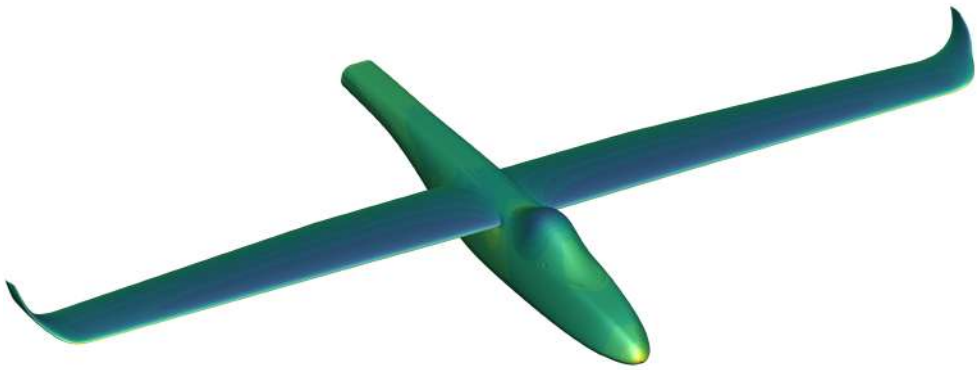
TÉCNICO LISBOA



Aerodynamic Design of a MAME UAV Wing Using High-Fidelity Numerical Tools

Rúben da Silva Gameiro

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor: Prof. André Calado Marta

Examination Committee

Chairperson: Prof. Afzal Suleman
Supervisor: Prof. André Calado Marta
Member of the Committee: Prof. Luís Rego da Cunha de Eça

November 2023

Dedicated to my parents

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisor, Professor André Marta, for the amazing guidance, expertise, availability and patience throughout this project.

I extend my gratitude to the rest of the work group: Nuno Matos, from Tekever, for always helping promptly, and Vítor Silva for all the mutual support during this last year. It really was a pleasure meeting and working with both of you.

I also want to acknowledge MDO Lab at the University of Michigan, particularly to Professor Joaquim Martins for providing access to the tools and all the documentation and to Dr. Eirikur Jonsson for taking the time to clarify and help with questions about the framework.

I would also like to thank Tekever UAS for not only funding and supporting the production of this work by providing all the needed data but also for the opportunity.

I am also extremely grateful to my parents, for always supporting me and ensuring I always had everything I needed throughout this journey, and my to my brother for always being there.

A special thanks also goes to all my friends who supported me during all this years and hopefully for years to come.

Finally, I would like to thank all the individuals, institutions and teams with which I was involved with during all this years and who have been a significant part of my student journey and all the professors who, through their professionalism, have left a mark on my journey.

Resumo

O mercado dos UAV é altamente competitivo, com o lançamento frequente de novos produtos e uma vasta gama de soluções já disponíveis, obrigando os fabricantes a um desenvolvimento mais rápido e eficiente que nunca. Uma abordagem económica é o desenvolvimento de novas gerações de um produto usando novas tecnologias e ferramentas de projeto. Algumas destas ferramentas incluem métodos de mecânica de fluidos computacional de alta fidelidade baseados em RANS e otimização com o método das variáveis adjuntas, usados neste trabalho numa ferramenta de otimização numérica para explorar a otimização aerodinâmica de uma asa, como parte do desenvolvimento da nova geração de um UAV para um fabricante português. É efetuada uma análise da asa do UAV atual, incluindo interferência com a fuselagem, e seguida de um processo de otimização com vista a minimizar a resistência para um coeficiente de sustentação prescrito. São consideradas três geometrias iniciais e parametrizadas com variáveis de projeto comuns, incluindo distribuição de torção e corda, flecha, diedro e forma do perfil aerodinâmico. O uso de duas asas simples como ponto de partida permitiu a verificação e configuração da ferramenta, com todas as otimizações testadas a aproximar a distribuição de circulação elíptica embora não exatamente tendo em conta os compromissos necessários entre resistência viscosa e de pressão. Desafios associados ao uso de uma geometria de asa complexa como ponto de partida são abordados e são efetuadas otimizações partindo da asa do UAV atual. Uma redução da resistência de $4,5\%$ é obtida considerando todas as variáveis de projeto.

Palavras-chave: Otimização aerodinâmica, otimização baseada em gradientes, método das variáveis adjuntas discreto, projeto de asa, mecânica de fluídos computacional, deformação de forma livre

Abstract

The UAV market is currently very competitive, with the frequent release of new products and a wide range of solutions already available, forcing manufacturers to explore the design space faster and more efficiently than ever. A cost effective approach is to develop growth versions, improving an existing product with new technologies and design tools. Some of these tools include RANS based high fidelity computational fluid dynamics methods and discrete adjoint gradient-based optimization, which are used in this work on a numerical design framework to explore the aerodynamic shape optimization of a wing, as part of the development of a growth version of a UAV of a leading Portuguese manufacturer. A comprehensive aerodynamic analysis of its current wing, including fuselage interference, is performed, followed by an optimization procedure to minimize drag subject to a prescribed lift coefficient constraint. To that end, three different starting geometries are considered and parameterized with common design variables, including twist and chord distributions, sweep, dihedral and airfoil shape. The use of two simple wings as starting geometries allowed the framework and set-up verification, with all optimizations considering different sets of design variables approaching an elliptical lift distribution, although not exactly considering the trade-offs needed between viscous and pressure drag. Challenges associated with the use of a complex wing geometry as a starting point are then addressed and optimizations considering the current UAV wing as a starting point are performed. Notably, a drag reduction of 4.5% is achieved considering all design variables.

Keywords: Aerodynamic optimization, gradient-based optimization, discrete adjoint method, wing design, computational fluid dynamics, free-form deformation

Contents

List of Tables

List of Figures

Glossary

3D	Three-dimensional
ADF	Auto-lead Data Format
AD	Automatic Differentiation
ANK	Approximate Newton Krylov
ANSI	American National Standards Institute
BFGS	Broyden–Fletcher–Goldfarb–Shanno algorithm
CAD	Computer Aided Design
CFD	Computational Fluid Dynamics
CGNS	CFD General Notation System
CPU	Central Processing Unit
D3ADI	Diagonalized diagonally dominant alternating direction implicit scheme
DADI	Diagonalized Alternating Direction Implicit method
DNS	Direct Numerical Simulation
DoF	Degrees of Freedom
FDM	Finite difference method
FEM	Finite element method
FFD	Free-form deformation
FVM	Finite volume method
GA	Genetic Algorithms
GMRES	Generalized Minimal Residual Method
HDF5	Hierarchical Data Format version 5
HLFC	Hybrid Laminar Flow Control
IGES	International Graphics Exchange Standards
ISO	International Organization for Standardization
ISR	Intelligence, Surveillance and Reconnaissance
LES	Large Eddy Simulation
MAME	Medium Altitude Medium Endurance
MDO	Multidisciplinary Design Optimization

MTOW	Maximum Take-Off Weight
NACA	National Advisory Committee for Aeronautics
NK	Newton Krylov
PDE	Partial differential equation
PETSc	Portable, Extensible Toolkit for Scientific Computation
PSO	Particle Swarm Optimization
PTC	Pseudo-Transient Continuation
RAM	Random Access Memory
RANS	Reynolds-Averaged Navier-Stokes
RK	Runge Kutta
SA	Spalart-Allmaras turbulence model
SLSQP	Sequential Least Squares Programming Algorithm
SNOPT	Sparse Nonlinear OPTimizer
SQP	Sequential Quadratic Programming
SST	Shear Stress Transport
STEP	Standard for the Exchange of Product Data
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
XDSM	Extended Design Structure Matrix

Chapter 1

Introduction

This first chapter is intended to provide a short overview of the evolution of Unmanned Aerial Vehicles (UAV) throughout the years, with special focus on recent times. With this, the motivation for this work is established, as part of a paradigm shift in the way UAV design is performed.

1.1 UAV evolution

UAVs have been around for more than a century [1]. Their development started taking off in the military industry in the form of aerial torpedoes and flying bombs [2], with the first successful demonstration being commonly attributed to the Hewitt–Sperry Automatic Airplane. For the initial development, many new technologies played an important role, such as gyroscopic devices and radio control [1], but major developments went hand in hand with the aviation industry, a branch that was still very recent in itself, which sometimes hindered up their development.

Although still under the military industry control, new applications started to be recognized for this type of vehicle, namely as a target drone for anti air-gunnery practice and as a way to gather intelligence, surveillance and recoinassance (ISR). This last application was used extensively in several wars during the 20th century and eventually became the main one [3], as the system could provide real-time information that could be used to support strategic planning without endangering the pilot life. The most notable example in this regard is the Ryan BQM-34A ”Firebee”, a jet-propelled, remotely controlled and recoverable UAV developed and built by Ryan Aeronautical Company in 1950 that was firstly used as a target drone for training purposes but later adapted to an ISR platform that performed missions over USSR [4]. Despite having been designed during a time when the industry was dominated by military applications, with very specific missions and requirements in mind, the ”Firebee” continues to inspire many modern UAV designs, several generations later.

Nowadays, UAVs can be found performing an ever increasing number of tasks, including real-time monitoring, remote sensing, search and rescue, delivery of goods, aerial photography, security and surveillance and civil infrastructure inspection [5], leading to an increasingly perceived usefulness and convenience of those vehicles in the civil applications, leading to a steady increase in sales every year.

According to recent estimates [6], the UAV industry is worth $US\$30.6$ billion as of 2022 and is projected to be worth $US\$55.8$ billion by the end of the decade, with Europe positioned to be one of the main markets. Figure 1.1 shows the expected evolution for a ten years period, and the expected division of the market by continent.

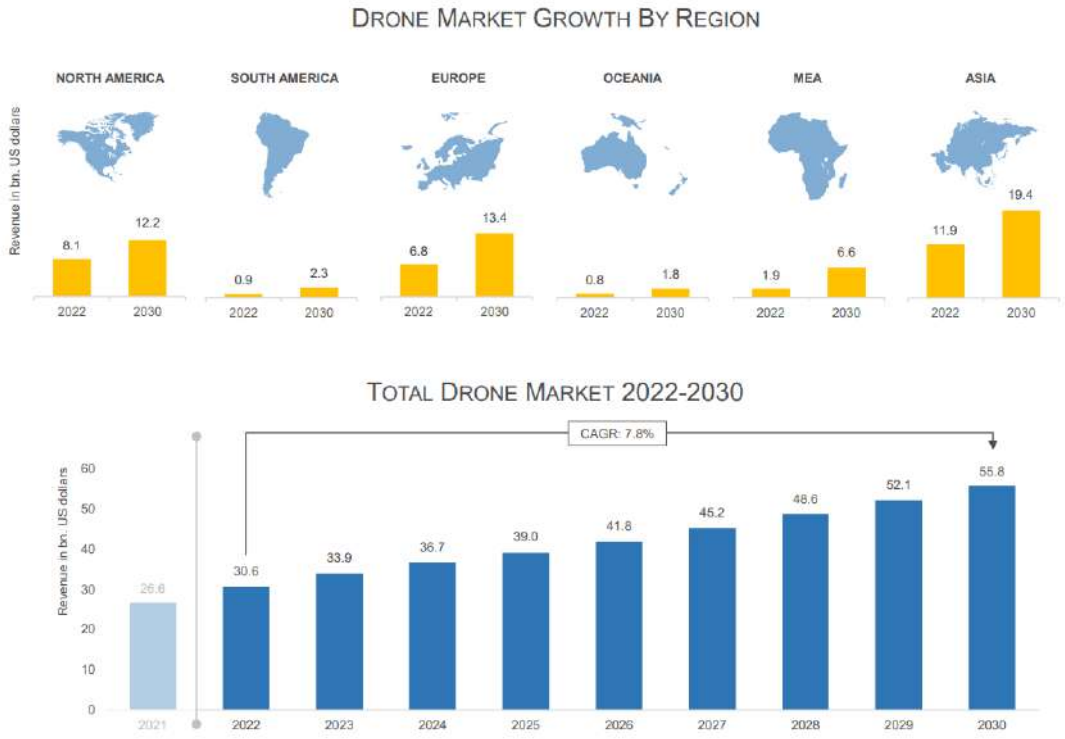


Figure 1.1: Projected global drone market growth for the next decade. (Source: Drone Industry Insights [6])

This puts the UAV market as one of the fastest growing and dynamically developing sectors of the world industry today, which naturally attracts investors attention. Proof of this are the records from the previous decade, shown in Figure 1.2, where it can be seen that the investment in this sector almost doubled from 2019 to 2020 and more than doubled from 2020 to 2021, with $US\$6.96$ billion invested into the industry during that year [6]. These large amounts of money flowing into the industry, however, need to be shared between an ever increasing competition. In 2019, Drone Industry Insights listed 945 key market players [7], meaning that nowadays, in order to capture external investment, companies need to add value by presenting differentiated and innovative products.

Other important drivers widening the design space in the industry are the range of tasks performed by UAVs nowadays, resulting in totally different sets of mission requirements arising for different products, a smaller set of constraints present when designing an UAV, being them geometric (in particular, for being unmanned), or from the relaxed set of regulations when compared with the design of a conventional manned aircraft [8] and to the opening of the industry to the civil market. All those imply that using a conventional design that worked well during the first decades may not be appropriate anymore.

The expanded design space available for the UAV manufacturers resulted in many different configurations presented to the market in the last years, which were strongly influenced by the drivers described

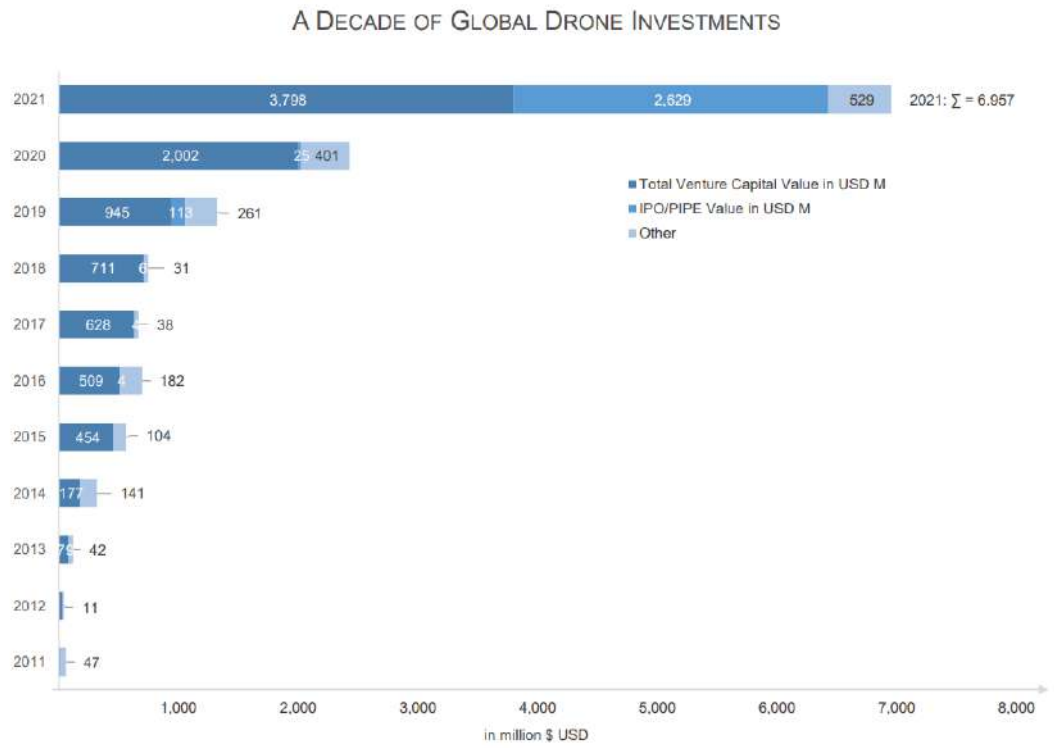


Figure 1.2: Investments in the global drone market during the last decade. (Source: Drone Industry Insights)

above. Some of them can be seen in Figure 1.3. These configurations reflect the innovative way each manufacturer found to tackle a specific market need and achieve specified performance indicators that enable their products to perform the respective missions.

(a) General Atomics MQ-9 Reaper. (Source: General Atomics)

(b) Tekever AR5. (Source: Tekever)

(c) UKRSpecsystems PD2. (Source: UKRSpecsystems)

Figure 1.3: Some of the recent UAVs.

In the conventional aircraft market, the project of a new aircraft can take years, with several iterations between the different disciplines, such as structures, aerodynamics and propulsion, and experts from the different fields required to have a strong presence in the entire process until convergence to a final design is achieved. However, as the presented data shows, the UAVs are currently a very fast paced market, forcing companies to explore the broad design space as quickly and efficiently as possible, reducing the design cycle time to obtain a product with better performance ahead of their competitors. This is where new design approaches, such as Optimal Design and Multidisciplinary Design Optimization (MDO) are

needed.

MDO is a methodology aimed for the design of complex systems with distinct but interacting physical phenomena through the use of numerical optimization [9], suiting well the aircraft design process. Expert input will always be needed. However, MDO allows a coupled analysis of the system considering the different constraints and interdisciplinary trade-offs all at once, allowing for an exploration and narrowing of the broad design space in a much more efficient way so that optimal designs can be found quicker, greatly reducing the design process time.

