$ is

$$\mathbf{f} = \mathbf{F}(\mathbf{x}, \mathbf{Y}(\mathbf{x})) \tag{3.10}$$

where the state variables depend implicitly on the independents through the solution of residual equations,

$$\mathbf{r} = \mathbf{R}(\mathbf{x}, \mathbf{Y}(\mathbf{x})) = 0 \tag{3.11}$$

where $\mathbf{R} = [R_1, ..., R_k]^T$ are the residual equations. According to the chain-rule, the total derivative of $\mathbf{f}$ with respect to $\mathbf{x}$ in Equation (3.10) yields

$$\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}} + \frac{\partial \mathbf{F}}{\partial \mathbf{y}} \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}} \tag{3.12}$$

being $\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}}$ the $m \times n$ jacobian matrix. Since the residual equations must be always verified, it is true that

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}\mathbf{x}} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \mathbf{y}} \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}} = 0 \tag{3.13}$$

which is equivalent to

$$\frac{\partial \mathbf{R}}{\partial \mathbf{y}} \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}} = -\frac{\partial \mathbf{R}}{\partial \mathbf{x}} \tag{3.14}$$

Solving Equation (3.14) to $\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}}$ and introducing into Equation (3.12) yields

$$\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \underbrace{- \frac{\partial \mathbf{F}}{\partial \mathbf{y}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{y}}\right]^{-1}}_{[\psi]^T} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\right] \tag{3.15}$$

Since the matrix product is associative, there are two ways of solving Equation (3.15): the adjoint and the direct method.

3.4.1 Direct Method

One way of solving Equation (3.15) is to solve directly the set of linear equations in Equation (3.14) to $\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}}$ and then substitute the result in Equation (3.12). This method is more advantageous when $m > n$, in other words, the number of output variables are bigger than the number of inputs. It should be noticed that solving Equation (3.14) corresponds to solve n linear systems of equations, one for each column of $-\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$.

3.4.2 Adjoint Method

Another option to solve Equation (3.15) is through adjoint equations. Considering the product of the first two matrix after the minus sign in Equation (3.15) and assigning it an adjoint matrix, as suggested by the under bracket, results

$$[\psi]^T = -\frac{\partial \mathbf{F}}{\partial \mathbf{y}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{y}}\right]^{-1} \tag{3.16}$$

which is equivalent to

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{y}}\right]^T \left[\psi\right] = \left[-\frac{\partial \mathbf{F}}{\partial \mathbf{y}}\right]^T \tag{3.17}$$

where $[\psi]$ is the adjoint matrix with dimensions of $k \times m$. Equation (3.15) can then be solved replacing $[\psi]^T$ after solving the linear system in Equation (3.17). This method is best suited when $n > m$, since now the adjoint equations are independent of the number of inputs $\mathbf{x}$. One can realize that Equation (3.17) represents m systems of equations, one for each column of $[\psi]$.

It is now easy to understand why is this method so attractive to estimate aerodynamic sensitivities. Typically, one is interested to calculate derivatives of very few output functions, such the lift and drag coefficients, which may be identified here by the output function $\mathbf{F}$, with respect to hundred or thousand design variables (wing shape), identified here by $\mathbf{x}$, using most likely millions of state variables (pressure, density...), identified here by $\mathbf{y}$.

3.5 Automatic Differentiation

Consider a computer program with n input, l intermediate and m output variables. Typically, the program is constructed based on elementary building blocks such unary and binary operations. Those can be combined to obtain more elaborated functions and so on until the desired final result is obtained. Thus, a computer program may be decomposed in elementary functions, such that: $t_i = T_i(t_1, ...t_{i-1})$, where t_i represents a computer variable and T_i an elementary function [41].

Automatic differentiation (AD) is an analytical method that allows the exact jacobian calculation in computer programs based on the idea of an elementary decomposable program. The procedure consists in a combination of differentiation of intrinsic functions and accumulation according to the chain-rule of differential calculus [51]. In automatic differentiation, the chain-rule for computing a generic $\frac{dt_i}{dt_j}$ is given by either

$$\frac{dt_i}{dt_j} = \delta_{ij} + \sum_{k=j}^{i-1} \frac{\partial T_i}{\partial t_k} \frac{dt_k}{dt_j} \tag{3.18a}$$

or

$$\frac{dt_i}{dt_j} = \delta_{ij} + \sum_{k=j+1}^{i} \frac{dt_i}{dt_k}\frac{\partial T_k}{\partial t_j} \tag{3.18b}$$

The first, given by Equation (3.18a), represents the forward mode, and the second, given by Equation (3.18b), represents the reverse mode. In the former mode, index j is kept fixed and i is incremented from $i = 1$ until ultimately to $i = n + l + m$. In the latter mode, index i is fixed and j varies from $j = n + l + m$ until ultimately to $j = 1$. Therefore, a sweep in forward mode corresponds to obtain a column of the matrix whose elements are $\frac{dt_i}{dt_j}$, meaning to choose a variable in which all the variables will be differentiated about. On the other hand, in the reverse mode, a sweep corresponds to obtain a row of the matrix whose elements are given by $\frac{dt_i}{dt_j}$. This means to choose a variable to differentiate with respect to all others. Thus, the reverse mode seems more efficient when $n > m$, which is a typical situation in many aerodynamic optimization problems. The opposite is true for the forward mode. Nevertheless, the reverse mode presents a clear disadvantage. It is expected that the memory requirements are higher when comparing to the forward mode since the program needs to be run once forward to store all intermediate variables and once backwards to calculate the derivatives and apply the chain-rule [51].

Implementation and Available Tools

Automatic Differentiation may be implemented by two different approach. One of them is the source-code transformation, a source-to-source, compiler based approach which intersperses the original source code and augments it with additional lines of code to perform the differentiation. As a consequence, the resulting program computes not just the function's jacobian but also the function's values. This approach presents some advantages such as the possibility of implementation in every computer-program language and code optimization, since the code parsing is done before compiling [51]. But since the original code is appended, large codes may become easily unreadable and difficult to debug. Several packages were developed using source-code transformation. Examples include ADIFOR [52] to FORTRAN and Tapenade [53] for both C/C++ and FORTRAN.

Another popular method is called operator-overloading. In this approach, variable data types are redefined to support both its own value and an associated derivative. Berland [54] calls them dual numbers. Operators are overloaded to return both the result and the derivative of such operation. In this latter approach, the code is unchanged, making easier to detect bugs during code development. Nevertheless, it presents the disadvantage of working only on object oriented languages, such as C/C++, FORTRAN or MATLAB®, and additionally, it runs slowly comparing to the previous approach. Developed tools using exclusively operator-overloading include ADOL-C [55] to C/C++ and ADOL-F [56] to FORTRAN.

ADiMat

ADiMat is the newest AD tool for MATLAB®. It is an hybrid framework that combines the best features of source-code transformation and operator-overloading [57]. In simplistic terms, when a new code have to be differentiated, a code parser analyses the all code and categorizes each of the present entities in variables, functions or unknowns. The latter category is necessary since MATLAB® syntax is powerful, allowing entities with the same name to be used with a different meaning in the same code.

An algorithm tries to discover the nature of each identifier by analyzing the search path and the builtin database, where these identifiers might be defined. Next, a dependence analysis is done to determine the active path, in other words, the set of variables involved in the intermediate computations from the independent to dependent variables. The latter step is also required since the user may be interested in computing the jacobian using only a portion of the inputs and outputs. Finally, the differentiation is performed and those statements are added to code.

3.6 Usage of Methods

In this work, every technique presented will be used, except the direct method, since they present unique features that make them well suited for a particular application. Symbolic differentiation will be used to calculate derivatives of simple explicit functions since it allows some algebraic simplifications. When applied to routines called many times in the code, this methodology is, by far, the most computationally efficient since it is possible to reduce the number of calculations as it will be clear in Chapter 5. The finite-differences method will be used exclusively to verify the sensitivity analysis framework since it is very easy to be implemented and it works for a broader set of built-in functions. The complex-step derivative will also be used for benchmarking purposes every time complex variable calculations are possible. It presents the advantage of accurate verification since for a step size small enough, the derivatives calculated using the CSD match the ones calculated using an analytical method. The adjoint method will be used to calculate the sensitivities of the disciplinary residual equations and output functions since the number of inputs is much higher than the number of outputs. Automatic differentiation will also be used since it is easy to apply, it calculates the derivatives exactly and both forward and reverse modes are implemented in ADiMat which allows to chose the best implementation according to the number of inputs and outputs of the function under consideration.

Chapter 4

Aerodynamic Model and Framework

This chapter presents both the theoretical support to justify the chosen aerodynamic model and its implementation. Firstly, the fundamental equations of fluid dynamics and related models will be presented. Incompressible flows will be primary considered since the speeds of operation are expected to be low. Incompressible flows are a good approximation typically when the cruise Mach number is less than 0.3. A major advantage when dealing with such flows is the large amount of knowledge and techniques already developed. Secondly, it will be shown that the potential flow model is accurate enough, much easier to implement and generate results faster, comparing with other models. Subsequently, a numerical technique to solve incompressible potential flows about complex geometries will also be presented, namely the panel method. Finally, the aerodynamic framework will be explained in detail since some modifications were made to improve the tool.

4.1 Fundamental Equations in Incompressible Flows

The governing equations of fluid flows are a mathematical description of such flows behavior. They are conservation laws since they state that quantities such as mass, momentum and energy are conserved [58].

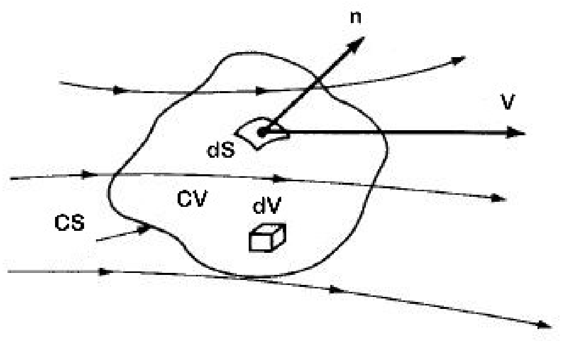


Figure 4.1: Fixed control volume in space (CV), bounded by the control surface (CS), (adapted from [58]).

Consider the control volume (CV), bounded by a closed surface (CS) and fixed in space, as depicted in Figure 4.1. The Reynolds transport theorem accounts for the extensive property rate of change as

$$\frac{dB}{dt} = \int_{CV} \frac{\partial(\beta\rho)}{\partial t} dV + \int_{CS} \beta\rho(\mathbf{V}.\mathbf{n})dS \tag{4.1}$$

where B is an extensive property of interest, such as mass (m) and linear momentum $(m\mathbf{V})$, β is an intensive property, given by B per unit mass (e.g. velocity: linear momentum per unit mass), $\mathbf{V}$ is the velocity vector, ρ is the density and $\mathbf{n}$ is the outwards control surface's normal vector.

Conservation of Mass

The conservation of mass principle states that mass cannot be created neither destroyed,

$$\frac{dm}{dt} = 0 \tag{4.2}$$

According to Ferziger and Peric [59], it follows from the definition of B and β that $\beta = 1$, when $B = m$. Substituting the results in Equation (4.1), applying the Gauss' theorem to the second term of the right hand side, and recognizing that it must hold for an arbitrary control volume, the differential form is obtained as

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \mathbf{V}) = 0 \tag{4.3}$$

where $\nabla.$ is the divergence operator. If the incompressible flow assumption is made, density is constant and Equation (4.3) simplifies to

$$\nabla.\mathbf{V} = 0 \tag{4.4}$$

Conservation of Momentum

Momentum equations states the second Newton's law of motion for a given control volume. For the present case $B = m\mathbf{V}$ and consequently, $\beta = \mathbf{V}$. Second Newton's law of motion may be written for a fixed control volume, through Equation (4.1) as

$$\int_{CV} \frac{\partial(\rho \mathbf{V})}{\partial t} dV + \int_{CS} \rho \mathbf{V}.(\mathbf{V}.\mathbf{n}) dS = \int_{CS} (\sigma.\mathbf{n}) dS + \int_{CV} \mathbf{f_b} dV \tag{4.5}$$

Following the same procedure as for obtaining the mass conservation equation, results

$$\frac{\partial(\rho \mathbf{V})}{\partial t} + \nabla.(\rho \mathbf{V}.\mathbf{V}) = \nabla.\sigma + \rho \mathbf{f_b} \tag{4.6}$$

Considering the right hand side of Equation (4.6), $\mathbf{f}_b$ represent the body forces (eg. gravity, Coriolis or electromagnetic). If the fluid is assumed to be Newtonian, the stress tensor σ depends only on the fluid velocity. Assuming also that the fluid is incompressible, the momentum equation simplifies to

$$\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V}(\nabla.\mathbf{V}) = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{V} + \mathbf{f}_b \tag{4.7}$$

where ∇ (without the dot) is the gradient operator, p is the static pressure, ∇^2 is the Laplacian operator and ν is the cinematic viscosity. Equation (4.3) and Equation (4.7) are a coupled system of nonlinear equations referred as the Navier-Stokes equations for incompressible flows. The unknowns are the static pressure p and the three components of the velocity vector. Only a few solutions are known for very simple cases, e.g. the Couette flow, therefore, sophisticated numerical techniques are required to solve those equations for most of the practical engineering situations.

4.2 Potential Flow Model

4.2.1 Laplace Equation

Since aircraft wings operating at a low angle of attack may be considered slender bodies, viscous effects are confined to thin boundary layers next to the surface, opposed to the essentially inviscid exterior flow. Thus, the analysis may be greatly simplified ignoring the boundary layers and assuming that the entire flow field is inviscid. This model is based on the assumption of inviscid and irrotational flow. The latter corresponds to a situation where the average angular velocity $\boldsymbol{\omega}$ is zero. This is true in many inviscid flows. In aerodynamics is usual to define the vorticity vector $\boldsymbol{\xi}$, defined as $\boldsymbol{\xi} = 2\boldsymbol{\omega}$. Based on the latter definition, the irrotational condition is

$$\boldsymbol{\xi} = \nabla \times \mathbf{V} = 0 \tag{4.8}$$

where $\nabla\times$ is the rotational operator. Due to the vector identity $\nabla \times \nabla(\phi) = 0$, it follows from Equation (4.8) that the velocity field can be written through a potential function ϕ as

$$\mathbf{V} = \nabla\phi \tag{4.9}$$

Substituting Equation (4.9) in Equation (4.4), the Laplace equation is readily obtained,

$$\nabla^2\phi = 0 \tag{4.10}$$

In order to solve the Laplace equation, two boundary conditions are needed,

$$\nabla\phi.\mathbf{n} = 0 \tag{4.11a}$$

$$\lim_{\mathbf{r}\to\infty}(\nabla\phi - \mathbf{V}_\infty) = 0 \tag{4.11b}$$

Equation (4.11a) states that the body is impermeable. That is, the velocity must be tangent to the body's surface. Equation (4.11b) says that the velocity field must tend to the unperturbed free-stream field, far away from the body. The Laplace equation presents a clear advantage comparing with the Navier-Stokes equations. The former is just one equation, as opposed to the couple system of equations, and it is linear. This means that if a collection of elementary solutions ϕ_i are known, any linear combination of those is also a solution,

$$\phi = \sum_i c_i\phi_i \tag{4.12}$$

4.2.2 Elementary Solutions

As stated in Equation (4.12), it is possible to obtain more complex solutions by the superposition of elementary ones. Typical elementary solutions includes the free-stream, source/sink and doublet potentials [60].

Free Stream

Since the Laplace equation is a second-order partial differential equation, any first-order polynomial is a solution. Considering, for example,

$$\phi(x, y, z) = u_\infty x + v_\infty y + w_\infty z \tag{4.13}$$

substituting the velocity potential in Equation (4.9) one obtains $\mathbf{V} = [u_\infty\ v_\infty\ w_\infty]^T$, an uniform flow of components u_∞, v_∞ and w_∞.

Sources/Sinks

Another elementary solution is the source/sink. The velocity potential is given in Cartesian coordinates by

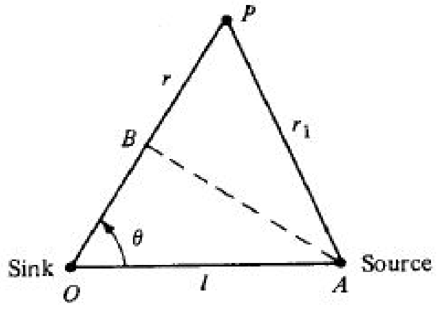
$$\phi(x, y, z) = -\frac{\sigma}{4\pi\sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}} \tag{4.14}$$

where $\mathbf{r}_0 = (x_0, y_0, z_0)$ is the position vector of the source/sink location, and $\mathbf{r} = (x, y, z)$ is the position vector. If $\sigma > 0$, fluid comes out from the source and the opposite is true for a sink, if $\sigma < 0$. For both cases, $|\sigma|$ is equal to the volumetric rate. Introducing the velocity potential into Equation (4.9), the velocity field is readily obtained as

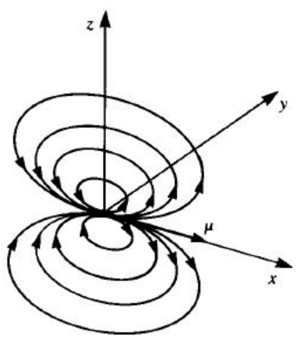
$$\mathbf{V}(x, y, z) = \frac{\sigma}{4\pi[(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2]^{\frac{3}{2}}} \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} \tag{4.15}$$

The velocity field is characterized by radial streamlines. It should be noticed that a singularity exists at $\mathbf{r}_0$ since $\lim_{\mathbf{r}\to\mathbf{r}_0} = \infty$.

Doublet



(a) Source and sink creating a doublet [58]



(b) Doublet oriented in the x direction [60]

Figure 4.2: The doublet element

The doublet may be obtained when a source and a sink are brought together and its intensity σ tends to infinity such that the product $\mu = l\sigma$ is constant. The distance between the source and the sink is l.

Based on that, it may be shown that the velocity potential is, in spherical coordinates

$$\phi(r,\theta,\psi) = \frac{\mu}{4\pi}\frac{\partial}{\partial n}\left(\frac{1}{|\mathbf{r}-\mathbf{r}_0|}\right) \tag{4.16}$$

where $\frac{\partial}{\partial n}$ is the derivative in the doublet direction, (direction of $\overrightarrow{OA}$, in Figure 4.2(a)), $\mathbf{r}_0 = (x_0, y_0, z_0)$ is the position vector of the doublet location, and $\mathbf{r} = (x, y, z)$ is the position vector. If $\overrightarrow{OA}$ is chosen to be along the x axis, the streamlines in plane $y = 0$ may be represented as in Figure 4.2(b), and Equation (4.16) simplifies to, now in Cartesian coordinates,

$$\phi(x,y,z) = -\frac{\mu}{4\pi}(x-x_0)\left[(x-x_0)^2+(y-y_0)^2+(z-z_0)^2\right]^{-\frac{3}{2}} \tag{4.17}$$

and consequently, the velocity field is

$$\mathbf{V} = -\frac{\mu}{4\pi}\left[(x-x_0)^2+(y-y_0)^2+(z-z_0)^2\right]^{-\frac{5}{2}}\begin{bmatrix}(y-y_0)^2+(z-z_0)^2-2(x-x_0)^2\\(x-x_0)(y-y_0)\\(x-x_0)(z-z_0)\end{bmatrix} \tag{4.18}$$

4.3 Panel Method

The panel method is a numerical technique to solve geometric complex aerodynamic problems. According to Hess [61], the method is based on finding the intensity of elementary solutions placed along the body's surface which satisfies the boundary conditions. From a computational point-of-view, this method is much less expensive than other CFD techniques since only discrete control points on the body's surface are required, unlike finite-differences or finite-volume methods that requires the entire flow field to be discretized [60]. For this reason, panel method codes are still used today in the preliminary stages of design.

Mathematical Background

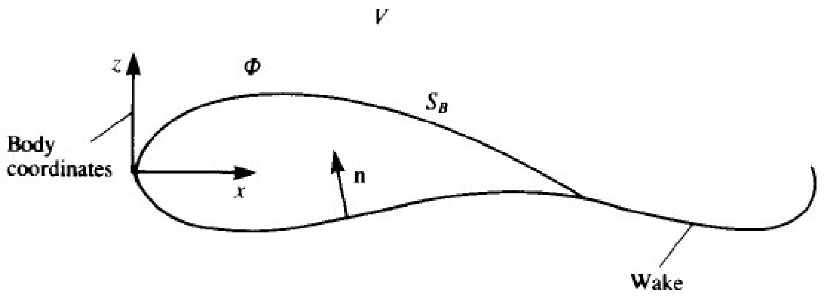


Figure 4.3: Potential flow over a closed body [60]

Consider the body depicted in Figure 4.3, surrounded by a fluid region of interest V, enclosed by a surface S_B and the normal vector $\mathbf{n}$, pointing inside of S_B. Based on the third Green's identity, a solution for the velocity potential for a generic point P, belonging to the interest region, is given by

$$\phi(P) = -\frac{1}{4\pi}\int_{S_B}\left[\sigma\left(\frac{1}{r}\right) - \mu\mathbf{n}.\nabla\left(\frac{1}{r}\right)\right]dS + \phi_\infty \tag{4.19}$$

where σ and μ are source/sink and doublet intensities, respectively, distributed along the surface S_B. They are the problem's unknowns. The free-stream potential is ϕ_∞ and the distance between the point P and a point on the surface S_B is r. Equation (4.19) gives a relation between the velocity potential ϕ at point P and the values of the unknowns at the boundaries, but gives no information about how to distribute those singularities. Since the source and doublets elements are used to simulate thickness and to simulate lifting bodies, respectively, Equation (4.19) may be expressed as

$$\phi(P) = \frac{1}{4\pi}\int_{Body+Wake}\mu\mathbf{n}.\nabla\left(\frac{1}{r}\right)dS - \frac{1}{4\pi}\int_{Body}\sigma\left(\frac{1}{r}\right)dS + \phi_\infty \tag{4.20}$$

where the distribution of singularities was rearranged.

Boundary Conditions

Towards the goal of obtaining a unique solution, the next logical step is the specification of boundary conditions. The impermeability condition may be set directly, through Equation (4.11a). Combining the impermeability condition with Equation (4.20) results in

$$\left[\frac{1}{4\pi}\int_{Body+Wake}\mu\nabla\left[\frac{\partial}{\partial\mathbf{n}}\left(\frac{1}{r}\right)\right]dS - \frac{1}{4\pi}\int_{Body}\sigma\nabla\left(\frac{1}{r}\right)dS + \nabla\phi_\infty\right].\mathbf{n} = 0 \tag{4.21}$$

which is called the *Neumann Problem*.

Consider now the point P inside the enclosed region by S_B. Since the interest region is now bounded, it is possible to prove that the impermeability condition forces the interior velocity potential to be constant within the region. Since the value of ϕ, given by Equation (4.20), is arbitrary, an equivalent and simpler expression for the impermeability condition may be obtained as

$$\frac{1}{4\pi}\int_{Body+Wake}\mu\mathbf{n}.\nabla\left(\frac{1}{r}\right)dS - \frac{1}{4\pi}\int_{Body}\sigma\left(\frac{1}{r}\right)dS = 0 \tag{4.22}$$

if $\phi = \phi_\infty$. The latter approach is called the *Dirichlet Problem*, since the impermeability boundary condition is satisfied indirectly, and it was chosen to be implemented by Cardeira [17]. The boundary condition given by Equation (4.11b) is automatically satisfied since the doublet and source influences vanish far away from the body.

Wake Model and Kutta Condition

Until now, neither Equation (4.21) or Equation (4.22) produce a unique solution since the amount of circulation is still undefined. To fix the problem, both the doublet strength and the shape of the wake must be defined. The two dimensional Kutta condition applied to the trailing edge allows to express the wake's doublet strength μ_W as a function of the trailing edge doublet strengths. Thus, this condition is written as

$$\mu_W = \mu_U - \mu_L \tag{4.23}$$

where μ_U and μ_L are the doublet strengths on the upper and lower surface, at the trailing edge (TE), respectively. The wake shape is such that no lift is produced by it. According to Kutta-Joukowski theorem, the condition for the wake geometry is

$$\mathbf{V} \times \boldsymbol{\xi} = 0 \tag{4.24}$$

where $\mathbf{V}$ and $\boldsymbol{\xi}$ are the local velocity and vorticity vectors on the wake, respectively.

Discretization

Since Equation (4.22) must hold for every point P on the body's surface, it is necessary to discretize the geometry into small regions, called *panels* to obtain an approximate solution. The discretization process corresponds to specify the boundary condition in a finite number of control points, called *collocation points*. For a given control point P, Equation (4.22) is approximated by

$$\sum_{k=1}^{N} \frac{1}{4\pi} \int_{BP_k} \mu \mathbf{n}.\nabla \left(\frac{1}{r}\right) dS + \sum_{l=1}^{N_w} \frac{1}{4\pi} \int_{WP_l} \mu \mathbf{n}.\nabla \left(\frac{1}{r}\right) dS - \sum_{k=1}^{N} \frac{1}{4\pi} \int_{BP_k} \sigma \left(\frac{1}{r}\right) dS = 0 \tag{4.25}$$

where the integration regions BP and WP stand for *Body Panel* and *Wake Panel*, respectively. Physically, Equation (4.25) represents the influence of all the collocation points in point P. Assuming μ and σ constant in each panel, Equation (4.25) simplifies to the linear equation

$$\sum_{k=1}^{N} C_k \mu_k + \sum_{l=1}^{N_w} C_l \mu_l = -\sum_{k=1}^{N} B_k \sigma_k \tag{4.26}$$

where N is the number of body panels, N_w is the number of wake panels and, C_k and B_k are the *influence coefficients*, defined as

$$\begin{aligned} C_k &= \frac{1}{4\pi} \int_{BP_k} \mathbf{n}.\nabla \left(\frac{1}{r}\right) dS \\ B_k &= -\frac{1}{4\pi} \int_{BP_k} \left(\frac{1}{r}\right) dS \end{aligned} \tag{4.27}$$

which depend only on geometrical quantities, as it will be shown later. The coefficient C_l, in Equation (4.26), is trivially obtained replacing the index in Equation (4.27). Since $\sigma = \mathbf{n}.\nabla\phi_\infty$, the right hand side of Equation (4.26) can be easily known. Through the Kutta condition, Equation (4.23), the wake doublets can be related with the unknown surface doublets. Introducing a new coefficient such that

$$\begin{aligned} A_k &= C_k \quad \text{if panel is not at TE} \\ A_k &= C_k \pm C_l \quad \text{if panel is at TE} \end{aligned} \tag{4.28}$$

Equation (4.26) can then be written as

$$\sum_{k=1}^{N} A_k \mu_k = -\sum_{k=1}^{N} B_k \sigma_k \tag{4.29}$$

Again, Equation (4.29) holds for a collocation point P. To obtain the solution μ_k, it must be written for every collocation point on the body. The result is a linear system of equations that may be represented as

$$A_\mu \boldsymbol{\mu} = \mathbf{b} \tag{4.30}$$

where A_μ is the matrix of aerodynamic influence coefficients and $\mathbf{b}$ is the vector of source influences.

4.4 Aerodynamic Framework Description

This section presents the aerodynamic framework since some modifications were performed to improve and revise the already developed tool. First, a general description will be performed to present how the already existing aerodynamic tool was divided into small modules, each of them with specific tasks. Then, each module will be explained in detail.

Figure 4.4 shows the mainstream data flow in the aerodynamic framework. The trapeziums, in red and green, are the design variables and the functions of interest, respectively. The first are represented by $\mathbf{x_{DV}}$ and the latter by f. The rectangles with rounded corners, in light blue, are routines (functions). The first component is called *Wing Parametrization*. This module translates the design variables into a discrete set of points which represent the wing's geometry. Next, the generated set of points are processed in a new module called *Panels Definition*. This routine is responsible for defining the panels' geometry as a function of those points. The panels are defined by the location of their corners. Knowing their locations, other relevant quantities can be obtained such as the panels' area, the location of the collocation points and local frames of reference where the local velocities will be evaluated. Next, a routine called *Change of Basis* is used to express the panels' corner points in their own frame of reference. After the panels have been defined, the module called *Aero Solver* runs an influence routine to construct the linear system in Equation (4.30) and then solves it to the unknown doublets intensities $\boldsymbol{\mu}$. Finally, a module called *Post-Process* calculates the aerodynamic coefficients. In the next sections, detailed information will be provided about the framework's constituting modules.

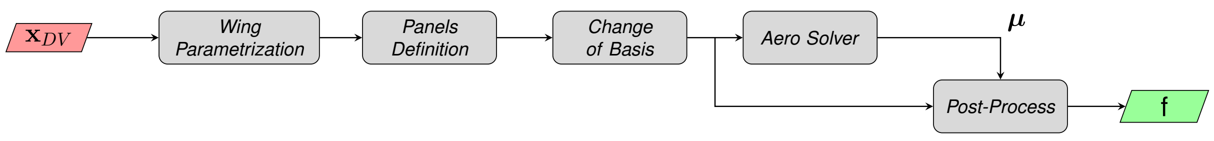


Figure 4.4: Flowchart illustrating the aerodynamic framework

4.4.1 Wing Parametrization

The first step towards obtaining a numerical solution is to translate the design variables to a discrete set of points representing the wing's geometry. To accomplish that, a MATLAB® function called `wing_geometry.m` was developed. A detailed list of the inputs and outputs are presented in Table 4.1. The wing is parameterized according to the exterior shape and the wing section airfoil. An improvement related with the previous developed framework is the implementation of an airfoil parametrization providing a set of design parameters to control the airfoil shape in each wing's cross section.