Although MDO have been around for some time, it had to surpass a series of obstacles in the past few years which prevented its widespread use in the industry. In the aerodynamic optimization domain in particular, some of those obstacles included solver robustness, scalability with the number of design variables, efficient and accurate gradient computation, mesh deformation and availability of specialized software [10]. Besides being one of the main subsystems of interest in an MDO approach of a full aircraft, aerodynamic optimization can also be performed in an uncoupled manner, which will be the main focus of this work.

Thanks to efforts in academia and research laboratories the majority of those problems were minimized or completely tackled, taking the approach to a point where it is starting to slowly be adopted by the industry. Figure 1.4 shows a particularly interesting application of aerodynamic optimization, from University of Michigan, where a supercritical profile is obtained from a sphere given as the initial shape in a matter of minutes [11].

Figure 1.4: Aerodynamic optimization of an airfoil for transonic flow, starting from a sphere. (Source: MDOLab, University of Michigan)

1.2 Framing and context

In this work, it is intended to explore one of the available frameworks from academia to perform aerodynamic optimization, in particular the MACH-Aero, developed by the MDO Lab at the University of Michigan [10].

MACH-Aero uses ADFlow [12] as a flow solver, which is a parallel solver for Euler and RANS equations with geometries discretized by structured grids that allows for the efficient computation of the gradients using adjoint methods, allowing for an efficient aeordynamic optimization of three-dimensional shapes given a set of constraints and an objective function [13].

This work is part of a collaborative project between industry (Tekever UAS) and academia that intends to explore aerostructural design optimization of UAVs using high-fidelity numerical tools, taking this new approach directly to the industry. For that, there is a need to develop in-house numerical aerodynamic and structural analysis and design capabilities, creating the need to establish a numerical high-fidelity aerostructural analysis and design framework. This approach will then be applied to the performance enhancement of the current Tekever AR5 UAS platform (see Figure 1.3 b)), a system currently on the market and mainly used for ISR applications and, on a later stage, to the development of future platforms.

To build the general analysis and design framework, there is a need to first develop and validate several of its components, namely, the aerodynamic and structural numerical tools. Figure 1.5 schematizes the overall project timeline. The blue boxes, representing the decoupled aerodynamic wing optimization, are the object of study of this work, which will evolve for aerostructural design of an entirely new fixed-wing Medium Altitude Medium Endurance (MAME) UAV over the span of three years.

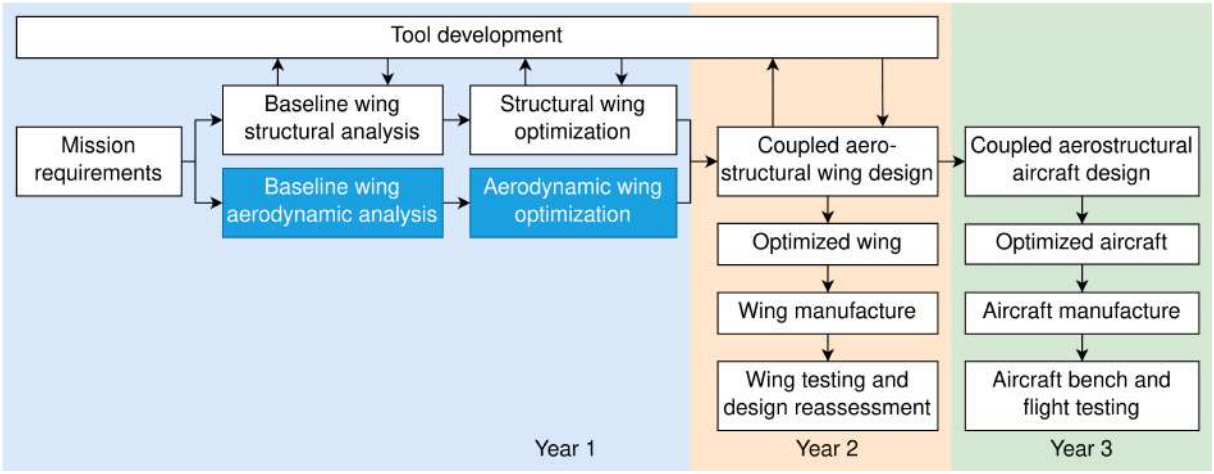


Figure 1.5: Overall project timeline.

1.3 Objectives and deliverables

This work aims to set a framework for high fidelity aerodynamic design optimization for wings/ aircraft that can be used in the industry and verify the results by optimizing the current wing of an existing UAV. MACH-Aero will be used for the aerodynamic optimization process. However, there is the need to set up with additional software for pre- and post-processing in an integrated manner to create a streamlined process.

The framework will be tested by firstly performing some grid studies and then analyzing the current Tekever AR5 wing and the wing-fuselage assembly and then by setting several optimization problems with the goal of minimizing drag subjected to a prescribed lift coefficient for different sets of design variables and different starting geometries of increasing complexity.

The simpler geometries will be used to test the process with more straightforward parameterizations and the challenges associated with the more complex Tekever AR5 wing will be discussed and tackled.

For this, some code will be modified and developed to generate high-quality body fitted Free-Form Deformation (FFD) boxes for arbitrarily complex geometries and to create new parameterizations to ensure meaningful design variables for the complex geometry. With this, the optimization of the Tekever AR5 wing will be performed and benchmarked against the original geometry.

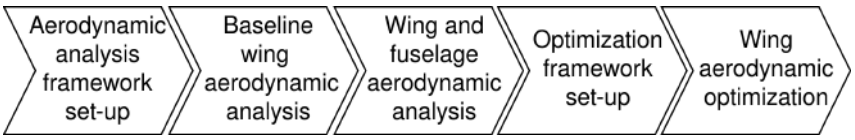


Figure 1.6: Process and deliverables of this work.

1.4 Thesis outline

After an introduction about the main topic, motivation and framework for this work, the remaining document is divided in the following way:

Chapter 2 introduces the aircraft design process with an overview of current methodologies, as well as the adaptions needed for UAVs. It then gets more specific for the problem at hand, focusing on aerodynamic design, particularly for a wing, finishing with a brief discussion about optimal design.

Chapter 3 gives a theoretical overview about several CFD aspects, including different fluid flow and turbulence models, equation discretization techniques, meshing and solver algorithms. With this discussion, the methods to be used in the remaining work are chosen and justified.

Chapter 4 presents a similar approach for optimization. Methods for optimization and for sensitivity analysis, as well as parameterization strategies are discussed and selected.

With the model choices justified, the implementation details required to perform a complete aerodynamic analysis and optimization are presented in Chapter 5 along with the respective tools, following a logical order: geometry definition; surface, volume and overset meshing; flow simulation; optimization framework, including parameterizations and constraints set-up and post-processing.

Chapter 6 presents the current Tekever AR5 wing aerodynamic analysis to check its performance. Some grid studies are performed and the wing performance is analyzed not only for cruise but also for a range of angles of attack. The influence of the fuselage on the flow is also studied.

Finally, on Chapter 7 results are shown and discussed for wing aerodynamic shape optimization starting from three different initial geometries: a rectangular wing, a simplification of the Tekever AR5 wing and the Tekever AR5 wing and considering different sets of design variables.

Chapter 8 presents the conclusions for this work, as well as some future work suggestions.

Chapter 2

Aircraft Design

In this chapter, an overview of the aircraft design process is provided. Firstly, the process as a whole and the current methodologies are presented. After that, it focuses on the aerodynamic design, with particular emphasis on the wing.

2.1 Conventional methodology

The conventional aircraft design methodology has been extensively described in the literature, by authors such as Roskam [14], Raymer [15], Torenbeek [16] and Anderson [17]. The process described, although initially developed for manned aircraft, is also commonly applied to UAV design, including those of unusual configurations, like those in Figure 2.1, provided that small adjustments are made to the process, as will be discussed further. Some examples of that design process applied to unusual configurations were discussed for a solar powered UAV [18], lift fan [19], tilt wings [20], and a tailless UAV [21]. Literature specifically directed towards design of UAV is also available in books such as [22, 23].

The complete development of a new product starts before the design phases, with the identification of a market need by a certain manufacturer, being the identification via direct market analysis and study of the competition or by a customer requirement. The identified need, together with top level management requirements, such as cost goals and the value of the new vehicle for customers, requirements derived from applicable regulations and even the technology state of art then generates a set of initial specifications and requirements for the vehicle to be developed.

The requirements encompass a lot of information about the new aircraft, including but not limited to basic information about the new vehicle, such as its purpose, the time frame of development and the performance requirements, which include parameters such as payload, range, speed and take-off field length. With the initial set of requirements defined, the manufacturer is capable of proceeding with the design of the vehicle, which is divided in three major phases, during which experts from different fields are involved and the level of detail is successively increased: conceptual design, preliminary design and detailed design.

During the conceptual design phase, the manufacturer starts by analyzing the defined set of require-

(a) Tailless UAV - Lockheed Martin X-44A. (Source: Lockheed Martin Corporation)

(b) Solar Powered UAV - Airbus Zephyr. (Source: Airbus SE)

(c) Ducted fan UAV - Boeing Phantom Works Swift. (Source: Boeing Company)

(d) Tilt rotor UAV - Rainbow CH-10. (Source: China Aerospace Science and Technology Corporation)

(e) NextGen Aeronautics MFX-2. (Source: NextGen)

(f) Aurora XV-24. (Source: Aurora Flight Sciences)

Figure 2.1: Some unusual UAV configurations.

ments to determine which ones are relevant for the design and if they can produce a viable and salable UAV. The requirements also help defining a rough idea of what the vehicle should look like and what trade offs are needed, after looking at a wide range of aircraft configuration concepts. If the initial set of requirements do not align with the manufacturers interests, they are changed and iterated successively until there is a viable set of requirements. In this phase, high level goals, such as weight and cost, can be defined.

Then follows the preliminary design phase, where the selected concept is refined, surfaces start to be defined and major items start to be designed. In this phase, it is already possible to statistically estimate the cost of the new aircraft which allows the sales effort to be started, marking the commitment of the company to the project. The result is a freeze in the overall design arrangement so major design changes are not expected anymore.

The detailed design follows, being the last phase of the design process, where all the parts composing the aircraft are designed in a final manner and its determined how they will be fabricated. Major items are also tested in this phase. As the design of each part is detailed, the vehicle weight and performance estimates are determined. This phase may imply some changes in the aircraft specification, as the company realizes that some parts may be unfeasible.

After completing those three major design phases, the aircraft can move into production. However, the project is still far from done. The design of a new UAV implies its flight test and, later, its operation. During operations, inputs from clients will be taken into account and, together with company drive to do so, growth versions usually appear, and the aircraft design restarts again, but now from the preliminary design, as major conceptual changes are not possible anymore. Figure 2.2 shows a schematic of the

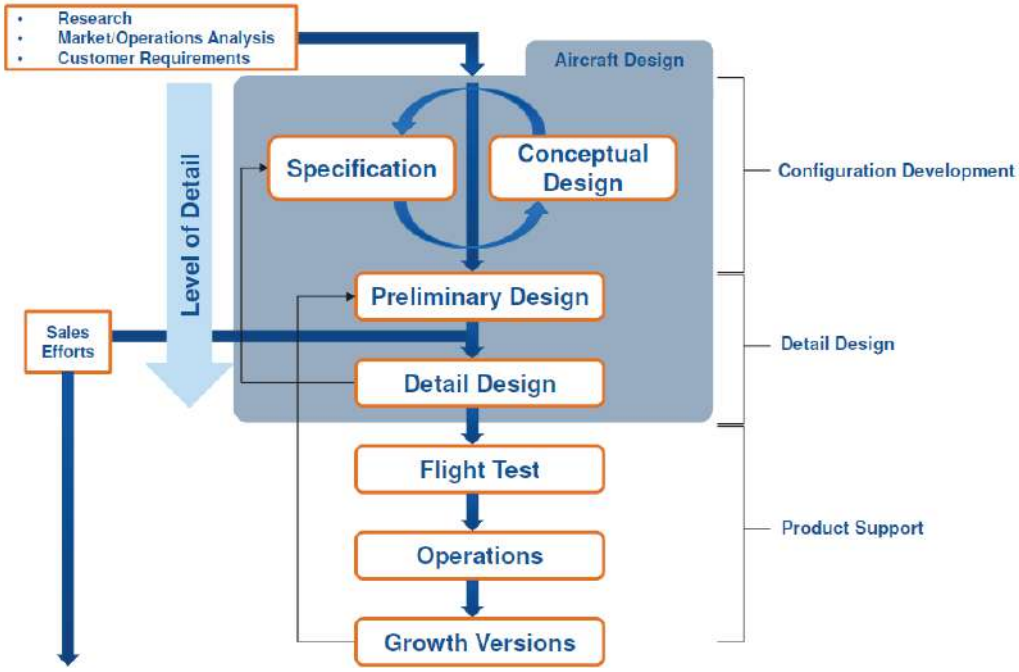


Figure 2.2: Aeronautical project process fluxogram. (Source: Torenbeek, E., Advanced Aircraft Design)

entire process.

It is important to note that, as the growth versions design process starts directly from the preliminary design, it is possible to obtain an UAV with improved performance, relying on previous know-how and new technologies without having to start the entire process again, particularly without the initial iterative loop between the conceptual design and specifications. This is often more advantageous for a company from an economical point of view than starting a new design from scratch, particularly if it is intended to perform the same kind of mission as a previous design already performed. Furthermore, the growth version does not have to go through the entire certification process again offering a better cost-benefit ratio for the manufacturer. For the operator, growth versions are also often more profitable too, as operation costs scale down when compared to an all new UAV. This work will focus on the possible development of a new growth version of the Tekever AR5, which already have a past one, with improved performance due to a new optimized wing.

All described, design phases are in fact divided into two components: the design layout, which includes drafting and modeling in Computer Aided Design (CAD) software, and the design analysis, in which the design must be verified via increasingly higher fidelity calculations and testing to assess performance and failure criteria and obtain a final product where the goals are fulfilled. Traditionally, this second step has been done manually, where the process is stopped when a satisfactory design is obtained. If this is not the case, the design is changed based on the aircraft designers intuition and heuristics. In smaller companies, the entire process may be performed by a single person, which does the design, analysis and optimization [15]. However, in larger projects this design phase includes one or more aircraft designers, dealing with the design layout, and several experts for each discipline, namely aerodynamics, structures, propulsion and so on, which are seen as different modulus of the project. The design is then iterated between them and there is a strong need for communication [15].

2.2 UAV design

Despite the previously described process being applicable to UAVs almost entirely, the correlations commonly found in literature which are useful in the conceptual design phase refer to manned aircraft. Those correlations are not the same for UAVs as a result of different mission requirements and constraints. However, it is still possible to establish correlations between several variables, as some recent research for the conceptual design of a fixed wing UAV shows [24]. An example of the obtained correlations between payload and maximum take-off weight (MTOW), and MTOW and wingspan, can be observed in Figure 2.3, and it can be verified that they fit the Tekever AR5. They were based on available data for 856 different fixed-wing platforms, primarily sourced from "Jane's All the World 's Aircraft: Unmanned", separated by propulsion type (piston, battery, fuel cell, turbine or solar), outlining the interdependences in certain design parameters between them from the obtained statistical trends. From the derived correlations in the form of a power law, a strongly linear correlation was observed when using a logarithmic scale. As the publication is still recent, it also provides a market overview of the available solutions. However, they are not separated by mission type and only include fixed wing solutions.

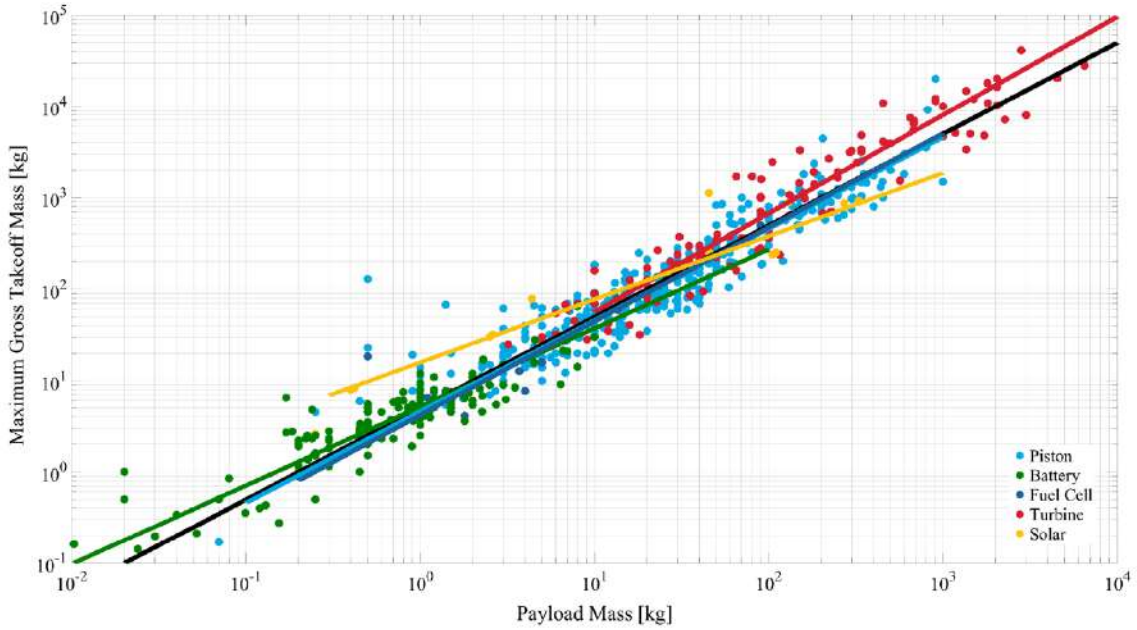
Figure 2.4 presents a smaller market overview only for some ISR solutions available, and it is not limited to fixed-wing solutions. The list of UAVs considered can be found in Table A.1 in Appendix A. It is important to note that there are not enough solutions available to derive meaningful conclusions but, as the systems needed onboard for UAVs of this type are different from those needed to perform other missions, some differences should be expected between them. From this figure, it is possible to see mainly two groups of UAVs: one with lower endurance and ceiling, and another with higher endurance and ceiling, allowing for longer range surveillance missions. It can also be observed that there is a tendency towards an endurance of 20 hours, indicating that it may be the maximum duration of a common surveillance mission. Finally, it is possible to see that Tekever AR5 lies approximately in the middle for both endurance and ceiling, emphasizing its classification as a MAME UAV.

2.3 Mission requirements

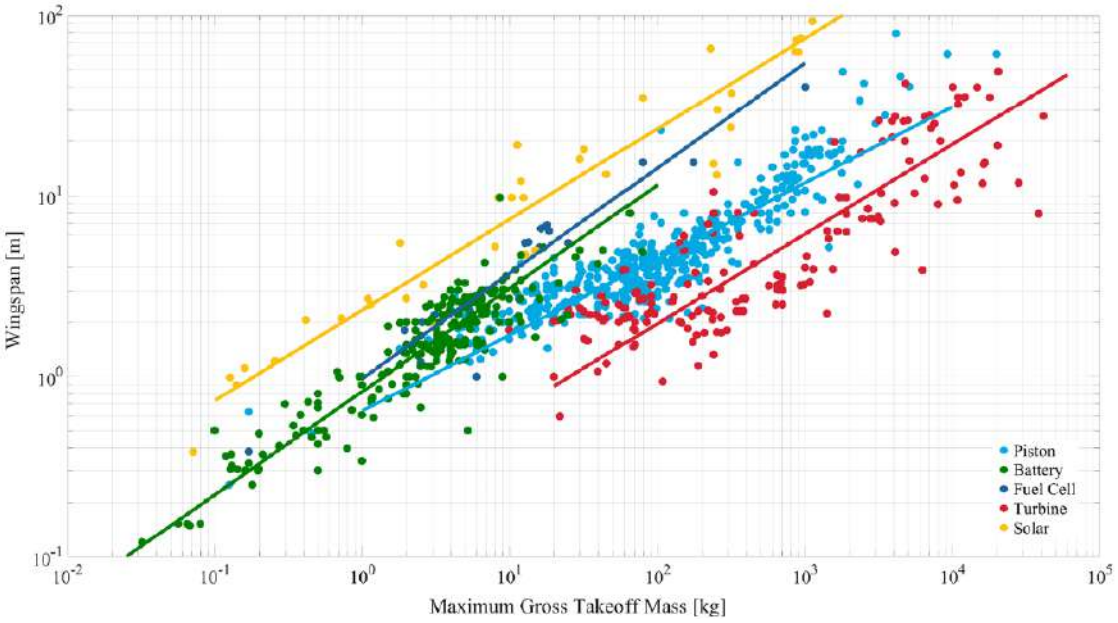
As discussed previously, a critical phase during aircraft design is the definition of mission requirements, as they influence the entire design process. Mission requirements define thresholds for several performance parameters to ensure a successful completion of the mission the UAV was designed for. From these requirements, using empirical correlations such as those found previously, it is possible to obtain estimates of some parameters to start the conceptual design phase.

The requirements to be considered are a decision of the manufacturer and they are project specific, but some common ones include:

- Maximum range: The maximum distance that the aircraft can fly between takeoff and landing. It determines the spatial operational window. It is usually limited by the energy the UAV can carry onboard and its ability to obtain energy in flight (as it is the case for solar powered UAVs for example) or by communication systems range (unless satellite communications are used). Range



(a) Correlation between payload mass and MTOW



(b) Correlation between MTOW and wingspan

Figure 2.3: Obtained correlations for UAVs. (Source: [24])

can usually be derived from the Breguet equation [25], which is also influenced by the specific fuel consumption of the propulsion system and the aerodynamic efficiency;

- Endurance: The maximum time that the aircraft can fly in cruise flight and thus determines the temporal operational window. It is related with range, but high endurance does not necessarily mean high range, as it depends on the flight speed and on the amount of energy that the aircraft can carry onboard. Endurance is typically more desirable for an ISR UAV than maximum range. However, this largely depends on the specifics of each mission;

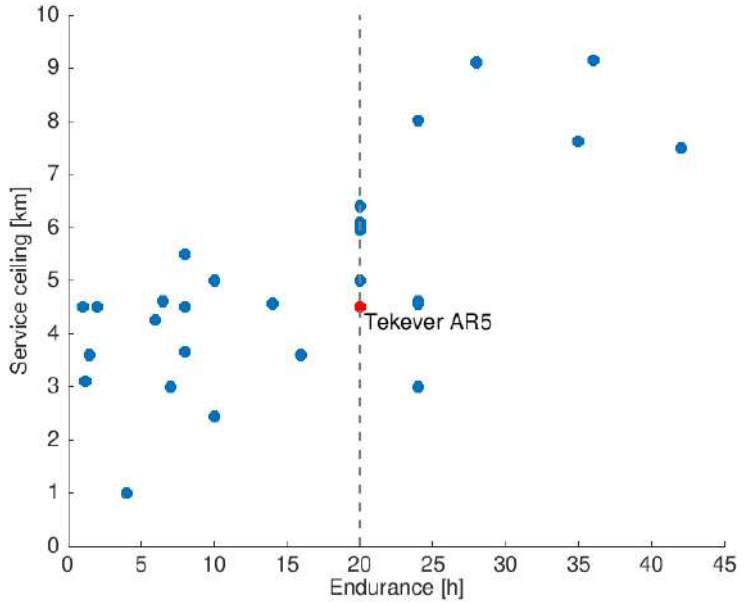


Figure 2.4: Service ceiling with endurance for current surveillance UAVs

- Maximum ceiling: The maximum altitude at which a certain aircraft can operate. It is usually defined by the air density, as it influences the lift that can be produced by the aircraft at a certain speed. Thus, it is limited by the maximum thrust that the engines can produce, the lift coefficient of the aircraft and the achievable speed, which dictates the minimum density were the aircraft is able to sustain its own weight. For certain engine types, namely combustion engines, altitude also influences engine performance, which should be taken into account. This value is important for example in some surveillance missions, as stealth aircraft require higher operating altitudes to avoid detection;

- Maximum payload: Refers to the maximum additional weight that an UAV can carry and, for an UAV, generally includes additional systems to perform a specific mission. For ISR applications, those systems typically include gimballs with cameras and additional sensors (some examples include high definition cameras, thermal cameras, iluminators, rangefinders, scientific equipment or ammunition in case of combat UAVs). Increasing the payload of an UAV also decreases its range, so both are typically related by a payload-range diagram provided by the manufacturer. In ISR applications, payload is typically not a critical parameter. However, additional payload provides margin to upgrades on the carried cameras and sensors and allows the flexibility to provide different missions at the operator desire;

- Maximum take off weight (MTOW) is the maximum weight that an UAV can have at take-off conditions and it is usually defined by structural airworthiness requirements, so that the aircraft structure can withstand the loads during flight and it meets the performance requirements, such as climbing rate. As shown in Figure 2.3, this requirement is strongly correlated with the UAV dimensions (typically represented by its wingspan) and the payload capacity.

2.4 Wing geometric parameters

The wing plays a major role in the design of a typical aircraft as it defines a significant percentage of its empty weight and creates the majority of the lift. This is also the case for fixed wing UAVs, playing a central role in the entire UAV design.

Considerations about the wing start during the conceptual phase, where the basic layout of the UAV is defined. To start defining the main wing parameters, it is important to have an estimation of the aircraft gross take-off weight, often based on historical data and mission requirements. This estimation allows to have a rough idea of the basic aerodynamic parameters of the vehicle using analytic methods and empirical correlations. Those parameters may include airfoil characteristics and wing geometric parameters. The design is then successively refined and detailed in every iteration, from which positions and shapes of the several aerodynamic devices, including the wing, can be defined. With the geometry, it is possible to obtain better estimates for the aerodynamic parameters.

When designing a wing, there are several parameters that can be changed. Generally, the wing is composed of an airfoil (two-dimensional) that is distributed on a certain planform (three-dimensional). The airfoil should be choosen having a certain application in mind, and thus, the expected operating conditions should be derived beforehand. Of particular importance to define the airfoil is the speeds and regime where the aircraft will operate at. Those operating conditions are typically derived in terms of non-dimensional parameters, such as the Mach number and the Reynolds number.

When it comes to the planform of a wing, it can be defined by several parameters, which will influence the lift distribution, and thus, the wing efficiency. For inviscid flow, the best theoretical lift distribution is elliptical, as it causes the downwash along the wingspan to be constant, minimizing the wingtip losses and thus, minimizing induced drag. For viscous flow, trade-offs with skin-friction drag need to be considered, but the optimal shape is expected to be close to the elliptical one. There are several ways of achieving such a distribution. Those parameters include the following:

- Chord (c): The distance between the trailing edge and the leading edge of a wing section. Except for rectangular wings, it usually varies across the wingspan.
- Wingspan (b): The distance from tip-to-tip of a wing. Varying the chord along the wingspan it is possible to obtain an elliptical lift distribution without any additional parameter;
- Aspect ratio ($\mathcal{AR}$): Given by $\mathcal{AR} = \frac{b^2}{S}$, where b is the wingspan and S is the wing area. For rectangular wings, it is the ratio of wingspan to mean aerodynamic chord. It is strongly coupled to the aerodynamic efficiency (L/D) of a wing and different aspect ratios can be found in aircraft designed for different missions;
- Sweep angle (Λ): The angle between the lateral axis and a chord line of the wing. The quarter-chord line is commonly used to define sweep angle, given that this is usually the approximate location of the aerodynamic center, where the pitching moment does not change with angle of attack. A positive sweep angle helps to mitigate transonic effects and is thus frequently used for commercial aircraft. However, it is not relevant for subsonic UAVs;

- Dihedral angle (Γ): The angle between the horizontal axis of the UAV and the wing. It is related to the dynamic stability provided by the wing, namely the roll moment produced with sideslip;
- Taper ratio (λ): The ratio between the wing chord at the root and at the tip. It is one for rectangular wings, but different from one to all other types of wings. It results from a trade-off between ease of manufacturing and aerodynamic and structural performance. Introducing taper, it is possible to approximate the wing to an elliptic lift distribution while still being relatively easy to manufacture;
- Twist angle (θ): A wing twist changes the local angle between the wing and the flow, thus it changes the lift distribution. This may be used for two reasons: to obtain an elliptical lift distribution; to tailor stall characteristics by reducing the incidence angle from root to tip of the wing, ensuring that the root region stalls before the tip and thus maintaining aileron authority. This distribution is known as washout.

Some parameters for the wing used in the Tekever AR5 are shown in Figure 2.5 and Table 2.1.

Furthermore, other aerodynamic devices may be added to the wing, such as winglets, vortex generators and high lift (such as slats and flaps) and control devices (such as ailerons and spoilers). However, in this work, the wing to be optimized will be regarded as a clean wing without deflected movable surfaces, representing the cruise condition.

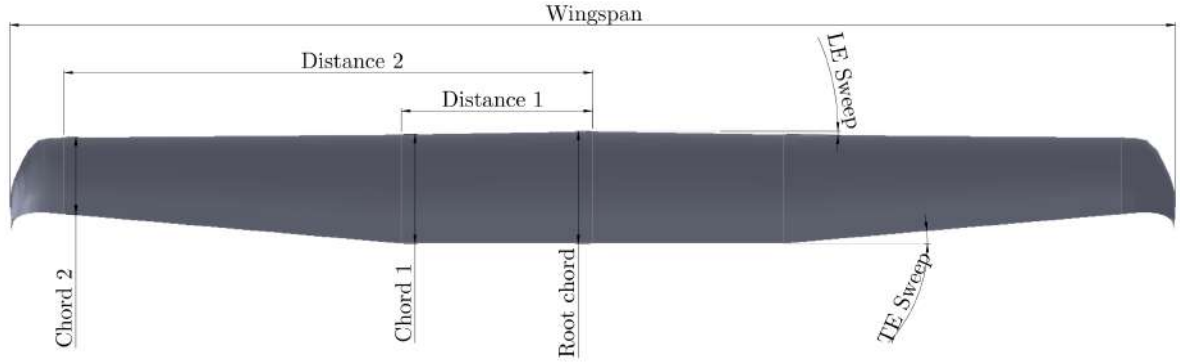


Figure 2.5: Planform of the Tekever AR5 wing.

Table 2.1: Some geometric parameters of Tekever AR5 wing.

Wingspan	b	7.565 m
Projected wingspan	b_{proj}	7.246 m
Root chord	c_{root}	0.70 m
Projected area	S_{proj}	2.169 m^2

2.5 Overview of wing designs

When it comes to the overall shape, wings may be classified as rectangular, tapered straight, elliptical (which meant to provide elliptical lift distribution through an elliptical shape), swept wings, which may or

may not be tapered or delta wings, which are triangular shaped and used mainly for supersonic aircraft, which is not the case in this work.

Despite most of the current UAVs being fitted with the conventional fixed wings, there are other more recent wing designs (most of which are still topics of research) that have potential to offer better aerodynamic performance but require different design methodologies. Some examples of that are provided.

Variable swept wings are wings fitted with a tilting mechanism that allows them to change the swept angle during flight and thus keep the optimal angle for each flight condition. However, the added complexity is usually not worth it and they are only found in some supersonic and transonic fighter jets. The design process for such wings has been discussed in the literature [26].

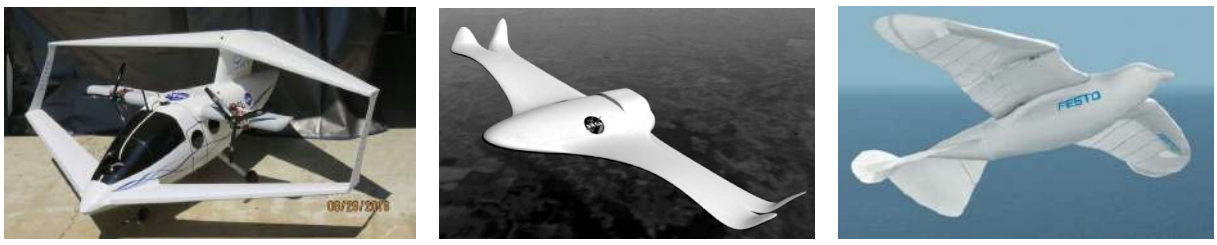
Closed wings usually describe two wing planes merged on their tip, as illustrated in Figure 2.6 a) avoiding wing tip losses due to induced drag and increasing the stiffness, bringing not only aerodynamic but also structural advantages. Those can be divided further based on their shape into box wings, annular box wings, annular cylindrical wings, annular planar wings and joined wings. The aerodynamic advantages of this kind of wings have been previously analysed by some authors [27].

Wings may also be flexible, when manufactured with a thin membrane. They are however most suited to sail planes and kites and see little to no application in UAVs. However, there have been some recent research in shape changes deliberately introduced in a wing during flight to change its shape depending on the flight conditions as illustrated in Figure 2.6 b). Those are known as morphing wings, and a discussion about their feasibility can be found in [28].

Some blended wing UAVs also appeared in the past years. They greatly differ from the shell and tube structure, having a smooth transition between the wing and the fuselage, reducing wetted area and interference drag on the junction between the wing and the fuselage, and thus improving the overall efficiency of the UAV. A review of the current progresses in that domain can be found in [29].

Nowadays, there is also active research in the domain of hybrid laminar flow control (HLFC) on wings, which intends to keep the flow laminar for as long as possible while avoiding separation, and thus reducing the higher drag caused by the turbulent flow. A state-of-art review can be found in [30].

Even more unusual configurations can be found, such as flapping wings that mimic the movement of birds wings, as illustrated in Figure 2.6 c) which, in theory, should be able to reduce the needed area to create lift. They also have the advantage of providing thrust and lift in the same movement. However, the performance advantages of this technology over common wings are yet to be proven. In [31] a review of the current progresses and challenges is provided.



(a) Closed wing. (Source: Elytron) (b) Morphing wing. (Source: NASA) (c) Flapping Wings. (Source: Festo)

Figure 2.6: Some less conventional wing designs.