Outputs	Inputs		
Planform Area (S)	Sweep Angle (Λ)	Tip Twist Angle (δ_t)	Taper Ratio (λ)
Mean Aerodynamic Chord (MAC)	Dihedral Angle (Γ)	Wing Span (b)	Airfoil Control Points ($[A, ..., L]_j$)
Wing Points ($\mathbf{WP}$)	Root Twist Angle (δ_r)	Root Chord (c_r)	Angle of Attack (α)

Table 4.1: List of inputs and outputs of function `wing_geometry.m`

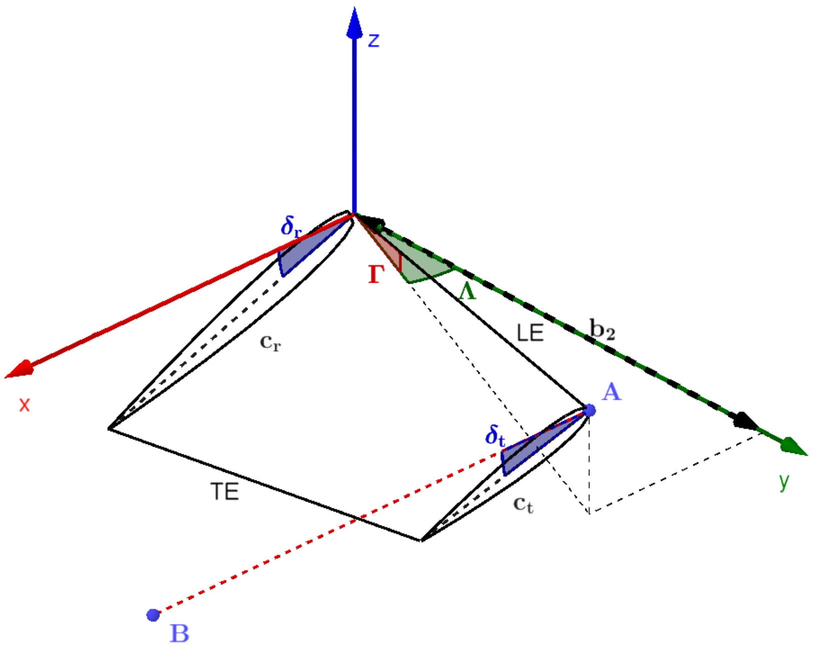


Figure 4.5: Geometrical description of the aircraft half wing

Exterior Wing Shape

The first design step is to define the leading edge. Observing Figure 4.5 one can conclude that the leading edge (LE) is fully defined by half of the wing span length, $\mathbf{b_2} = \frac{b}{2}$, and by the sweep and dihedral angles, Λ and Γ, respectively. The next step corresponds to define the root and the tip of the wing. These extremities are characterized by an airfoil shape, a root and tip chord lengths, c_r and c_t, and their respective twist angles, δ_r and δ_t. Alternatively to specify c_t, the function `wing_geometry.m` accepts the taper ratio λ, since $\lambda = \frac{c_t}{c_r}$. According to the figure, the root's twist angle is the angle measured between the root's chord line, represented by a dark dashed segment, and the x axis. Similarly, the tip's twist angle is the angle measured between the segment AB which is parallel to the x axis, and the the tip's chord line, also depicted by a dark dashed segment. The local chord lengths and twist angles in the spanwise direction are automatically defined since they are assumed to vary linearly between their respective root and tip values. The wing is then discretized in the spanwise direction according to the user specifications. Each discretized cross-section receives a set of points representing the airfoil shape, which may change from section to section, if desired. Finally, a rigid rectangular wake is added to the model, oriented according to the angle of attack α.

After the wing has been fully defined, both the wing's planform area, S, and mean aerodynamic chord, MAC, are required to the non-dimensionalisation of the aerodynamic loads. For a trapezoidal wing, these quantities are calculated as

$$S = \left(\frac{1+\lambda}{2}\right) b\, c_r \tag{4.31a}$$

$$\mathrm{MAC} = \frac{2}{3}\left(\frac{\lambda^2+\lambda+1}{\lambda+1}\right) c_r \tag{4.31b}$$

Airfoil Definition

As addressed in the previous section, it is necessary to generate the airfoil shape to be used in a wing section. The implementation procedure follows the work of Venkataraman [62], which corresponds to solve a constrained least-squares problem. The main advantages of using this parametrization is the easiness of setting construction constraints such as continuity, tangency and, optimization constraints such as maximum thickness. The airfoil is parametrized using four cubic bezier curves such that two of them represents the upper surface and the other two represent the bottom one.

A cubic bezier curve is a parametric curve which is generated by a set of four vertexes forming a polygon distributed in space. The curve is parametrized as

$$\mathbf{P}(t) = \begin{bmatrix} p(t) \\ q(t) \end{bmatrix} = \sum_{i=0}^{3} \begin{bmatrix} x \\ y \end{bmatrix}_i J_i^3(t) \tag{4.32}$$

where $[x_i \; y_i]^T$ are the two dimensional coordinates of the vertex i and will be used as airfoil design variables. The parameter t is defined as $t \in [0,1]$ and J_i^3 is the third order i^{th} Bernstein's basis function, defined as

$$J_i^3(t) = \binom{3}{i} t^i \,(1-t)^{3-i} \tag{4.33}$$

Consider now a subset of the airfoil data set as an ordinated set of points $U = \{(u_k, v_k)\}_{k=1}^n$. The ultimate goal is to solve a least-squares problem casted as

$$\begin{aligned} &\text{minimize} && \sum_{k=1}^{n} \left\{ \left(u_k - \sum_{i=0}^{3} J_i^3(t_k) x_i \right)^2 + \left(v_k - \sum_{i=0}^{3} J_i^3(t_k) y_i \right)^2 \right\}, \\ &\text{w.r.t.} && \{x_i, y_i\} \end{aligned} \tag{4.34}$$

where $t = \{t_0, t_1, ..., t_n\}$ was chosen to be a prescribed vector to simplify the problem. For illustration purposes, a symmetrical NACA 0010 airfoil was chosen to be parametrized. A MATLAB® function called `fit_baseline.m` was developed to read the airfoil coordinates from a text file and return the set of bezier control points. The airfoil data was first divided in four groups: the first corresponds to the data from the trailing edge until the point of maximum thickness, starting from the lower surface; the second corresponds to the set formed from the previous point until the leading edge; the third allocates all the points from the leading edge to the point of maximum thickness, in the upper surface; and the fourth allocates the remaining points. For each set, the least-squares problem is solved to find the control points with the aid of `fmincon`, a MATLAB® constrained nonlinear programming solver. The result is presented in Figure 4.6.

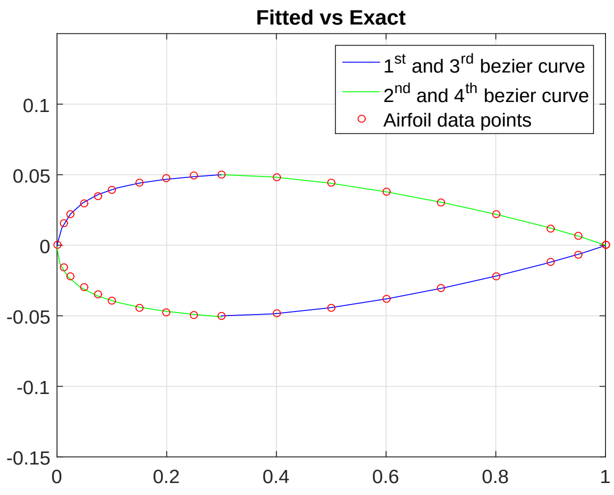


Figure 4.6: Airfoil NACA 0010 fitted by bezier curves

Figure 4.7 illustrates the imposed constraints during the airfoil NACA 0010 parametrization. The control points in red, A and G were not allowed to move, corresponding to the trailing and leading edge, respectively. Similarly, the points D and J were also fixed, corresponding to the points of maximum thickness. The points in yellow B and L, were allowed to move in any direction. The points in green, C, E, I and K are constrained to move relatively to D and J in the horizontal direction, as suggested by the horizontal red lines. Similarly, the points in blue, F and H were constrained to move in the vertical direction relatively to the point G, as suggested by the vertical red lines.

The applied constraints are also necessary to parametrize acceptable airfoil shapes during the optimization process. However, to obtain shapes with camber and different thicknesses, the points D and J must be free to move in any direction. Moreover, control about these features may be achieved by bounding the control points appropriately.

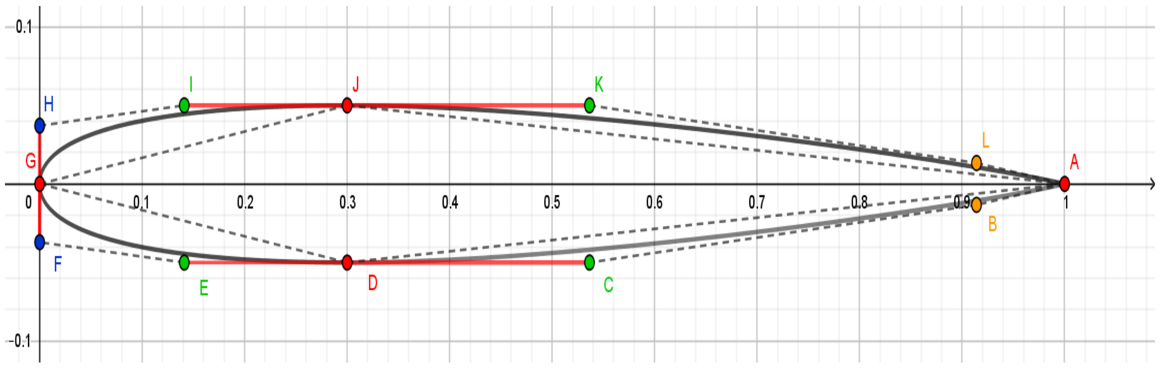


Figure 4.7: Airfoil NACA 0010 defined by bezier polygons and respective control points constraints

4.4.2 Panels Definition

A panel is fully characterized by a set of four points which represents its vertexes. In addition, it is necessary to determine a collocation point where the boundary conditions will be enforced, its area and a local frame of reference where the velocity field will be measured. Since three points define a plane, there is no guarantee that four nearest points belonging to the wing's geometry are coplanar. Therefore, it is necessary to define an average plane where the element lies on, bounded by the edges formed by its vertexes. To achieve this task, a MATLAB® function called `panels.m` was developed. A detailed list of the inputs and outputs are presented in Table 4.2.

Outputs		Inputs
Panels Corner Points $(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4)$	Panels Areas (DS)	Input Points $(\mathbf{WP}_1, \mathbf{WP}_2, \mathbf{WP}_3, \mathbf{WP}_4)$
Collocation Points $(\mathbf{CP})$	Panels Basis Vectors $(\mathbf{l}, \mathbf{m}, \mathbf{n})$	

Table 4.2: List of inputs and outputs of function `panels.m`

Typically, when the wing has no twist, the taper ratio is small and the airfoil shape is the same for the all wing, it is possible to assume that four wing points closest to each other are almost coplanar. Therefore, there is no need to define an average plane to the panel. In the tool developed by Cardeira [17], the latter approach was assumed and thus, modified in this work. The followed methodology is based on the work of Hess [61].

Before proceeding further some nomenclature needs to be introduced. The generated set of wing points are represented by the vector $\mathbf{WP}$, as already suggested in Table 4.1. It is defined as

$$\mathbf{WP} = \begin{bmatrix} \mathbf{WP_1}^T & \mathbf{WP_2}^T & \mathbf{WP_3}^T & \mathbf{WP_4}^T \end{bmatrix}^T \tag{4.35}$$

with e.g. $\mathbf{WP_1}$ given by

$$\mathbf{WP_1} = \begin{bmatrix} \cup_{ij} WP_{1_{ij1}} \mid & \cup_{ij} WP_{1_{ij2}} \mid & \cup_{ij} WP_{1_{ij3}} \end{bmatrix}^T, \forall i \in \{1, ..., M\}, \forall j \in \{1, ..., N\} \tag{4.36}$$

where $\cup_{ij}$ means concatenation, a compact way to indicate that the indexes are unrolled, and M and N are the chordwise and semi spanwise number of panels. The same variable $\mathbf{WP_1}$, with indexes, is defined as

$$\mathbf{WP_1}_{ij} = \begin{bmatrix} WP_{1_{ij1}} & WP_{1_{ij2}} & WP_{1_{ij3}} \end{bmatrix}^T \tag{4.37}$$

where $WP_{1_{ij1}}$, $WP_{1_{ij2}}$ and $WP_{1_{ij3}}$ are the three dimensional coordinates of the point $\mathbf{WP_1}_{ij}$, belonging to the mesh location (i, j), as stated by Equation (4.37). Points $\mathbf{WP_1}_{ij}$ to $\mathbf{WP_4}_{ij}$ are the inputs to form the panel (i, j) as depicted in Figure 4.8. The expressions for $\mathbf{WP_2}_{ij}$ until $\mathbf{WP_4}_{ij}$ are obtained by changing the variable name in Equation (4.37). When no indexes are present in a panel related variable, one is assuming that the quantity in question is represented for all the computational mesh, with all the indexes (i, j) unrolled, as stated by Equation (4.36). This nomenclature holds for both scalar quantities such as the panel area, or vectors (represented in bolt) such as position vectors, and it will be used from here forward.

Consider the input points from the wing's discretization $\mathbf{WP_1}_{ij}$, $\mathbf{WP_2}_{ij}$, $\mathbf{WP_3}_{ij}$, $\mathbf{WP_4}_{ij}$, and the panel (i, j), depicted in Figure 4.8. Two parallel edges are formed by connecting the points $\mathbf{WP_1}_{ij}$ with

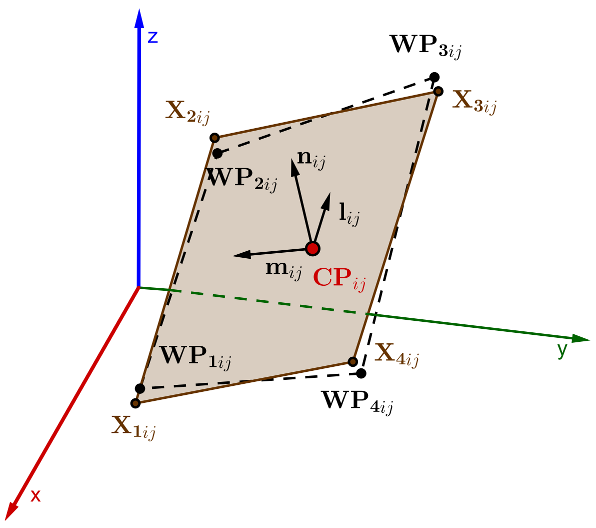


Figure 4.8: Panel construction through a set of four non-coplanar nearest points.

$\mathbf{WP_2}_{ij}$ and $\mathbf{WP_3}_{ij}$ with $\mathbf{WP_4}_{ij}$. Thus, four auxiliary vectors may be defined as

$$\mathbf{P_f}_{ij} = \mathbf{WP_2}_{ij} - \mathbf{WP_1}_{ij} \tag{4.38a}$$

$$\mathbf{P_s}_{ij} = \mathbf{WP_3}_{ij} - \mathbf{WP_4}_{ij} \tag{4.38b}$$

$$\mathbf{X_f}_{ij} = \frac{1}{2}\left(\mathbf{WP_1}_{ij} + \mathbf{WP_2}_{ij}\right) \tag{4.39a}$$

$$\mathbf{X_s}_{ij} = \frac{1}{2}\left(\mathbf{WP_3}_{ij} + \mathbf{WP_4}_{ij}\right) \tag{4.39b}$$

Using the definitions from Equation (4.38a) and Equation (4.38b), the first basis vector, which lies on the panel's plane, can be defined as

$$\mathbf{l}_{ij} = \frac{\left(\mathbf{P_f}_{ij} + \mathbf{P_s}_{ij}\right)}{\|\mathbf{P_f}_{ij} + \mathbf{P_s}_{ij}\|} \tag{4.40}$$

In this manner, the panel's corner points $\mathbf{X_1}_{ij}$, $\mathbf{X_2}_{ij}$, $\mathbf{X_3}_{ij}$ and $\mathbf{X_4}_{ij}$ are automatically defined as a function the vectors $\mathbf{X_f}_{ij}$, $\mathbf{X_s}_{ij}$, $\mathbf{P_f}_{ij}$, $\mathbf{P_s}_{ij}$ and $\mathbf{l}_{ij}$ as

$$\mathbf{X_{1,2}}_{ij} = \mathbf{X_f}_{ij} \mp \frac{1}{2}\left(\|\mathbf{P_f}_{ij}\|.\mathbf{l}_{ij}\right) \tag{4.41a}$$

$$\mathbf{X_{3,4}}_{ij} = \mathbf{X_s}_{ij} \pm \frac{1}{2}\left(\|\mathbf{P_s}_{ij}\|.\mathbf{l}_{ij}\right) \tag{4.41b}$$

where the minus sign corresponds to the assignment of $\mathbf{X_1}_{ij}$ and $\mathbf{X_4}_{ij}$, and the plus sign to $\mathbf{X_2}_{ij}$ and $\mathbf{X_3}_{ij}$, in Equations (4.41a) and (4.41b).

Next, the unit normal of the panel is defined as

$$\mathbf{n}_{ij} = \frac{\mathbf{N}_{ij}}{\|\mathbf{N}_{ij}\|} \tag{4.42}$$

where $\mathbf{N}_{ij}$ is a normal vector which is a function of the corner points

$$\mathbf{N}_{ij} = \left(\mathbf{X_3}_{ij} - \mathbf{X_1}_{ij}\right) \times \left(\mathbf{X_4}_{ij} - \mathbf{X_2}_{ij}\right) \tag{4.43}$$

As already said, a local frame of reference is required to measure the velocity field in each panel. Two perpendicular unit vectors are already defined, thus, the last one may be determined based on the last two as

$$\mathbf{m}_{ij} = \mathbf{n}_{ij} \times \mathbf{l}_{ij} \tag{4.44}$$

Finally, it is necessary to calculate the panel's area and place the collocation point in the center of the panel. Both can easily be calculated as a function of the corner points as

$$\mathbf{CP}_{ij} = \frac{1}{4}\left(\mathbf{X_1}_{ij} + \mathbf{X_2}_{ij} + \mathbf{X_3}_{ij} + \mathbf{X_4}_{ij}\right) \tag{4.45}$$

and

$$DS_{ij} = \frac{1}{2}\left[\|\mathbf{X_B}_{ij} \times \mathbf{X_A}_{ij}\| + \|\mathbf{X_C}_{ij} \times \mathbf{X_B}_{ij}\|\right] \tag{4.46}$$

where $\mathbf{X_A}_{ij} = \mathbf{X_2}_{ij} - \mathbf{X_1}_{ij}$, $\mathbf{X_B}_{ij} = \mathbf{X_3}_{ij} - \mathbf{X_1}_{ij}$, $\mathbf{X_C}_{ij} = \mathbf{X_4}_{ij} - \mathbf{X_1}_{ij}$. Additionally, $\mathbf{CP}_{ij}$ is the collocation point location and DS_{ij} is the panel's area.

4.4.3 Change of Basis

Lately it will be required to write some points in the panel's frame of reference. Consider a point $\mathbf{P}$ written in the global frame of reference with coordinates $\mathbf{P} = [P_1, P_2, P_3]^T$. The same point may be written on the (i, j) panel's frame of reference whose origin is $\mathbf{CP}_{ij}$ and the basis vectors are the set $\{\mathbf{l}, \mathbf{m}, \mathbf{n}\}_{ij}$ as

$$\begin{bmatrix} P_1^{'} \\ P_2^{'} \\ P_3^{'} \end{bmatrix} = \begin{bmatrix} l_{ij1} & l_{ij2} & l_{ij3} \\ m_{ij1} & m_{ij2} & m_{ij3} \\ n_{ij1} & n_{ij2} & n_{ij3} \end{bmatrix} \begin{bmatrix} P_1 - CP_{ij1} \\ P_2 - CP_{ij2} \\ P_3 - CP_{ij3} \end{bmatrix} \tag{4.47}$$

Each row of the matrix is composed by the components of each basis vectors. To implement Equation (4.47), a generic function called `convert.m` was developed. This function accepts any basis vectors, origin and point P. As it will be clear next, its also useful to express the panel's corner points in its own frame of reference. The implementation routine is called `write_local_corners.m` and a list of outputs and inputs of that function is presented in Table 4.3.

Outputs	Inputs		
Local Panels Points $(\mathbf{X}_1^{'}, \mathbf{X}_2^{'}, \mathbf{X}_3^{'}, \mathbf{X}_4^{'})$	Collocation Points $(\mathbf{CP})$	Panels Basis Vectors $(\mathbf{l}, \mathbf{m}, \mathbf{n})$	Panels Corner Points $(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4)$

Table 4.3: List of inputs and outputs of function `write_local_corners.m`

4.4.4 Aerodynamic Solver

At this point, all the necessary requirements to obtain a solution to Equation (4.30) are met. The aerodynamic solver consists on a routine to calculate the coefficients of influence, assemble and solve the resulting system. The implementation was carried in a MATLAB® function called `aero_solver.m`. A detailed list of the inputs and outputs are presented in Table 4.4.

Outputs	Inputs		
Doublet Intensities (μ)	Local Panels Points ($\mathbf{X}_1', \mathbf{X}_2', \mathbf{X}_3', \mathbf{X}_4'$)	Collocation Points ($\mathbf{CP}$)	Angle of Attack (α)
Residuals ($\mathbf{R}$)	Panel Basis Vectors ($\mathbf{l}, \mathbf{m}, \mathbf{n}$)	Free-stream Velocity (V_∞)	

Table 4.4: List of inputs and outputs of function `aero_solver.m`

Consider the dual $M \times N$ mesh as displayed in Figure 4.9. The latter is characterized by three main features. The vertical red line divides the effectively modeled grid from their image, thus it may be thought as a mirror. The reason behind it is because the wing was assumed symmetric with respect to the Oxz plane of Figure 4.5 and therefore only half was actually modeled. The grid's image plays a role in the implementation of the method of images. The horizontal red line divides the wake from the wing body panels where the boundary conditions will be enforced.

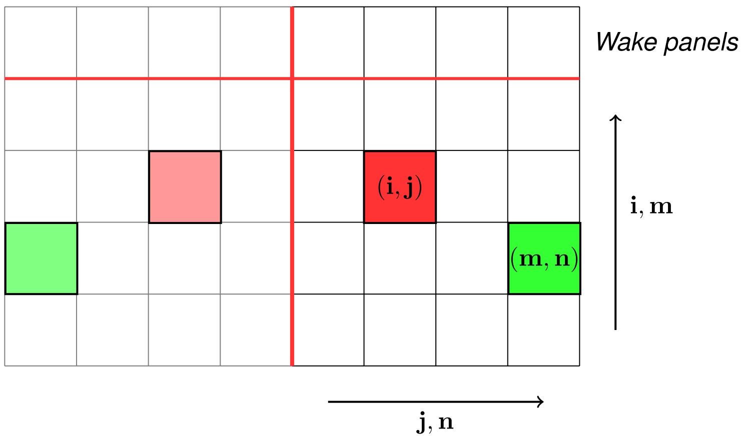


Figure 4.9: $M \times N$ computational mesh. Influence panel (m, n), influenced panel (i, j) and respective images

The influence routine consists of setting Equation (4.29) for every wing body collocation point. Rewriting the equation in residual form and suited for the computational mesh, yields

$$R_{ij} = \sum_{n=1}^{N} \left[\sum_{m=1}^{M-1} \left(\mathcal{C}_{ijmn}\mu_{mn} + \mathcal{B}_{ijmn}\sigma_{mn} \right) + \mathcal{C}_{ijMn} \left(\mu_{(M-1)n} - \mu_{1n} \right) \right] = 0 \tag{4.48}$$

where R_{ij} is the residual associated with the body panel (i, j), depicted in red, in Figure 4.9. The quantities $\mathcal{C}_{ijmn}$ and $\mathcal{B}_{ijmn}$ are the total dipole and source influences from panel (m, n) on panel (i, j), respectively. These quantities also take in account the influence of the image's panel (m, n) on the panel (i, j). It should also be noticed that there is no need to go through the process of defining a new image's panel, as described earlier. The influence of the latter on panel (i, j) is exactly equivalent to the influence of panel (m, n) on the image of panel (i, j). The great advantage is because panel (m, n) is already defined and only the collocation point's position of image panel (i, j) is additionally required to calculate the contribution to $\mathcal{C}_{ijmn}$ and $\mathcal{B}_{ijmn}$. In this manner, the total influence coefficient $\mathcal{B}_{ijmn}$ is composed by the sum of $\mathcal{B}^1_{ijmn}$ with $\mathcal{B}^2_{ijmn}$, where the first term is actually the influence of panel (m, n) on panel (i, j) and the second, the influence of panel (m, n) on panel (i, j) image. The same principle and nomenclature applies to the coefficient $\mathcal{C}_{ijmn}$, thus $\mathcal{C}_{ijmn} = \mathcal{C}^1_{ijmn} + \mathcal{C}^2_{ijmn}$. In this manner, it is clear that the only unknown in Equation (4.48) is the doublet vector, since the source intensity is known and equal to

$$\sigma_{mn} = \mathbf{n}_{mn}.\mathbf{V}_\infty \tag{4.49}$$

The influence coefficients present in Equation (4.48) are integral terms which can be proven to be a function of the influence panel's corner points expressed in their own local frame of reference, and the influenced panel collocation point's location expressed in that frame of reference. According to Katz and Plotkin [60], those influence coefficients are generically defined as

$$\mathcal{B} = -\frac{1}{4\pi}\left\{[s_{12} + s_{23} + s_{34} + s_{41}] + |z|\,[q_{12} + q_{23} + q_{34} + q_{41}]\right\} \tag{4.50a}$$

$$\mathcal{C} = \frac{1}{4\pi}[q_{12} + q_{23} + q_{34} + q_{41}] \tag{4.50b}$$

where the contributions s_{ij} and q_{ij} are expressed as

$$s_{ij} = \frac{(x - x_i)(y_j - y_i) - (y - y_i)(x_j - x_i)}{d_{ij}} \ln\left(\frac{r_i + r_j + d_{ij}}{r_i + r_j - d_{ij}}\right) \tag{4.51a}$$

$$q_{ij} = \arctan\left(\frac{m_{ij}e_i - h_i}{zr_i}\right) - \arctan\left(\frac{m_{ij}e_j - h_j}{zr_j}\right) \tag{4.51b}$$

and additionally

$$m_{ij} = \frac{y_j - y_i}{x_j - x_i} \tag{4.52a}$$

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{4.52b}$$

$$r_k = \sqrt{(x - x_k)^2 + (y - y_k)^2 + z^2} \tag{4.52c}$$

$$e_k = (x - x_k)^2 + z^2 \tag{4.52d}$$

$$h_k = (x - x_k)(y - y_k) \tag{4.52e}$$

where (x, y, z) is the influenced panel collocation point's location and (x_k, y_k) the influence panel corner point's location, both expressed on the influence panel's frame of reference.

Writing Equation (4.48) for every collocation point (i, j) is equivalent to set an influence matrix and a right hand side, as expressed in Equation (4.30). The next step to obtain a solution is to solve the system. This is accomplished using a MATLAB® function called `linsolve`. The solver uses LU factorization with partial pivoting, according to the documentation available on the company's website [63].

4.4.5 Post-Processing

Another advantage of using the panel method is concerned with the easiness of computing the pressure coefficients directly from the previous obtained solution [60]. In this manner, the calculation of the aerodynamic coefficients is quite straightforward as explained next.

The total velocity vector on the panel (ij) is the algebraic sum of the free-stream plus the perturbation velocity vectors, $\mathbf{V}_\infty$ and $\mathbf{v}$, respectively. In the panel's frame of reference, the previous statement is written as

$$\mathbf{V}_{ij} = (V_{\infty l}, V_{\infty m}, V_{\infty n})_{ij} + (v_l, v_m, v_n)_{ij} \tag{4.53}$$

where the perturbation velocities are calculated for each panel as

$$v_{l_{ij}} = -\left.\frac{\partial \mu}{\partial l}\right|_{ij} \tag{4.54a}$$

$$v_{m_{ij}} = -\left.\frac{\partial \mu}{\partial m}\right|_{ij} \tag{4.54b}$$

These partial derivatives are estimated using second-order central finite-differences if the panel is not at the mesh's boundary and backward or forward first-order finite differences otherwise, depending which boundary is being considered. Since the velocity vector is tangent to the panel's surface, there is no need to compute the perturbation velocity in the normal direction because the normal components of the free-stream and perturbation velocity vectors cancels out.