2.6 Evaluation parameters

When it comes to the aerodynamic performance of a low-speed aircraft, there are several parameters which are important to take into account, namely:

- Aerodynamic efficiency (L/D). As lift and drag varies with angle of attack, aerodynamic efficiency also varies, thus it is usually referred to at its maximum value, along with the corresponding angle of attack. It can be improved by either reducing the drag produced for a certain lift, or increasing the lift produced without changing drag. It is also influenced by the Reynolds and Mach number, related to the flight condition. This quantity is present in the Breguet equation [25], influencing the range that a certain UAV can fly in a linear way, directly impacting the mission requirements;

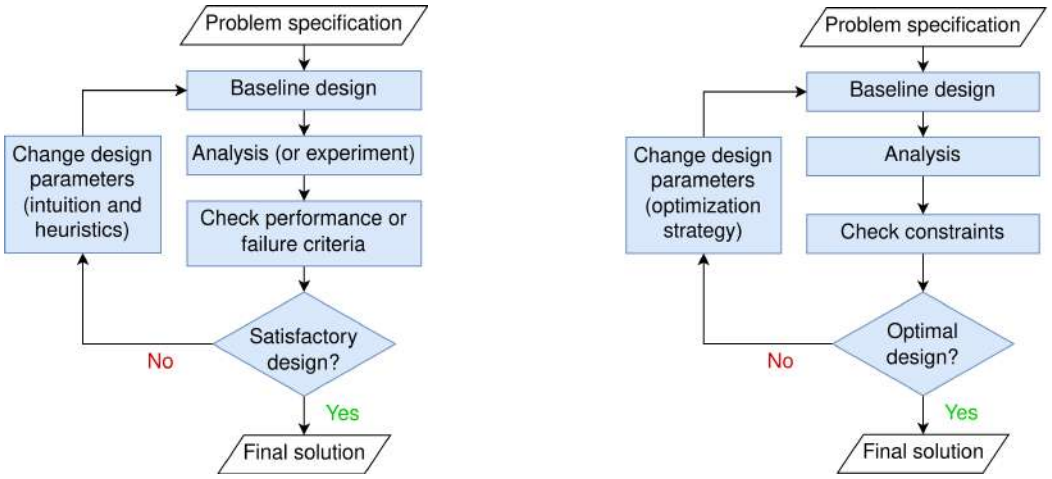
- Lift coefficient, C_L. This parameter assesses the total lift generated by the wing and for low angles of attack, it can be written as C_L = C_{L_0} + $C_{L_\alpha} \cdot \alpha$. C_{L_α} is the lift variation with angle of attack and it is useful to assess how much lift coefficient can be obtained for a certain angle of attack. Both parameters should be defined in such a way that on its standard mission, the aircraft flies with an angle of attack as close to its maximum aerodynamic efficiency as possible. This value is usually related with the used airfoil, but is also defined for the overall wing and even for the entire aircraft;

- Total drag coefficient (C_D). It can be divided into zero-lift drag and lift-dependent drag. The zero-lift drag coefficient C_{D_0} is the drag produced when there is no lift being generated. It consists of the skin friction drag, influenced by Reynolds number, surface finish and Mach number and of pressure drag, dependent on fuselage slenderness, wing and empennage airfoil, sweep angles and other appendix used and on component configuration due to interference drag. It can also include wave drag, which is not relevant for this work. The lift-dependent drag considers lift-dependent form drag and induced drag, which arises due to the finite wingspan and can be minimized with an elliptical lift distribution, as explained before. As a result, it is possible to write $C_D = C_{D_{\text{skin friction}}} + C_{D_{\text{interference}}} + C_{D_{\text{induced}}} + C_{D_{\text{form}}} + C_{D_{\text{wave}}}$ However, in the solver used in this work, drag contributions are simply split as $C_D = C_{D_{\text{pressure}}} + C_{D_{\text{viscous}}}$, where $C_{D_{\text{viscous}}}$ corresponds to the skin friction drag coefficient.

2.7 Optimal aircraft design

Contrasting with the conventional design process described, there is the optimal design process. It starts in the same way, with the problem specification and the baseline design being analyzed. However, the performance and failure criteria are now objectives and constraints of the optimization problem, respectively, and are checked automatically. If the constraints are verified, it represents a valid design, and so, it is checked if the design is optimal, in which case the design is stopped. If it is not optimal, the design parameters are automatically changed with an optimization strategy, and the process is repeated iteratively until convergence is achieved. Both design processes are represented in Figure 2.7.

Unlike the conventional design method, this method ensures that not only a certain design is satisfactory, but it is also the optimal solution for a specified mission or set of missions. Despite the fact



(a) Conventional design (b) Optimal design

Figure 2.7: Flowcharts depicting two different approaches to the design process.

that the optimization using this methodology does not require a constant input from the intuition of an experienced aircraft designer in every iteration to achieve an optimum, it does not discard the need for one, as the results need to be checked and understood, given that optimizers are notable for exploiting any problem formulation weaknesses [32]. Furthermore, there is the need to define the design variables and constraints in a sensible way to obtain good results.

In the field of aircraft design, optimizing various aspects of aircraft performance is a central concern. Optimal design methodologies are applied across different disciplines, including aerodynamics, structures, and control. The following articles provide insights into these optimization efforts:

In a study by Chau and Zing ([33]), aerodynamic shape optimization was conducted for a strut-braced-wing regional aircraft in transonic conditions. They employed the Jetstream framework [34] developed at the University of Toronto Institute for Aerospace Studies. Their approach utilized free-form deformations (FFD) to parameterize twist, taper, and section shape, effectively mitigating shock formation and boundary-layer separation at the wing-strut junction. The result was an optimized strut-braced wing, illustrated in Figure 2.8, where Lovely-Haimes [35] was used to detect the shock discontinuities present. Structured overset grids were also used in this work.

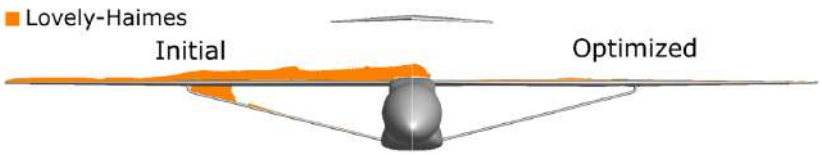


Figure 2.8: Optimized strut-braced wing. (Source: [33])

In another study by Seraj and Martins ([36]), numerical optimization was used to address the trade-off between high-speed performance and low-speed stability. Their approach, integrated within the MACH-Aero framework developed by the MDO Lab at the University of Michigan, minimized drag at a supersonic cruise condition while adhering to a pitch stability constraint. The optimized wing achieved stability in subsonic flight conditions, albeit with increased drag during supersonic flight.

Optimal design principles extend beyond fixed-wing aircraft, as demonstrated in a study by Lu et al. ([37]). Here, the focus was on optimizing the aerodynamic layout of a helicopter, aiming to minimize the adverse interaction between the rotor wake and other aerodynamic components. Due to the prohibitive cost of computational fluid dynamics (CFD) simulations, the authors developed an optimization model to calculate helicopter aerodynamic interactions. Through a combined genetic/sequential quadratic programming algorithm, they optimized the shape and relative position of components, leading to a significant improvement in hovering efficiency.

While the previous examples highlight decoupled optimization efforts, it should be noted that the optimum obtained for one discipline may be considerably different from the optimum obtained for another. In particular, aircraft design verifies a tight coupling and interaction between disciplines, especially aerodynamics and structures and thus, aerostructural considerations should ideally be taken. In [38], aerostructural optimization of drooped wings was performed for cruise and $2.5g$ load condition using high-fidelity aerodynamic and structural methods using gradient-based optimization. A successful range increase of the configuration was achieved. Furthermore, in the same study, several random initial geometries were used, with the final optimized design being similar between all of them.

In [39], the winglet configuration of a solar aircraft with a wingspan of 15 m was considered. Four types of constraints were used: geometry, aerodynamic, energy and stability and a genetic algorithm is used for the optimization process. The energy constraint took into account the shadowing effect of the winglet on the solar cells during the day and the optimized geometry, although with more drag and mass than a traditional winglet, provided a $24\ h$ cruise capability.

Moving beyond aerodynamics and structures, Denieul coupled control and aerodynamics in [40] for a blended wing-body aircraft to minimize the control surface areas, in order to reduce power consumption and actuator mass penalty, while ensuring adequate closed-loop handling qualities. Longitudinal and lateral flight control laws were optimized simultaneously with a control allocation module and the control surface areas, subjected to handling quality constraints, and a 60% reduction on the control surfaces span for the Airbus blended wing body was achieved.

Optimal design can be used as soon as the conceptual design phase. An example of this is given in [41], where multidisciplinary optimization was used for the conceptual design of a general aviation aircraft with fuel cells and hydrogen using the SUAVE framework [42] to optimize aircraft range with both wing shape and maximum fuel cell power, which influences the weight. A similar performance to conventional general aviation aircraft was achieved through optimization of a fuel cell aircraft.

More recently, machine learning techniques had also seen some applications to aerodynamic optimization. In [43], an online data-based framework is developed for the optimal morphing of the airfoil using airfoil shape control points with vertical displacements, which allows to search for an airfoil shape that minimizes a performance metric without full knowledge of the aerodynamic parameters.

The presented papers represent some interesting applications of aircraft optimal design in the recent years. Closer applications to the problem being studied in this work will be presented, particularly at Chapter 4 to support model and algorithm selection.

Chapter 3

Computational Fluid Dynamics

In the previous chapter, the aircraft design process was discussed. As an important part of that process, different models that can be used to computationally solve fluid mechanics problems are discussed here. After analyzing the advantages and disadvantages of each one, particular emphasis is placed in those to be used in this work.

3.1 CFD in aircraft design

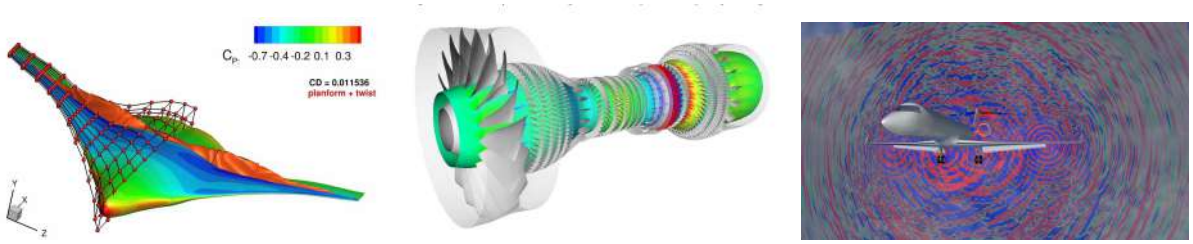
Before CFD simulations were a reality in the aerospace industry, empirical correlations and analytical methods, which resulted from simplifications of the flow governing equations, such as the Kutta-Joukowski theorem and Prandtl's lifting line theory, were used in conjunction with wind tunnel and flight testing [44]. However, in the 1960s, as computational resources increased and more efficient numerical methods were developed, CFD tools began to be explored in the industry to obtain more accurate insights about the flow around 3D geometries [45].

The use of CFD was further promoted with the paradigm shift in the early 1990s, to a design-to-cost perspective in an effort to reduce development costs while achieving higher-performance aircraft with previously unattainable solutions [46]. This minimized the need for wind tunnel and flight testing only to the final versions of the design [47], reducing the quantity of prototypes that needed to be built and allowing for quick and cost-effective comparison of alternative designs.

The advantages shown by CFD quickly made it become a competitive necessity for aerospace companies. Throughout the years, accuracy and complexity of CFD models have been moving along with computational power. Its evolution, with particular emphasis on the European market, can be found in [48]. Nowadays, CFD simulations are used in all stages of the aircraft design process for multiple purposes, including:

- External aerodynamic shape analysis, with the computation of the airflow distribution over the aircraft, as shown in [49, 50]; An example of the expected result is shown in Figure 3.1 a).
- Turbomachinery design, used for the propulsion system of certain aircraft. The use of CFD in that field is discussed in [51]; A visual representation is shown in Figure 3.1 b).

- Cooling analysis in which internal flows on the aircraft are considered, as exemplified in [52];
- Computation of noise and vibrations around the aircraft which, although in a less mature state, is exemplified in articles such as [53] and presented in Figure 3.1 c).
- Determination of stability and control characteristics, exemplified in [54];
- Wind tunnel data augmentation, by providing Reynolds number corrections and the analysis of wind tunnel wall interference in flow solutions, as discussed in [55].



(a) Pressure coefficient around a blended wing-body configuration. (Source: MDOLab at the University of Michigan)

(b) Axial velocity simulation of an entire turbofan jet engine. (Source: Center for Turbulence Research at Stanford University)

(c) Unsteady simulation to calculate flow around the airframe. (Source: NASA Glenn Research Center)

Figure 3.1: Some applications of CFD to aircraft design.

3.2 Mathematical models

3.2.1 Fluid flow models overview

Today, thanks to the accumulation of several years of research in the field, there are many models available to simulate fluid flow, which vary in complexity and accuracy. A detailed overview of the various models available can be found in the literature, for example in [56] or [57]. However, a short description of the main ones is provided here. The hierarchy of fluid flow models is depicted in Figure 3.2, representing an increase in complexity, accuracy and computational cost from the base to top. The main characteristics of each model and a general timeline are also depicted.

One of the simplest available models is the potential flow model, which characterizes the fluid flow as non-viscous and irrotational, which allows the three-dimensional velocity field to be described with a single scalar function. Despite that, for steady potential flows, circulation on a closed curve around a lifting body needs to be accounted to achieve non-zero lift, which is achieved by the addition of a free vortex singularity whose circulation is determined by the Kutta condition [58]. Potential flow equations can be further simplified by linearization for thin airfoils which, for subsonic flows, allows to model the flow as a superposition of known elementary flows. This observation gave rise to the creation of several simple models including vortex lattice method and panel methods. Despite being initially developed in the 1960s, panel methods in particular had seen extensive use in the aerospace industry only a few years later, during the 1980s and 1990s, thanks to code development contributions for the computation of external flow around an arbitrary shape under subsonic [59] or supersonic [60] flow with good results.

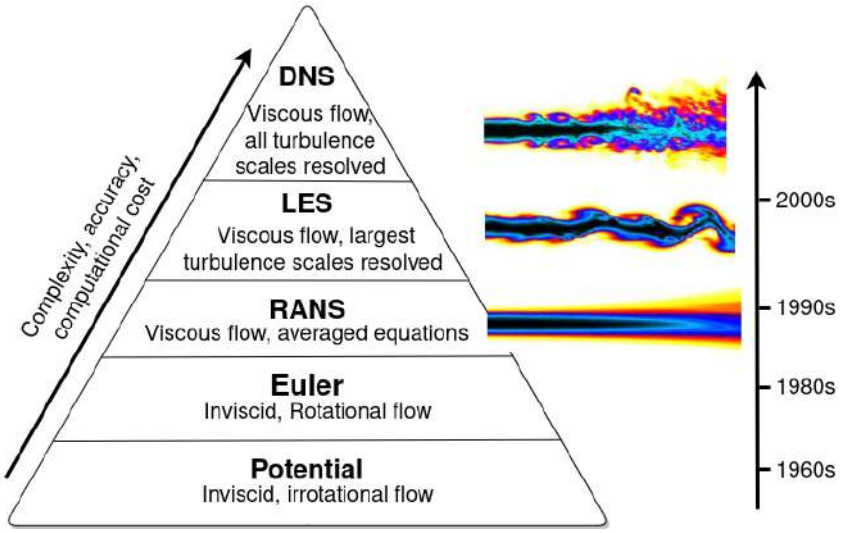


Figure 3.2: Hierarchy of fluid flow models.

Being currently still one of the less computationally demanding options, potential flow equations are a great option to evaluate multiple designs quickly during the conceptual phase.

Another option is the use of Euler equations, applicable to adiabatic (non-heat conducting) and inviscid flow, originally described in [61]. These equations constitute a first-order system of partial differential equations (PDEs), derived from fundamental equations in continuum mechanics. They encompass convective, source, and transient terms. Euler equations are well-suited to describe low viscosity fluids or for applications where viscosity is not relevant and the boundary layer is small in relation to the body, namely flows at high Reynolds numbers outside viscous regions developing near solid surfaces. For external flows around a wing, this model typically provides good estimates for lift produced. However, these equations do not calculate the contribution of the skin friction component to the overall drag. They have seen extensive use in wing design and optimization. Examples of that are given in [62, 63].

3.2.2 Navier-Stokes equations

Navier-Stokes equations are a generalization of Euler equations that can handle viscosity and thermal conductivity on the flow, composing a second order system of PDEs that describe the viscous fluid flow in a general manner, by expressing the conservation laws of mass, momentum and energy. In differential conservative form, they are given by

$$\frac{\partial}{\partial t}\begin{vmatrix} \rho \\ \rho\vec{v} \\ \rho E \end{vmatrix} + \vec{\nabla}\cdot\begin{vmatrix} \rho\vec{v} \\ \rho\vec{v}\otimes\vec{v} + p\bar{\bar{I}} - \bar{\bar{\tau}} \\ \rho\vec{v}H - \bar{\bar{\tau}}\cdot\vec{v} - k\vec{\nabla}T \end{vmatrix} = \begin{vmatrix} 0 \\ \vec{f_{\mathrm{e}}} \\ W_{\mathrm{f}} + q_H \end{vmatrix}, \tag{3.1}$$

where ρ is the density, p is the pressure, $\vec{v}$ is the velocity vector, E is the total energy, $\bar{\bar{I}}$ is the unit tensor, $\bar{\bar{\tau}}$ is the viscous shear stress tensor, k is the thermal conductivity, H is the enthalpy and T is the flow temperature. The right hand side represents the source term, where $\vec{f_e}$ are the external forces acting on the flow, W_{f} the work performed by them and q_H is the effect of heat sources [57].

For laminar flows, it is possible to compute the Navier-Stokes equation in a deterministic way. However, due to the non-linear convection terms, after a critical Reynolds number, a spontaneous instability of the flow appears, creating statistical fluctuations of all variables around their mean, and the flow becomes turbulent [57]. To tackle this, several models with varying complexity are available.