Since the flow is potential and irrotational, the pressure coefficient at the panel (ij) is expressed as a function of the velocity vector as

$$C_{p_{ij}} = 1 - \frac{|\mathbf{V}_{ij}|^2}{|\mathbf{V}_\infty|^2} \tag{4.55}$$

Knowing the pressure coefficient along the wing's surface, it is possible to obtain the aerodynamic coefficients through numerical integration. Therefore, the lift, drag and moment coefficients are calculated as

$$C_L = -\frac{2}{S}\sum_{i}^{M-1}\sum_{j}^{N} C_{p_{ij}} DS_{ij}\left(\mathbf{n}_{ij}.\mathbf{e}_L\right) \tag{4.56a}$$

$$C_D = -\frac{2}{S}\sum_{i}^{M-1}\sum_{j}^{N} C_{p_{ij}} DS_{ij}\left(\mathbf{n}_{ij}.\mathbf{e}_D\right) \tag{4.56b}$$

$$\mathbf{C_M} = -\frac{2}{S.\,l_0}\sum_{i}^{M-1}\sum_{j}^{N} C_{p_{ij}} DS_{ij}\left(\mathbf{CP}_{ij} \times \mathbf{n}_{ij}\right) \tag{4.56c}$$

where S is the wing planform area, DS_{ij} and $\mathbf{n}_{ij}$ are the panel's (ij) area and normal vector and l_0 is an appropriated reference length. From the moment coefficient vector $\mathbf{C_M}$, one is interested in the first and second components, corresponding to the rolling and pitching moment coefficients, C_{M_x} and C_{M_y}, respectively. For those cases, the reference dimension l_0 is equal to the wing span b and the mean aerodynamic chord, MAC, respectively. The vectors $\mathbf{e}_L$ and $\mathbf{e}_D$ are the unit vectors with the same direction as the lift and drag forces. They are a function of the angle of attack as

$$\mathbf{e}_L = [\cos\alpha, 0, \sin\alpha]^T \tag{4.57a}$$

$$\mathbf{e}_D = [-\sin\alpha, 0, \cos\alpha]^T \tag{4.57b}$$

Computationally, the post process of the solution was carried in a MATLAB® function called `post_process.m`. A detailed list of inputs and outputs is presented in Table 4.5.

Outputs	**Inputs**	
Lift Coefficient (C_L)	Doublet Intensities (μ)	Panel Basis Vectors ($\mathbf{l}, \mathbf{m}, \mathbf{n}$)
Drag Coefficient (C_D)	Panels Corner Points ($\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4$)	Angle of Attack (α)
Rolling Moment Coefficient (C_{M_x})	Collocation Points ($\mathbf{CP}$)	Mean Aerodynamic Chord (MAC)
Pitching Moment Coefficient (C_{M_y})	Panels Areas (DS)	Planform Area (S)
Aerodynamic Efficiency ($\frac{C_L}{C_D}$)		Free-stream Velocity (V_∞)

Table 4.5: List of inputs and outputs of function `post_process.m`

4.5 Code Verification

A benchmark between the reformulated tool, the results obtained by Cardeira [17] and an airfoil/wing design tool, the XFLR5 [64], was also performed. The test case corresponds to an unswept and untwisted rectangular wing with an aspect ratio of 4 and a NACA 0010 airfoil. Since the three dimensional panel method code used in XFLR5 is not available, an additional verification is made using the lifting-line theory. According to Corke [65], the lift and drag coefficients for an unswept wing with an uncambered airfoil in inviscid potential flow are

$$\begin{aligned} C_L =& \frac{2\pi \mathit{Æ}\, \alpha}{2+\sqrt{4+\mathit{Æ}^2}} \\ C_D =& \frac{C_L^2}{\pi \mathit{Æ}\, e} \end{aligned} \tag{4.58}$$

where $\mathit{Æ}$ is the aspect ratio, α is the angle of attack and e is the efficiency factor which accounts with the taper ratio effect. For rough calculations, Corke [65] recommends e equal to 0.8. For a flight configuration of: $\alpha = 6^\circ$ and $V_\infty = 75\, m/s$, the resulting aerodynamic coefficients are presented in Table 4.6. The results presented were obtained using a mesh of 64×34 panels in the chordwise and

	C_L	C_D	C_{M_y}
Lifting-line theory [65]	0.40665	0.0165	-
XFLR5 [64]	0.39502	0.0122	-0.08975
Cardeira [17]	0.09572	0.0074	-
Rodrigues	0.39161	0.0128	-0.08910

Table 4.6: Aerodynamic coefficients benchmark for the test case with $\alpha = 6^\circ$ and $V_\infty = 75\, m/s$

spanwise directions, respectively. The largest deviation is observed for Cardeira's results because errors were found in the post-processing code part, where the aerodynamic coefficients were being calculated wrongly. Taking the values from the XFLR5 inviscid analysis as reference, the relative errors in lift, drag and pitching moment coefficients of the modified framework are 0.86%, 4.92% and 0.72%, respectively.

Chapter 5

Sensitivity Analysis Framework

This chapter presents the sensitivity analysis framework. Figure 5.1 presents a schematic representation of the new and already existent modules. Each module presented in Chapter 4 has its own sensitivity analysis block, represented by the gray dashed rectangles with rounded corners, labeled with the method used for sensitivity analysis, where the jacobian of the outputs relative to the inputs are evaluated. After those derivatives are determined, the chain-rule will be used to ultimately calculate the derivatives of the interest functions with respect to the design variables.

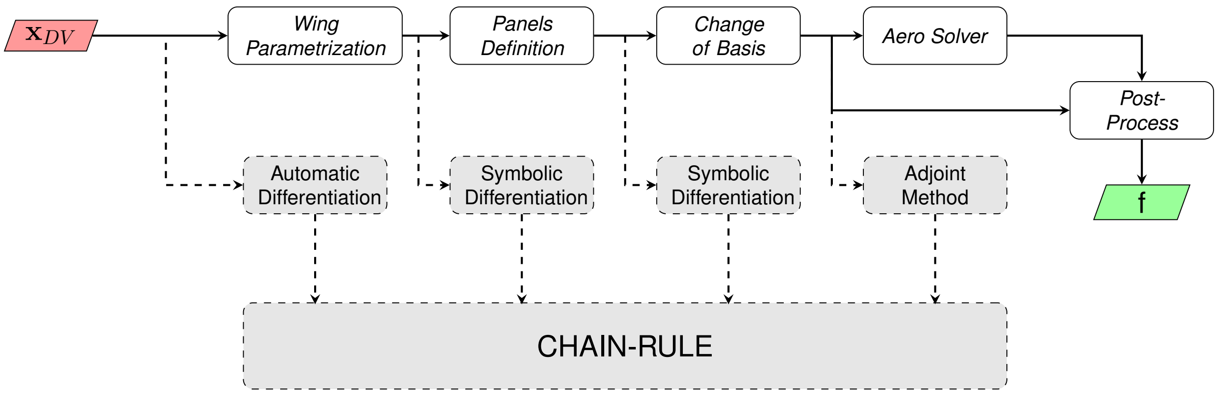


Figure 5.1: Flowchart illustrating the sensitivity analysis framework

When calculating the jacobians, one is concerned about two major requirements:

1. **Accuracy**: The derivatives have to be estimated with as much precision possible since exact gradient calculation allows to reduce the number of iterations during the optimization process, clearly affecting the convergence behavior of the algorithm;

2. **Computational cost**: The number of intermediate variables are about $\mathcal{O}(10^3)$, even for relatively course meshes, when the aerodynamic coefficients are still not converged. Thus, the intermediate jacobians have about $\mathcal{O}(10^6)$ elements, for those cases. Thus, an effort has to be made concerning with efficient algorithms and good programming practices.

The chapter will begin with a mathematical formulation including a series of variable definitions and the application of the chain-rule to the aerodynamic framework. Next, a detailed presentation will be made for the sensitivity analysis of each framework's module, justifying the method used, exhibiting the implementation procedures and the verification of the results with other sensitivity analysis methods available. At the end of the chapter, a comparison in accuracy and computational cost will be made between the new sensitivity analysis framework and the one obtained using finite-differences, assuming the aerodynamic framework as a unique function of the design variables.

5.1 Mathematical Formulation

5.1.1 Design Variables

The first point one must consider is to define the vector of design variables. Those are the inputs of the function `wing_geometry.m`, as described in Table 4.1. The corresponding design vector is characterized by a segment containing only the wing's planform representation, another segment containing the control points for each wing section and the angle of attack. Thus, the design vector is defined as

$$\mathbf{x_{DV}} = \begin{bmatrix}\alpha & \mathbf{x_{geo}}^T & \mathbf{x_{airfoil}}^T\end{bmatrix}^T \tag{5.1}$$

where each of the right hand side vectors are

$$\mathbf{x}_{geo} = \begin{bmatrix}\Lambda & \Gamma & \delta_r & \delta_t & b & c_r & \lambda\end{bmatrix}^T \tag{5.2a}$$

$$\mathbf{x_{airfoil}}^T = \bigcup_j \begin{bmatrix}A_x & ... & L_x & A_y & ... & L_y\end{bmatrix}_j \ , \forall j \in \{1, ..., N+1\} \tag{5.2b}$$

The points A through L are the airfoil control points' coordinates, as defined in section 4.4.1, j is the wing's section index and N is the number of the semi spanwise number of panels. Thus, the resulting design vector $\mathbf{x}_{DV}$, has the size of $8 + 24(N + 1)$.

5.1.2 Intermediate Variables

After defining the design variables, several intermediate others need to be presented since they will provide the necessary information to calculate the jacobians to be used in the chain-rule. The first auxiliary vector is called $\mathbf{PP}$, short for "Panels Points", which results from the concatenation of the four corners of each panel, as stated by

$$\mathbf{PP} = \begin{bmatrix}\mathbf{X_1}^T & \mathbf{X_2}^T & \mathbf{X_3}^T & \mathbf{X_4}^T\end{bmatrix}^T \tag{5.3}$$

One should remember the reader to the notation presented in the beginning of section 4.4.2 for the meaning of using variables with and without index notation when those are directly related with the panels. Those concepts apply vastly on this chapter. The second auxiliary vector results from the concatenation of the three panel's basis vectors. It is named $\mathbf{LV}$, short for "Local Vectors", and is defined as

$$\mathbf{LV} = \begin{bmatrix}\mathbf{l}^T & \mathbf{m}^T & \mathbf{n}^T\end{bmatrix}^T \tag{5.4}$$

According to Table 4.2, the vectors $\mathbf{PP}$, $\mathbf{CP}$, DS and $\mathbf{LV}$ depend explicitly on $\mathbf{WP}$ exclusively. The third auxiliary vector is obtained from the concatenation of the panel points expressed in its own frame of reference. Similarly to Equations (5.3) and (5.4), the variable $\mathbf{LPP}$, short for "Local Panel Points" is defined as

$$\mathbf{LPP} = \begin{bmatrix}\mathbf{X_1'}^T & \mathbf{X_2'}^T & \mathbf{X_3'}^T & \mathbf{X_4'}^T\end{bmatrix}^T \tag{5.5}$$

and according to Table 4.3, it depends explicitly on the variables $\mathbf{PP}$, $\mathbf{CP}$ and $\mathbf{LV}$.

5.1.3 Adjoint Method

Paying now close attention to the input variables in Table 4.4 and Table 4.5, one may define an intermediate information vector $\mathbf{x_2}$, as the concatenation of several auxiliary variables presented before

$$\mathbf{x_2} = \begin{bmatrix}\mathbf{x_1}^T & DS^T & \mathbf{LPP}^T & S & \mathrm{MAC} & \alpha & V_\infty\end{bmatrix}^T \tag{5.6}$$

where $\mathbf{x_1}$ is defined as

$$\mathbf{x_1} = \begin{bmatrix}\mathbf{PP}^T & \mathbf{CP}^T & \mathbf{LV}^T\end{bmatrix}^T \tag{5.7}$$

such that $\mathbf{R} = \mathbf{R}(\mathbf{x_2}, \boldsymbol{\mu})$ and $f = f(\mathbf{x_2}, \boldsymbol{\mu})$, where f is a constraint or objective function. It should be clear that no explicit dependence exists between f and $\mathbf{LPP}$ or between $\mathbf{R}$ and $\mathbf{PP}$, DS, MAC and S. This only means that the respective jacobians are null. Since the length of $\mathbf{x_2}$ is $37(M \times N) + 4$, thus much larger then the number of outputs, the adjoint method is the most appropriated to calculate $\frac{df}{d\mathbf{x_2}}$ as

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x_2}} = \frac{\partial f}{\partial \mathbf{x_2}} + \begin{bmatrix}\boldsymbol{\psi}\end{bmatrix}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x_2}} \tag{5.8a}$$

$$\begin{bmatrix}\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\end{bmatrix}^T \begin{bmatrix}\boldsymbol{\psi}\end{bmatrix} = -\begin{bmatrix}\frac{\partial f}{\partial \boldsymbol{\mu}}\end{bmatrix}^T \tag{5.8b}$$

5.1.4 Chain-Rule

The last step in the sensitivity analysis framework corresponds to the assembly of the intermediate jacobians and to the application of the chain-rule. Since an intermediate information vector $\mathbf{x_2}$ is already defined, the final jacobian is given by

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x_{DV}}} = \frac{\mathrm{d}f}{\mathrm{d}\mathbf{x_2}}\frac{\mathrm{d}\mathbf{x_2}}{\mathrm{d}\mathbf{x_{DV}}} \tag{5.9}$$

where $\frac{\mathrm{d}\mathbf{x_2}}{\mathrm{d}\mathbf{x_{DV}}}$ corresponds to

$$\frac{\mathrm{d}\mathbf{x_2}}{\mathrm{d}\mathbf{x_{DV}}} = \begin{bmatrix}\begin{bmatrix}\frac{\mathrm{d}\mathbf{x_1}}{\mathrm{d}\mathbf{x_{DV}}}\end{bmatrix}^T & \begin{bmatrix}\frac{\mathrm{d}DS}{\mathrm{d}\mathbf{x_{DV}}}\end{bmatrix}^T & \begin{bmatrix}\frac{\mathrm{d}\mathbf{LPP}}{\mathrm{d}\mathbf{x_{DV}}}\end{bmatrix}^T & \begin{bmatrix}\frac{\partial S}{\partial \mathbf{x_{DV}}}\end{bmatrix}^T & \begin{bmatrix}\frac{\partial \mathrm{MAC}}{\partial \mathbf{x_{DV}}}\end{bmatrix}^T & \begin{bmatrix}\frac{\partial \alpha}{\partial \mathbf{x_{DV}}}\end{bmatrix}^T & [0]\end{bmatrix}^T \tag{5.10}$$

Note that some entries of the matrix $\frac{\mathrm{d}\mathbf{x_2}}{\mathrm{d}\mathbf{x_{DV}}}$ were already replaced by their respective partial derivatives since explicit dependence is observed for those cases. The reason is because S and MAC are outputs from the first module, *Wing Geometry*, and α is a design variable. The zero matrix, in the last entry, corresponds to $\frac{\partial V_\infty}{d\mathbf{x_{DV}}}$, since V_∞ does not depend on the design variables. Moreover, $\frac{\partial \alpha}{d\mathbf{x_{DV}}}$ is given by

$$\frac{\partial \alpha}{\partial \mathbf{x_{DV}}} = \begin{bmatrix}1 & 0 & \ldots & 0 & 0\end{bmatrix} \tag{5.11}$$

The remaining matrix entries are still required to be expressed in terms of the partial derivatives, obtained directly from the modules of sensitivity analysis. That task can be easily accomplished attending the fact that

$$\frac{\mathrm{d}\mathbf{x_1}}{\mathrm{d}\mathbf{x_{DV}}} = \frac{\partial \mathbf{x_1}}{\partial \mathbf{WP}} \frac{\partial \mathbf{WP}}{\partial \mathbf{x_{DV}}} \tag{5.12}$$

$$\frac{\mathrm{d}DS}{\mathrm{d}\mathbf{x_{DV}}} = \frac{\partial DS}{\partial \mathbf{WP}} \frac{\partial \mathbf{WP}}{\partial \mathbf{x_{DV}}} \tag{5.13}$$

$$\frac{\mathrm{d}\mathbf{LPP}}{\mathrm{d}\mathbf{x_{DV}}} = \frac{\partial \mathbf{LPP}}{\partial \mathbf{x_1}} \frac{\partial \mathbf{x_1}}{\partial \mathbf{WP}} \frac{\partial \mathbf{WP}}{\partial \mathbf{x_{DV}}} \tag{5.14}$$

5.2 Sensitivities of *Wing Parametrization* Module

The sensitivity analysis for the module *Wing Parametrization* will now be presented. Its goal corresponds to the calculation of three jacobian matrices, $\frac{\partial S}{\partial \mathbf{x_{DV}}}$, $\frac{\partial \mathrm{MAC}}{\partial \mathbf{x_{DV}}}$ and $\frac{\partial \mathbf{WP}}{\partial \mathbf{x_{DV}}}$. The first two matrices are calculated by hand since the respective analytical expressions are simple. The last one is calculated with the aid of automatic differentiation since the dependence of $\mathbf{WP}$ on the design variables is much more complex than the dependence of S and MAC on the same variables.

5.2.1 Partial Derivatives by Symbolic Differentiation

Observing Equation (4.31a) one may conclude that the planform area depends only on the taper ratio, wing span and the root chord. Thus, the partial derivatives different from zero are given by

$$\frac{\partial S}{\partial c_r} = \left(\frac{1+\lambda}{2}\right) b \tag{5.15a}$$

$$\frac{\partial S}{\partial b} = \left(\frac{1+\lambda}{2}\right) c_r \tag{5.15b}$$

$$\frac{\partial S}{\partial \lambda} = \frac{b\, c_r}{2} \tag{5.15c}$$

Observing now Equation (4.31b), only explicit dependence on the root chord and taper ratio are observed. Therefore, the partial derivatives different from zero are given by

$$\frac{\partial \mathrm{MAC}}{\partial c_r} = \frac{2}{3}\left(\frac{\lambda^2+\lambda+1}{\lambda+1}\right) \tag{5.16a}$$

$$\frac{\partial \mathrm{MAC}}{\partial \lambda} = \frac{2}{3} c_r \frac{\lambda(\lambda+2)}{(\lambda+1)^2} \tag{5.16b}$$

5.2.2 Partial Derivatives by Automatic Differentiation

The calculation of $\frac{\partial \mathbf{WP}}{\partial \mathbf{x_{DV}}}$ was carried using automatic differentiation (AD), more precisely with ADiMat. A decision had to be made regarding the method of propagating the intermediate derivatives since both forward and reverse modes are implemented in the AD tool. The choice was based on the number of outputs and inputs ratio. The number of inputs in `wing_discretization.m` are $8 + 24(N+1)$ and the number of outputs are $3(N \times M)$, where M and N are the chordwise and semi-spanwise number of panels. It is worth to use the forward mode of AD if the number of outputs is bigger than the number of inputs. In other words, forward mode is justified for the present case if the condition $\frac{M}{N} > \frac{10.67}{N^2} + \frac{8}{N}$ holds. Since the required number of panels in the chord direction is expected to be higher than the number of panels in the span direction and $N > 10$, most likely, it seems plausible to use forward mode. ADiMat provides a high-level function which implements the forward mode of AD, called `admDiffFor`. The routine takes as arguments an handle to the function to be differentiated, the function's input values and a seed matrix S. The output is a matrix equal to JS, where J is the jacobian of all the outputs with respect to all the inputs. One may choose the right seed matrix to calculate the partial derivatives of interest. Alternatively to the seed matrix, `admDiffFor` accepts options where one may choose which variables are independent and dependent, where the remaining function's variables are assumed as constants. The latter approach was chosen in the implementation.

When calling `admDiffFor` with an handle to `wing_geometry.m`, an additional function is created with the differentiated code named with the same name of the original function added by the prefix `g_`. A great advantage of using AD is the easiness of implementation from the users perspective, being the main reason why it was used in this module, as it will be seen next.

5.2.3 Benchmark - Complex-Step Derivative

Since the complex-step derivative may also produce results with the same precision as automatic differentiation, both methods were implemented and compared to chose the best. First, a baseline wing configuration was chosen. The choice was quite arbitrary since one is only interested in comparing the execution time and the derivatives accuracy, for both methods. Nevertheless, the configuration chosen was a plane rectangular wing with $A\!\!\!R = 6$, without twist, sweep and dihedral. The airfoil is a NACA 0010 and it is the same along the wing span. Also, the flight condition is characterized by $V_\infty = 75\, m/s$ and $\alpha = 1.5°$. This wing's configuration will be adopted along this chapter for all the benchmark purposes.

First, one must guarantee that the derivatives are being calculated correctly. Table 5.1, provides a representative benchmark of the derivatives calculated in this module. The table presents the absolute error of the partial derivatives of a wing point P components with respect to the components of $\alpha \cup \mathbf{x_{geo}}$. The point P is an element of $\mathbf{WP}$, located on the lower surface of the wing. According to the table, the absolute error is really small, being the greatest error about $\mathcal{O}(10^{-16})$. This analysis was carried for all wing points, with respect to all the inputs, although the results are not presented here. In that situation, the worst error is $\mathcal{O}(10^{-14})$.

Additionally, the execution time of each implementation was also benchmarked and presented in Table 5.2. The elapsed time was measured using the `tic` and `toc` commands, using an *Intel® Core™ i5-2410M CPU @ 2.30GHz* processor, which was used to obtain all the results presented in this work. According to the same table, the execution times are quite similar differing in a few seconds, for all the mesh sizes. Nevertheless, automatic differentiation is slightly faster with savings up to 60%, thus, it was chosen to be implemented.

5.3 Sensitivities of *Panels Definition* Module

The jacobian matrices generated in the second sensitivity analysis module are presented next. They correspond to a set of four jacobians: $\frac{\partial \mathbf{PP}}{\partial \mathbf{WP}}$, $\frac{\partial \mathbf{CP}}{\partial \mathbf{WP}}$, $\frac{\partial \mathbf{LV}}{\partial \mathbf{WP}}$ and $\frac{\partial DS}{\partial \mathbf{WP}}$. These matrices were constructed with the aid of smaller ones, as demonstrated next. All of the latter matrices are obtained using symbolic

	$E\left(\frac{\partial \mathbf{P}_i}{\partial \Lambda}\right)$	$E\left(\frac{\partial \mathbf{P}_i}{\partial \Gamma}\right)$	$E\left(\frac{\partial \mathbf{P}_i}{\partial \delta_r}\right)$	$E\left(\frac{\partial \mathbf{P}_i}{\partial \delta_t}\right)$	$E\left(\frac{\partial \mathbf{P}_i}{\partial b}\right)$	$E\left(\frac{\partial \mathbf{P}_i}{\partial c_r}\right)$	$E\left(\frac{\partial \mathbf{P}_i}{\partial \lambda}\right)$	$E\left(\frac{\partial \mathbf{P}_i}{\partial \alpha}\right)$
$\mathbf{i=x}$	5.55E-17	0	6.94E-18	4.34E-19	0	0	6.94E-18	0
$\mathbf{i=y}$	0	0	0	0	6.94E-18	0	0	0
$\mathbf{i=z}$	0	5.55E-17	0	6.94E-18	0	0	4.34E-19	0

Table 5.1: Absolute error of point P derivatives with respect to $\alpha \cup \mathbf{x_{geo}}$ components

No. panels	50	200	450	800	1250	1800
CS time [s]	0.639	3.011	9.332	19.815	38.970	64.553
AD time [s]	0.380	1.151	4.191	10.400	27.219	55.999
Savings [%]	40.5	61.8	55.1	47.5	30.2	13.3

Table 5.2: Computational cost of the *Wing Parametrization* sensitivity analysis module for $M = N$.

differentiation, with the aid of the *Symbolic Math Toolbox*™ from MATLAB® and coded by hand. The process was quite lengthy since a large number of intermediate calculations were necessary to build the jacobians but, the price to pay was fair since large savings in the computational runtime were obtained.

5.3.1 Partial Derivatives by Symbolic Differentiation

The process of obtaining the jacobians was performed through differentiating *Panels Definition* module. Taking the partial derivatives of Equations (4.38a) and (4.38b) with respect to the wing points $\mathbf{WP}_{kh}$ and considering the derivatives different from zero only, one can write

$$\frac{\partial \mathbf{P_f}_{ij}}{\partial \mathbf{WP_2}_{ij}} = \frac{\partial \mathbf{P_s}_{ij}}{\partial \mathbf{WP_3}_{ij}} = [I] = -\frac{\partial \mathbf{P_f}_{ij}}{\partial \mathbf{WP_1}_{ij}} = -\frac{\partial \mathbf{P_s}_{ij}}{\partial \mathbf{WP_4}_{ij}} \tag{5.17}$$

and doing the same to Equations (4.39a) and (4.39b)

$$\frac{\partial \mathbf{X_f}_{ij}}{\partial \mathbf{WP_1}_{ij}} = \frac{\partial \mathbf{X_f}_{ij}}{\partial \mathbf{WP_2}_{ij}} = \frac{1}{2}[I] = \frac{\partial \mathbf{X_s}_{ij}}{\partial \mathbf{WP_3}_{ij}} = \frac{\partial \mathbf{X_s}_{ij}}{\partial \mathbf{WP_4}_{ij}} \tag{5.18}$$

where $[I]$ is the 3×3 identity matrix. Observing now the dependency of $\mathbf{l}_{ij}$ on $\mathbf{P_f}$ and $\mathbf{P_s}$ in Equation (4.40) it is possible to state that

$$\frac{\mathrm{d}\mathbf{l}_{ij}}{\mathrm{d}\mathbf{WP}_{kh}} = \begin{bmatrix} \frac{\partial \mathbf{l}_{ij}}{\partial \mathbf{P_f}_{mn}} & \frac{\partial \mathbf{l}_{ij}}{\partial \mathbf{P_s}_{mn}} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{P_f}_{mn}}{\partial \mathbf{WP_1}_{kh}} & \frac{\partial \mathbf{P_f}_{mn}}{\partial \mathbf{WP_2}_{kh}} & [0] & [0] \\ [0] & [0] & \frac{\partial \mathbf{P_s}_{mn}}{\partial \mathbf{WP_3}_{kh}} & \frac{\partial \mathbf{P_s}_{mn}}{\partial \mathbf{WP_4}_{kh}} \end{bmatrix} \tag{5.19}$$

where the non-zero entries for the matrices $\frac{\partial \mathbf{l}_{ij}}{\partial \mathbf{P_f}_{mn}}$ and $\frac{\partial \mathbf{l}_{ij}}{\partial \mathbf{P_s}_{mn}}$ in Equation (5.19) are obtained for $m = i$ and $n = j$, given by

$$\frac{\partial l_{ijk}}{\partial P_{f_{ijh}}} = \frac{\partial l_{ijk}}{\partial P_{s_{ijh}}} = -\frac{\left(P_{f_{ijk}} + P_{s_{ijk}}\right)\left(P_{f_{ijh}} + P_{s_{ijh}}\right)}{\|\mathbf{P_f}_{ij} + \mathbf{P_s}_{ij}\|^3}, \quad k \neq h \tag{5.20a}$$

$$\frac{\partial l_{ijk}}{\partial P_{f_{ijh}}} = \frac{\partial l_{ijk}}{\partial P_{s_{ijh}}} = \frac{P^2_{f_{ijp}} + P^2_{f_{ijq}} + P^2_{s_{ijp}} + P^2_{s_{ijq}} + 2\left(P_{f_{ijp}}P_{s_{ijp}} + P_{f_{ijq}}P_{s_{ijq}}\right)}{\|\mathbf{P_f}_{ij} + \mathbf{P_s}_{ij}\|^3}, \quad k = h, p \neq q \neq k \tag{5.20b}$$

Observing now Equations (4.41a) and (4.41b) one may realize they are quite similar, in fact, the equations have the same form if one consider that $\mathbf{X}_{\mathbf{1}ij}$ corresponds to $\mathbf{X}_{\mathbf{4}ij}$, $\mathbf{X}_{\mathbf{2}ij}$ to $\mathbf{X}_{\mathbf{3}ij}$, $\mathbf{X}_{\mathbf{f}ij}$ to $\mathbf{X}_{\mathbf{s}ij}$ and $\mathbf{P}_{\mathbf{f}ij}$ to $\mathbf{P}_{\mathbf{s}ij}$. Taking this advantage one may realize that the different from zero partial derivatives of $\mathbf{X}_{\mathbf{1}ij}$ and $\mathbf{X}_{\mathbf{2}ij}$ with respect to $\mathbf{X}_{\mathbf{f}ij}$, $\mathbf{P}_{\mathbf{f}ij}$ and $\mathbf{l}_{ij}$ are given by

$$\frac{\partial \mathbf{X}_{\mathbf{1}ij}}{\partial \mathbf{X}_{\mathbf{f}ij}} = \frac{\partial \mathbf{X}_{\mathbf{2}ij}}{\partial \mathbf{X}_{\mathbf{f}ij}} = [I] \tag{5.21a}$$

$$\frac{\partial X_{1ijm}}{\partial P_{f_{ijn}}} = -\frac{\partial X_{2ijm}}{\partial P_{f_{ijn}}} = -\frac{P_{f_{ijn}}.l_{ijm}}{2\|\mathbf{P}_{\mathbf{f}ij}\|} \tag{5.21b}$$

$$\frac{\partial \mathbf{X}_{\mathbf{1}ij}}{\partial \mathbf{l}_{ij}} = -\frac{\partial \mathbf{X}_{\mathbf{2}ij}}{\partial \mathbf{l}_{ij}} = -\frac{\|\mathbf{P}_{\mathbf{f}ij}\|}{2}[I] \tag{5.21c}$$