To capture the fluctuations accurately, direct numerical simulation (DNS) of turbulence should be used, where all relevant physical scales are solved in space and time. Although developments are still being made, it requires enormous computational power, making it unusable for industrial applications with the current available hardware, so it is a research topic mainly inside academia that can provide insights about the fundamental mechanisms of turbulence. A state of art review of this model can be found in [64, 65]. Information obtained from DNS simulations can also be useful to establish a database of information that may be used to augment lower level models. Examples of such applications are discussed in [66, 67].

To reduce the computational power needed and make viscous turbulent fluid simulations possible at an industrial level, some approximated models were derived. The closest one to DNS is Large Eddy Simulation (LES), where turbulent fluctuations are filtered and only those above a certain length scale are solved in space and time. The remaining fluctuations, corresponding to smaller lenght scales, are modeled. [68] describes this approach in detail.

LES is, however, still too computationally expensive for most practical applications. The currently most widely used alternative is the Reynolds Averaged Navier-Stokes (RANS), which is considered an high-fidelity method in the context of this work, where turbulent fluctuations influence is removed and only an averaged flow over the whole turbulent fluctuations spectrum is calculated. This relies on Reynolds decomposition, introduced in [69], in which every turbulent quantity is separated into its mean value (time averaged) and a fluctuation about it with stochastic nature. More information about this method can be found in [70]. RANS was compared with LES in [71] and recently, hybrid approaches with RANS and LES were also suggested, as reviewed in [72] and [73].

The equations obtained for RANS are similar to those presented in Equation (3.1). For incompressible flow, quantities are time-averaged, represented by an overbar. However, for compressible flow solvers as it is the case with ADFlow, density-weighted averaging of the quantities with Favre averaging is used for all variables except pressure and density, represented by a tilde, in order to avoid products of fluctuations between density and other variables [57]. As a result of the averaging process, a new term arises for momentum equations - the Reynolds stresses - defined by $\bar{\bar{\tau}}^{\mathrm{R}} = -\overline{\rho\vec{v}'' \otimes \vec{v}''}$, which define an apparent stress due to the fluctuating velocity field and for energy equations - the turbulent heat flux vector - given by $\bar{\vec{q}}^{R} = -C_p\overline{\rho\vec{v}'T'}$ [57]. The final set of equations is given by

$$\frac{\partial}{\partial t}\begin{vmatrix} \bar{\rho} \\ \bar{\rho}\tilde{\vec{v}} \\ \bar{\rho}\tilde{E} \end{vmatrix} + \vec{\nabla}\cdot\begin{vmatrix} \bar{\rho}\tilde{\vec{v}} \\ \bar{\rho}\tilde{\vec{v}}\otimes\tilde{\vec{v}} + \bar{p}\bar{\bar{I}} - \tilde{\bar{\tau}} + \bar{\bar{\tau}}^{\mathrm{R}} \\ \bar{\rho}\tilde{\vec{v}}\tilde{E} + \tilde{\vec{v}}\cdot\bar{p} + \tilde{\vec{v}}\cdot\bar{\bar{\tau}}^{\mathrm{R}} - \bar{\vec{q}}^{R} \end{vmatrix} = \begin{vmatrix} 0 \\ \rho\vec{f}_{\mathrm{e}} \\ W_{\mathrm{f}} + q_H \end{vmatrix}, \tag{3.2}$$

However, as the relation between those new terms and the mean flow quantities is unknown, the system of equations ends up with more unknowns than equations. Reynolds stresses must thus be modeled

based on theoretical considerations and empirical or semi-empirical information on the turbulence structure and its relation to the averaged flow to close the system. Some of the models are discussed in Section 3.2.3.

3.2.3 Turbulence models

Over the years, several models to predict the effects of turbulence have been proposed and a comprehensive review can be found in the literature [74–76]. The use of turbulence models is a trade-off between computational power and accuracy, as they save resources in computational power when using them with Navier-Stokes equations, while still giving a fairly accurate way of modeling turbulence.

Here, only the models implemented in the CFD solver used in this work - ADFlow - are mentioned, namely the Spalart-Allmaras (SA), Wilcox $k-\omega$, $k-\tau$, Menter shear stress transport (SST) and $\bar{v}^2-f$.

Most models rely on the Boussinesq hypothesis, which, under the conjecture that turbulent eddies dominate the momentum transfer on turbulent flow introduced the eddy viscosity (μ_t) concept, a linear proportionality factor that relates the turbulent shear stress with mean strain rate [77]. With this, Reynolds stresses for RANS equations may be given by

$$\bar{\bar{\tau}}_{ij}^{\mathrm{R}} = -\rho\overline{v_i'' v_j''} = \mu_T \left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i} \right) - \frac{2}{3}\rho k \delta_{ij} \,, \tag{3.3}$$

where k is the turbulent kinetic energy.

One of the simplest models based on the Boussinesq hypothesis is the Spalart-Allmaras model. Designed for aerospace applications, it uses only one equation, given by

$$\begin{aligned} \frac{\partial \tilde{v}}{\partial t} + \frac{\partial}{\partial x_j}\left(\tilde{v} v_j\right) =& C_{b1}\left(1-f_{t2}\right)\tilde{S}\tilde{v} \\ &+ \frac{1}{\sigma}\left\{ \frac{\partial}{\partial x_j}\left[\left(v_{\mathrm{L}}+\tilde{v}\right)\frac{\partial \tilde{v}}{\partial x_j}\right] + C_{b2}\frac{\partial \tilde{v}}{\partial x_j}\frac{\partial \tilde{v}}{\partial x_j}\right\}, \\ &- \left[C_{w1} f_w - \frac{C_{b1}}{\kappa^2} f_{t2}\right]\left(\frac{\tilde{v}}{d}\right)^2 + f_{t1}\|\Delta \vec{v}\|_2^2 \end{aligned} \tag{3.4}$$

that solves a modeled transport equation for the undamped kinematic eddy-viscosity parameter, $\tilde{\nu}$. In Equation (3.4), c_{b1}, c_{w1} and c_{b2} are constants obtained based on empiricism, dimensional analysis and Galilean invariance for external flows [77] and f_{t1}, f_{t2} are functions used for modeling the laminar-turbulent transition. Their values, along with a more detailed description of the model, can be found in the original manuscript [78].

The turbulent eddy viscosity can then be obtained as

$$\mu_{\mathrm{T}} = f_{v1}\rho\tilde{\nu} \,, \tag{3.5}$$

where $f_{\nu 1} = \frac{\chi^3}{\chi^3 + C_{\nu 1}^3}$ and $\chi = \frac{\tilde{\nu}}{\nu}$. This model gives good results for external flows, with boundary layers subjected to adverse pressure gradients at low Reynolds numbers. However, it should be used with caution when there is shear flow, separated flow and decaying turbulence, as it tends to deliver inaccurate results in those cases [74]. Furthermore, it is generally regarded as computationally efficient.

Two-equation models have been gaining popularity as they incorporate substantially more turbulence physics, dispensing the need to use as many empirical correlations. The $k-\omega$ turbulence model also uses the Boussinesq approximation and uses two PDEs to find the two variables, k (kinetic energy) and ω (specific dissipation rate), allowing to account for history effects. It is also a low Reynolds number model, but it has sucessfully predicted separation and reattachment, as described in [79].

Another two equation model based on the Boussinesq approximations is the $k-\tau$ [80], where τ, the turbulence time scale, is the second variable. Its accuracy for simple flows is comparable to that of $k-\omega$ but it lacks validation for more complex cases [74]. Application examples can be found in [80].

The Menter SST is also a two equations model combining the strengths of both $k-\omega$ (used in the inner region of the boundary layer, as it can predict well flow with high gradients and even separation) and $k-\epsilon$ (used in the free shear flow, as this model presents good results when predicting flow far from the walls) models. Between the two regions separating the use of $k-\omega$ and $k-\epsilon$ model, a blending function, F_1, is used. More details about the 2003 version of the model can be found in [81].

Finally, the $\bar{v}^2-f$ model [82] uses a slightly different approach. It considers the velocity scale, $\bar{v^2}$, instead of k and models anisotropic wall effects by solving an elliptic relaxation function, f. By modeling the near-wall, it performs well for heat transfer applications and flows with separation [75].

When choosing the turbulence model to use, it is important to note that only the Spalart-Allmaras turbulence model has been differentiated in the ADFlow solver, which is required if gradient based optimization is envisioned, as detailed in Chapter 4.

Besides this fact, comparisons for external flows around an Onera M6 wing concluded that the Spalart-Allmaras and SST $k-\omega$ turbulence models gave the closest results to the experimental data [83]. Also, using the DPW7 CRM configuration, it was shown that both the k-ω and Spalart-Allmaras models yield great results [84]. The main drawback of the Spalart-Allmaras model is poor accuracy for separated flows and free-shear flows, including jets, performing well for external flows otherwise [85], where the SST model performs better. A similar conclusion was obtained in [86] when comparing various turbulence models agains several experimental external flow fields. Spalart-Allmaras was bad for round and plane jets, which will not be the case in this work The flow solution using ADFlow with the Spalart-Allmaras turbulence model was also compared with two other solvers and to experimental data for the CRM wing, successfully validating RANS using the Spalart-Allmaras model on the ADFlow solver [87]. Considering the good results obtained with Spalart-Allmaras for external flow computations and the need for the model differentiation, it is the most suited for the optimization of a wing.

3.3 Discretization techniques

3.3.1 Equation discretization methods

The model presented in Equation (3.2) is a system of PDEs with very few exact solutions so, for general cases numerical methods must be applied to transform it into an algebraic system of equations. For that, three main discretization methods are available: Finite Difference Method (FDM), Finite Volume

Method (FVM) and Finite Element Method (FEM).

FDM is the oldest and simplest of the three implementation wise. This approach hinges on the direct discretization of flow variable derivatives through a Taylor series expansion [77]. Besides its simplicity, another merit of this model lies in its flexibility to obtain and implement higher order approximations by considering more terms on the Taylor expansion. However, its main shortcomings are not allowing the use of unstructured grids, making irregular geometries difficult to handle [88] and not being conservative.

Nowadays, most CFD models are based on the FVM, particularly with second order accuracy. This method starts with the integral formulation of the flow governing equations. Then, Gauss divergence theorem is applied to convert volume integrals with divergence terms to surface integrals, resulting in

$$\frac{\partial}{\partial t}\int_V U dV + \oint_s \vec{F}\cdot d\vec{S} = \int_V Q dV\,, \tag{3.6}$$

where U is a scalar quantity, Q are volume sources and $\vec{F}$ are the fluxes across the faces. Then, based on the provided grid, control volumes are created around either each cell center or each node of the grid, depending on the scheme used [77], where the conservation law represented by Equation (3.6) should be verified. To this end, the surface and volume integrals are transformed into discrete ones and integrated numerically. The fluxes associated with a certain face are determined by the control volume center values of the elements sharing it. Thus, the flux leaving one element face is equal to the flux entering the face of a neighbour element [89], ensuring conservative discretization throughout the domain. Besides this property, FVM also benefits from its conceptual simplicity and ease of implementation for both structured and unstructured grids and it is the method used in ADFlow. The method is, however, limited in the order of accuracy that can be easily and efficiently achieved [44].

Lastly, FEM method is based on the weak form of the equations and on the Galerkin approximations and it is nowadays very common for problems in solid mechanics. It also starts with a discretization of the space into smaller volumes (elements), where integrals will be formulated generally in the weak form. A linear combination of shape functions is then used to approximate the field variables in each element and the number of unknowns is the degree of freedom of said node [57]. This method shares some advantages with FVM, namely the integral formulation and the affinity with unstructured grids [77]. Other advantages include multiphysics problems, given that it can combine different kinds of functions to approximate the solution in each element in an easier way than the other methods. This method is also currently being explored for higher order of accuracy solutions in CFD, which would potentially reduce the computational time needed for CFD problems [44], with some codes already being verified [90]. Given this, developments with this method for CFD are to be expected in the future. The main disadvantage of the method is the implementation difficulty and the smaller flexibility when compared with the other two [91]. Furthermore, it is not locally conservative.

3.3.2 Meshing methods

Meshing methods can be generally classified as structured or unstructured. With structured grids, nodes are placed at the intersection of lines from three different families (one for each spatial direction

(i, j, k)) creating hexahedral elements with neighbors easily identifiable as $(i \pm 1, j \pm 1, k \pm 1)$ [92]. They can be further divided into cartesian, non-uniform cartesian and body-fitted and although cartesian grids yield high accuracy, they are typically not suitable for external flow computations around a body. Body fitted grids follow the solid surfaces with a curvilinear grid [57]. Structured grids are better in accuracy, CPU time and memory requirements [93] and their drawbacks are the time and user input necessary for generation, considering their rigid structure and the difficulty to perform local refinements.

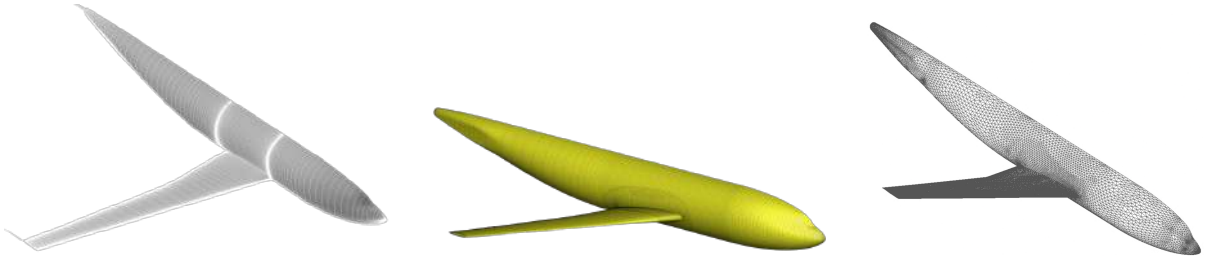
To alleviate some of these problems, there are two approaches available. The first is the multi-block grid, depicted in Figure 3.3 (a), in which multiple structured grid blocks defining different regions are combined and connected, allowing different levels of refinement and topologies between regions. Some solver algorithms are even able to use blocks with non-matching lines between them by using interpolation routines [94]. With this, it is possible to achieve higher grid quality while minimizing the number of elements needed. However, extensive user input is still required and it is almost impossible to automate the generation of multi-block grids on arbitrary geometries [57], with the main challenge being the decomposition in blocks.

Overset grids also combine multiple structured (or unstructured) grids into one single file but here they are allowed to overlap (shown in Figure 3.3 b) between the wing and body of the DLR-F6 configuration), adding further flexibility to their generation as component grids can be created relatively independent from each other, which is particularly useful when dealing with structured meshes around complex geometries. Furthermore, this also allows to change individual components without the need to regenerate the whole mesh, useful not only to iterate component shapes but also for simulations involving bodies in relative motion. The use of overset grids require a solver capable of handling accurately the interpolation between the component grids but they still suffer from the same automation difficulty as multi-block grids (although progress as been made, as shown in [95] for example). More details about their use can be found in [96].

Unstructured grids gained popularity during the 1990s with the increase in computational power. Despite requiring more memory and computational time for the CFD solution, they are the most flexible option and automation of the generation process is considerably easier, with software for this purpose widely available today [92]. Within them, nodes follow arbitrary distributions with the connectivity information explicitly saved and different types of elements can be used, including hexahedral, tetrahedral and polyhedral elements. An example using tetrahedral cells is shown in Figure 3.3 (c). Hexahedral tend to offer significant advantages compared to tetrahedral elements regarding memory requirements and accuracy especially for flow aligned problems due to its significantly smaller ratio of number of cells to number of vertices. Polyhedral elements offer higher accuracy for complex geometries at the expense of computational resources due to the higher number of neighbors to approximate the gradients [94].

Independently of the type of mesh, when performing viscous simulations using RANS, the boundary layer region must have enough resolution, with the specific requirements depending on the turbulence model used. The use of hexahedral cells with high aspect ratio near the viscous walls is common and more details are discussed in [97].

The best suited grid type is always strongly tied to the flow solver to be used, as not all flow solvers



(a) Multi-block structured grid. (Source: Chris Rumsey)

(b) Overset structured grid. (Source: MDOLab at the University of Michigan)

(c) Unstructured grid. (Source: Chittur Venkatasubban)

Figure 3.3: Different grids applied to the DLR-F6 wing-body model.

can deal with every type of mesh, and to the complexity of the geometry. Independently of the chosen grid type, the created grid should always have a smooth cell size variation and adequate refinement to resolve the relevant flow features, including boundary layer and wingtip effects [57]. In this work, multi-block structured grids are used for the wing discretization. For the wing-fuselage assembly, overset grids are used. Those are the two types of grids accepted by ADFlow, the flow solver used.

3.3.3 Boundary conditions

This section provides a brief overview of the four types of boundary conditions employed throughout this work: wall, symmetry, farfield, and overset boundary conditions.