The different from zero partial derivatives of $\mathbf{X}_{\mathbf{3}ij}$ and $\mathbf{X}_{\mathbf{4}ij}$ with respect to $\mathbf{X}_{\mathbf{s}ij}$, $\mathbf{P}_{\mathbf{s}ij}$ and $\mathbf{l}_{ij}$ are obtained easily using the correspondence presented previously. Since all the building blocks to construct $\frac{\mathrm{d}\mathbf{PP}_{ij}}{\mathrm{d}\mathbf{WP}_{\mathbf{kh}}}$ are defined, the assembly according to the chain-rule is

$$\frac{\mathrm{d}\mathbf{PP}_{ij}}{\mathrm{d}\mathbf{WP}_{kh}} = \begin{bmatrix} \frac{\partial \mathbf{X}_{\mathbf{1}ij}}{\partial \mathbf{X}_{\mathbf{f}mn}} & \frac{\partial \mathbf{X}_{\mathbf{1}ij}}{\partial \mathbf{P}_{\mathbf{f}mn}} & \frac{\partial \mathbf{X}_{\mathbf{1}ij}}{\partial \mathbf{l}_{mn}} & [0] & [0] \\ \frac{\partial \mathbf{X}_{\mathbf{2}ij}}{\partial \mathbf{X}_{\mathbf{f}mn}} & \frac{\partial \mathbf{X}_{\mathbf{2}ij}}{\partial \mathbf{P}_{\mathbf{f}mn}} & \frac{\partial \mathbf{X}_{\mathbf{2}ij}}{\partial \mathbf{l}_{mn}} & [0] & [0] \\ [0] & [0] & \frac{\partial \mathbf{X}_{\mathbf{3}ij}}{\partial \mathbf{l}_{mn}} & \frac{\partial \mathbf{X}_{\mathbf{3}ij}}{\partial \mathbf{X}_{\mathbf{s}mn}} & \frac{\partial \mathbf{X}_{\mathbf{3}ij}}{\partial \mathbf{P}_{\mathbf{s}mn}} \\ [0] & [0] & \frac{\partial \mathbf{X}_{\mathbf{4}ij}}{\partial \mathbf{l}_{mn}} & \frac{\partial \mathbf{X}_{\mathbf{4}ij}}{\partial \mathbf{X}_{\mathbf{s}mn}} & \frac{\partial \mathbf{X}_{\mathbf{4}ij}}{\partial \mathbf{P}_{\mathbf{s}mn}} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{X}_{\mathbf{f}mn}}{\partial \mathbf{WP}_{\mathbf{1}kh}} & \frac{\partial \mathbf{X}_{\mathbf{f}mn}}{\partial \mathbf{WP}_{\mathbf{2}kh}} & [0] & [0] \\ \frac{\partial \mathbf{P}_{\mathbf{f}mn}}{\partial \mathbf{WP}_{\mathbf{1}kh}} & \frac{\partial \mathbf{P}_{\mathbf{f}mn}}{\partial \mathbf{WP}_{\mathbf{2}kh}} & [0] & [0] \\ \frac{\mathrm{d}\mathbf{l}_{mn}}{\mathrm{d}\mathbf{WP}_{\mathbf{1}kh}} & \frac{\mathrm{d}\mathbf{l}_{mn}}{\mathrm{d}\mathbf{WP}_{\mathbf{2}kh}} & \frac{\mathrm{d}\mathbf{l}_{mn}}{\mathrm{d}\mathbf{WP}_{\mathbf{3}kh}} & \frac{\mathrm{d}\mathbf{l}_{mn}}{\mathrm{d}\mathbf{WP}_{\mathbf{4}kh}} \\ [0] & [0] & \frac{\partial \mathbf{X}_{\mathbf{s}mn}}{\partial \mathbf{WP}_{\mathbf{3}kh}} & \frac{\partial \mathbf{X}_{\mathbf{s}mn}}{\partial \mathbf{WP}_{\mathbf{4}kh}} \\ [0] & [0] & \frac{\partial \mathbf{P}_{\mathbf{s}mn}}{\partial \mathbf{WP}_{\mathbf{3}kh}} & \frac{\partial \mathbf{P}_{\mathbf{s}mn}}{\partial \mathbf{WP}_{\mathbf{4}kh}} \end{bmatrix} \tag{5.22}$$

Considering Equation (4.45), one may quickly conclude that the collocation point location only depends on the panel corner's locations. Therefore, the partial derivatives of $\mathbf{CP}_{ij}$ different from zero are

$$\frac{\partial \mathbf{CP}_{ij}}{\partial \mathbf{X}_{\mathbf{1}ij}} = \frac{\partial \mathbf{CP}_{ij}}{\partial \mathbf{X}_{\mathbf{2}ij}} = \frac{\partial \mathbf{CP}_{ij}}{\partial \mathbf{X}_{\mathbf{3}ij}} = \frac{\partial \mathbf{CP}_{ij}}{\partial \mathbf{X}_{\mathbf{4}ij}} = \frac{1}{4}[I] \tag{5.23}$$

combining the results from Equation (5.23) and Equation (5.22) one may calculate the assembly

$$\frac{\mathrm{d}\mathbf{CP}_{ij}}{\mathrm{d}\mathbf{WP}_{kh}} = \frac{\partial \mathbf{CP}_{ij}}{\partial \mathbf{PP}_{mn}} \frac{\mathrm{d}\mathbf{PP}_{mn}}{\mathrm{d}\mathbf{WP}_{kh}} \tag{5.24}$$

The next interest partial derivative is the panel's areas DS with respect to the input points $\mathbf{WP}$. Observing Equation (4.46), it can be shown that $\frac{\mathrm{d}DS_{ij}}{\mathrm{d}\mathbf{WP}_{kh}}$ may be given by

$$\frac{\mathrm{d}DS_{ij}}{\mathrm{d}\mathbf{WP}_{kh}} = \frac{\partial DS_{ij}}{\partial \mathbf{U}_{mn}} \frac{\partial \mathbf{U}_{mn}}{\partial \mathbf{PP}_{rs}} \frac{\mathrm{d}\mathbf{PP}_{rs}}{\mathrm{d}\mathbf{WP}_{kh}} \tag{5.25}$$

where $\mathbf{U}_{mn}^T = [\mathbf{X_A}\,\mathbf{X_B}\,\mathbf{X_C}]_{mn}$. Also, through Equation (4.46), it can be concluded that the non zero components of $\frac{\partial \mathbf{U}_{mn}}{\partial \mathbf{PP}_{rs}}$ are given by

$$\frac{\partial \mathbf{U}_{mn}}{\partial \mathbf{PP}_{mn}} = \begin{bmatrix} -[I] & [I] & [0] & [0] \\ -[I] & [0] & [I] & [0] \\ -[I] & [0] & [0] & [I] \end{bmatrix} \tag{5.26}$$

The partial derivatives for $\frac{\partial DS_{ij}}{\partial \mathbf{U}_{mn}}$ may be found in section A.1 in Appendix A since each matrix entry is calculated individually.

Considering now Equations (4.42) and (4.43), it can easily be noticed that the partial derivatives of the unit normal $\mathbf{n}$ with respect to the input wing points $\mathbf{WP}$ are given by

$$\frac{\mathrm{d}\mathbf{n}_{ij}}{\mathrm{d}\mathbf{WP}_{kh}} = \frac{\partial \mathbf{n}_{ij}}{\partial \mathbf{N}_{mn}} \frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{PP}_{rs}} \frac{\mathrm{d}\mathbf{PP}_{rs}}{\mathrm{d}\mathbf{WP}_{kh}} \tag{5.27}$$

Finally, from Equation (4.44), it can be observed that the second basis vector depends on the first and third, for each panel. Therefore, it can be written that

$$\frac{\mathrm{d}\mathbf{m}_{ij}}{\mathrm{d}\mathbf{WP}_{kh}} = \begin{bmatrix} \frac{\partial \mathbf{m}_{ij}}{\partial \mathbf{l}_{mn}} & \frac{\partial \mathbf{m}_{ij}}{\partial \mathbf{n}_{mn}} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}\mathbf{l}_{mn}}{\mathrm{d}\mathbf{WP_1}_{kh}} & \frac{\mathrm{d}\mathbf{l}_{mn}}{\mathrm{d}\mathbf{WP_2}_{kh}} & \frac{\mathrm{d}\mathbf{l}_{mn}}{\mathrm{d}\mathbf{WP_3}_{kh}} & \frac{\mathrm{d}\mathbf{l}_{mn}}{\mathrm{d}\mathbf{WP_4}_{kh}} \\ \frac{\mathrm{d}\mathbf{n}_{mn}}{\mathrm{d}\mathbf{WP_1}_{kh}} & \frac{\mathrm{d}\mathbf{n}_{mn}}{\mathrm{d}\mathbf{WP_2}_{kh}} & \frac{\mathrm{d}\mathbf{n}_{mn}}{\mathrm{d}\mathbf{WP_3}_{kh}} & \frac{\mathrm{d}\mathbf{n}_{mn}}{\mathrm{d}\mathbf{WP_4}_{kh}} \end{bmatrix} \tag{5.28}$$

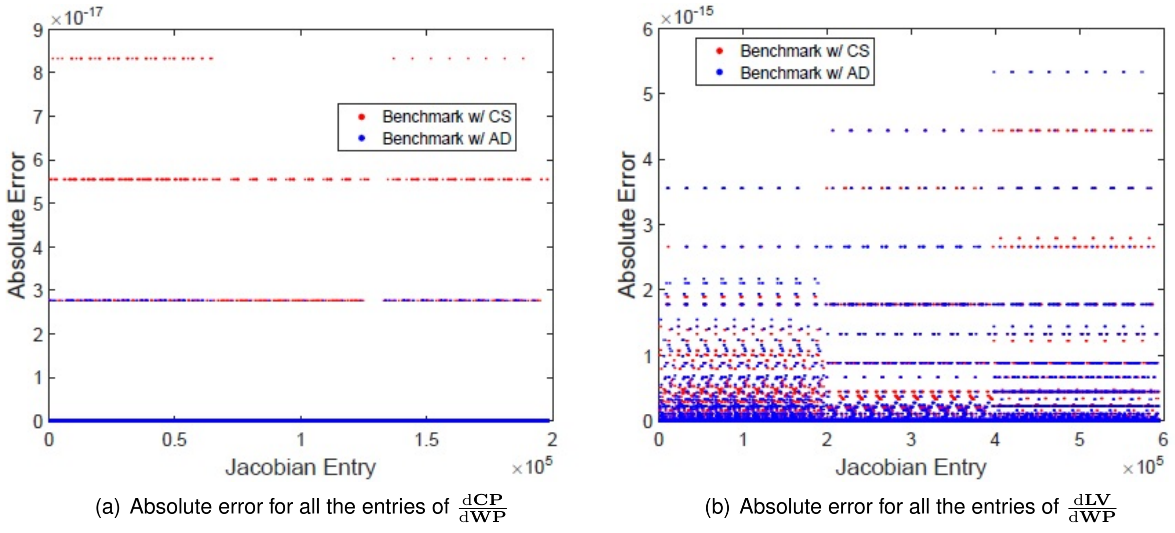
Since large expressions are present in the intermediate jacobians in Equations (5.27) and (5.28), the expressions for $\frac{\partial \mathbf{n}_{ij}}{\partial \mathbf{N}_{mn}}$, $\frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{PP}_{rs}}$ and the first matrix in the right hand side of Equation (5.28) are also presented in section A.1 in the Appendix A.

5.3.2 Benchmark - AD and the Complex-Step Derivative

Since this module was constructed using symbolic differentiation, it was firstly required to guarantee the resulting jacobians were being calculated correctly. To accomplish that task, the same wing configuration was used to verify the previous module . Only two benchmark cases are presented here for the jacobians $\frac{\mathrm{d}\mathbf{CP}}{\mathrm{d}\mathbf{WP}}$ and $\frac{\mathrm{d}\mathbf{LV}}{\mathrm{d}\mathbf{WP}}$ although the analysis was carried for all the calculated jacobians.

To verify this sensitivity analysis module, the resulting jacobians were compared using both automatic differentiation and the complex-step derivative with a step-size of $h = 10^{-50}$. Consider Figures 5.2 (a) and 5.2 (b). Each of the jacobians were unrolled, column by column, and for each entry on the jacobian, labeled as the horizontal axis in both plots, the absolute error was calculated using both AD and the CSD as reference, represented by the red and blue dots, respectively. According to the same figures, this error is bounded for both cases and very small, $< \mathcal{O}(10^{-15})$, proving the derivatives were calculated correctly and therefore are accurate.

As already said, one of the greatest benefits of using symbolic differentiation (SD) on this module relates not only with the derivatives calculated accurately but also with the huge amount of computational time saving. According to Table 5.3, the elapsed time using the developed module is about a thousand times faster than using AD or the CSD. One of the reasons relates with algebraic simplifications, reducing the number of calculations, and attention to jacobian sparsity, being only the nonzero entries calculated.



(a) Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{CP}}{\mathrm{d}\mathbf{WP}}$ (b) Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{LV}}{\mathrm{d}\mathbf{WP}}$

Figure 5.2: Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{CP}}{\mathrm{d}\mathbf{WP}}$ and $\frac{\mathrm{d}\mathbf{LV}}{\mathrm{d}\mathbf{WP}}$

The same could be done using sparsity exploration on the AD tool and a gain would be obtained. Nevertheless, coding this module by hand, one could also use MATLAB® vectorization techniques to improve even further the framework's performance.

No. panels	200	450	800	1250	1800
CSD time [s]	20.05006	109.4079	289.9539	761.1905	2821.0137
AD time [s]	22.93737	148.1591	599.1266	2296.589	8623.2376
SD time [s]	0.117399	0.294621	0.554629	1.005761	1.822698
Savings CSD [%]	99.4	99.7	99.8	99.9	99.9
Savings AD [%]	99.5	99.8	99.9	100	100

Table 5.3: Computational cost of the *Panels Definition* sensitivity analysis module for $M = N$.

5.4 Sensitivities of *Change of Basis* Module

The partial derivatives of *Change of Basis* module were also calculated using symbolic differentiation and coded by hand. The main reason is because the change of basis matrix equation is very simple to differentiate although the goal here was not to particularly achieve less computational expense. No benchmark with other methods will be provided here for the reason stated previously. This benchmark was naturally done though. Remember Equation (4.47) which allows to write any point in the panel's (i,j) frame of reference. The partial derivative of the output point $\mathbf{P}^{'}$ with respect to $\mathbf{P}$ is equal to minus the partial derivative of the same point with respect to the origin $\mathbf{CP}_{ij}$, as

$$\frac{\partial \mathbf{P}^{'}}{\partial \mathbf{P}} = -\frac{\partial \mathbf{P}^{'}}{\partial \mathbf{CP}_{ij}} = \begin{bmatrix} l_{ij1} & l_{ij2} & l_{ij3} \\ m_{ij1} & m_{ij2} & m_{ij3} \\ n_{ij1} & n_{ij2} & n_{ij3} \end{bmatrix} \tag{5.29}$$

Similarly, taking the partial derivative with respect to $\mathbf{LV}_{ij}^T = [\mathbf{l}\,\mathbf{m}\,\mathbf{n}]_{ij}$, it will result in

$$\frac{\partial \mathbf{P}}{\partial \mathbf{LV}_{ij}} = \begin{bmatrix} P_1-CP_{ij1} & P_2-CP_{ij2} & P_3-CP_{ij3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & P_1-CP_{ij1} & P_2-CP_{ij2} & P_3-CP_{ij3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & P_1-CP_{ij1} & P_2-CP_{ij2} & P_3-CP_{ij3} \end{bmatrix} \tag{5.30}$$

All the building blocks are now determined to build the jacobians $\frac{\partial \mathbf{LPP}}{\partial \mathbf{PP}}$, $\frac{\partial \mathbf{LPP}}{\partial \mathbf{CP}}$ and $\frac{\partial \mathbf{LPP}}{\partial \mathbf{LV}}$. These matrices are easily constructed attending the fact each corner point (i, j) written in the panel's frame of reference only depends on the panel's collocation point $\mathbf{CP}_{ij}$, the panel's basis vectors $\mathbf{LV}_{ij}$ and its coordinates in the global frame of reference.

5.5 Sensitivities of *Aero Solver* Module

It is now required to calculate all the intermediate jacobians which play a role in the adjoint method, presented in the beginning of this chapter. Observing Table 4.4 one can remember that the residuals depend explicitly on the variables $\mathbf{CP}$, $\mathbf{LPP}$, $\mathbf{LV}$, α and V_∞. Moreover, solving the residual system of equations leads to the aerodynamic solution in terms of the doublet intensities $\boldsymbol{\mu}$, which depend implicitly of the last five variables. Therefore, the sensitivity analysis for this module consists in calculating six jacobian matrices. As already shown in section 5.3, symbolic differentiation and post coding can be a great advantage, specially in MATLAB® where vectorization is possible. For this reason and similarly to section 5.3, all of these jacobians were calculated by hand with the aid of symbolic differentiation.

5.5.1 Partial Derivatives w.r.t. Collocation Points

Remember the residual associated with the panel (i, j), given by Equation (4.48). Taking the partial derivative with respect to the collocation points and after some algebra yields

$$\frac{\partial R_{ij}}{\partial \mathbf{CP}_{kh}} = \begin{cases} \displaystyle\sum_{n=1}^{N}\left[\sum_{m=1}^{M-1}\left(\frac{\partial \mathcal{C}_{ijmn}}{\partial \mathbf{CP}_{kh}}\mu_{mn} + \frac{\partial \mathcal{B}_{ijmn}}{\partial \mathbf{CP}_{kh}}\sigma_{mn}\right)\right] & \text{if } (k,h) = (i,j) \\ \displaystyle\frac{\partial \mathcal{C}_{ijkh}}{\partial \mathbf{CP}_{kh}}\mu_{kh} + \frac{\partial \mathcal{B}_{ijkh}}{\partial \mathbf{CP}_{kh}}\sigma_{kh} & \text{if } (k,h) \neq (i,j) \text{ and } k \neq M \\ \displaystyle\frac{\partial \mathcal{C}_{ijkh}}{\partial \mathbf{CP}_{kh}}\left(\mu_{(M-1)h} - \mu_{1h}\right) & \text{if } (k,h) \neq (i,j) \text{ and } k = M \end{cases} \tag{5.31}$$

where the partial derivative of $\frac{\partial \mathcal{C}_{ijmn}}{\partial \mathbf{CP}_{kh}}$ is defined as

$$\begin{aligned} \frac{\partial \mathcal{C}_{ijmn}}{\partial \mathbf{CP}_{kh}} = \frac{\partial \mathcal{C}^1_{ijmn}}{\partial \mathbf{r^1}_{ijmn}}\left(\frac{\partial \mathbf{r^1}_{ijmn}}{\partial \mathbf{CP}_{ij}}\frac{\partial \mathbf{CP}_{ij}}{\partial \mathbf{CP}_{kh}} + \frac{\partial \mathbf{r^1}_{ijmn}}{\partial \mathbf{CP}_{mn}}\frac{\partial \mathbf{CP}_{mn}}{\partial \mathbf{CP}_{kh}}\right) + \\ \frac{\partial \mathcal{C}^2_{ijmn}}{\partial \mathbf{r^2}_{ijmn}}\left(\frac{\partial \mathbf{r^2}_{ijmn}}{\partial \mathbf{CP^2}_{ij}}\frac{\partial \mathbf{CP^2}_{ij}}{\partial \mathbf{CP}_{ij}}\frac{\partial \mathbf{CP}_{ij}}{\partial \mathbf{CP}_{kh}} + \frac{\partial \mathbf{r^2}_{ijmn}}{\partial \mathbf{CP}_{mn}}\frac{\partial \mathbf{CP}_{mn}}{\partial \mathbf{CP}_{kh}}\right) \end{aligned} \tag{5.32}$$

where, additionally, $\mathbf{r^1}_{ijmn}$ and $\mathbf{r^2}_{ijmn}$ are the collocation point and respective image's locations from panel (i, j), written in the (m, n) panel's frame of reference. With this in mind, the partial derivatives of $\mathbf{r^1}_{ijmn}$ and $\mathbf{r^2}_{ijmn}$ with respect to the respective collocation points, in Equation (5.32), are calculated using Equation (5.29). Since $\mathbf{CP^2}_{ij}$ is the image of $\mathbf{CP}_{ij}$, the respective partial derivative is given by

$$\frac{\partial \mathbf{CP^2}_{ij}}{\partial \mathbf{CP}_{ij}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.33}$$

It should also be noticed that the source influence $\mathcal{B}$ has the same variable dependence as $\mathcal{C}$, thus, $\frac{\partial \mathcal{B}_{ijmn}}{\partial \mathbf{CP}_{kh}}$ is calculated using Equation (5.32) replacing $\mathcal{C}$ by $\mathcal{B}$. All the partial derivatives of the source and dipole influences with respect to all their dependencies, already and lately presented, are introduced in section A.2 in Appendix A.

5.5.2 Partial Derivatives w.r.t. Local Corner Points

Each residual associated with panel (i, j) depends also on the influence panel's corner points, written in its own frame of reference. Taking the partial derivative with respect to the points $\mathbf{X}'_{\mathbf{1}kh}$, $\mathbf{X}'_{\mathbf{2}kh}$, $\mathbf{X}'_{\mathbf{3}kh}$, $\mathbf{X}'_{\mathbf{4}kh}$, or simply with respect to $\mathbf{LPP}_{kh}$

$$\frac{\partial R_{ij}}{\partial \mathbf{LPP}_{kh}} = \begin{cases} \dfrac{\partial \mathcal{C}_{ijkh}}{\partial \mathbf{LPP}_{kh}} \left(\mu_{(M-1)h} - \mu_{1h}\right) & \text{if } k = M \\ \dfrac{\partial \mathcal{C}_{ijkh}}{\partial \mathbf{LPP}_{kh}} \mu_{kh} + \dfrac{\partial \mathcal{B}_{ijkh}}{\partial \mathbf{LPP}_{kh}} \sigma_{kh} & \text{if } k \neq M \end{cases} \tag{5.34}$$

5.5.3 Partial Derivatives w.r.t. Basis Vectors

The residuals also present explicit dependence on the panel's basis vectors. Taking now the partial derivatives with respect to $\mathbf{LV}_{kh}$, leads to

$$\frac{\partial R_{ij}}{\partial \mathbf{LV}_{kh}} = \begin{cases} \dfrac{\partial \mathcal{C}_{ijkh}}{\partial \mathbf{LV}_{kh}} \left(\mu_{(M-1)h} - \mu_{1h}\right) & \text{if } k = M \\ \dfrac{\partial \mathcal{C}_{ijkh}}{\partial \mathbf{LV}_{kh}} \mu_{kh} + \dfrac{\partial \mathcal{B}_{ijkh}}{\partial \mathbf{LV}_{kh}} \sigma_{kh} + \mathcal{B}_{ijkh} \dfrac{\partial \sigma_{kh}}{\partial \mathbf{LV}_{kh}} & \text{if } k \neq M \end{cases} \tag{5.35}$$

where $\frac{\partial \mathcal{C}_{ijkh}}{\partial \mathbf{LV}_{kh}}$ is given by

$$\frac{\partial \mathcal{C}_{ijkh}}{\partial \mathbf{LV}_{kh}} = \frac{\partial \mathcal{C}^1_{ijkh}}{\partial \mathbf{r^1}_{ijmn}} \frac{\mathbf{r^1}_{ijmn}}{\partial \mathbf{LV}_{kh}} + \frac{\partial \mathcal{C}^2_{ijkh}}{\partial \mathbf{r^2}_{ijmn}} \frac{\mathbf{r^2}_{ijmn}}{\partial \mathbf{LV}_{kh}} \tag{5.36}$$

and $\frac{\mathbf{r^1}_{ijmn}}{\partial \mathbf{LV}_{kh}}$ and $\frac{\mathbf{r^2}_{ijmn}}{\partial \mathbf{LV}_{kh}}$ are calculated using Equation (5.30). As previously, $\frac{\partial \mathcal{B}_{ijkh}}{\partial \mathbf{LV}_{kh}}$ is calculated easily replacing $\mathcal{C}$ by $\mathcal{B}$ in Equation (5.36).

5.5.4 Partial Derivatives w.r.t. Angle-of-Attack and Airspeed

The residuals depend also on the angle of attack and on the absolute value of the free-stream velocity vector. Taking the partial derivatives with respect to these two variables yields

$$\frac{\partial R_{ij}}{\partial \alpha} = \sum_{n=1}^{N} \sum_{m=1}^{M-1} \mathcal{B}_{ijmn} \frac{\partial \sigma_{mn}}{\partial \alpha} \tag{5.37a}$$

$$\frac{\partial R_{ij}}{\partial V_\infty} = \sum_{n=1}^{N} \sum_{m=1}^{M-1} \mathcal{B}_{ijmn} \frac{\partial \sigma_{mn}}{\partial V_\infty} \tag{5.37b}$$

Partial Derivatives w.r.t. Source and Doublet Intensities

As observed in this section and also in section 5.5.3, it is required to calculate the partial derivatives of the source intensities with respect to its dependencies. Observing Equation (4.49), it may be concluded that those dependencies are on the panel's unitary normal and the free-stream velocity vectors. Taking the partial derivatives with respect to $\mathbf{LV}_{kh}$, α and V_∞, results

$$\frac{\partial \sigma_{kh}}{\partial \mathbf{LV}_{kh}} = \begin{bmatrix} [0] & [0] & \frac{\partial \sigma_{kh}}{\partial \mathbf{n}_{kh}} \end{bmatrix} = V_\infty \cdot \begin{bmatrix} [0 \quad 0 \quad 0] & [0 \quad 0 \quad 0] & [\cos\alpha \quad 0 \quad \sin\alpha] \end{bmatrix} \tag{5.38}$$

$$\frac{\partial \sigma_{kh}}{\partial \alpha} = V_\infty \cdot [n_{kh3}\cos\alpha - n_{kh1}\sin\alpha] \tag{5.39}$$

$$\frac{\partial \sigma_{kh}}{\partial V_\infty} = \frac{\sigma_{kh}}{V_\infty} \tag{5.40}$$

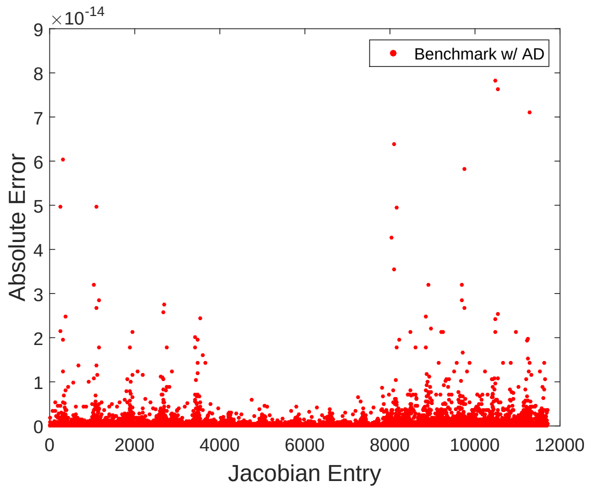
5.5.5 Partial Derivative w.r.t. the Doublet Intensities

Finally, the residuals' partial derivatives with respect to the double intensities still need to be calculated. Considering once more Equation (4.48) and taking the partial derivative with respect to μ_{kh} one may write

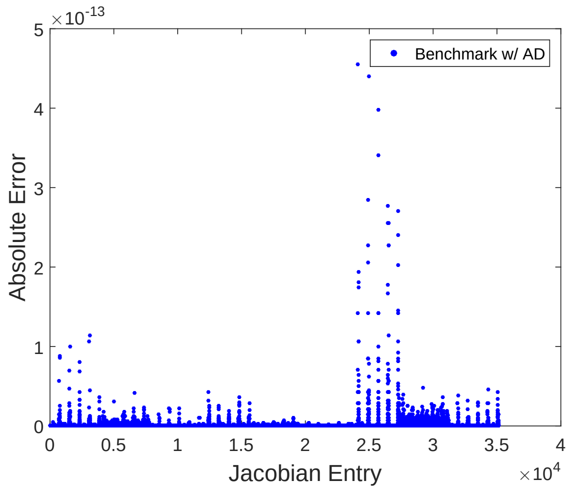
$$\frac{\partial R_{ij}}{\partial \mu_{kh}} = \begin{cases} \mathcal{C}_{ij1h} - \mathcal{C}_{ijMh} & \text{if } k = 1 \\ \mathcal{C}_{ij(M-1)h} + \mathcal{C}_{ijMh} & \text{if } k = M-1 \\ \mathcal{C}_{ijkh} & \text{if } k \neq 1 \wedge k \neq M-1 \end{cases} \tag{5.41}$$

5.5.6 Benchmark - Automatic Differentiation

It is now required to verify the sensitivity analysis of the present module since the differentiation was carried by hand with the aid of symbolic differentiation. Assuming the same wing configuration as previously, a benchmark with the results obtained with the forward mode of automatic differentiation will be presented.