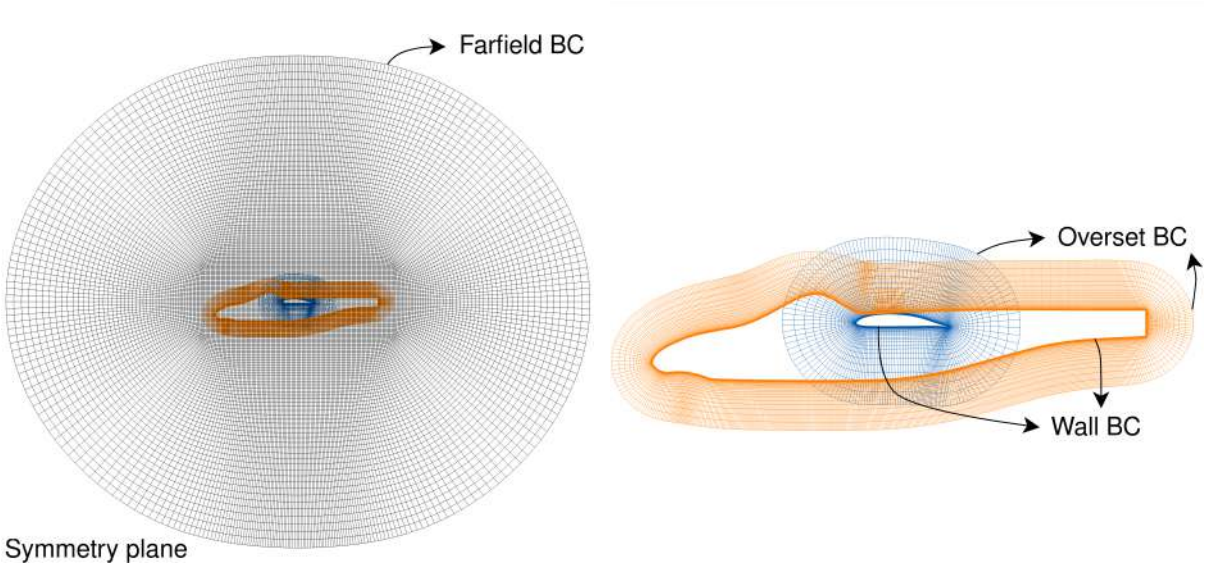
- Wall boundaries represent solid surfaces in the computational domain where the fluid is assumed to have zero normal velocity. When dealing with viscous RANS simulations, a no-slip condition will also be applied, ensuring the velocity of the flow in the tangential direction is the same as that of the wall [89].

- Symmetry boundaries are used on the symmetry plane of the aircraft to allow for the use of only half geometry, saving computational resources by reducing the domain size without compromising accuracy. This condition forces the fluxes and normal components of all variables to be zero across the boundary [89].

- Farfield boundaries are used at the computational domain limit (the ”far” boundary). When using this condition, ADFlow will determine if a region is inflow or outflow based on the dot-product of the surface normal with the free stream velocity direction. For the outflow, zero Neumann boundary conditions are applied (the gradient of the variables is prescribed as zero [57]). For inflow, variables are prescribed depending if subsonic or supersonic flow is present [98]. Possibilities for prescription of inflow conditions are discussed in Section 5.3.

- Overset boundaries are needed when using overset grids. They are applied to the outer surface of the nearfield domain of each component and setting it as an overset boundary condition will allow ADFlow to apply the implicit hole cutting algorithm on that region to find interpolation cells between the different grids [98].

Figure 3.4 shows the overset mesh of the wing and fuselage with the respective boundary conditions, with the farfield grid in black and the wing and fuselage nearfields in blue and orange, respectively.



(a) Overset mesh farfield (b) Overset mesh nearfield

Figure 3.4: Boundary conditions for the wing and fuselage overset mesh.

3.4 Solver algorithms

As discussed before, the RANS equations must be solved numerically after discretization. The solver algorithms relevant for ADFlow are presented in this section, including multigrid, Approximate Newton-Krylov (ANK) and Newton-Krylov (NK) methods.

Multigrid methods leverage the faster convergence of iterative methods when coarser grids are used. The iterations then smooth out the solution error. They require the solution initialization with the coarsest grid level and after a prescribed convergence is achieved by the smoother, switch to successively finer grids, performing a multigrid cycle after convergence of the finer grid. A use of three to five different grid levels is typical and more details can be found in [99]. The multigrid method in ADFlow specifically uses either Runge-Kutta (RK) or Diagonalized diagonally dominant alternating direction implicit scheme (D3ADI) smoother algorithms (with the second one being typically faster than the first but lacking robustness [98]) to update the flow variables while a segregated Diagonalized Alternating Direction Implicit (DADI) method is used to solve the turbulence model [12]. The main advantage of this method is the speed of the solution process without penalizing accuracy. They do not work well with overset grids and when separation regions are present in the flow [98]. This method can also be used between optimization iterations, when the external shape of the geometry is not changed drastically, in order to get a fast initial guess for the solution of the new geometry, before proceeding with other solution algorithms.

NK is a popular algorithm for CFD applications. It uses Newton's method for the solution of the nonlinear system of equations, benefiting from its favorable convergence rates and a Krylov subspace method (GMRES is a popular choice) to solve the resulting large linearized systems on each nonlinear

iteration, mitigating the bad problem-size scaling provided by Newton methods alone [100]. ADFlow offers a fully coupled Jacobian-free NK algorithm used for the final stages of convergence. Despite its favorable nonlinear convergence rates, NK algorithms tend to perform poorly during the initial stages of convergence, especially when the initial guess is far away from the solution, leading the algorithm to fail. To overcome this, some kind of globalization method is needed [101].

Some ANK algorithms have been proposed to solve that, usually by using a matrix-based approximate Jacobian lagged for a pre-determined number of iterations. Although this was found to work for Euler equations [102], when attempting to apply them to RANS [103], it was observed that this lag had serious implications in the robustness of the algorithm, so an update of the approximate Jacobian was required at every iteration, increasing greatly the computational cost of the method. To solve those issues, a Jacobian-free ANK algorithm using an approximate residual formulation was developed for use in ADFlow, which allowed the Jacobian to stay updated at every iteration [104]. The ADFlow ANK solver uses a pseudo-transient continuation (PTC) method and a backward Euler time-stepping scheme to approximate the Jacobian.

In ADFlow, a cell centered approach is used with the Jameson-Schmidt-Turkel scheme with scalar dissipation [105] to compute inviscid fluxes and a Green-Gauss approach is used for the viscous flux gradients, resulting in a 33 point stencil used in the NK solver [104]. The default approximate residual formulation used in the ANK solver omits the fourth-order dissipation fluxes, whilst still keeping a second-order accurate dissipation term, providing a good trade-off between accuracy and robustness [104].

3.5 Associated errors

Independently of the fidelity level of the chosen approximation, CFD results always have an associated error. Those errors may be classified as physical modeling or numerical errors.

Physical model errors arise from the fact that the considered models only approximate the real world. This errors are typically higher for lower fidelity methods, as more simplifications are made to obtain simpler sets of equations. However, even DNS has physical modeling errors, as it considers the fluid equation of state for example. Turbulence models for the RANS equations are also sources for this kind of error. Deliberate simplifications of the models are not the only source of physical modeling errors, as other uncertainties are always present when a model is formulated. This errors can be quantified through uncertainty quantification. A review of the current methods for CFD is given in [106].

When it comes to numerical errors associated with CFD simulations, they may be classified in one of the following three categories [107]:

- Round off errors, resulting from the computer finite precision. If double-precision is used, they are typically very small and thus not a concern for typical engineering problems;
- Iterative convergence errors, arising as a result of the iterative methods used to solve the non-linear equations until convergence is reached. Residuals and differences between iterations can give an idea about the magnitude of this type of error;

- Discretization errors are due to both equation (to transform the systems of PDEs on an algebraic systems of equations) and domain discretization in the meshing process, which is strongly dependent on the grid quality. This error can be estimated via mesh refinment studies and subsequent treatment of the obtained results by using, for example, a Richardson extrapolation.

Generally, numerical errors can be minimized by increasing the use of computational power (except for round off errors) [107]. This can be achieved for example with more iterations, leading to a better converged solution, or a finer grid, reducing the discretization errors. However, the needed hardware for this is not always available and the increase in computational time may not be worth the additional accuracy obtained, which highlights the importance of performing refinement studies. Those aspects are taken into account in Section 6.1, where grid studies are performed to assess the errors.

3.6 ADFlow

ADFlow is an open-source multi-block and overset structured, parallel, finite-volume flow solver for Euler and RANS equations initially developed in the Stanford University under the sponsorship of the Department of Energy Advanced Strategic Computing (ASC) Initiative with the purpose of computing the flow in the rotating components of jet engines [98].

It is being developed and maintained by the MDO Lab at the University of Michigan since 2006. Although ADFlow is applicable to a broad range of aerodynamics problems, it is primarily being used for aerodynamic and aerostructural optimization. It is useful for these applications as it provides the derivatives to the functions of interest with respect to the specified design variables using a discrete adjoint method with partial derivatives computed with automatic differentiation (AD), allowing its use for gradient-based optimization. Besides computing the gradients, ADFlow offers other advantages which make it ideal for optimization problems. The ability to switch algorithms during the solution procedure based on the relative reduction in the residual L_2 norm and in particular the use of the robust ANK algorithm (which may penalize performance at the expense of robustness) followed by a switch to the NK algorithm makes it accurate and very robust at the same time, much needed characteristics when considering non-physical intermediate geometries given by the optimizer, which may lead the solver to fail and consequently, stop the whole optimization problem. Although RANS is not expected to give accurate predictions in this case, iterative convergence is important to obtain gradients and continue the optimization process. Furthermore, the RANS solutions are expected to become valid as the optimum shape is approached, rendering non-physical intermediate results irrelevant [10].

ADFlow also provides direct memory access that is helpful to couple to other analysis codes and offers a Python API for greater flexibility, while keeping its high-performance routines in Fortran 90 [12].

Typically, the use of ADFlow requires the use of a Linux based operating system. However, a Docker container can be used for installation on other platforms. The installation of additional common packages is also required, including codes like CGNS, as it is the file format used for the input and solution files, OpenMPI for parallelization of the code and PETSc for linear algebra operations. The reader is referred to the documentation [98] for additional information on installation, tutorials and examples.

Chapter 4

Optimization

In this chapter, optimization methods are discussed from a theoretical point of view. An overview of available methods is first provided, and followed by an analysis of sensitivity methods for gradient-based optimization. Finally, the implications of both aspects for aerodynamic shape optimization are discussed.

4.1 Optimization methods

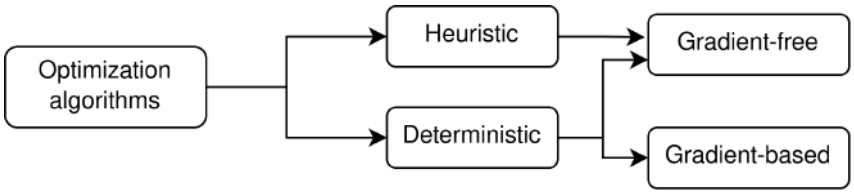


Figure 4.1: Overview of optimization methods.

Several optimization algorithms are available and regarding stochasticity they may be categorized as heuristic or deterministic. Whilst heuristic methods are always gradient-free, deterministic methods may be classified as either gradient-free or gradient-based. A schematic of this classification is presented in Figure 4.1.

According to [108], heuristics are simple procedures meant to provide good but not necessarily optimal solutions to difficult problems quickly, as they cannot prove that a point is a local optimum. They often mimic behaviors found in nature and do not rely on strong assumptions about the optimization problem, such as the continuity of the objective function and the availability of gradient information, allowing them to work on a wide range of optimization problems and often providing an easier problem setup. There are numerous heuristic methods, including but not limited to genetic algorithms [109], particle swarm optimization [110] and simulated annealing [111], with new ones being frequently presented. Comprehensive reviews of these methods are given in [112, 113]. Their main disadvantages include the solution quality, due to their stochastic nature, and the computational power needed to repeatedly evaluate the objective function [108, 114].

Deterministic methods, on the other hand, do not have randomness associated and the models rely

solely on mathematics [115]. Gradient-free methods are generally much easier to implement and include methods such as DIRECT [116], Nelder-Mead [117] and interior-point [118]. Examples of gradient-based deterministic methods are bisection methods [119], SQP [120] and trust region [121]. Higher order deterministic methods requiring second derivative information are also available [114]. Gradient based algorithms are usually composed of two main steps [122]:

1. Finding a suitable search direction.

2. Minimizing along that direction (line search).

Figure 4.2 shows a gradient vector field for a two-variable function. It is possible to see that the gradients point to the maximum, and this information will be used to point the optimization direction. Thus, these methods will typically converge to the optimal solution faster than their gradient-free counterpart, making them more efficient particularly as the number of design variables increases. Gradient-based methods also allow a more rigorous definition of optimality as it is based on gradient information [10].

To achieve successful convergence, however, some characteristics need to be verified by the objective and constraint functions, namely continuity, smoothness (at least C_1 continuous, like the function shown in Figure 4.2) and low-modality throughout the design space [123] as otherwise, they tend to get stuck in local minimums and multi-start strategies may be needed, hindering their advantages in regard to function evaluations needed when compared with gradient-free methods. These properties also make them susceptible to noise in the functions, making some gradient-free methods more robust when said properties are not verified [114].

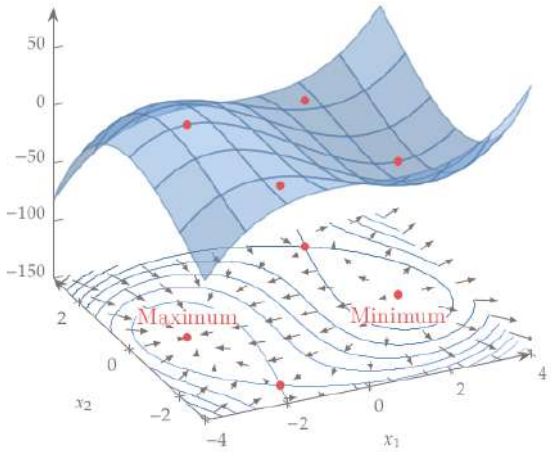


Figure 4.2: Gradient vector field. (Source: Martins and Ning, Engineering Design Optimization)

Comparative studies of both gradient-free and gradient-based methods were performed in [124] and [125], with similar conclusions between them: both can converge to a solution, but gradient-free methods are considerably more computationally expensive, with the difference increasing dramatically with the number of design variables. Despite this, gradient-free methods do not require continuity or predictability over the design space, becoming a viable alternative for some specific aerodynamic shape optimization

applications. In [126], those methods were used to optimize an airfoil using LES and in [127], gradient-free methods were applied to supersonic conceptual aircraft design problems using multi-fidelity models. MACH-Aero framework also allows the use of gradient-free methods, namely ARMOGA [128] and NSGA-II [129].

Examples of gradient-based optimization have been performed where both refinement and exploratory studies were done using both inviscid and viscous models for the optimization of a wing using the MACH-Framework and the SNOPT optimizer [130]. In [131], a more complex problem of multi-component aerodynamic optimization for a wing propeller coupling was also solved using gradient-based methods within the OpenMDAO/MPhys framework. Furthermore, in the same studies, concerns about continuity and modality of the design space were analyzed and it was concluded that there is no evidence that aerodynamic optimization problems are multi-modal and when multi-modality is present, multi-start methods were found to be enough and that the functions involved are smooth and C_1-continuous. Considering this and the large number of design variables that will be involved in the problem, deterministic gradient-based methods were chosen for this work.

Regarding the specific optimization algorithm, the SLSQP (Sequential Least Squares Quadratic Programming) will be used due to its open source nature and robustness. It has been used in the past for aerodynamic shape optimization [132, 133]. The SLSQP algorithm solves constrained nonlinear optimization problems, which corresponds to the nature of the problem being studied in this work, using the Han-Powell quasi-Newton method with BFGS update of the B-matrix and an $L1$-test function in the step length algorithm [134].

4.2 Sensitivity methods

As stated in the previous section, gradient based optimization methods rely on the availability of derivatives across the entire design space. Despite the potential to make more efficient algorithms by providing the gradients, obtaining them is generally neither trivial nor computationally cheap [135]. Symbolic differentiation, finite differences, complex-step method, algorithmic differentiation and semi-analytic methods are the main methods available.

Pure symbolic differentiation can be performed with symbolic math software but it is limited to cases with relatively simple functions with explicit governing equations, making them inadequate for most engineering problems, frequently governed by PDEs.

Finite difference methods are simple to implement without modifications needed on the solver. In fact, they are the only method discussed here that do not require any information on the model other than the outputs coming from function evaluations. The main drawbacks of this method is the lack of robustness coming from the step size dilemma (when the step size gets too small, subtractive cancellation errors become predominant) and their cost, which is proportional to the number of design variables as their calculation on each point requires the perturbation of every variable [136].

Another alternative is the complex step method, an approach that is also easy to implement. The initial formulation is similar to the finite difference one but a complex-step is used in the Taylor expansion,

resulting in the elimination of the difference operation and consequentially, of the subtractive cancellation error, allowing for the use of smaller steps without loosing accuracy. However, they still need a perturbation to be applied to every design variable, also making them too expensive for use with larger sets of design variables. More information on this method can be found in [137].

A more complex approach is algorithmic differentiation (AD), which differentiates basic operations throughout the code and accumulates the derivatives based on the systematic application of the differentiation chain rule [138] using two possible modes. In forward mode, function and partial derivative values are computed in a forward pass. Using this approach, the derivative of all other function outputs with respect to the chosen input variable comes as a by-product, resulting in a calculation cost proportional to the number of inputs, and thus, being more efficient when it is exceeded by the number of outputs; In reverse mode, a forward pass is used to store the values of all variables and partial derivatives and a backwards pass is then used to calculate the total derivatives by backwards propagation of an output, allowing to obtain the derivatives of a certain objective function with respect to all the inputs in a single pass, making it the more efficient approach when there are more inputs than outputs. The main drawback of this method are the huge memory requirements; Contrary to symbolic differentiation, this is a procedure rather than a symbolic expression and the implementation is simpler than semi-analytic methods, thanks to various AD tools that automate the process like Tapenade [139]. These methods can be implemented either by source code transformation or by operator overloading.

Semi-analytic methods solve a linearized system of equations resulting from the model governing equations to obtain the desired derivatives [114], thus requiring knowledge of the governing equations and corresponding state variables but not of the algorithms involved in the solution. Using the chain rule, the derivatives of a function of interest, f with respect to a design variable x_n can be written as

$$\frac{\mathrm{d}f}{\mathrm{d}x_n} = \frac{\partial f}{\partial x_n} + \frac{\partial f}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}x_n}\,, \tag{4.1}$$

where u are state variables, which depend implicitly on x_n through the governing PDEs. All partial derivatives can be evaluated using previously described methods except $\frac{\mathrm{d}u}{\mathrm{d}x_n}$. To solve this, the discretized numerical model, written as $\mathcal{R}_k\left(x_n, u\left(x_n\right)\right) = 0$, where $\mathcal{R}_k$ are the residuals. Discretizing this equation, it is possible to replace the total derivative in Equation 4.1 and re-write it as

$$\frac{\mathrm{d}f}{\mathrm{d}x_n} = \frac{\partial f}{\partial x_n} - \underbrace{\frac{\partial f}{\partial u}\overbrace{\left[\frac{\partial \mathcal{R}_k}{\partial u}\right]^{-1}}^{-\mathrm{d}u/\mathrm{d}x_n}}_{-\Psi_k}\frac{\partial \mathcal{R}_k}{\partial x_n}\,. \tag{4.2}$$

Semi-analytic methods can be divided into two. Direct methods calculate $\frac{\mathrm{d}u}{\mathrm{d}x_n}$ directly using the differentiated form of the residuals equation. This, however, needs to be done for every design variable, which is costly especially when iterative solutions are required to avoid the expensive memory requirements of storing the inverse Jacobian. Despite this, the cost is independent of the number of outputs, making it ideal for functions with more outputs than inputs.

Adjoint methods, on the other hand, initially applied by Jameson [140] to solve optimal control prob-

lems in aerospace, compute $-\Psi_k$ by solving the adjoint equations

$$\frac{\partial \mathcal{R}_k}{\partial u}\Psi_k = -\frac{\partial f}{\partial u}\,, \tag{4.3}$$

where a linear system with no dependence on x_n appears. This means that the total derivative computation is independent of the number of design variables, making this method ideal when it exceeds the number of output functions. Considering that, this method poses itself as an alternative to the reverse-mode AD that is not affected by the memory issue with the drawback being the considerable implementation effort [13, 141].

As discussed before, aerodynamic optimization problems typically have a reduced set of objective functions but many design variables, rendering gradient-free methods almost useless. Even when considering gradient-based methods, attention must be paid to the sensitivity analysis method to be used. In the recent years, adjoint methods have been the most promising option for aerodynamic optimization [123, 142], given the higher number of inputs (design variables) than outputs (objective functions). When using gradient-based optimizers, this is the approach used to compute derivatives throughout the MACH-Aero framework used in this work. Partial derivatives are computed with AD whenever possible, although finite difference methods are also used.

4.3 Design and parameterization

Parameterization is an important part of an aerodynamic shape optimization framework, as it has an impact in the results that are possible and on the computational time required. There are two main options for geometry parameterization: constructive and deformative methods. The first ones benefit from being directly connected to variables on the CAD geometry, resulting in an easier and more intuitive optimization setup as the design variables for the optimizer are directly related to parametric CAD variables. This also eliminates the need to convert the geometry back to CAD format at the end of the optimization process. Disadvantages include the computational cost to obtain sensitivity information and the need for the continuous parameterization of the geometry [143]. Furthermore, the parametric variables used to construct the CAD model are not necessarily directly correlated to those important for aerodynamic shape optimization and so, attention should be paid to that when building the CAD geometry [144], as a bad definition of those variables can severely limit the optimization space. Those methods have been used for aerodynamic shape optimization for example in [145] where constructive methods using Engineering SketchPad [146] were explored for aerodynamic optimization using the MACH-Aero framework and in [147] were the FreeCAD API was used and coupled with the SU2 framework [148].

As an alternative, deformative methods modify the geometry without requiring CAD or even the geometry itself, as only changes to the baseline geometry are parametrized [143]. One of the most popular approaches is Free-Form Deformation (FFD), which uses a box that completely embeds the surface mesh and that will be referred to in this work simply as the FFD box. The surface mesh nodes are then mapped to the FFD box with an $\mathcal{R}^3 \rightarrow \mathcal{R}^3$ mapping, determined by performing a Newton

search. After that, surface mesh nodes can be deformed by performing deformations on the FFD box nodes, given their mapping. A visual representation of this is given in Figure 4.3.

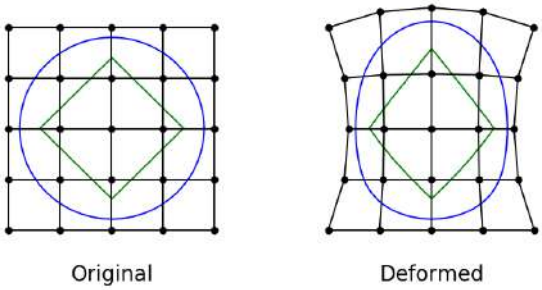


Figure 4.3: Free-form deformation and its influence on the grid. (Source: MDO Lab, University of Michigan)

The surface mesh deformations are then used to perturb the volume mesh using an hybrid algebraic-linear-elasticity mesh perturbation scheme [143] due to its resulting high grid quality and low computational effort. Figure 4.4 shows an FFD box enclosing a wing surface mesh. Compared to CAD based approaches, the FFD method is generally easier to setup and use with an already existing geometry and has freedom to parameterize multiple design variables easily. Furthermore, it is easier to impose constraints on the movement of individual nodes of the mesh by deforming it directly, ensuring an higher quality of the resulting grids.

Figure 4.4: FFD box used to deform a wing surface mesh.

Another important advantage of FFD methods is that they also allow efficient computation of analytical derivatives, much needed for gradient based optimization. For the cited reasons, deformation methods are usually the preferred choice to perform aerodynamic shape optimization, with several different cases tested during the years. For example, in [149] a Blender plug-in was used to perform free-form deformation and return the deformed geometry and sensitivities. In [150], FFD methods were applied to aerodynamic shape optimization of turbomachinery.

Given that in this work, optimization is to be performed using high fidelity numerical analysis with a relatively large set of design variables, a good and efficient parameterization is needed for every intermediate geometry, so a deformative method using FFD is used.

Chapter 5

Implementation

In previous chapters, various models to perform a flow simulation and compute sensitivities, as well as optimization algorithms were discussed. Here, it is intended to discuss some implementation details and give an overview of the process to perform a complete aerodynamic analysis and optimization. Figure 5.1 schematizes the process followed to optimize the aerodynamic shape of a wing. The blue shaded boxes represent the modules developed by the MDOLab at the University of Michigan and the gray box encompasses the MACH-Aero framework. The white text represents parameters that are automatically updated by the framework and the black text represents parameters that require user input.

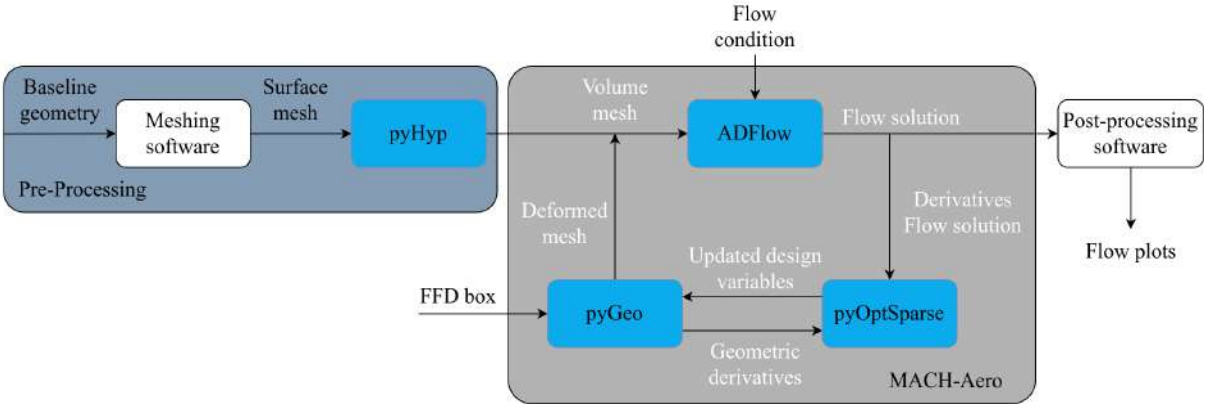


Figure 5.1: MACH-Aero optimization framework.

5.1 Geometry definition

The first step is to create a baseline geometry which will serve as a starting point for the optimizer. Considering that this geometry will be an input to the MACH-Aero framework, two main options are available for generating the geometry: using a typical CAD software or using pyGeo.

In the first option, the body should be created using a typical CAD or geometry generation software, like OpenVSP [151], a parametric aircraft geometry generation tool. The geometry should then be exported to a neutral CAD format (data exchange formats) that is accepted by the mesh generator.

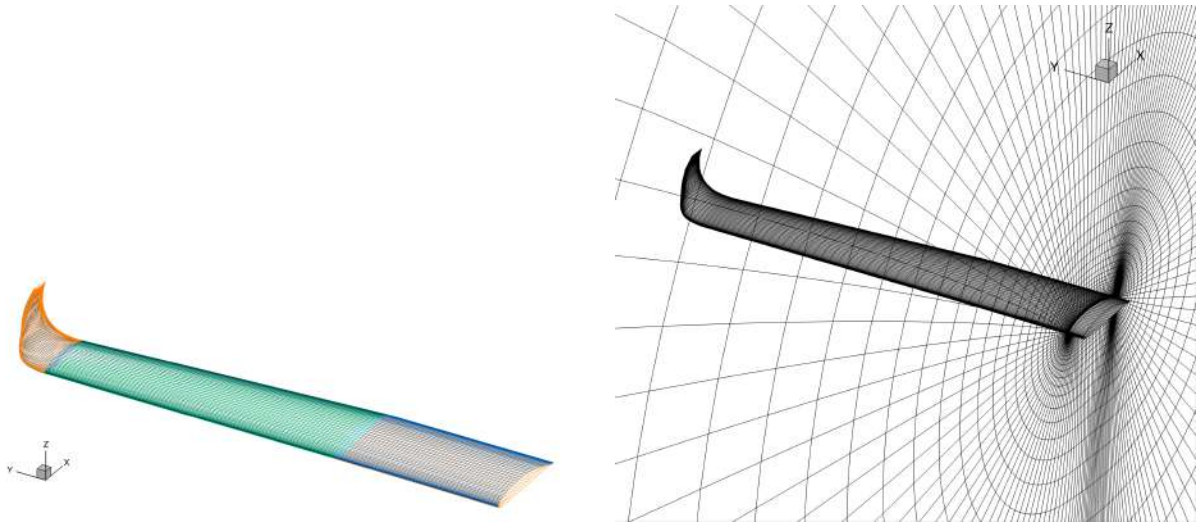
The second option makes use of pyGeo, a package which is part of the MACH-Aero Framework and performs geometry generation and manipulation. With pyGeo, it is possible to create a wing geometry by lofting surfaces between a provided set of airfoils in a code-based way. Independently of the selected method, pyGeo will always be used to handle the deformations during the optimization process, given that it uses a CAD-free approach and so, is agnostic to the geometry generation method. Considering this, for this work, OpenVSP was the main tool used to obtain the starting points for the optimizer given its intuitive yet powerful interface for wing geometry generation.

5.2 Meshing process

5.2.1 Surface mesh

From the geometry CAD file, a surface mesh must be obtained. For that, it is possible to choose from a variety of different software. However, the chosen grid generator should be able to generate a multi-block and/or overset 2D mesh, as those are the grid types accepted by the flow solver, and output the resulting grid to a CGNS (CFD General Notation System) file.

The most popular options are ICEM CFD® and Pointwise®. Some open source solutions for mesh generation, such as GMSH [152] are also widely available. However, the first requirement limits their usage in this work. Figure 5.2 a) shows an example of a surface mesh of the Tekever AR5 wing with the multiple blocks (totaling 17) used in this work. The topology employed is considered of O-type.



(a) Mesh blocks depicted with different colors.

(b) Volume mesh obtained using pyHyp.

Figure 5.2: Used mesh for the Tekever AR5 wing.

5.2.2 Volume mesh

Having a surface mesh in the appropriate format, an hyperbolic volume mesh is then extruded using pyHyp, an hyperbolic mesh generator that also automatically applies the necessary boundary conditions for a wing if desired [153]. Figure 5.2 b) shows a volume mesh obtained with pyHyp, as well as the

respective surface mesh. To generate a volume mesh from a surface mesh, pyHyp requires the user to provide a few parameters: the first layer height, the total height of the volume mesh and the number of layers to extrude. Several default parameters are available to change the algorithm used to generate the mesh. However, the default values were mainly used, as they are the recommended for the extrusion of a volume mesh around a wing [154]. The main exception is `cMax`, which control the maximum permissible ratio of the step in the marching direction. It should be reduced for more complex geometries, such as the case of the wing with the winglet, where a value of 0.1 was used in order to internally split the marching steps and ensure a more robust and higher quality mesh generation.

A volume mesh will be valid for use with ADFlow as long as there are no negative volumes present. However, ideally no negative quality cells should be present either. Cells quality is measured by the normalized determinant of the Jacobian, which measures the difference to a perfect hexahedral cube. During the extrusion process, pyHyp shows in the terminal the minimum quality at each step, facilitating the debug process. Furthermore, using `nTruncate` option, it is possible to stop the extrusion at a specified step to analyze potential quality issues. Although pyHyp is a robust volume mesh generator, particulary for wings, sometimes tweaking of the parameters may be needed to achieve optimal results.

Both the surface and the volume mesh should, naturally, be sufficiently refined, so grid dependence studies must be carried out before starting the aerodynamic analysis and optimization procedure. For this, the reader is referred to Section 6.1. As discussed in Section 3.2.3, the Spallart-Allmaras turbulence model will be used, which requires an $y+$ value of approximately one for the near-wall cells [155]. From the flat-plate boundary layer theory, it is possible to estimate a value for the initial wall spacing. Considering

$$\Delta s = \frac{\mathrm{y}^{+}\mu}{\mathrm{U}_{\mathrm{fric}}\,\rho}\,, \tag{5.1}$$

where $\mathrm{U}_{\mathrm{fric}} = \sqrt{\frac{\tau_{\mathrm{wall}}}{\rho}}$, $\tau_{\mathrm{wall}} = \frac{\mathrm{C_f}\rho{\mathrm{U}_\infty}^2}{2}$ and $\mathrm{C_f} = \frac{0.027}{\mathrm{Re_x}^{1/7}}$ [156] and the Reynolds number is given by

$$Re = \frac{\rho U_\infty c_{MAC}}{\mu}\,, \tag{5.2}$$

where c_{MAC} is the mean aerodynamic chord, U_∞ is the free-stream velocity, defined by the cruise speed (Table 5.2) and air density, ρ and dynamic viscosity, μ, which can be computed for the cruise altitude using the 1976 U.S. Standard atmosphere [157]. With this, a value for Δs of approximately $1.3e-5$ is obtained. For this reason, volume meshes were generated with an initial wall spacing $\Delta s = 1.0e-5$.

5.2.3 Overset mesh

The overset meshing approach was also used in this work to obtain a volume mesh including wing and fuselage to study the effects of the latter on the optimized wing. Its creation involved three main steps:

1. Creation of the volume grids for each individual component;
2. Creation of a collar grid;

3. Generation of a background grid and assembly of the individual grids.

In the first step, the process is similar to that described Section 5.2, although overset boundary conditions should be assigned to the outer faces, instead of farfield and the march distance should be significantly smaller, as the farfield will be filled with the background grid. Attention should be paid when building individual grids in order to keep the cell sizes similar between them in the intersection region.

In the second step, with the goal of avoiding gaps between intersections, a collar mesh should be generated between the wing and fuselage. It should typically be slightly more refined than the other two and ensure sufficient overlap. To this end, the full Tekever AR5 configuration was loaded into a mesh generation tool and the wing surface was trimmed at its intersection with the fuselage, resulting in an edge from which surface grids were extruded along the surfaces of the fuselage and wing. The resulting surface collar grid can be seen in Figure 5.3, represented in orange, along with th fuselage and wing surface meshes. For the volume extrusion of the collar grid, there is the need to specify how much the floating edges splay outwards at the extrusion through the `splay` parameter in order to obtain a better overlap with the other component grids. Furthermore, both the near-wall distance, Δs, and the growth ratio should be slightly lower than that used for the component grids. This ensures that the collar grid elements are selected as computation cells by the overset algorithm, although large size differences should be avoided, as they will result in orphan cells. For the case of collar grids, the default values for wings do not tend to produce good quality volume meshes. Some recommendations are to reduce `volBlend` parameter to a minimum as it does not improve volume meshes for collar grids. Volume coefficient (`volCoef`) should be increased to values near one to improve grid quality near the wall and finally, explicit, implicit and Kinsey-Barth smoothing parameters (`epsE`, `epsI`, `theta` respectively) should all be greatly increased to help prevent grid lines from crossing. Table 5.1 shows the values used for each of the components. If needed, specific boundary conditions can also be applied on each edge. Here, for instance, as Tekever AR5 has a high wing, it was necessary to create an edge on the collar surface grid in the symmetry plane region to define a symmetry boundary condition as otherwise the volume mesh would cross this plane due to the splay added.