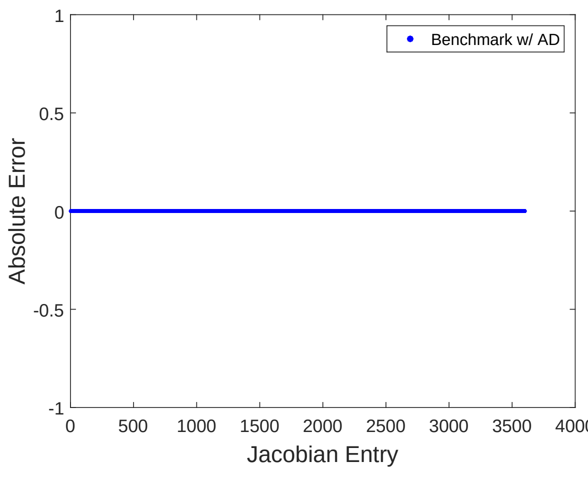


(a) Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\mathbf{CP}}$

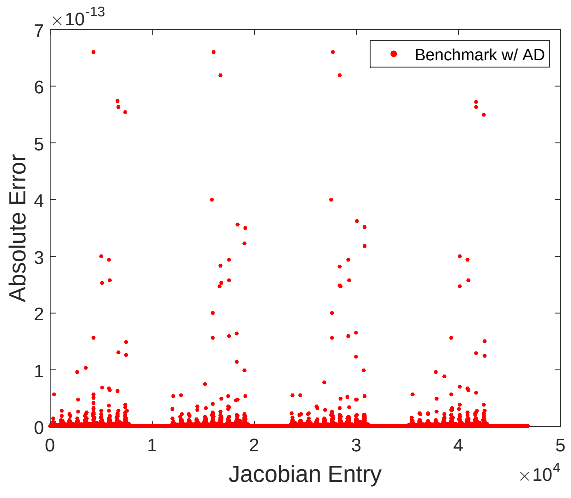


(b) Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\mathbf{LV}}$

Figure 5.3: Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\mathbf{CP}}$ and $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\mathbf{LV}}$

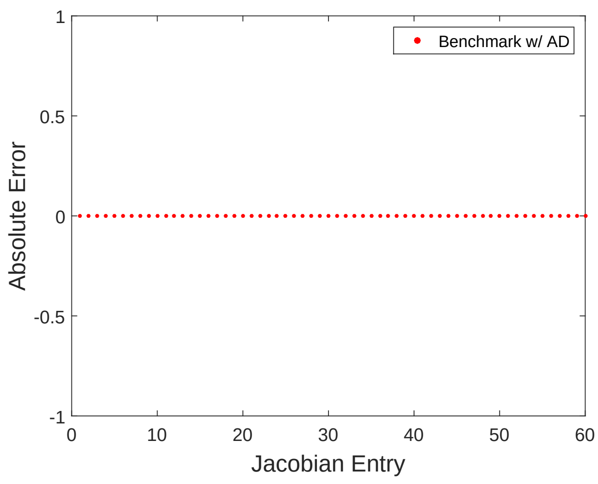


(a) Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\mu}$

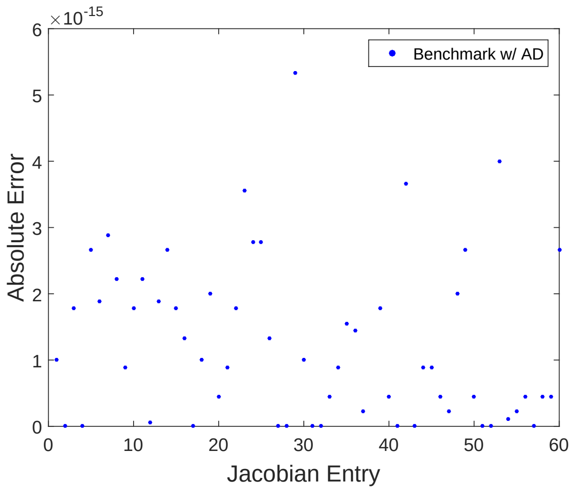


(b) Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\mathbf{LPP}}$

Figure 5.4: Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\mu}$ and $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\mathbf{LPP}}$



(a) Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}V_\infty}$



(b) Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\alpha}$

Figure 5.5: Absolute error for all the entries of $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}V_\infty}$ and $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\alpha}$

Figures 5.3 through 5.5 presents the absolute error associated with the six jacobian matrices produced in this module, for all the jacobian entries, taking the results obtained with AD as reference and for a mesh with 200 panels. The calculated errors are bounded and, for the worst case scenario, some components are about 10^{-12}, proving the constructed framework's validity.

The developed module was also benchmarked with AD, according to the computational cost. The run time spent by the AD tool corresponds to the time to generate the differentiated code with respect to each of the independent variables, and the time to run it. The former corresponds to the smaller portion of time and took about 10 seconds. As observed in Table 5.4, the new developed module is about 200 times faster than using AD with savings above 99%. Moreover, the AD tool run times are absolutely unacceptable, even for course meshes, proving the efficiency of the new developed module.

No. panels	50	200	450
SD time [s]	3.796	40.797	222.411
AD time [s]	749.964	8653.022	40582.198
savings [%]	99.5	99.5	99.5

Table 5.4: Computational cost of the *Aero Solver* sensitivity analysis module for $M = N$.

5.6 Sensitivities of *Post Process* Module

Paying close attention to Table 4.5, one may conclude that the number of inputs is much larger than the number of outputs. In fact, the number of outputs is at most equal to five, corresponding to the aerodynamic coefficients. Due to this characteristic, the application of the reverse mode of automatic differentiation is well suited for implementation.

In order to apply the reverse-mode of AD, ADiMat provides another function called `admDiffRev.m` for that effect. The function's signature is the same as for `admDiffFor.m`. The result of using this function associated with an handle to function `post_process.m` is an augmented differentiated code, able to calculate both the function values and associated derivatives, with the same name as the original function but appended with the prefix `a_`.

Figure 5.6 presents the computational runtime as a function of the `post_process.m` number of inputs. According to the figure, the measured data is well fitted by a linear function and is quite inexpensive even for large number of inputs. Thus, this implementation is almost insensible to the mesh size.

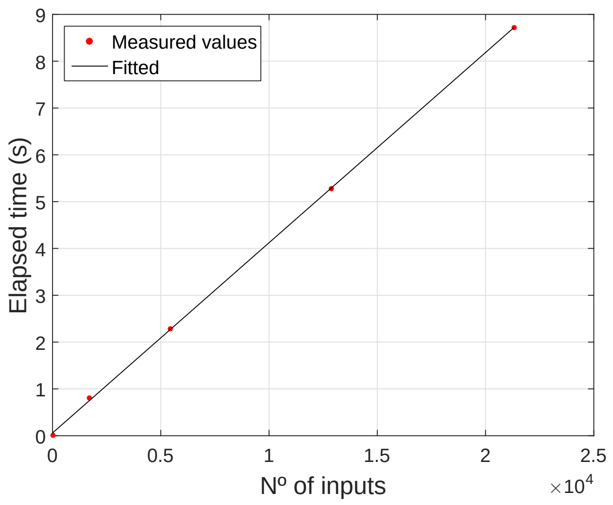


Figure 5.6: Computational cost of the *Post-Process* sensitivity analysis module as a function of the number of inputs

5.7 Summary of the Chain Rule

Since all the jacobians have been derived, the detailed procedure to assemble the chain-rule in Equation (5.9) will now be clarified. The procedure is composed by two major steps. The first corresponds to assembly the intermediate jacobian $\frac{\mathrm{d}\mathbf{x_2}}{\mathrm{d}\mathrm{x_{DV}}}$ and the second corresponds to apply the adjoint method and the chain-rule. The algorithm is as follows:

1. **Assembly of $\frac{\mathrm{d}\mathbf{x_2}}{\mathrm{d}\mathbf{x}_{\mathbf{DV}}}$**

   (a) Use the forward mode of AD applied to the function `wing_geometry.m` to calculate $\frac{\partial \mathbf{WP}}{\partial \mathbf{x}_{\mathbf{DV}}}$. Replace the result in the respective entries in Equations (5.12), (5.13) and (5.14);

   (b) Use Equations (5.15a) up to (5.15c) to construct $\frac{\partial S}{\partial \mathbf{x}_{\mathbf{DV}}}$. Use also Equations (5.16a) and (5.16b) to build $\frac{\partial \mathrm{MAC}}{\partial \mathbf{x}_{\mathbf{DV}}}$. Introduce both vectors in the respective entries in Equation (5.10);

   (c) Follow the procedure presented in section 5.3.1 to calculate the intermediate jacobians $\frac{\partial \mathbf{PP}}{\partial \mathbf{WP}}$, $\frac{\partial \mathbf{CP}}{\partial \mathbf{WP}}$, $\frac{\partial \mathbf{LV}}{\partial \mathbf{WP}}$ and $\frac{\partial DS}{\partial \mathbf{WP}}$. Assembly the jacobian $\frac{\partial \mathbf{x_1}}{\partial \mathbf{WP}}$ according to its definition and replace the results in Equations (5.12) and (5.14). Use also $\frac{\partial DS}{\partial \mathbf{WP}}$ to finish the assembly of $\frac{\mathrm{d}DS}{\mathrm{d}\mathbf{x}_{\mathbf{DV}}}$ in Equation (5.13);

   (d) Follow the procedure presented in section 5.4 to calculate $\frac{\partial \mathbf{LPP}}{\partial \mathbf{PP}}$ and $\frac{\partial \mathbf{LPP}}{\partial \mathbf{CP}}$ using Equation (5.29), and use also Equation (5.30) to calculate $\frac{\partial \mathbf{LPP}}{\partial \mathbf{LV}}$. Construct $\frac{\partial \mathbf{LPP}}{\partial \mathbf{x_1}}$ according to its definition using the results just calculated;

   (e) Replace finally the jacobians $\frac{\mathrm{d}\mathbf{x_1}}{\mathrm{d}\mathbf{x}_{\mathbf{DV}}}$, $\frac{\mathrm{d}DS}{\mathrm{d}\mathbf{x}_{\mathbf{DV}}}$ and $\frac{\mathrm{d}\mathbf{LPP}}{\mathrm{d}\mathbf{x}_{\mathbf{DV}}}$ in the respective entries of $\frac{\mathrm{d}\mathbf{x_2}}{\mathrm{d}\mathbf{x}_{\mathbf{DV}}}$ in Equation (5.10).

2. **Assembly of $\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}_{\mathbf{DV}}}$ according to the chain-rule**

   (a) Follow the procedure presented in section 5.5.1 up to section 5.5.4 to calculate the non zero entries of $\frac{\partial \mathbf{R}}{\partial \mathbf{x_2}}$ according to its definition. Remember that the entries corresponding to $\frac{\partial \mathbf{R}}{\partial \mathbf{PP}}$, $\frac{\partial \mathbf{R}}{\partial DS}$, $\frac{\partial \mathbf{R}}{\partial S}$ and $\frac{\partial \mathbf{R}}{\partial \mathrm{MAC}}$ are null. Next, use Equation (5.41) to calculate $\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}$;

   (b) Use the reverse mode of AD to the function `post_process.m` and obtain $\frac{\partial f}{\partial \mathbf{x_2}}$ and $\frac{\partial f}{\partial \boldsymbol{\mu}}$, where f is an interest function. Remember that the entry corresponding to $\frac{\partial f}{\partial \mathbf{LPP}}$ is null;

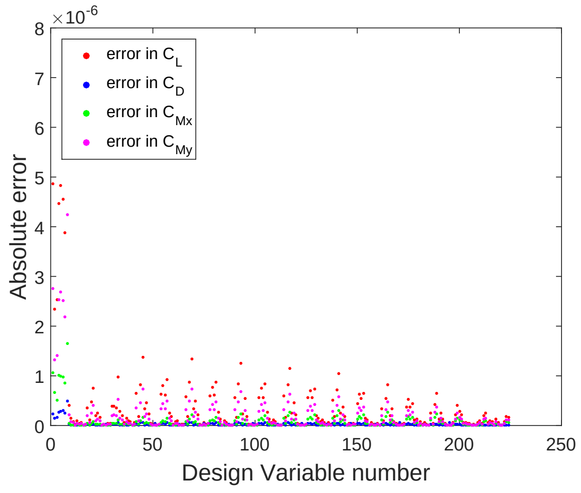
   (c) Apply the first step of the adjoint method by calculating the adjoint matrix $[\psi]$ using Equation (5.8b). Apply the last step of the method introducing $[\psi]$ in Equation (5.8a) to obtain $\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x_2}}$;

   (d) Use the chain-rule of differential calculus to calculate $\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}_{\mathbf{DV}}}$ according to Equation (5.9).

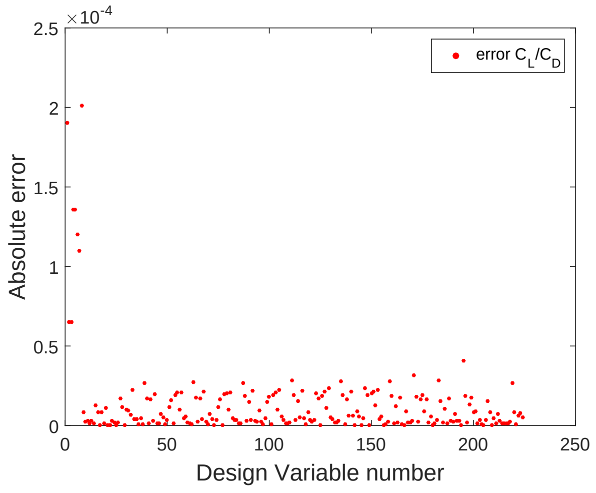
5.8 Final Benchmark with Finite Differences

After the sensitivity analysis framework had been constructed, it was necessary to validate it, thus guarantying the framework was free of programming errors and also measure its performance. To accomplish this task, the resulting sensitivities of the aerodynamic coefficients with respect to the design variables was compared with the same sensitivities calculated using finite-differences being the elapsed time for both frameworks registered in Table 5.5. The parameter t_{model} is the aerodynamic analysis tool run time, t_{sens} is the new sensitivity analysis framework time, and t_{FD} is the measured time using the FD method. It may be observed that the new developed framework becomes more efficient as the number of design variables increases, since the percentage of time saved using the sensitivity analysis framework increases with increasing number of design variables, proving its efficiency for accurate wing discretization.

It should also be shown the absolute error's calculation for the sensitivities using both methods since it provides the necessary confidence in the obtained results. The benchmark procedure started with choosing a suited step size. After some blind attempts trying different step sizes it became evident that $h = 10^{-7}$ was suited. After isolating the aerodynamic analysis tool from its sensitivity analysis, the finite-differences method was applied. The results are presented in Figure 5.7. According to the figures, the sensitivities are calculated correctly since the absolute error is about $\mathcal{O}(10^{-6})$ for the values associated with C_L, C_D, C_{M_x} and C_{M_y} and about $\mathcal{O}(10^{-4})$ for the aerodynamic efficiency $\frac{C_L}{C_D}$.



(a) Absolute error of the aerodynamic coefficients sensitivities w.r.t. $\mathbf{x_{DV}}$, benchmarked with FD



(b) Absolute error of the aerodynamic efficiency $\frac{C_L}{C_D}$ sensitivities w.r.t. $\mathbf{x_{DV}}$, benchmarked with FD

Figure 5.7: Benchmark of the developed sensitivity analysis framework with the finite-differences method

No. DV	32	80	128	224	320	392
t_{model} [s]	0.037	0.083	0.239	0.876	1.946	3.037
t_{sens}/t_{model} [-]	53.69	41.65	34.56	27.77	25.07	24.01
t_{FD}/t_{model} [-]	18.31	61.47	120.07	224.78	316.24	390.19
savings [%]	-193.2	32.2	71.1	87.6	92.1	93.8

Table 5.5: Computational cost benchmark between the sensitivity analysis framework and the finite-differences method

Chapter 6

Parametric Study

This chapter is devoted to survey the impact of changing both the wing's geometrical characteristics and flight condition on the wing's aerodynamic behavior. First, a baseline wing configuration will be chosen providing a representative benchmark for the new wing configurations. A convergence study is also conducted to find a suitable mesh to obtain results. Special effort will be made to conclude about the impact on the aerodynamic efficiency $\frac{C_L}{C_D}$ on the pitching and rolling moments C_{M_y} and C_{M_x}, about a reference point located at the root chord, in the trailing edge of the baseline wing configuration. The main reason behind it is related with the importance of those parameters on aircraft performance, structural safety and flight equilibrium.

- **Aerodynamic Efficiency**: Directly related with performance measures such as maximum range and maximum flight time;
- **Rolling Moment about x axis**: Directly related with the bending moment applied on the wing's root, which is related with the normal stress applied. For safety reasons, this may have to be no higher than a certain value;
- **Pitching Moment about y axis**: Directly related with the moment produced about the aircraft CG. For operational reasons, this value may have to be constant or no higher than a certain value.

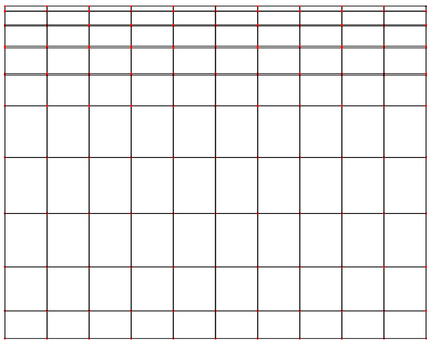
For benchmark purposes, a plane rectangular wing with $AR = 6$ and constant airfoil section was chosen. The geometrical characteristics and flight condition are presented in Table 6.1.

$\Lambda\,[^\circ]$	$\Gamma\,[^\circ]$	$\delta_r\,[^\circ]$	$\delta_t\,[^\circ]$	$b\,[\mathrm{m}]$	$c_r\,[\mathrm{m}]$	λ	section airfoil	$V[\mathrm{m/s}]$	$\alpha\,[^\circ]$
0	0	0	0	6	1	1	NACA 0010	75	5

Table 6.1: Baseline wing configuration for the parametric study

6.1 Convergence Study

To find a suitable mesh, a convergence study was conducted for the baseline wing configuration. The aerodynamic model was executed several times for increasing number of panels measuring the values of the aerodynamic coefficients. The coarser and finer meshes are depicted in Figure 6.1. The convergence behavior of the outputs C_L and C_D, and C_{M_x} and C_{M_y} was measured and it is presented in Figure 6.2 (a) and 6.2 (b), respectively. All the values are converged in 3 decimal points. Since the solver takes about 30 minutes to run in the finer mesh, all the following studies are carried in a courser


(a) Coarser mesh

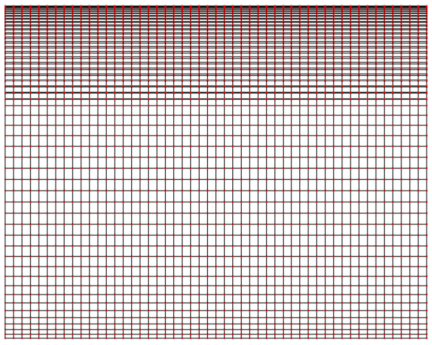
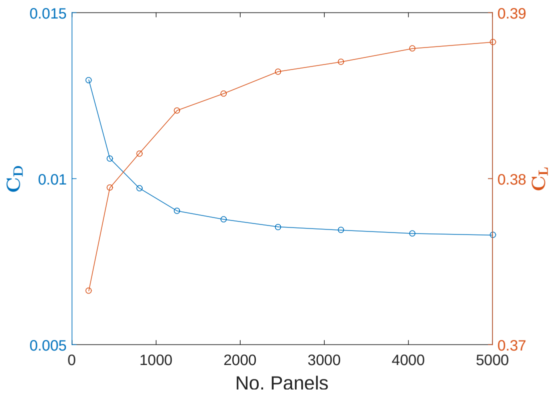
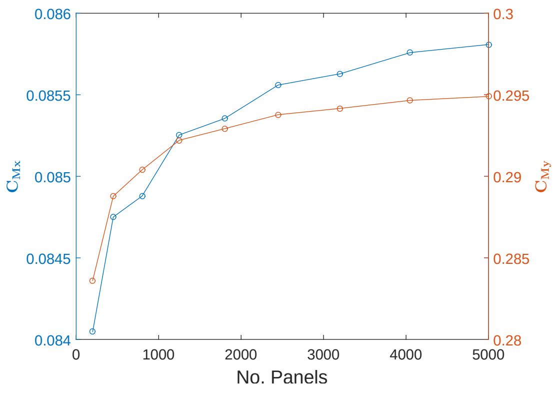

(b) Finer mesh

Figure 6.1: Top view of the baseline wing configuration for the coarser and finer meshes

mesh, 50×10 panels, with an error no larger that 10% relatively to the best results for all the aerodynamic coefficients.



(a) Values of C_L and C_D for increasing number of panels



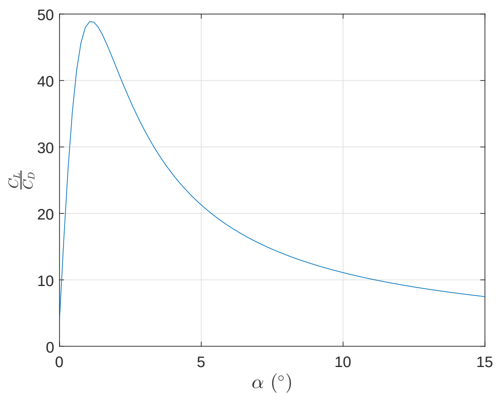
(b) Values of C_{M_x} and C_{M_y} for increasing number of panels

Figure 6.2: Convergence of the aerodynamic coefficients

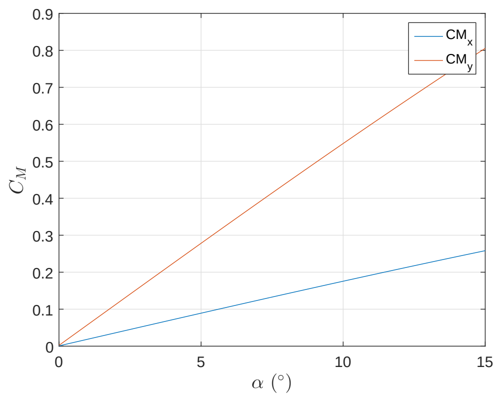
6.2 Angle of Attack

First, the angle of attack effect on the aerodynamic coefficients is explored for the baseline wing configuration. Figure 6.3 (a) shows the evolution of the aerodynamic efficiency $\frac{C_L}{C_D}$, as a function of the angle of attack. According to this picture, the aerodynamic efficiency increases from zero to about 49 for an optimal angle of attack of $\alpha_{opt} = 1°$ and then it smoothly decreases for higher values of α. Thus, one may quickly conclude that the angle of attack has a great impact on the aerodynamic efficiency and that a given value for the angle of attack that maximizes the aerodynamic efficiency exists.

The angle of attack has also an impact on the moment coefficients, as it can be observed in Figure 6.3 (b). According to the picture, it can be observed that moment coefficients increase with the angle of attack. The reason is because the aerodynamic forces increase with the angle of attack leading the moments about the reference point to increase.



(a) Aerodynamic efficiency variation with the angle of attack



(b) Moment coefficent variations with the angle of attack

Figure 6.3: Variation of the aerodynamic coefficients with the angle of attack

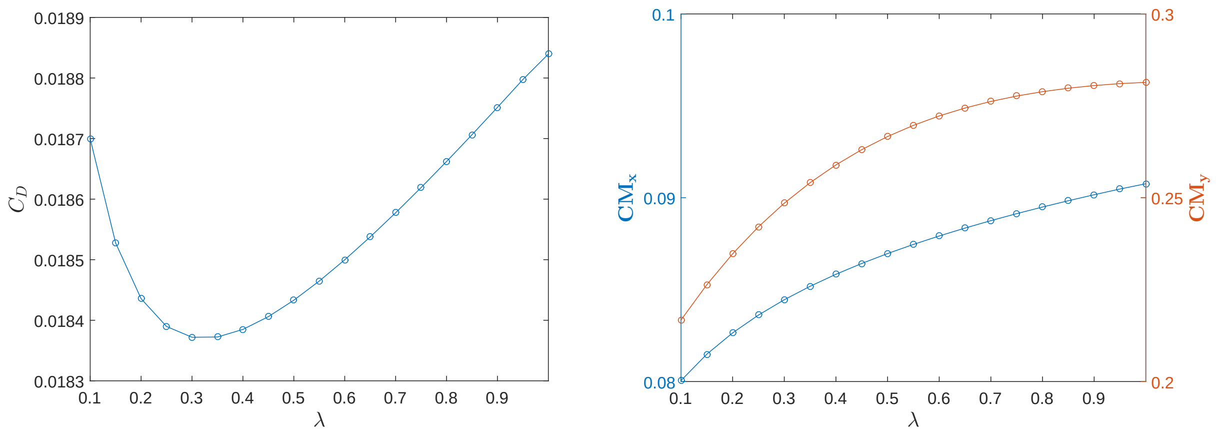
6.3 Taper Ratio

The influence of the taper ratio in the relevant aerodynamic coefficients will now be presented. To evaluate the effect of the taper ratio, one may force the wing's to work at the same lift coefficient and with the same aspect ratio. Additionally, the wing span b will also be constrained to the baseline value. This situation corresponds to evaluate the taper ratio's effect on the Oswald efficiency factor with the constraint

$$c_r = \frac{const}{1+\lambda} \tag{6.1}$$

Figure 6.4 presents the taper ratio effect on the drag and moment coefficients. Considering first the evolution of the drag coefficient in Figure 6.4 (a), it is observed that for a decreasing value of taper ratio, the drag coefficient decreases to about $\lambda = 0.35$ and then it increases. Therefore, one may conclude there must exist some value of taper ratio which minimizes C_D for a given operating C_L. Considering now Figure 6.4 (b), it is observed that the moment coefficients decreases with decreasing taper ratio. The decreasing tendency of the pitching moment is because as the taper ratio decreases, the root chord increases and therefore the streamwise component of the center of pressure is closer to the reference point. The decreasing tendency of the bending moment is due to the increase of the lift per unit span near the wing's root for decreasing taper ratio.

It is possible to reach the same conclusion observing Figure 6.5 where the normalized spanwise lift distribution is plotted for different taper ratios. The elliptical lift distribution is also depicted for reference. Comparing with the latter, it may observed that the load distribution increases near the wing root and, decreases near the wing tip, when the taper ratio is diminished. Thus, for small taper ratio, more lift is being generated near the root and therefore, weaker bending moments are present at the wing root. It is also easy to understand why the drag coefficient is smaller for $\lambda = 0.35$ since the lift distribution curve for that wing configuration is the closest to a wing with an elliptical lift distribution.



(a) Variation of the drag coefficient with the taper ratio

(b) Variation of the moment coefficients with the taper ratio

Figure 6.4: Variation of the aerodynamic coefficients with the taper ratio

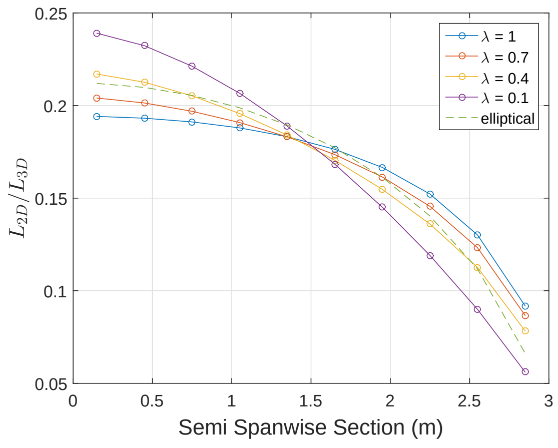


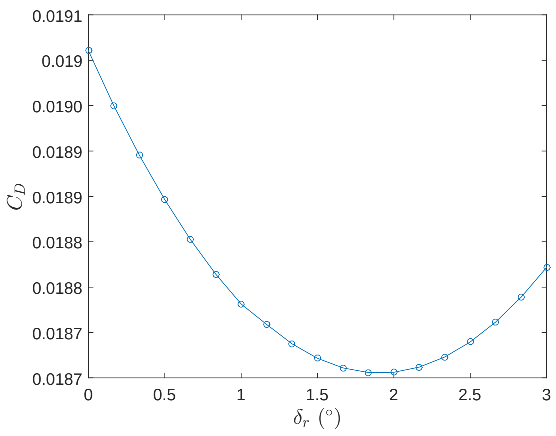
Figure 6.5: Normalized spanwise lift distribution for different taper ratios

6.4 Twist Distribution

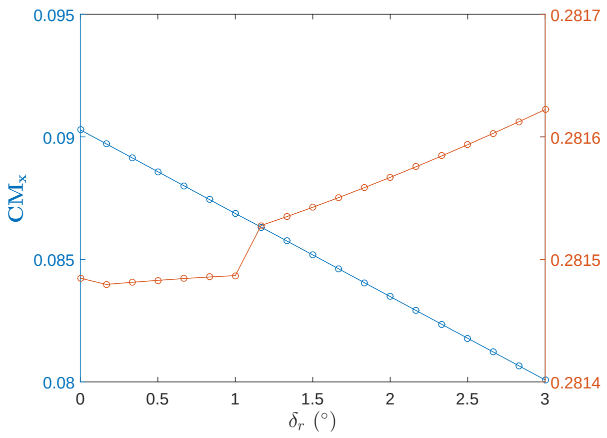
Next, the influence of the twist distribution is evaluated. Twisted wings are obtained through applying twist to the wing's root and tip. For this study, the tip twist is imposed to be symmetrical to the root twist angle, for all wing configurations, with a linear variation in between.

Figure 6.6 (a) and 6.6 (b) presents the variation of the aerodynamic coefficients with the wing twist, for a fixed lift coefficient. Observing Figure 6.6 (a) one may observe that the drag coefficient is minimum about $\delta_r = 1.9°$ and $\delta_t = -1.9°$. Considering now Figure 6.6 (b), it is observed that the pitching moment is insensible to the wing twist and the bending moment decreases with increasing twist. Considering the overall results, one may conclude that the twist and taper ratio effects are quite similar, except for the pitching moment. This is intuitively explained since the local twist angle changes the local angle of attack, adjusting the lift being generated at each wing section, similarly to the taper ratio effect.

An additional comment should be made about the wing's twist distribution. The choice of keeping the root's twist angle higher than the tip's was intentional. Typically, it is desirable to work with negative torsion to guarantee the wing tips are the last wing part to stall. Thus, even if the wing's root begins to lose lift, the pilot is still able to control the aircraft once the ailerons are located near the wing tips.



(a) Variation of the drag coefficient with the wing twist

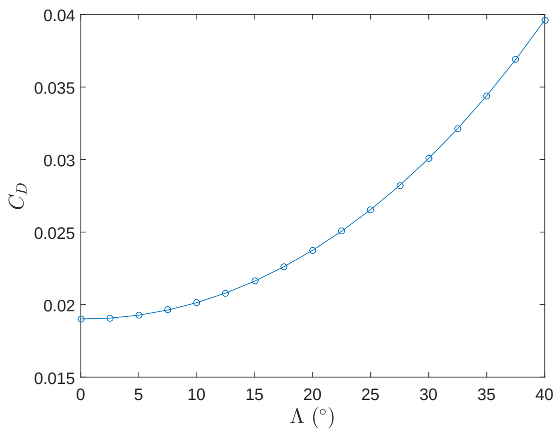


(b) Variation of the moment coefficients with the wing twist

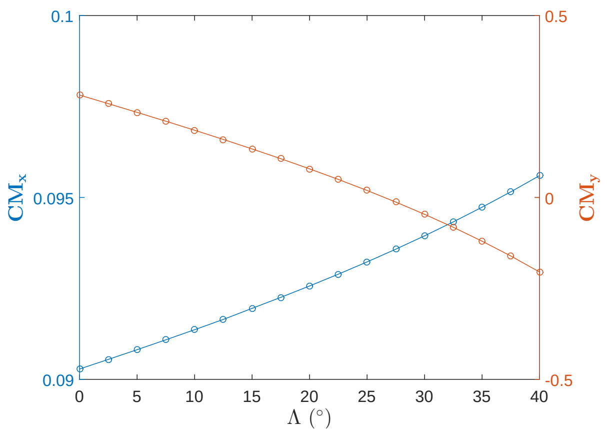
Figure 6.6: Variation of the aerodynamic coefficients with the wing twist

6.5 Sweep Angle

Next, the effect of the sweep angle on the aerodynamic coefficients is studied. The parametric study is similar to the previous cases studies where the lift coefficient was kept constant. Departing from the baseline configuration, the values of drag and moment coefficients were tracked for successively increasing sweep angles. These results are presented in Figure 6.7 (a) and 6.7 (b). All the wings present the same area and aspect ratio. It can be observed that increasing the sweep angle results in an increase in the aerodynamic drag and rolling moment and a decrease in the pitching moment. The decrease of the latter is easily explained because increasing the sweep angle shifts the center of pressure backwards. The increase in drag and rolling moment may be explained through Figure 6.8. According to the figure, increasing the sweep angle increases the wing loading near the wing tips, thus the lift distribution is deviated from the elliptical reference case.



(a) Variation of the drag coefficient with the wing sweep



(b) Variation of the moment coefficients with the wing sweep

Figure 6.7: Variation of the aerodynamic coefficients with the wing sweep

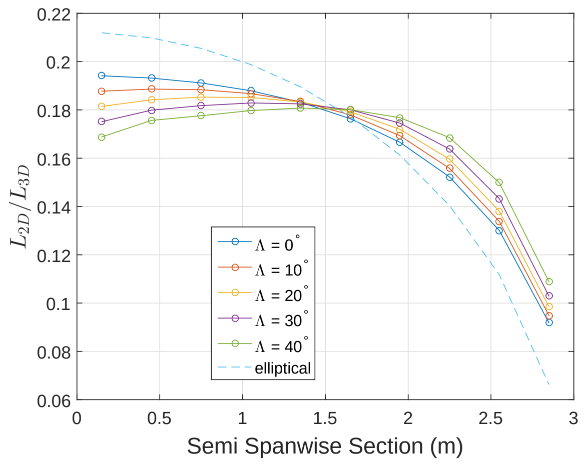


Figure 6.8: Normalized spanwise lift distribution for different sweep angles

6.6 Dihedral Angle

Next, the impact of varying the dihedral angle on the aerodynamic coefficients will be studied. The new wing configurations are obtained from the baseline wing configuration keeping all the parameters equal except the dihedral angle. Therefore, the projected xy wing area is the same for all test cases and is used as reference. The study was carried again at constant lift-coefficient, where the remaining aerodynamic coefficients were tracked as a function of the dihedral angle. These results are presented in Figure 6.9 (a) and (b). According to Figure 6.9 (a), C_D is optimal for a dihedral angle near $17°$. Brederode [66] justifies this trend stating the Munk's second theorem, which shows that non-planar wings can be more efficient than elliptical ones. Such wings can be approximated by employing dihedral. Considering now Figure 6.9 (b), one may conclude that the pitching moment coefficient decreases increasing the sweep angle and the reverse situation is observed for the rolling moment coefficient .

Brederode [66] also states that dihedral is not primarily used to enhance aerodynamic efficiency but to provide rolling stability. He also recommends dihedral angles no higher than $5°$. For this reason, the dihedral will be bounded between $0°$ and $5°$.

6.7 Airfoil Section

This section studies the airfoil shape effects on the aerodynamic coefficients. The baseline wing configuration was kept the same as in the previous case studies. Two tests were conducted.

In the first study, the effect of airfoil camber was evaluated. According to Brederode [66], it is also possible to obtain an elliptical lift distribution for maximum aerodynamic efficiency applying different airfoil shapes, with different cambers along the wing's span. In an attempt to produce such wing configurations, four wings were tested, where the camber varies linearly along the wing span, being the higher cambered airfoil located at the wing root. The airfoil shape at the tip was kept uncambered, with the same shape as the baseline. The airfoil shapes at the root and the baseline may visualized in Figure 6.10 (a). As it can be seen in Table 6.2, increasing the airfoil root camber results in a drag and rolling moment coefficients

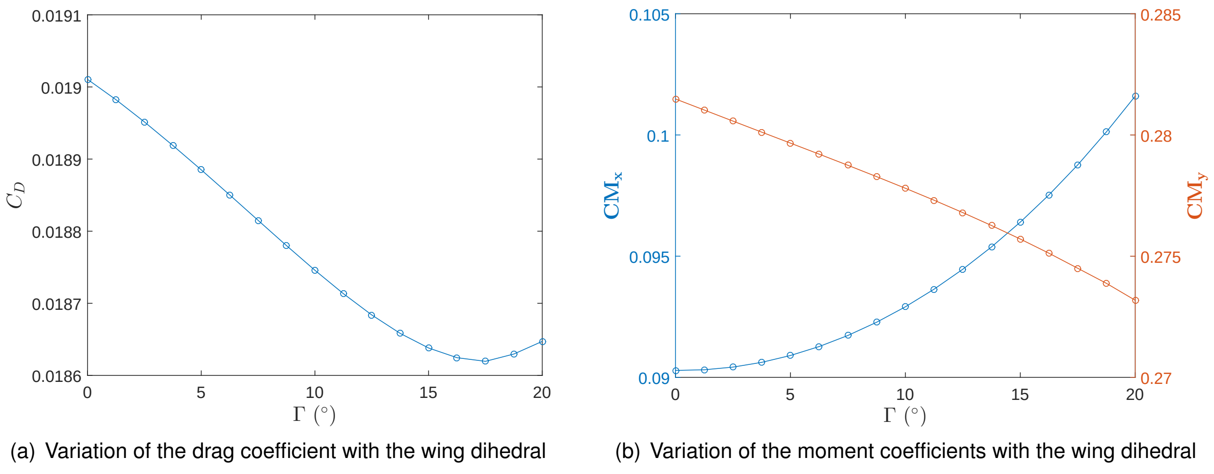


Figure 6.9: Variation of the aerodynamic coefficients with the wing dihedral

reduction. This behavior is quite similar to the taper ratio effect on the same two variables, thus one may conclude about an approximation to the elliptical lift distribution.

The second study focus on the effect of airfoil thickness in the aerodynamic coefficients. Two additional wing configurations were tested in which the airfoil shape was kept constant along the wing span, differing only on the airfoil thickness. To accomplish that, two symmetrical NACA 0015 and NACA 0018 were simulated, respectively. Their relative size and shape may be found in Figure 6.10 (b). The results are shown in Table 6.3. As it can be seen, the increased thickness translate into a significant drag increase, for the same lift. Also, it can be observed that the rolling moment does not change and the pitching moment is reduced. The rolling moment stays the same since no spanwise lift redistribution is made. On the other hand, the pitching moment is reduced since a reduction in the angle of attack was required to obtain the same lift-coefficient.

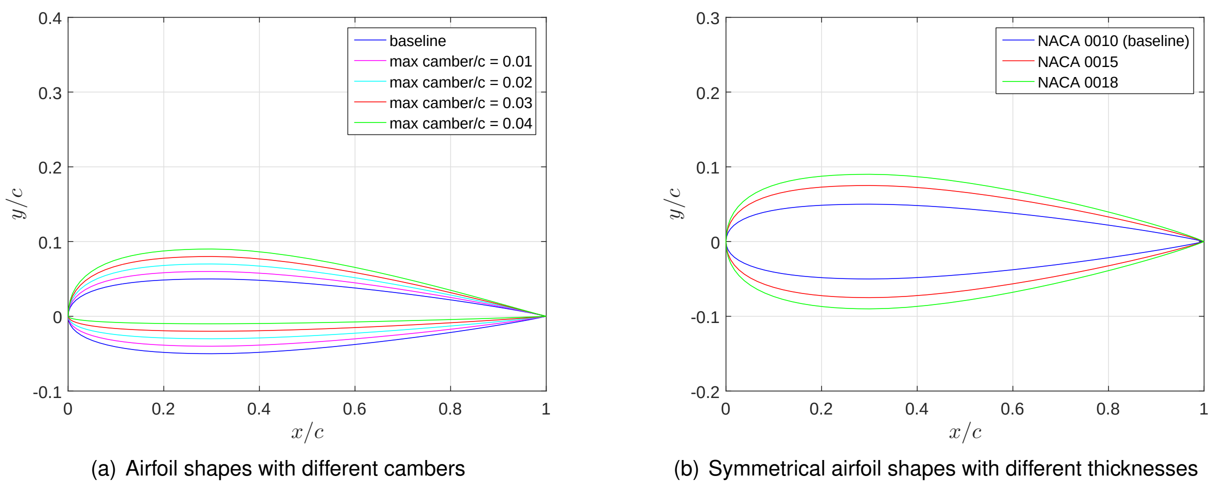


Figure 6.10: Airfoil shapes used for parametric study

max camber	$\alpha\,(^\circ)$	C_L	C_D	C_{M_x}	C_{M_y}
0	5.061	0.4000	0.0190	0.0903	0.2815
0.01	4.602	0.4000	0.0183	0.0888	0.2771
0.02	4.141	0.4000	0.0178	0.0874	0.2728
0.03	3.678	0.4000	0.0175	0.0859	0.2686
0.04	3.211	0.4000	0.0173	0.0844	0.2643

Table 6.2: Effect of airfoil camber on the aerodynamic coefficients, operating at the same lift-coefficient

airfoil	$\alpha\,(^\circ)$	C_L	C_D	C_{M_x}	C_{M_y}
NACA 0010	5.061	0.4000	0.0190	0.0903	0.2815
NACA 0015	4.886	0.4000	0.0223	0.0902	0.2721
NACA 0018	4.786	0.4000	0.0242	0.0902	0.2670

Table 6.3: Effect of airfoil thickness on the aerodynamic coefficients, operating at the same lift-coefficient

6.8 Remarks on Wing Parameters

It is now necessary to conclude about which parameters should be used as design variables in the aerodynamic optimization problems that will be addressed in Chapter 7. In these problems, one is typically interested in reducing the induced drag for a given target lift with eventual constraints in the moment coefficients. That task can be achieved mainly by increasing the aspect ratio and maximizing the Oswald efficiency factor. For this reason, parameters that affect these quantities should be used as design variables.

In an attempt to solve the described type of aerodynamic design problems, the angle of attack will be used as design variable since it is the main mechanism to adjust the lift and moment coefficients. Moreover, it was shown in the respective parametric study that an optimal value that maximizes the aerodynamic efficiency exists, although it will not be attempted to find it. The aspect ratio may be increased by simply increasing the wing span and decreasing the root chord, therefore these parameters will be also used as design variables. Although a decrease in the taper ratio also increases the aspect ratio, this parameter plays an important role in the adjustment of the spanwise lift distribution which can be used to minimize the drag coefficient. As it was also shown, if the wing twist is chosen appropriately, the same goal can be achieved and therefore it is indifferent to operate with the taper ratio or wing twist. Nevertheless, since the wing twist does not change the planform shape, it does not influence much the pitching moment as opposed to the taper ratio. Due to these arguments two optimization problems will be solved using each of these parameters separately. The influence of the airfoil shape in the aerodynamic coefficients is up to some degree unknown since it is modeled using 24 degrees of freedom per cross section as explained in section 4.4.1. Nevertheless, it can be stated with confidence that exists a combination of camber and thickness that minimizes the wing drag coefficient by maximizing the Oswald efficiency factor. An optimization problem will also be solved to explore the effect of the airfoil shape in the wing drag.

Chapter 7

Wing Aerodynamic Optimization

This chapter presents the benefits of gradient-based optimization with efficient gradient estimation compared with the traditional approach of forward finite-differences to sensitivity analysis. Towards that goal, two representative aerodynamic optimization problems are solved using both approaches. A third problem is presented in Appendix B for completeness, similar to the first. For each problem, the optimization time, the number of iterations and function evaluations are compared using both implementations. In the first problem, a small subset of design variables are selected from all the possible available to minimize the wing's drag coefficient, at constant lift and wing area, subject to bounds. In the second problem, all available design variables are selected, but in addition to the first, the pitching moment coefficient was kept fixed.

The mainstream data flow in the aerodynamic optimization problem is well illustrated in Figure 7.1, where the connection between the reformulated aerodynamic model, the new sensitivity analysis framework and the optimizer is depicted. First, some design parameters from the baseline configuration are chosen, corresponding to a point in the design space. Then, both aerodynamic information and respective sensitivities are generated and provided to the optimizer. Finally, the optimizer uses this information to find a search direction in the design space and an acceptable step size so a new improved design is obtained. The process in then repeated until the Karusch-Kuhn-Tucker optimality conditions are satisfied.

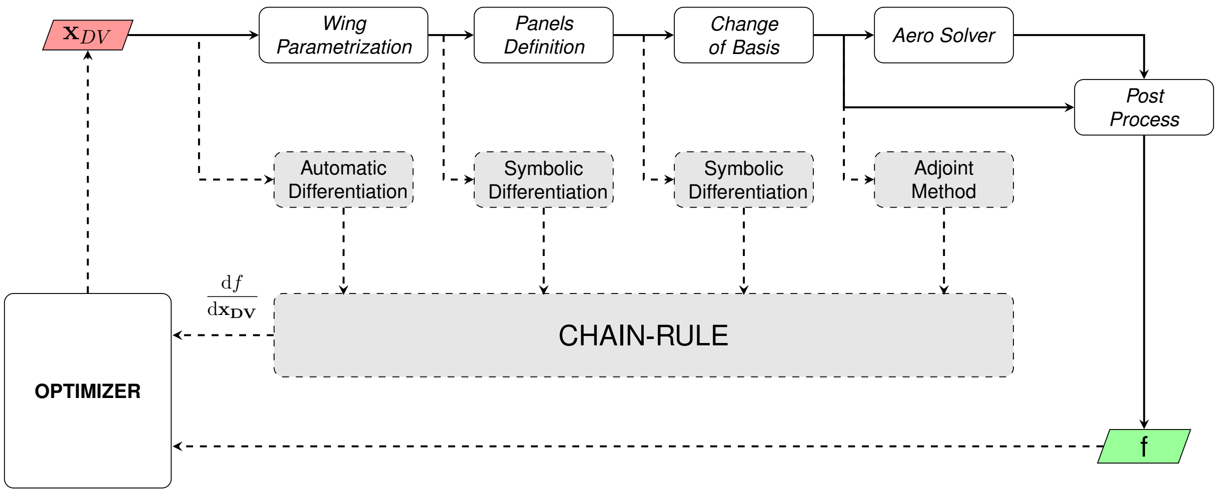


Figure 7.1: Flowchart illustrating the aerodynamic optimization framework

7.1 Optimizer

An important step related with the aerodynamic optimization is the choice of the optimizer. For this effect, MATLAB® provides a powerful toolbox called *Optimization Toolbox* [67], which allows the user to solve a vast set o problems, namely linear programming (LP), mixed-integer linear programming (MILP), quadratic programming (QP), nonlinear programming (NLP), among other solvers. For the present case of nonlinear programming with constraints, MATLAB® provides a function called `fmincon`. The nonlinear programming mathematical statement is slightly different, although equivalent from the one expressed in Equation (2.1). Thus, `fmincon` requires the problem to be formulated as

$$\begin{array}{lll} \text{minimize} & f(\mathbf{x}) & \\ \text{w.r.t.} & \mathbf{x} & \\ \text{subject to} & [A]\mathbf{x} \leq \mathbf{b}, & [A_{eq}]\mathbf{x} = \mathbf{b_{eq}} \\ & \mathbf{c}(\mathbf{x}) \leq 0 & \mathbf{c_{eq}}(\mathbf{x}) = 0 \\ & \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U & \end{array} \tag{7.1}$$

After formulating the problem in this manner, `fmincon` accepts any combination of constraints. In order to deal with linear equality and/or inequality constraints, the respective matrices and vectors should be provided. Both nonlinear objective and constraint functions should be supplied to `fmincon` through a function handle to their respective MATLAB® functions. Additionally, if the gradient is calculated externally by the user, it should be provided as a second output in those functions.

Several optimization related options may also be provided to the solver through a structure array. To the effect, MATLAB® provides a function called `optimoptions`, which creates that array according to the user preferences. Examples of available options for `fmincon` are the optimization algorithm, the maximum number of function evaluations and iterations, first-order optimality measure and the option to provide analytical gradients for objective and constraints.

MATLAB® R2015a version provides four different optimization algorithms for `fmincon`:`'active-set'`, `'sqp'`, `'interior-point'` and `'trust-region-reflective'`. The method `'trust-region-reflective'` is immediately discarded since it only accepts problems with bounds or linear equality constraints, but not simultaneously. For medium size problems, MATLAB® recommends to start first with `'interior-point'` and lastly with `'active-set'`. After some informal test cases in a course mesh, the method `'sqp'` proved to be the best suited since it converged faster than the other methods, therefore, the method was kept the same for all the remaining optimization problems. Since both gradient and function evaluations are expensive computationally, the first order optimality condition was changed to 10^{-4} against the default value of 10^{-6}.

7.2 Wing Planform Optimization

The first optimization problem corresponds to minimize the wing drag coefficient with respect to the design variables, subject to constant lift and wing area, and bounds. The design variables correspond to a small subset from all the design parameters available. They correspond to planform related variables: wing span b, root chord c_r and taper ratio λ, and also the angle of attack α. With this in mind, this optimization problem may be casted as

$$\begin{aligned} &\text{minimize} && C_D \\ &\text{w.r.t.} && \mathbf{x_{DV}} \\ &\text{subject to} && C_L = 0.3, \quad S = S_0 \\ & && \mathbf{x}^L \leq \mathbf{x_{DV}} \leq \mathbf{x}^U \end{aligned} \tag{7.2}$$

where S_0 is the baseline wing area. After posing the problem in mathematical terms, it is necessary to define the baseline wing configuration and flight condition. The former was kept the same as in Chapter 6 but, the angle of attack has changed though. Both baseline wing configuration and flight condition are tabulated in Table 7.1.

$\Lambda\,[^\circ]$	$\Gamma\,[^\circ]$	$\delta_r\,[^\circ]$	$\delta_t\,[^\circ]$	$b\,[\mathrm{m}]$	$c_r\,[\mathrm{m}]$	λ	section airfoil	$V[\mathrm{m/s}]$	$\alpha\,[^\circ]$
0	0	0	0	6	1	1	NACA 0010	75	4

Table 7.1: Baseline wing configuration to the first optimization problem

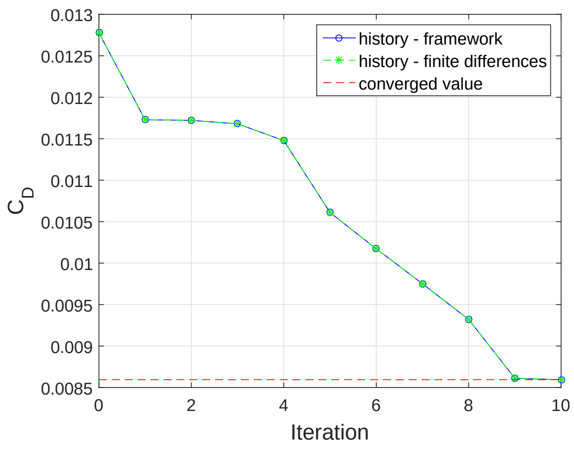
A baseline plane wing was chosen as the initial design and the resulting optimized design will also be a plane wing since only planform related design variables will change. This imposition could be important if it is desired an easy to build wing. The bounds and constraints values were chosen arbitrarily although care was taken to keep the problem realistic. Table 7.2 summarizes both the bounds and initial design vector for the current problem.

Design Vector ($\mathbf{x_{DV}}$)	Lower Bound ($\mathbf{x}^L$)	Initial Design ($\mathbf{x_{DV_0}}$)	Upper Bound ($\mathbf{x}^U$)
$\alpha\,[^\circ]$	1	4	10
$b\,[\mathrm{m}]$	5	6	8
$c_r\,[\mathrm{m}]$	1	1	3
$\lambda\,[-]$	0.1	1	1

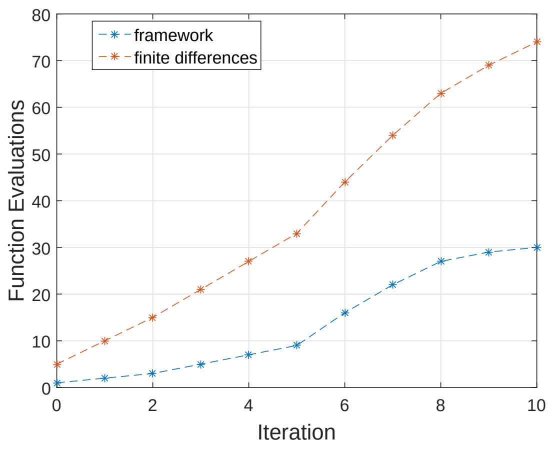
Table 7.2: Initial values of the design vector and respective bounds for the first optimization problem

After both the problem and baseline wing configuration have been defined, the optimization takes place. Figure 7.2 (a) shows the objective function history as a function of the iteration number, showing that the optimization is concluded after 11 iterations, using either finite-differences or the developed sensitivity analysis framework to estimate the gradient information required by the optimizer. In addition to the number of iterations, also the number of function evaluations and computational time were tracked and benchmarked. A summary of these results are presented in Table 7.3. Considering the optimization times, it can be observed that using finite-differences was about 3 times faster than using the differentiation tool. Nevertheless, the number of function evaluations were about 2.5 times higher. The cumulative number of function evaluations are represented in Figure 7.2 (b), using both FD and the sensitivity framework. Since the minimum number of cumulative function evaluations may be given by

$(N_{DV}+1)(i+1)$ for FD, and $i+1$ for the sensitivity framework, where N_{DV} is the number of design variables and i is the iteration number as given in Figure 7.2 (b), one may conclude that the optimizer spent 20 and 24 additional function evaluations using FD and the differentiation tool, respectively, during line-search. Moreover, the same conclusion could be reached noticing that the plots in Figure 7.2 (b) are not straight lines.



(a) Convergence history of C_D



(b) Function evaluations

Figure 7.2: Convergence process and number of function evaluations during the first optimization problem

These results show that the new sensitivity analysis framework is less efficient when dealing with a small number of design variables. This conclusion could be inferred through the results of Table 5.5. As it will be seen next, that is not the case when considering all the design variables available.

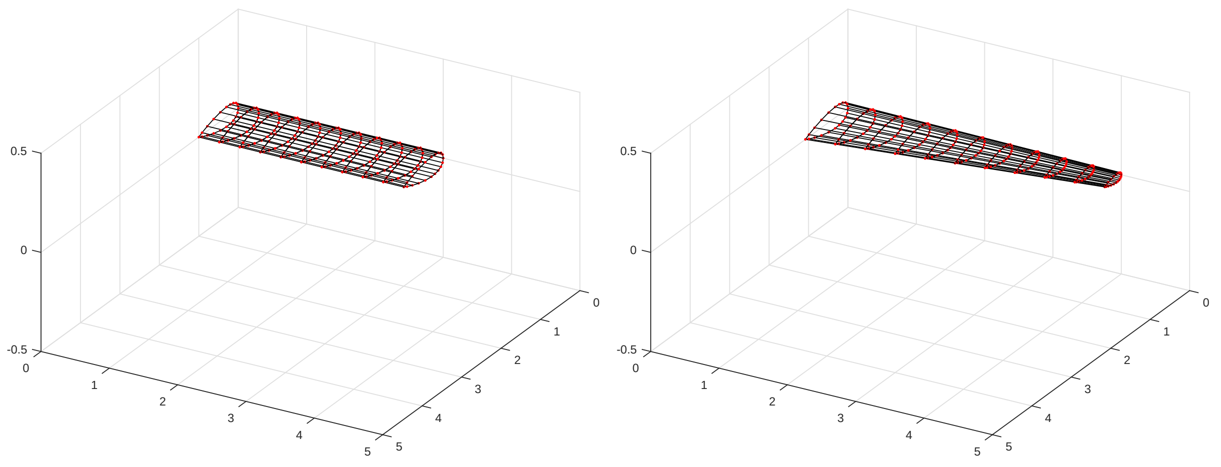
Gradient Calculation Method	Time [s]	Iterations	Function Evaluations
Sensitivity Framework	4681.8	11	31
Forward Finite Differences	1314.7	11	79

Table 7.3: Benchmark of the first optimization case performance between different sensitivity analysis methods

Table 7.4 presents the initial and optimized values for both the design vector, output functions, wing area and aspect ratio, using both forward finite-differences and the sensitivity framework. The results were obtained with a 40×10 panels mesh, in the chordwise and spanwise directions, respectively. Considering the accuracy of the obtained results, it is observable that, in worst case scenario, the taper ratio and root chord values obtained with finite-differences differ from the values obtained using the framework at the fifth decimal place. All the other results obtained with FD match the ones obtained with the developed framework. Considering the resulting wing drag coefficient, a reduction of about 33% is observed in comparison to the baseline wing. That reduction was possible through an angle of attack decrease, a wing span increase (bound limited), a slight increase in the root chord and substantial decrease in the taper ratio. Additionally, the drag coefficient reduction comes along with an increase in the pitching moment coefficient and a slight reduction in the rolling moment coefficient. Similarly to the parametric study, the reference point is located at the trailing-edge, at the baseline configuration root chord. The reduction in the angle of attack was necessary to adjust the value of C_L and, the combined change of the planform design variables was such that it produced a substantial increase in the aspect ratio and Oswald efficiency. A geometrical comparison between the baseline and optimized configuration is shown in Figures 7.3 (a) and 7.3 (b).

Design variables	Baseline	Optimized FW	Optimized FD	Outputs	Baseline	Optimized FW	Optimized FD
$\alpha\,[^\circ]$	4	3.177316	3.177317	C_D	0.012777	0.008595	0.008595
$b\,[\mathrm{m}]$	6	8.000000	8.000000	C_L	0.314576	0.300000	0.300000
$c_r\,[\mathrm{m}]$	1	1.079303	1.079325	C_{M_x}	0.071026	0.064134	0.064133
λ	1	0.389785	0.389757	C_{M_y}	0.221416	0.287423	0.287421
				S	6.000000	6.000000	6.000000
				$Æ$	6.000000	10.66667	10.666667

Table 7.4: Baseline, optimized design vector and output values in the first optimization problem



(a) Baseline wing (b) Optimized wing

Figure 7.3: Geometrical comparison between wing configurations in the first problem

7.3 Full Wing Optimization

The second optimization problem is similar to the first except that all the design parameters available were used and the pitching moment coefficient was forced to be equal to the baseline value. The design vector is defined as in section 5.1.1. In addition to the nonlinear equality constraints found in the first problem, the pitching moment was forced to be fixed to simulate cruise conditions, where the trim condition and required lift are specified since the dynamic pressure is assumed constant during the optimization. Thus, the problem may be mathematically expressed as

$$\begin{aligned} &\text{minimize} && C_D \\ &\text{w.r.t.} && \mathbf{x_{DV}} \\ &\text{subject to} && C_L = 0.3, \quad S = S_0, \quad C_{M_y} = C_{M_{y0}} \\ &&& \mathbf{x_{airfoil}} \in \Omega_{eq}, \quad \mathbf{x}^L \leq \mathbf{x_{DV}} \leq \mathbf{x}^U \end{aligned} \tag{7.3}$$

where the $\mathbf{x_{airfoil}} \in \Omega_{eq}$ stands for the equality constraints imposed to the bezier control points. These constraints are the same as in section 4.4.1 and they are suggested by the green lines, in Figure 7.4. The bounds may be defined in terms of the design vector partition:$[\alpha\ \mathbf{x_{geo}} \,|\, \mathbf{x_{airfoil}}]$. Table 7.5 provides these bounds for $[\alpha\ \mathbf{x_{geo}}]$ and Figure 7.4 shows how the airfoil control points were bounded. As already stated in Chapter 4, those bounding boxes are required to produce acceptable airfoil shapes. The box is centered at the baseline control point. The upper and lower edge ordinates are given by $P_y \pm 0.3P_y$, where P_y is a generic control point ordinate. The abscissas of the side edges are given by $P_x \pm 0.01$, $P_x \pm 0.02$ and $P_x \pm 0.03$ for the red, blue and black boxes, respectively, where P_x is the generic control point abscissa.

	$\alpha\,[^\circ]$	$\Lambda\,[^\circ]$	$\Gamma\,[^\circ]$	$\delta_r\,[^\circ]$	$\delta_t\,[^\circ]$	$b\,[\mathrm{m}]$	$c_r\,[\mathrm{m}]$	λ
$\mathbf{x}^L$	1	0	0	0	-3	5	1	0.1
$[\alpha\ \mathbf{x_{geo}}]$	4	0	0	0	0	6	1	1
$\mathbf{x}^U$	10	30	5	3	0	8	3	1

Table 7.5: Bounds and design vector for α and $\mathbf{x_{geo}}$

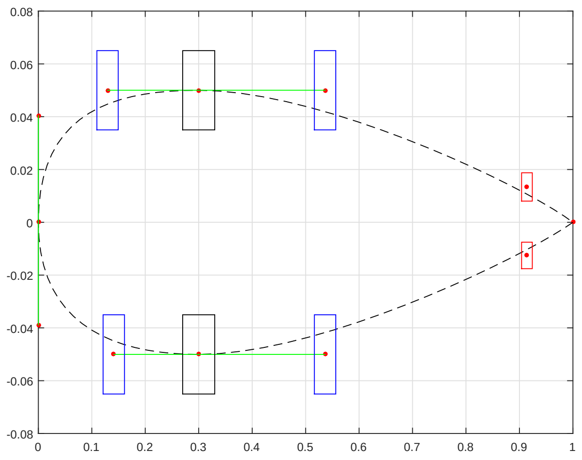
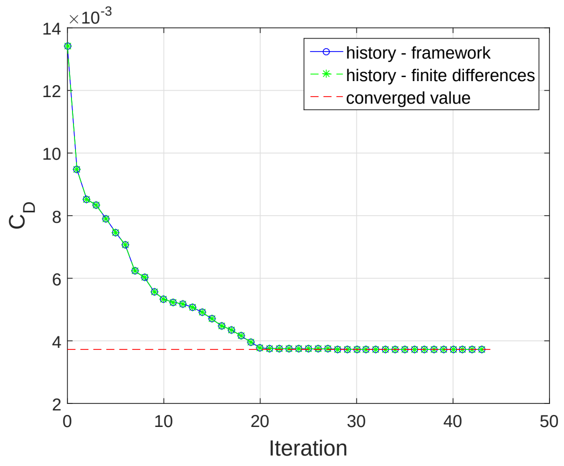
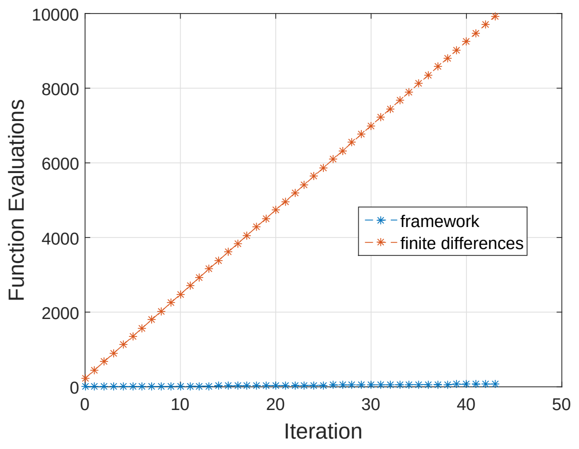


Figure 7.4: Bounding boxes of the airfoil control points

After stating the problem, the baseline wing configuration and respective flight condition must be defined and it was chosen to be the same as in the first optimization problem (Table 7.1). Similarly to the first problem, the optimization was performed using both the sensitivity analysis framework and forward finite-differences. Again, these problems were compared in terms of number of iterations, number of function evaluations and optimization time. A summary of these results are presented in Table 7.6. The objective function history was tracked for both optimizations and it is represented in Figure 7.5 (a). As depicted, the objective function had converged almost identically in 44 iterations, regardless of the method to estimate the sensitivities. However, the number of function evaluations using finite-differences is much higher than the same number using the new framework, as illustrated in Figure 7.5 (b). A direct consequence of those many function evaluations was an increase in the optimization time. As observed in Table 7.6, one may realize that the optimization using the new differentiation tool was about 9 times faster than the one using finite-differences. Using the new sensitivity analysis framework has provided a great advantage in terms of computational time since it took approximately 2 hours as opposed to the 19 hours optimization using FD.



(a) Convergence history of C_D



(b) Function evaluations

Figure 7.5: Convergence process and number of function evaluations during the second optimization problem

Gradient Calculation Method	Time [s]	Iterations	Function Evaluations
Sensitivity Framework	7607.1	44	75
Forward Finite Differences	68726.9	44	10155

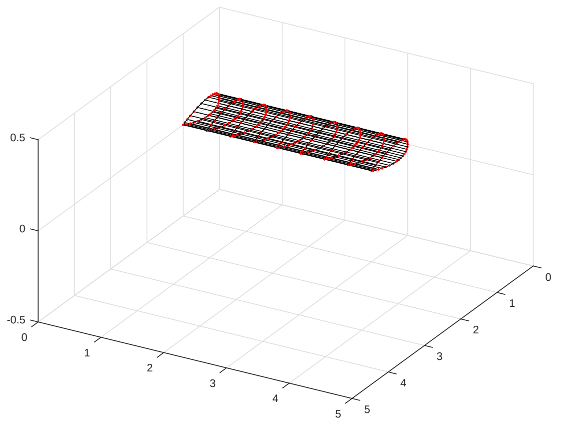
Table 7.6: Benchmark of the second optimization case performance between different sensitivity analysis methods

The optimization results are presented in Table 7.7 where the baseline and optimized values for the design variables and output functions are benchmarked using FD and the new framework. The results were obtained using a mesh with 30×8 panels in the chordwise and spanwise directions, respectively. The results were obtained in a relative course mesh due exclusively to extensive computational times using finite-differences. As it can be observed, the objective function was reduced by approximately 72%. This reduction was possible through a decrease in the lift coefficient, an increase in the aspect ratio and Oswald efficiency, achieved mainly through a combined effect of increasing the wing span and adjusting the twist, chord and camber distributions. A comparison between the baseline and optimized configurations is presented in Figure 7.6.

Figure 7.7 presents also a comparison between the airfoil shapes at their respective incidences (angle of attack $+$ local twist angle) and respective pressure distributions for different semi-spanwise wing sections. The baseline shapes are drawn in blue and the optimized in dashed red. A common feature between the optimized shapes is a relatively flat lower surface and much more curved upper surface. As a result, the pressure coefficient distribution is smoother with smaller suction peaks. This may present an advantage if viscous effects would be considered since high suction peaks usually means adverse pressure gradients in the upper surface leading to boundary layer separation, increasing drag.

Design variables	Baseline	Optimized FW	Optimized FD	Outputs	Baseline	Optimized FW	Optimized FD
$\alpha\,[^\circ]$	4	1.000000	1.000000	C_D	0.013429	0.003724	0.003725
$\Lambda\,[^\circ]$	0	6.011407	6.007157	C_L	0.313735	0.300000	0.300000
$\Gamma\,[^\circ]$	0	5.000000	5.000000	C_{M_x}	0.071235	0.065081	0.065076
$\delta_r\,[^\circ]$	0	2.230745	2.227752	C_{M_y}	0.220788	0.220788	0.220788
$\delta_t\,[^\circ]$	0	-0.124150	-0.124778	S	6.000000	6.000000	6.000000
$b\,[\mathrm{m}]$	6	8.000000	8.000000	$Æ$	6.000000	10.666667	10.666667
$c_r\,[\mathrm{m}]$	1	1.000000	1.000000				
λ	1	0.500000	0.500000				

Table 7.7: Baseline, optimized design vector and output values in the second optimization problem


(a) Baseline wing

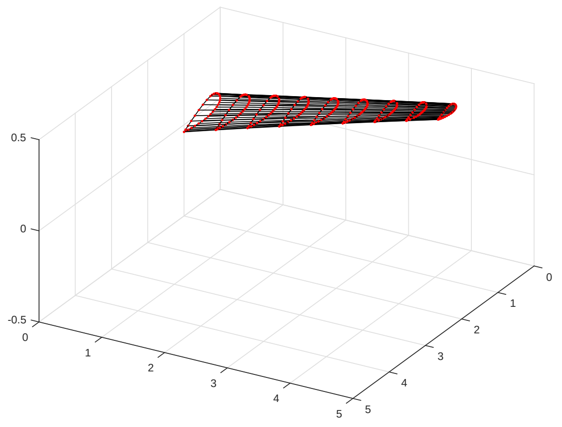
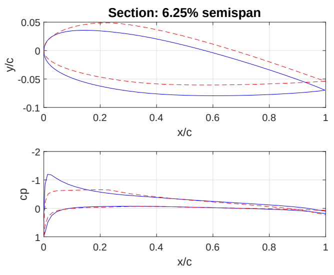
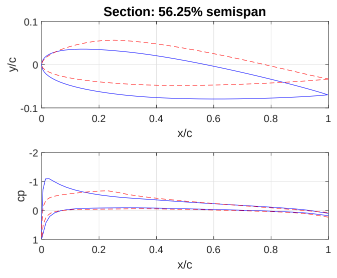

(b) Optimized wing

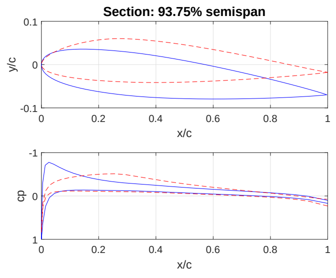
Figure 7.6: Geometrical comparison between wing configurations in the second optimization problem



(a) Airfoils at 6.25% semi-span



(b) Airfoils at 56.25% semi-span



(c) Airfoils at 93.75% semi-span

Figure 7.7: Baseline and optimized airfoil shapes and respective coefficient of pressure distributions

Chapter 8

Conclusions

8.1 Achievements

The main goal of this work was to develop an efficient aerodynamic optimization framework to be incorporated in an existing aeroelastic tool with static aero-structural capabilities. As a result, the new tool includes a fully revised panel method code for aerodynamic analysis and an efficient sensitivity analysis framework that relies on exact gradient calculation.

Firstly, a review on optimization algorithms was conducted to provide sufficient background for a conscious choice between MATLAB® optimization algorithms. It was concluded that the SQP method, a gradient-based optimization algorithm, is efficient enough since it presents a quadratic convergence rate and it is one of the most powerful algorithms available to solve nonlinear constraint optimization problems. Secondly, a survey on sensitivity analysis methods was presented since the optimization algorithm required the gradient evaluation of the objective function and constraints. It was shown that the adjoint method is the most efficient tool to estimate the sensitivities of aerodynamic models since gradient information may be calculated exactly and independently of the number of inputs. Moreover, both automatic differentiation and the complex-step derivative represent an alternative since the gradient is also estimated exactly.

Sequentially, the aerodynamic tool developed by Cardeira [17] was firstly divided into modules and then reformulated since the model was not compatible with efficient sensitivity analysis and errors where found in the calculation of the aerodynamic coefficients. The module responsible for translating the design variables into a discrete set of points representing the discretized wing was improved. A new feature was added to the model corresponding to an airfoil parametrization using bezier curves whose control points correspond to the airfoil design variables. This feature allows the optimizer to explore different airfoil configurations along the wingspan, adding more degrees of freedom to the wing and therefore, reduce the wing drag even further. The module in charge of translating the previously generated set of points into panels was reformulated. The respective original code was modified since the panel vertexes were obtained directly from the generated discrete set of points and there was no guarantee that four nearest points were coplanar. The module responsible for obtaining the aerodynamic coefficients was revised since the formula to calculate the moment coefficients was wrong.

After, the sensitivity analysis framework was developed according to the criterion of exact derivative calculation with the lowest computational cost possible. To meet this criterion, both modes of automatic differentiation, symbolic differentiation and the adjoint method were employed. It was concluded that the largest savings in computational runtime were achieved when symbolic differentiation was used since algebraic simplifications, sparsity exploration and code vectorization were possible. In that sense, it was observed that the respective sensitivity analysis of the module responsible for the panels generation and the aerodynamic solver were about 1000 and 200 times faster when compared to the implementation using automatic differentiation, respectively. In order to pre-assemble the derivatives from the sensitivity analysis of the aerodynamic solver and the post-process module, the semi analytical adjoint method was used since the combined number of inputs was much larger than the number of outputs. As a result, the sensitivity analysis framework proved to be more than 9 times faster when comparing to the implementation of the finite-differences method to the aerodynamic model without compromising the accuracy of the final jacobian.

Finally, two representative aerodynamic optimization problems were solved using the new tool and a similar implementation using forward finite-differences. The first was a simple planform shape optimization with respect to four design variables, subject to constant lift coefficient and wing area. The second was a complete wing optimization with respect to all the design variables, subject to the same constraints as the first, but in addition, the pitching moment coefficient was fixed to the baseline value. From the first problem, it can be observed that the optimized design did not changed much using one method over the other. Moreover, the new tool proved to be inefficient when dealing with a small number of design variables since the cost of evaluating the aerodynamic model a few times is less computational costly than evaluating the new sensitivity analysis framework. From the last problem, it is also observed that the final design parameters between implementations did not change much. However, the new tool proved to be considerably more efficient than the implementation using finite-differences since the optimization process using the first was about 9 times faster, reducing an 19 hour optimization to one lasting only 2 hours. According to these results, one can conclude that the new tool provides a much better computational efficiency for designs using many design variables.

8.2 Future Work

When developing this work, a few ideas to be tested and implemented arose. The first relates to the aerodynamic model. An improvement would be obtained if the wake was modeled to its exact shape since it affects the inviscid solution. Another improvement could be to include viscous flow calculations since skin friction represents a considerable percentage of the total drag, specially in cruise conditions. A possible and simple solution could be the simulation of a displacement thickness through a transpiration velocity in each panel. A second idea is to improve the wing modeling adding more design variables. This could be achieved by controlling both the local chord lengths and twist angles at specific locations along the wing span. The improvement would be obtained easily since the respective sensitivity analysis is performed using automatic differentiation. The last idea corresponds to develop an efficient aero-structural framework. A possible path to follow could be to couple the structural tool designed by Freire [18] with the framework developed in the scope of this work.

References

[1] J. D. Anderson Jr. *The airplane, a history of its technology*. Aiaa, 2002.

[2] Towpilot. Douglas dc-3, se-cfp, operated by non-profit organisation ”flygande veteraner” in sweden, October 1989. URL https://commons.wikimedia.org/wiki/File%3ADouglas_DC-3%2C_SE-CFP.jpg. Accessed: 16/January/2018.

[3] F. Cabrol. Ray hanna aux commandes du spitfire mh434 lors du flying legends, October 2006. URL https://commons.wikimedia.org/wiki/File%3ARay_Flying_Legends_2005-1.jpg. Accessed: 16/January/2018.

[4] S. Loff. Nasa's x-57 electric research plane, June 2016. URL https://www.nasa.gov/image-feature/nasas-x-57-electric-research-plane. Accessed: 17/January/2018.

[5] C. S. Tang, J. D. Zimmerman, and J. I. Nelson. Managing new product development and supply chain risks: The boeing 787 case. In *Supply Chain Forum: An International Journal*, volume 10, pages 74–86. Taylor & Francis, 2009.

[6] Artist's concept of nasa's x-57 maxwell aircraft, June 2016. URL https://commons.wikimedia.org/wiki/File%3AX57-Maxwell-CGI.jpg. Accessed: 17/January/2018.

[7] G. USAFE AFAFRICA from Ramstein Air Base. Vietnam airlines boeing 787-9 at paris air show 2015. https://www.flickr.com/photos/usafe/18884817591/, June 2015. URL https://commons.wikimedia.org/wiki/File%3AVietnam_Airlines%2C_Boeing_787-9_Dreamliner%2C_VN-A861.jpg. Accessed: 17/January/2018.

[8] D. Hulst. Current market outlook: 2015–2034. *Boeing Commercial Airplanes TR Market Analysis, Seattle, WA*, 2015.

[9] J. Sobieski and R. T. Haftka. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural optimization*, 14(1):1–23, 1997.

[10] A. C. Marta. Multidisciplinary design optimization of aircrafts. Instituto Superior Técnico, 2014.

[11] R. T. Haftka. Optimization of flexible wing structures subject to strength and induced drag constrains. *AIAA Journal*, Vol. 15(No.8), August 1977.

[12] A. Jameson. Aerodynamic design via control theory. In *Recent advances in computational fluid dynamics*, pages 377–401. Springer, 1989.

[13] J. Kennedy. *Aerostructural analysis and design optimization of composite aircraft*. PhD thesis, University of Toronto, 2012.

[14] J. R. R. A. Martins. *A coupled-Adjoint Method for High-Fidelity Aero-Structural Optimization*. PhD thesis, Stanford University, October 2002.

[15] J. R. R. A. Martins and A. B. Lambe. Multidisciplinary design optimization: a survey of architectures. *AIAA journal*, 2013.

[16] J. Almeida. Structural dynamics for aeroelastic analysis. Master's thesis, Instituto Superior Técnico, November 2015.

[17] A. Cardeira. Aeroelastic analysis of aircraft wings. Master's thesis, Instituto Superior Técnico, December 2014.

[18] T. Freire. Efficient structural optimization of aircraft wings. Master's thesis, Instituto Superior Técnico, February 2017.

[19] L. Liberti and S. Kucherenko. Comparison of deterministic and stochastic approaches to global optimization. *International Transactions in Operational Research*, 12(3):263–285, 2005.

[20] A. D. Belegundu and T. R. Chandrupatla. *Optimization concepts and applications in engineering*. Cambridge University Press, 2011.

[21] M. Gilli and P. Winker. A review of heuristic optimization methods in econometrics, June 2008. Swiss Finance Institute Research Paper No. 08-12. Available at SSRN: https://ssrn.com/abstract=1140655.

[22] A.-L. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées [Translated: (2010)]. *Compte Rendu des S'eances de L'Acad'emie des Sciences*, 25(2):536–538, 1847.

[23] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964.

[24] A. R. Conn, N. I. Gould, and P. L. Toint. *Trust region methods*. SIAM, 2000.

[25] G. Zoutendijk. *Methods of feasible directions: A study in linear and nonlinear programming*. Elsevier, 1960. ASIN: B0006AWLPG.

[26] P. Wolfe. Methods of nonlinear programming. In R. L. G. P.Wolfe, editor, *Recent Advances in Mathematical Programming*, pages 67–86, New York, 1963. McGraw-Hill.

[27] K. Schittkowski and Y.-x. Yuan. Sequential quadratic programming methods. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[28] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[29] R. Hooke and T. A. Jeeves. "direct search"solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961.

[30] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4): 308–313, 1965.

[31] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 79(1):157–181, 1993.

[32] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[33] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[34] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[35] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995.

[36] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, et al. Optimization by simulated annealing. *science*, 220 (4598):671–680, 1983.

[37] P. Moscato and J. Fontanari. Stochastic versus deterministic update in simulated annealing. *Physics Letters A*, 146(4):204–208, 1990.

[38] G. Dueck and T. Scheuer. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of computational physics*, 90(1):161–175, 1990.

[39] M. Dorigo. Optimization, learning and natural algorithms. *Italian PhD dissertationPolitecnico di MilanoMilan*, 1992.

[40] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[41] J. R. R. A. Martins and J. Hwang. Review and unification of methods for computing derivatives of multidisciplinary computational models. *AIAA Journal*, 51(11):2582–2599, November 2013. DOI: 10.2514/1.J052184.

[42] H. M. Adelman and R. T. Haftka. Sensitivity analysis of discrete structural systems. *AIAA Journal*, 24(5):823–832, May 1986.

[43] J. R. R. A. Martins. Sensitivity analysis. *AA222-Multidisciplinary Design Optimization*, 2001.

[44] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262, 2003.

[45] J. N. Lyness. Numerical algorithms based on the theory of complex variable. In *Proceedings of the 1967 22nd national conference*, pages 125–133. ACM, 1967.

[46] J. N. Lyness and C. B. Moler. Numerical differentiation of analytic functions. *SIAM Journal on Numerical Analysis*, 4(2):202–210, 1967.

[47] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *Siam Review*, 40(1):110–112, 1998.

[48] J. E. Peter and R. P. Dwight. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, 39(3):373–391, 2010.

[49] D. R. Bristow and J. D. Hawk. Subsonic panel method for the efficient analysis of multiple geometry perturbations. Technical report, NASA, 1982.

[50] D. R. Bristow and J. D. Hawk. Subsonic panel method for designing wing surfaces from pressure distribution. Technical report, NASA, 1983.

[51] C. H. Bischof, P. D. Hovland, and B. Norris. On the implementation of automatic differentiation tools. *Higher-Order and Symbolic Computation*, 21(3):311–331, 2008.

[52] A. Carle and M. Fagan. Adifor 3.0 overview. *Rice University Report CAAM-TR-00-02*, 2000.

[53] L. Hascoët and V. Pascual. *Tapenade 2.1 user's guide*. PhD thesis, INRIA, 2004.

[54] H. Berland. Automatic differentiation. Department of Mathematical Sciences, NTNU, September 2006.

[55] A. Griewank, D. Juedes, and J. Utke. Algorithm 755: Adol-c: a package for the automatic differentiation of algorithms written in c/c++. *ACM Transactions on Mathematical Software (TOMS)*, 22(2): 131–167, 1996.

[56] D. Shiriaev and A. Griewank. Adol-f: Automatic differentiation of fortran codes. *Computational Differentiation: Techniques, Applications, and Tools*, pages 375–384, 1996.

[57] C. H. Bischof, H. M. Bucker, B. Lang, A. Rasch, and A. Vehreschild. Combining source transformation and operator overloading techniques to compute derivatives for matlab programs. In *Source Code Analysis and Manipulation, 2002. Proceedings. Second IEEE International Workshop on*, pages 65–72. IEEE, 2002.

[58] J. D. Anderson Jr. *Fundamentals of aerodynamics*. Tata McGraw-Hill Education, 2010.

[59] J. H. Ferziger and M. Peric. *Computational methods for fluid dynamics*. Springer Science & Business Media, 2012.

[60] J. Katz and A. Plotkin. *Low-speed aerodynamics*, volume 13. Cambridge university press, 2001.

[61] J. L. Hess. Calculation of potential flow about arbitrary three-dimensional lifting bodies. Technical report, Douglas Aircraft Co Long Beach CA, 1972.

[62] P. Venkataraman. A new procedure for airfoil definition. *AIAA Paper*, pages 95–1875, 1995.

[63] Matlab documentation for the linspace function. URL `https://www.mathworks.com/help/matlab/ref/linsolve.html`. Accessed: 24/February/2018.

[64] A. Deperrois. Xflr5 analysis of foils and wings operating at low reynolds numbers. *Guidelines for XFLR5*, 2009.

[65] T. C. Corke. *Design of aircraft*. Pearson College Division, 2003.

[66] V. d. Brederode. Fundamentos de aerodinâmica incompressível. *Edição do autor*, 1997.

[67] Matlab documentation for the optimization toolbox. URL `https://www.mathworks.com/help/optim/index.html`. Accessed: 25/February/2018.

Appendix A

Details of Partial Derivatives

A.1 *Panels Definition* Module

The non-zero components of $\frac{\partial DS_{ij}}{\partial \mathbf{U}_{mn}}$ in Equation (5.25) are given by

$$\begin{aligned}\frac{\partial DS_{ij}}{\partial X_{A_{1ij}}} = & \left[X_{B_{ij2}}\left(X_{A_{ij1}}X_{B_{ij2}} - X_{A_{ij2}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij2}} - X_{B_{ij2}}X_{C_{ij1}}\right) + \right. \\ & \left. X_{B_{ij3}}\left(X_{A_{ij1}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij1}}\right)\right]/aux_{ij}\end{aligned} \tag{A.1}$$

$$\begin{aligned}\frac{\partial DS_{ij}}{\partial X_{A_{ij2}}} = & -\left[X_{B_{ij1}}\left(X_{A_{ij1}}X_{B_{ij2}} - X_{A_{ij2}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij2}} - X_{B_{ij2}}X_{C_{ij1}}\right) - \right. \\ & \left. X_{B_{ij3}}\left(X_{A_{ij2}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij2}} + X_{B_{ij2}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij2}}\right)\right]/aux_{ij}\end{aligned} \tag{A.2}$$

$$\begin{aligned}\frac{\partial DS_{ij}}{\partial X_{A_{ij3}}} = & -\left[X_{B_{ij1}}\left(X_{A_{ij1}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij1}}\right) + \right. \\ & \left. X_{B_{ij2}}\left(X_{A_{ij2}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij2}} + X_{B_{ij2}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij2}}\right)\right]/aux_{ij}\end{aligned} \tag{A.3}$$

$$\begin{aligned}\frac{\partial DS_{ij}}{\partial X_{B_{ij1}}} = & -\left[\left(X_{A_{ij2}} - X_{C_{ij2}}\right)\left(X_{A_{ij1}}X_{B_{ij2}} - X_{A_{ij2}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij2}} - X_{B_{ij2}}X_{C_{ij1}}\right) + \right. \\ & \left. \left(X_{A_{ij3}} - X_{C_{ij3}}\right)\left(X_{A_{ij1}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij1}}\right)\right]/aux_{ij}\end{aligned} \tag{A.4}$$

$$\begin{aligned}\frac{\partial DS_{ij}}{\partial X_{B_{ij2}}} = & \left[\left(X_{A_{ij1}} - X_{C_{ij1}}\right)\left(X_{A_{ij1}}X_{B_{ij2}} - X_{A_{ij2}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij2}} - X_{B_{ij2}}X_{C_{ij1}}\right) - \right. \\ & \left. \left(X_{A_{ij3}} - X_{C_{ij3}}\right)\left(X_{A_{ij2}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij2}} + X_{B_{ij2}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij2}}\right)\right]/aux_{ij}\end{aligned} \tag{A.5}$$

$$\frac{\partial DS_{ij}}{\partial X_{B_{ij3}}} = \left[\left(X_{A_{ij1}} - X_{C_{ij1}}\right)\left(X_{A_{ij1}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij1}}\right) + \right.$$
$$\left.\left(X_{A_{ij2}} - X_{C_{ij2}}\right)\left(X_{A_{ij2}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij2}} + X_{B_{ij2}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij2}}\right)\right]/aux_{ij} \quad \text{(A.6)}$$

$$\frac{\partial DS_{ij}}{\partial X_{C_{ij1}}} = -\left[X_{B_{ij2}}\left(X_{A_{ij1}}X_{B_{ij2}} - X_{A_{ij2}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij2}} - X_{B_{ij2}}X_{C_{ij1}}\right) + \right.$$
$$\left.X_{B_{ij3}}\left(X_{A_{ij1}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij1}}\right)\right]/aux_{ij} \quad \text{(A.7)}$$

$$\frac{\partial DS_{ij}}{\partial X_{C_{ij2}}} = \left[X_{B_{ij1}}\left(X_{A_{ij1}}X_{B_{ij2}} - X_{A_{ij2}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij2}} - X_{B_{ij2}}X_{C_{ij1}}\right) - \right.$$
$$\left.X_{B_{ij3}}\left(X_{A_{ij2}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij2}} + X_{B_{ij2}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij2}}\right)\right]/aux_{ij} \quad \text{(A.8)}$$

$$\frac{\partial DS_{ij}}{\partial X_{C_{ij3}}} = \left[X_{B_{ij1}}\left(X_{A_{ij1}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij1}}\right) + \right.$$
$$\left.X_{B_{ij2}}\left(X_{A_{ij2}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij2}} + X_{B_{ij2}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij2}}\right)\right]/aux_{ij} \quad \text{(A.9)}$$

where aux_{ij} is defined as

$$aux_{ij} = 2\left[\left(X_{A_{ij1}}X_{B_{ij2}} - X_{A_{ij2}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij2}} - X_{B_{ij2}}X_{C_{ij1}}\right)^2\right.$$
$$\left(X_{A_{ij1}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij1}} + X_{B_{ij1}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij1}}\right)^2$$
$$\left.\left(X_{A_{ij2}}X_{B_{ij3}} - X_{A_{ij3}}X_{B_{ij2}} + X_{B_{ij2}}X_{C_{ij3}} - X_{B_{ij3}}X_{C_{ij2}}\right)^2\right]^{\frac{1}{2}} \quad \text{(A.10)}$$

The non-zero components of $\frac{\partial \mathbf{n}_{ij}}{\partial \mathbf{N}_{mn}}$, in Equation (5.27), are given by

$$\frac{\partial \mathbf{n}_{ij}}{\partial \mathbf{N}_{ij}} = \frac{1}{\|\mathbf{N}_{ij}\|^3}\begin{bmatrix} N_{ij2}^2 + N_{ij3}^2 & -\left(N_{ij1}N_{ij2}\right) & -\left(N_{ij1}N_{ij3}\right) \\ -\left(N_{ij1}N_{ij2}\right) & N_{ij1}^2 + N_{ij3}^2 & -\left(N_{ij2}N_{ij3}\right) \\ -\left(N_{ij1}N_{ij3}\right) & -\left(N_{ij2}N_{ij3}\right) & N_{ij1}^2 + N_{ij2}^2 \end{bmatrix} \quad \text{(A.11)}$$

Additionally, in Equation (5.27), $\frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{PP}_{rs}}$ is defined as

$$\frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{PP}_{rs}} = \begin{bmatrix} \frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{X_1}_{rs}} & \frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{X_2}_{rs}} & \frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{X_3}_{rs}} & \frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{X_4}_{rs}} \end{bmatrix} \quad \text{(A.12)}$$

where the non-zero components are given by

$$\frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{X_1}_{mn}} = \begin{bmatrix} 0 & X_{2_{mn3}} - X_{4_{mn3}} & X_{4_{mn2}} - X_{2_{mn2}} \\ X_{4_{mn3}} - X_{2_{mn3}} & 0 & X_{2_{mn1}} - X_{4_{mn1}} \\ X_{2_{mn2}} - X_{4_{mn2}} & X_{4_{mn1}} - X_{2_{mn1}} & 0 \end{bmatrix} \tag{A.13}$$

$$\frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{X_2}_{mn}} = \begin{bmatrix} 0 & X_{3_{mn3}} - X_{1_{mn3}} & X_{1_{mn2}} - X_{3_{mn2}} \\ X_{1_{mn3}} - X_{3_{mn3}} & 0 & X_{3_{mn1}} - X_{1_{mn1}} \\ X_{3_{mn2}} - X_{1_{mn2}} & X_{1_{mn1}} - X_{3_{mn1}} & 0 \end{bmatrix} \tag{A.14}$$

$$\frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{X_3}_{mn}} = \begin{bmatrix} 0 & X_{4_{mn3}} - X_{2_{mn3}} & X_{2_{mn2}} - X_{4_{mn2}} \\ X_{2_{mn3}} - X_{4_{mn3}} & 0 & X_{4_{mn1}} - X_{2_{mn1}} \\ X_{4_{mn2}} - X_{2_{mn2}} & X_{2_{mn1}} - X_{4_{mn1}} & 0 \end{bmatrix} \tag{A.15}$$

$$\frac{\partial \mathbf{N}_{mn}}{\partial \mathbf{X_4}_{mn}} = \begin{bmatrix} 0 & X_{1_{mn3}} - X_{3_{mn3}} & X_{3_{mn2}} - X_{1_{mn2}} \\ X_{3_{mn3}} - X_{1_{mn3}} & 0 & X_{1_{mn1}} - X_{3_{mn1}} \\ X_{1_{mn2}} - X_{3_{mn2}} & X_{3_{mn1}} - X_{1_{mn1}} & 0 \end{bmatrix} \tag{A.16}$$

Similarly, in Equation (5.28)

$$\begin{bmatrix} \frac{\partial \mathbf{m}_{ij}}{\partial \mathbf{l}_{ij}} & \frac{\partial \mathbf{m}_{ij}}{\partial \mathbf{n}_{ij}} \end{bmatrix} = \left[\begin{array}{ccc|ccc} 0 & -n_{ij3} & n_{ij2} & 0 & l_{ij3} & -l_{ij2} \\ n_{ij3} & 0 & -n_{ij1} & -l_{ij3} & 0 & l_{ij1} \\ -n_{ij2} & n_{ij1} & 0 & l_{ij2} & -l_{ij1} & 0 \end{array}\right] \tag{A.17}$$

A.2 *Aero Solver* Module

Observing Equation (4.50b), it can be shown that

$$\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\xi} = \begin{cases} \frac{1}{4\pi}\left(\frac{\mathrm{d}q_{ik}}{\mathrm{d}\xi} + \frac{\mathrm{d}q_{kj}}{\mathrm{d}\xi}\right) & \xi = x_k, y_k \\ \frac{1}{4\pi}\left(\frac{\mathrm{d}q_{12}}{\mathrm{d}\xi} + \frac{\mathrm{d}q_{23}}{\mathrm{d}\xi} + \frac{\mathrm{d}q_{34}}{\mathrm{d}\xi} + \frac{\mathrm{d}q_{41}}{\mathrm{d}\xi}\right) & \xi = x, y, z \end{cases} \tag{A.18}$$

After some algebra, Equation (4.51b) can be manipulated to

$$q_{ij} = \operatorname{atan2}(u_{ij}, v_{ij}) \tag{A.19}$$

where $\operatorname{atan2}$ is the arc-tangent function with two arguments. The variables u_{ij} and v_{ij} are defined as

$$u_{ij} = z(x_j - x_i)(f_{ij}r_j - g_{ij}r_i) \tag{A.20a}$$

$$v_{ij} = z^2(x_j - x_i)^2 r_i r_j + f_{ij}g_{ij} \tag{A.20b}$$

and additionally

$$f_{ij} = (y_j - y_i)e_i - (x_j - x_i)h_i \tag{A.21a}$$

$$g_{ij} = (y_j - y_i)e_j - (x_j - x_i)h_j \tag{A.21b}$$

Taking the total derivative of contribution q_{ij} with respect to an arbitrary variable ξ it is obtained

$$\frac{\mathrm{d}q_{ij}}{\mathrm{d}\xi} = \frac{v_{ij}}{u_{ij}^2 + v_{ij}^2}\frac{\mathrm{d}u_{ij}}{\mathrm{d}\xi} - \frac{u_{ij}}{u_{ij}^2 + v_{ij}^2}\frac{\mathrm{d}v_{ij}}{\mathrm{d}\xi} \tag{A.22}$$

Observing the explicit dependence of u_{ij} and v_{ij} on the variables z, x_i, x_j, r_i, r_j, g_{ij} and f_{ij}, one can write

$$\frac{\mathrm{d}u_{ij}}{\mathrm{d}\xi} = \frac{\partial u_{ij}}{\partial z}\frac{\mathrm{d}z}{\mathrm{d}\xi} + \frac{\partial u_{ij}}{\partial x_j}\frac{\mathrm{d}x_j}{\mathrm{d}\xi} + \frac{\partial u_{ij}}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}\xi} + \frac{\partial u_{ij}}{\partial r_i}\frac{\mathrm{d}r_i}{\mathrm{d}\xi} + \frac{\partial u_{ij}}{\partial r_j}\frac{\mathrm{d}r_j}{\mathrm{d}\xi} + \frac{\partial u_{ij}}{\partial f_{ij}}\frac{\mathrm{d}f_{ij}}{\mathrm{d}\xi} + \frac{\partial u_{ij}}{\partial g_{ij}}\frac{\mathrm{d}g_{ij}}{\mathrm{d}\xi} \tag{A.23}$$

$$\frac{\mathrm{d}v_{ij}}{\mathrm{d}\xi} = \frac{\partial v_{ij}}{\partial z}\frac{\mathrm{d}z}{\mathrm{d}\xi} + \frac{\partial v_{ij}}{\partial x_j}\frac{\mathrm{d}x_j}{\mathrm{d}\xi} + \frac{\partial v_{ij}}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}\xi} + \frac{\partial v_{ij}}{\partial r_i}\frac{\mathrm{d}r_i}{\mathrm{d}\xi} + \frac{\partial v_{ij}}{\partial r_j}\frac{\mathrm{d}r_j}{\mathrm{d}\xi} + \frac{\partial v_{ij}}{\partial f_{ij}}\frac{\mathrm{d}f_{ij}}{\mathrm{d}\xi} + \frac{\partial v_{ij}}{\partial g_{ij}}\frac{\mathrm{d}g_{ij}}{\mathrm{d}\xi} \tag{A.24}$$

Re-writing equations (A.23) and (A.24), and unrolling the partial derivatives, one obtains

$$\begin{aligned}\frac{\mathrm{d}u_{ij}}{\mathrm{d}\xi} = (f_{ij}r_j - g_{ij}r_i)&\left[-(x_i - x_j)\frac{\mathrm{d}z}{\mathrm{d}\xi} + z\left(\frac{\mathrm{d}x_j}{\mathrm{d}\xi} - \frac{\mathrm{d}x_i}{\mathrm{d}\xi}\right)\right] \\ +z(x_i - x_j)&\left[g_{ij}\frac{\mathrm{d}r_i}{\mathrm{d}\xi} + f_{ij}\frac{\mathrm{d}r_j}{\mathrm{d}\xi} - r_j\frac{\mathrm{d}f_{ij}}{\mathrm{d}\xi} + r_i\frac{\mathrm{d}g_{ij}}{\mathrm{d}\xi}\right]\end{aligned} \tag{A.25}$$

$$\begin{aligned}\frac{\mathrm{d}v_{ij}}{\mathrm{d}\xi} =& 2r_ir_jz(x_i - x_j)\left[(x_i - x_j)\frac{\mathrm{d}z}{\mathrm{d}\xi} + z\left(\frac{\mathrm{d}x_i}{\mathrm{d}\xi} - \frac{\mathrm{d}x_j}{\mathrm{d}\xi}\right)\right] + \\ &z^2(x_i - x_j)^2\left[r_j\frac{\mathrm{d}r_i}{\mathrm{d}\xi} + r_i\frac{\mathrm{d}r_j}{\mathrm{d}\xi}\right] + g_{ij}\frac{\mathrm{d}f_{ij}}{\mathrm{d}\xi} + f_{ij}\frac{\mathrm{d}g_{ij}}{\mathrm{d}\xi}\end{aligned} \tag{A.26}$$

Considering the dependence of r_i on the variables x_i, y_i, x, y and z, it can be stated that

$$\frac{\mathrm{d}r_i}{\mathrm{d}\xi} = \frac{\partial r_i}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}\xi} + \frac{\partial r_i}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}\xi} + \frac{\partial r_i}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}\xi} + \frac{\partial r_i}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}\xi} + \frac{\partial r_i}{\partial z}\frac{\mathrm{d}z}{\mathrm{d}\xi} \tag{A.27}$$

where the partial derivatives are given by

$$\frac{\partial r_i}{\partial x_i} = -\frac{\partial r_i}{\partial x} = -\frac{(x - x_i)}{r_i} \tag{A.28}$$

$$\frac{\partial r_i}{\partial y_i} = -\frac{\partial r_i}{\partial y} = -\frac{(y - y_i)}{r_i} \tag{A.29}$$

$$\frac{\partial r_i}{\partial z} = \frac{z}{r_i} \tag{A.30}$$

The derivatives of r_j with respect to the generic variable ξ are easily obtained replacing the index i by j on the previous equations.

Observing now the expressions for f_{ij} and g_{ij} in equations (A.21a) and (A.21b), and applying the chain-rule to those, one gets

$$\frac{\mathrm{d}f_{ij}}{\mathrm{d}\xi} = \frac{\partial f_{ij}}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}\xi} + \frac{\partial f_{ij}}{\partial x_j}\frac{\mathrm{d}x_j}{\mathrm{d}\xi} + \frac{\partial f_{ij}}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}\xi} + \frac{\partial f_{ij}}{\partial y_j}\frac{\mathrm{d}y_j}{\mathrm{d}\xi} + \frac{\partial f_{ij}}{\partial e_i}\frac{\mathrm{d}e_i}{\mathrm{d}\xi} + \frac{\partial f_{ij}}{\partial h_i}\frac{\mathrm{d}h_i}{\mathrm{d}\xi} \tag{A.31}$$

$$\frac{\mathrm{d}g_{ij}}{\mathrm{d}\xi} = \frac{\partial f_{ij}}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}\xi} + \frac{\partial f_{ij}}{\partial x_j}\frac{\mathrm{d}x_j}{\mathrm{d}\xi} + \frac{\partial f_{ij}}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}\xi} + \frac{\partial f_{ij}}{\partial y_j}\frac{\mathrm{d}y_j}{\mathrm{d}\xi} + \frac{\partial f_{ij}}{\partial e_j}\frac{\mathrm{d}e_j}{\mathrm{d}\xi} + \frac{\partial f_{ij}}{\partial h_j}\frac{\mathrm{d}h_j}{\mathrm{d}\xi} \tag{A.32}$$

The partial derivatives in both equations are given by

$$\frac{\partial f_{ij}}{\partial x_i} = -\frac{\partial f_{ij}}{\partial x_j} = h_i \tag{A.33}$$

$$\frac{\partial f_{ij}}{\partial y_i} = -\frac{\partial f_{ij}}{\partial y_j} = -e_i \tag{A.34}$$

$$\frac{\partial f_{ij}}{\partial e_i} = \frac{\partial g_{ij}}{\partial e_j} = (y_j - y_i) \tag{A.35}$$

$$\frac{\partial f_{ij}}{\partial h_i} = \frac{\partial g_{ij}}{\partial h_j} = -(x_j - x_i) \tag{A.36}$$

$$\frac{\partial g_{ij}}{\partial x_i} = -\frac{\partial g_{ij}}{\partial x_j} = h_j \tag{A.37}$$

$$\frac{\partial g_{ij}}{\partial x_i} = -\frac{\partial g_{ij}}{\partial x_j} = h_j \tag{A.38}$$

$$\frac{\partial g_{ij}}{\partial y_i} = -\frac{\partial g_{ij}}{\partial y_j} = -e_j \tag{A.39}$$

Finally, through equations (4.52d) and (4.52e), one can write

$$\frac{\mathrm{d}e_i}{\mathrm{d}\xi} =2\left[(x - x_i)\left(\frac{\mathrm{d}x}{\mathrm{d}\xi} - \frac{\mathrm{d}x_i}{\mathrm{d}\xi}\right) + z\frac{\mathrm{d}x}{\mathrm{d}\xi}\right] \tag{A.40}$$

$$\frac{\mathrm{d}h_i}{\mathrm{d}\xi} =(x - x_i)\left[\frac{\mathrm{d}y}{\mathrm{d}\xi} - \frac{\mathrm{d}y_i}{\mathrm{d}\xi}\right] + (y - y_i)\left[\frac{\mathrm{d}x}{\mathrm{d}\xi} - \frac{\mathrm{d}x_i}{\mathrm{d}\xi}\right] \tag{A.41}$$

Again, the respective derivatives for e_j and h_j are obtained replacing the index i by j.

Considering now the source influence in Equation (4.50a), it is possible to relate it with the dipole influence $\mathcal{C}$ as

$$\mathcal{B} = -\frac{1}{4\pi}\left[(s_{12} + s_{23} + s_{34} + s_{41}) + 4\pi\|z\|\mathcal{C}\right] \tag{A.42}$$

Taking the derivative with respect to the generic variable ξ

$$\frac{\mathrm{d}\mathcal{B}}{\mathrm{d}\xi} = \begin{cases} -\frac{1}{4\pi}\left(\frac{\mathrm{d}s_{ik}}{\mathrm{d}\xi} + \frac{\mathrm{d}s_{kj}}{\mathrm{d}\xi} + 4\pi\|z\|\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\xi}\right) & \text{for } \xi = x_k, y_k \\ -\frac{1}{4\pi}\left(\frac{\mathrm{d}s_{12}}{\mathrm{d}\xi} + \frac{\mathrm{d}s_{23}}{\mathrm{d}\xi} + \frac{\mathrm{d}s_{34}}{\mathrm{d}\xi} + \frac{\mathrm{d}s_{41}}{\mathrm{d}\xi} + 4\pi\|z\|\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\xi}\right) & \text{for } \xi = x, y \\ -\frac{1}{4\pi}\left(\frac{\mathrm{d}s_{12}}{\mathrm{d}\xi} + \frac{\mathrm{d}s_{23}}{\mathrm{d}\xi} + \frac{\mathrm{d}s_{34}}{\mathrm{d}\xi} + \frac{\mathrm{d}s_{41}}{\mathrm{d}\xi} + 4\pi\left(-\mathcal{C} + \|z\|\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\xi}\right)\right) & \text{for } \xi = z < 0 \\ -\frac{1}{4\pi}\left(\frac{\mathrm{d}s_{12}}{\mathrm{d}\xi} + \frac{\mathrm{d}s_{23}}{\mathrm{d}\xi} + \frac{\mathrm{d}s_{34}}{\mathrm{d}\xi} + \frac{\mathrm{d}s_{41}}{\mathrm{d}\xi} + 4\pi\left(\mathcal{C} + \|z\|\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\xi}\right)\right) & \text{for } \xi = z > 0 \end{cases} \tag{A.43}$$

Observing now Equation (4.51a) and paying attention to the dependencies of q_{ij} one may write

$$\begin{aligned}\frac{\mathrm{d}s_{ij}}{\mathrm{d}\xi} =&\frac{\partial s_{ij}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}\xi} + \frac{\partial s_{ij}}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}\xi} + \frac{\partial s_{ij}}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}\xi} + \frac{\partial s_{ij}}{\partial x_j}\frac{\mathrm{d}x_j}{\mathrm{d}\xi} + \frac{\partial s_{ij}}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}\xi} + \\ &\frac{\partial s_{ij}}{\partial y_j}\frac{\mathrm{d}y_j}{\mathrm{d}\xi} + \frac{\partial s_{ij}}{\partial r_i}\frac{\mathrm{d}r_i}{\mathrm{d}\xi} + \frac{\partial s_{ij}}{\partial r_j}\frac{\mathrm{d}r_j}{\mathrm{d}\xi} + \frac{\partial s_{ij}}{\partial d_{ij}}\frac{\mathrm{d}d_{ij}}{\mathrm{d}\xi}\end{aligned} \tag{A.44}$$

Re-writing the equation and explaining the partial derivatives

$$\begin{aligned}\frac{\mathrm{d}s_{ij}}{\mathrm{d}\xi} = \frac{1}{d_{ij}} \ln\left(\frac{r_i + r_j + d_{ij}}{r_i + r_j - d_{ij}}\right) \Bigg[(y_j - y_i) \frac{\mathrm{d}x}{\mathrm{d}\xi} + (x_i - x_j) \frac{\mathrm{d}y}{\mathrm{d}\xi} + (y - y_j) \frac{\mathrm{d}x_i}{\mathrm{d}\xi} - \\ (y - y_i) \frac{\mathrm{d}x_j}{\mathrm{d}\xi} - (x - x_j) \frac{\mathrm{d}y_i}{\mathrm{d}\xi} + (x - x_i) \frac{\mathrm{d}y_j}{\mathrm{d}\xi} \Bigg] \\ + \frac{2(xy_i - x_i y - xy_j + x_j y + x_i y_j - x_j y_i)}{r_i^2 + r_j^2 + 2r_i r_j - d_{ij}^2} \left[\frac{\mathrm{d}r_i}{\mathrm{d}\xi} - \frac{\mathrm{d}r_j}{\mathrm{d}\xi} \right] + \frac{\partial s_{ij}}{\partial d_{ij}} \frac{\mathrm{d}d_{ij}}{\mathrm{d}\xi}\end{aligned} \tag{A.45}$$

where $\frac{\partial s_{ij}}{\partial d_{ij}}$ is given by

$$\begin{aligned}\frac{\partial s_{ij}}{\partial d_{ij}} = \frac{1}{d_{ij}^2} \ln\left(\frac{r_i + r_j + d_{ij}}{r_i + r_j - d_{ij}}\right) \left[(x - x_i)(y_i - y_j) - (x_i - x_j)(y - y_i) \right] - \\ \frac{1}{d_{ij}(d_{ij} + r_i + r_j)} \left[\frac{1}{r_i + r_j - d_{ij}} + \frac{r_i + r_j + d_{ij}}{(r_i + r_j - d_{ij})^2} \right] \left[(x - x_i)(y_i - y_j) - (x_i - x_j)(y - y_i) \right] (r_i + r_j - d_{ij})\end{aligned} \tag{A.46}$$

and $\frac{\mathrm{d}d_{ij}}{\mathrm{d}\xi}$ is given by

$$\frac{\mathrm{d}d_{ij}}{\mathrm{d}\xi} = \frac{1}{d_{ij}} \left[(x_i - x_j) \left(\frac{\mathrm{d}x_i}{\mathrm{d}\xi} - \frac{\mathrm{d}x_j}{\mathrm{d}\xi} \right) + (y_i - y_j) \left(\frac{\mathrm{d}y_i}{\mathrm{d}\xi} - \frac{\mathrm{d}y_j}{\mathrm{d}\xi} \right) \right] \tag{A.47}$$

Appendix B

Twist and Planform Optimization

This problem is similar to the first, presented in section 7.2, and it corresponds to minimize the wing drag coefficient subject to constant lift coefficient and wing area, and bounds. Opposed to the first problem, the angle of attack was kept constant and the lift distribution is expected to be rearranged through wing twist. Therefore, the design variables were chosen to be the root and tip twist angles, δ_r and δ_t, the wing span b, and the root chord c_r. The optimization problem is mathematically described by

$$\begin{aligned} \text{minimize} \quad & C_D \\ \text{w.r.t.} \quad & \mathbf{x_{DV}} \\ \text{subject to} \quad & C_L = C_{L0}, \quad S = S_0 \\ & \mathbf{x}^L \leq \mathbf{x_{DV}} \leq \mathbf{x}^U \end{aligned} \tag{B.1}$$

where S_0 is the baseline wing area, C_{L0} is the baseline lift coefficient, $\mathbf{x}^L$ is the lower bound vector and $\mathbf{x}^U$ is the upper bound vector. Table B.1 shows both the initial design variables and respective bounds.

Design Vector ($\mathbf{x_{DV}}$)	Lower Bound ($\mathbf{x}^L$)	Initial Design ($\mathbf{x_{DV_0}}$)	Upper Bound ($\mathbf{x}^U$)
$\delta_r\,[^\circ\,]$	0	0	6
$\delta_t\,[^\circ\,]$	-6	0	0
$b\,[\mathrm{m}]$	5	6	8
$c_r\,[\mathrm{m}]$	1	1.5	3

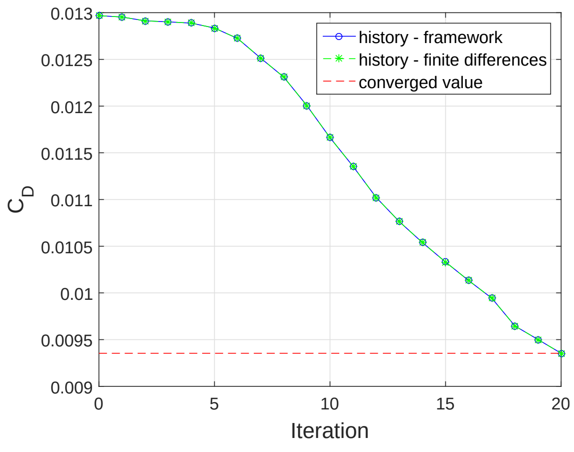
Table B.1: Initial values of the design vector and respective bounds for the additional optimization problem

After the design variables have been defined, it is necessary to define a baseline wing configuration and respective flight condition. The wing was chosen to be rectangular, without sweep and dihedral, with constant airfoil shape along the wing span, as shown in Table B.2.

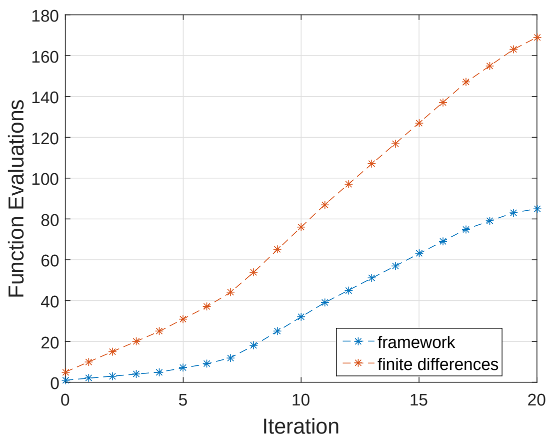
$\Lambda\,[^\circ\,]$	$\Gamma\,[^\circ\,]$	$\delta_r\,[^\circ\,]$	$\delta_t\,[^\circ\,]$	$b\,[\mathrm{m}]$	$c_r\,[\mathrm{m}]$	λ	section airfoil	$V[\mathrm{m/s}]$	$\alpha\,[^\circ\,]$
0	0	0	0	6	1.5	1	NACA 0010	75	4

Table B.2: Baseline wing configuration to the additional optimization problem

The optimization process was solved using both the new sensitivity analysis framework and finite differences as previously. Figure B.1 (a) shows the optimization process using both methods. As depicted, using either finite differences or the new framework, the optimizations converge in 21 iterations. During the process, the number of function evaluations using finite differences is roughly double than using the framework. However, the optimization using finite differences was about 3 times faster than using the new tool. According to these results, one may conclude that using finite differences is preferable if very few design variables are being used, as one may suspect looking at Table 5.5. A summary of the optimization process performance is tabulated in Table B.3.



(a) Convergence history of C_D



(b) Function evaluations

Figure B.1: Convergence process and number of function evaluations during the additional optimization problem

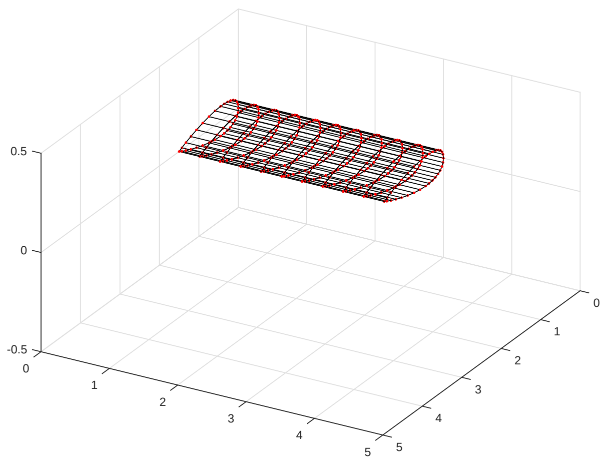
Gradient Calculation Method	Time [s]	Iterations	Function Evaluations
Sensitivity Framework	6053.3	21	86
Forward Finite Differences	1928.1	21	174

Table B.3: Benchmark of the additional optimization case performance between different sensitivity analysis methods

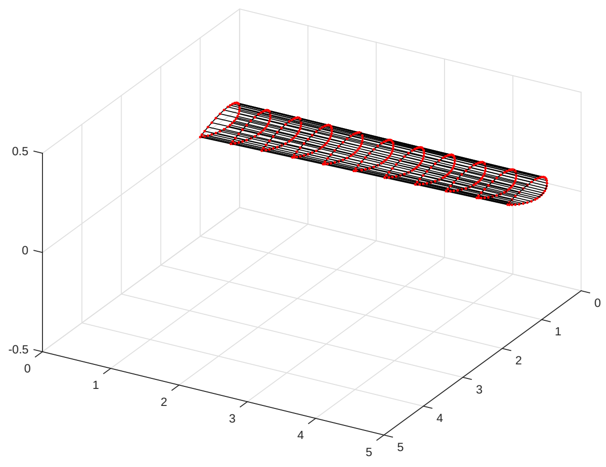
The optimization results are presented in Table B.4 and a visual comparison between the baseline and optimized wing configurations are presented in Figure B.2. Observing the objective function values before and after the optimization one may observe a 28% reduction while satisfying the lift and area equality constraints. The reduction was possible by adjusting the tip angle of twist to approach an elliptical lift distribution and an increase in the wing's aspect ratio, increasing b and decreasing c_r.

Design variables	Baseline	Optimized FW	Optimized FD	Outputs	Baseline	Optimized FW	Optimized FD
δ_r [$^\circ$]	0	0.000000	0.000000	C_D	0.012968	0.009354	0.009354
δ_t [$^\circ$]	0	-2.128537	-2.128537	C_L	0.266394	0.266394	0.266394
b [m]	6	9.000000	9.000000	C_{M_x}	0.059249	0.056615	0.056615
c_r [m]	1.5	1.000000	1.000000	C_{M_y}	0.098610	0.182938	0.182938
				S	9.000000	9.000000	9.000000
				AR	4.000000	9.000000	9.000000

Table B.4: Baseline, optimized design vector and output values of the additional optimization problem



(a) Baseline wing



(b) Optimized wing

Figure B.2: Geometrical comparison between baseline and optimized wing configurations of the additional optimization problem