Table 5.1: Values used for several pyHyp parameters.

	`epsE`	`epsI`	`theta`	`volCoef`	`volBlend`	`volSmoothIter`	`slExp`	`cMax`
Wing	1.0	2.0	3.0	0.2	0.0005	25	0.15	0.1
Fuselage	1.0	2.0	3.0	0.2	0.0005	20	0.15	1.0
Collar	10.0	20.0	6.0	0.9	$1 \cdot 10^{-8}$	20	0.35	1.0

In the last step, a background mesh needs to be generated to enclose all the nearfield grids. pyHyp has a simple function, `simpleOCart`, that creates an uniform cartesian grid around the nearfield meshes and then marches with an O-type topology grid to the farfield distance.

Having the component nearfield grids, the collar grid for the intersection and the background mesh, they should be merged together into a single CGNS file, which can be easily done using the tool cgnsutilities, also from MDO Lab at the University of Michigan. With all grids on a single file, the Implicit Hole Cutting (IHC) algorithm present in ADFlow will automatically assign the overset connectivities and

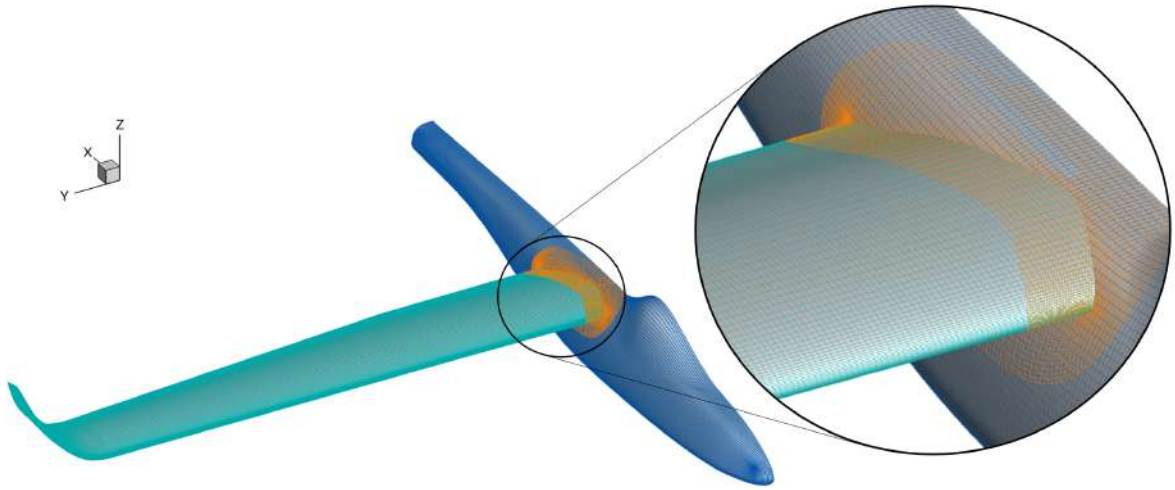
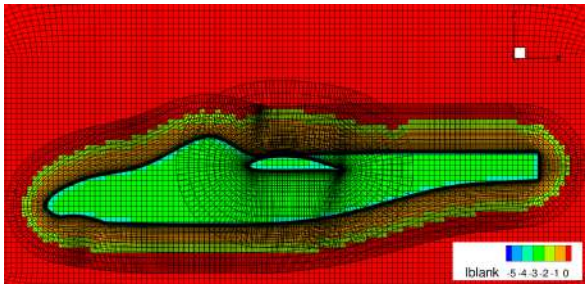
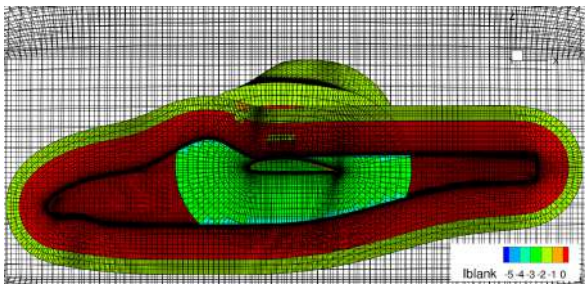


Figure 5.3: Overset surface grids of the Tekever AR5 fuselage and wing.

define the role of each cell (which can be compute, blanked, interpolate or flooded cells). A flooding process is applied to automatically determine which is the inner or outer side of the wall to blank cells inside the geometry and form an overtight outer mold line representation of the geometry. The algorithm is successful if no orphan cells are present in the domain, although attention should also be paid to the number of flood cells, as a high number may indicate a leak between the inner and outer side of the geometry. Figure 5.4 shows the resultant overset grid, with the role of each cell. Red cells are compute cells, orange are blanked cells, lime are interpolation cells, shown around the blanked cells, green are flooded cells, shown inside the fuselage and light blue are flood seeds, around the fuselage.



(a) Types of cells on the background mesh.



(b) Types of cells on the nearfield mesh.

Figure 5.4: Types of cells on the overset grid.

5.3 Flow simulation

As stated in Section 1, the Tekever AR5 wing is to be optimized. For this reason, the first step is to assess the performance of the current. Its cruise conditions are summarized in Table 5.2.

The analysis should be performed for a specific angle of attack, which can be determined by optimization with a single design variable and a prescribed lift coefficient as constraint or with a secant method provided by ADFlow to find the initial angle of attack for a certain lift coefficient. It should be

Table 5.2: Performance parameters of Tekever AR5 [158].

Cruise speed	U_∞	$100\ km/h$
Cruise altitude	h	$305\ m$
Maximum Take-off Weight	$MTOW$	$180\ kg$
Endurance	E	$20\ h$

noted that the application of angles such as angle of attack and sideslip angle change the inflow angle and not the mesh.

Tekever provided the wing lift coefficient $C_{L_{wing}} = 0.8932$, considering the projected area of the Tekever AR5 half wing, $S_{wing} = 2.1691m^2$, and the cruise conditions presented in Table 5.2.

It is important to estimate the Reynolds number to understand what kind of simulation will be performed and choose the models adequately. Using Equation (5.2), it is possible to obtain a Reynolds number of $1.1 \cdot 10^6$. This is a moderate Reynolds number, and it indicates that turbulent flow will be dominant. The effects of viscous forces on total drag force will be much smaller than those caused by pressure forces, but they should not be ignored.

Mach number is another dimensionless parameter that should be calculated, not only because it provides an insight into the kind of flow to be expected, but also because it is an input for the AeroProblem. It is given by

$$M = \frac{U_\infty}{\sqrt{\gamma RT}}\,, \tag{5.3}$$

where γ is the adiabatic constant, R is the gas constant and T is the temperature, calculated also with the 1976 U.S. Standard Atmosphere for the cruise altitude (T=$286.169K$). This results in a Mach number of 0.08164, which is very low, so no transonic effects are to be expected. Considering that ADFlow is a compressible flow solver, and despite the robustness of the ANK algorithm, some care must be taken when setting up the simulation. In [159], the application of ADFlow to incompressible flow was discussed and some options were added for this purpose. The first one is the application of characteristic-based preconditioning to the ANK time step matrix, which should be used in conjunction with the option `acousticScaleFactor`, which defines a factor that multiplies the acoustic contribution in the spectral radius computation, set to approximately the freestream Mach number [98]. The preconditioner can be either Turkel or van Leer-Lee-Roe. Finally, the number of linear solver iterations should be increased.

Table 5.3 summarizes the main parameters used for the CFD simulation. Other parameters were kept at their default values [98].

Table 5.3: CFD solver parameters used for the simulation.

Discretization	Central difference with JST scalar dissipation
Equation type	RANS
Equation mode	Steady
Turbulence model	Spalart-Allmaras

For the JST dissipation scheme, the default values were set to 0.67 for the exponent factor, 0.0156

for the coefficient of the fourth order dissipation and 0.25 for the second order dissipation, although it may be set to 0.0 for fully subsonic simulations.

To solve the linear equations, the size of the GMRES subspace is 60 for the NK solver and the Eisenstat-Walker algorithm was used to determine linear convergence at each iteration. For both the adjoint, NK and ANK solvers, the additive Schwarz method was used as a global pre-conditioner. The Jacobian used for the pre-conditioner is lagged by 20 iterations for NK and 10 for ANK solver.

A DD-ADI solution methodology with first order numerical accuracy was used for the turbulence model, as this facilitates the adjoint systems solution. Although different versions of the Spalart-Allmaras turbulence model are available in ADFlow, the noft2 variant without the rotation correction was used and so, the f_{t2} tripping term on (Equation 3.4) is not included. A decoupled formulation was used in the ANK solver and a coupled one was used for the NK solver.

Regarding the stopping criteria, either a maximum of 10000 iterations or a convergence of the residuals L2-norm to $1 \cdot 10^{-6}$ in relation to the residuals obtained in the first iteration will be used. Furthermore, both ANK and NK solvers were used, with the switch occurring at a residuals decay of $1 \cdot 10^{-4}$.

5.4 Optimization process

Thus far, implementation details for an aerodynamic analysis process were discussed. This section, is focused on optimization set-up. Firstly, the framework used is described, then its implementation to the problem at hand is detailed.

5.4.1 MACH-Aero framework

The MACH-Aero Framework [160] is composed by the following main modules: ADFlow, the CFD solver which was already discussed in Section 3.6, pyGeo, the geometry manipulation tool that will be described in Section 5.4.3, IDWarp, which is used by pyGeo for the volume mesh deformation and pyOptSparse, which handles the constrained nonlinear optimization problem [161] with a variety of algorithms.

As depicted in Figure 5.5 using a XDSM diagram, a baseline design and the corresponding volume mesh is needed to perform the optimization process and the process to obtain them was described in Section 5.2. The optimizer itself (wrapped with pyOptSparse) is the driver for the procedure and it receives information about geometric constraints and derivatives, given that gradient-based optimization is to be performed, to update the values of the FFD design variables, which will be used to deform the surface mesh with pyGeo and then propagated to the volume mesh with IDWarp. The new volume mesh is then analyzed with the flow solver (ADFlow) and state variables for the current iteration are computed. The adjoint solvers recalculate the derivatives to pass to the optimizer and the loop continues until the desired convergence is achieved and an optimized design is found.

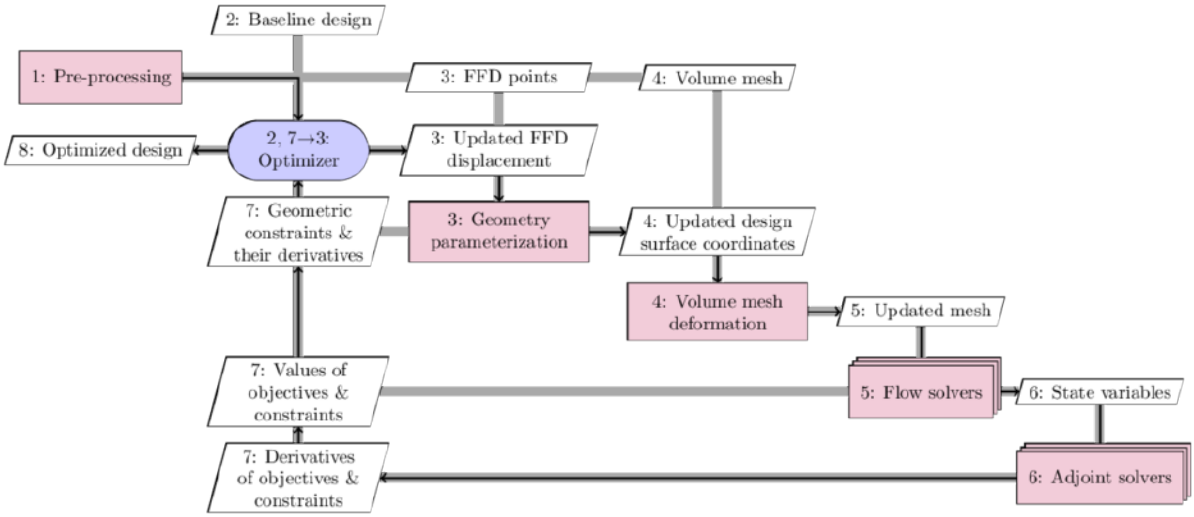


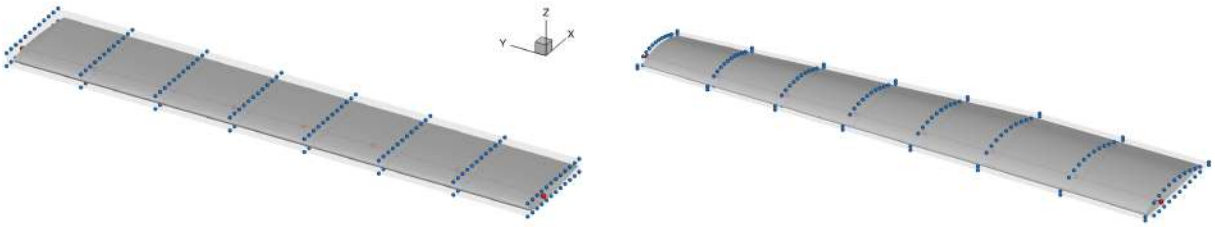
Figure 5.5: Overview of the aerodynamic optimization process. (Source: MDOLab at the University of Michigan)

5.4.2 FFD box generation

As discussed before, the FFD method is used for the geometry parameterization, implying the creation of an FFD box. For simple geometries, a Python script can be written to generate the set of points that makes up the FFD box. An example of the result is shown in Figure 5.6 (a). However, the FFD box should be centered with the wing, as the reference axis position is calculated based on the FFD box and not on the geometry itself.

With enough parameterization freedom, it would be possible to obtain the same optimized design with different FFD boxes, even when they are not centered with the wing. However, in this work, it is intended to provide physical meaning to the design variables and correlate them with common wing shape parameters. Furthermore, as the starting geometry will be an existing wing, it is important that the parameterized design variables match those previously used for the original wing design. For this reason, it is important that the reference axis is located at the quarter-chord of the wing, so a pyGeo function to create an FFD box fitted to the wing (`createFittedWingFFD`) was used. The function takes as inputs the wing geometry, a list of leading edge and trailing edge points, the number of points in each direction and the margin to the wing surface. The function starts by creating a 2D grid with the user-specified number of points in each direction between the leading edge and the trailing edge, which can be composed of several sections. Then it projects the points to the geometry surface in the normal direction to the 2D grid using another pyGeo function developed for geometric constraints and applies the specified margin in each direction. A result obtained with this approach can be seen in Figure 5.6 (b).

This second approach works well for mostly straight wings with clearly defined sections and moderate dihedral and twist angles. However, for complex wing geometries, like the current Tekever AR5 wing, it does not work at all, as the margins are applied according to the global coordinate system, thus always in the x, y, z directions, which differ greatly from a local reference axis at the winglet. Figure



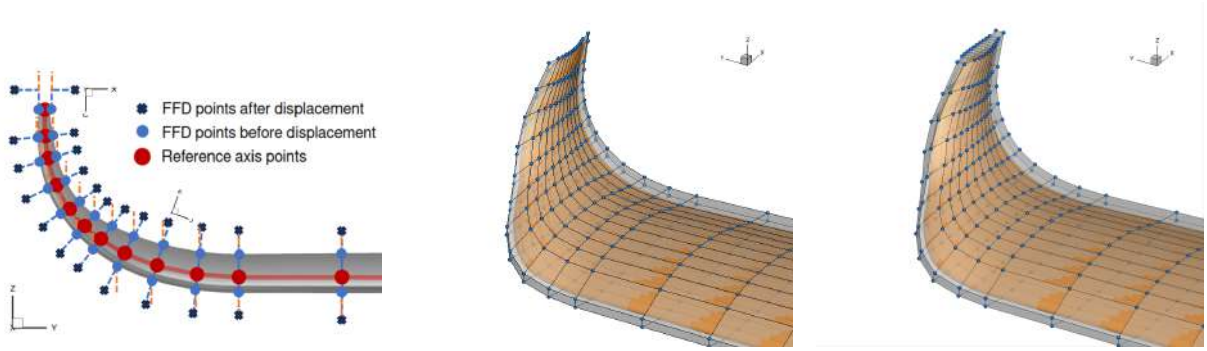
(a) Rectangular wing with symmetric airfoil.

(b) Simplified Tekever AR5 wing.

Figure 5.6: FFD boxes around different wings.

5.7 (a) illustrates these differences. Blue points represent the projection of the points to the surface, according to the normals of the 2D surface. Dashed orange lines then represent the direction of the points displacement when the margin is applied on the current implementation: in the planar region of the wing, the local axis is aligned with the reference axis and so, the margins are applied correctly, but in the winglet region, the local axis is rotated almost 90° relative to the global axis and the margins are applied in a completely wrong direction. Dashed blue lines represent the corrected displacement direction, along the k-direction of each local section.

To deal with this, the `createFittedWingFFD` function was modified to generalize it. The normals at each point were already computed by the function that does the projections onto the surface so they were added as one of the function outputs. The normals were then used to rotate the margin vector from the global reference frame to the local reference frame at every point, ensuring that the margins would be added in the appropriate directions. With this modification, Figure 5.7 compares the FFD box that was obtained with the original function and that obtained with the new one, with visibly higher quality. With this, a generalized function to create FFD boxes around arbitrarily complex wings with minimal user intervention was obtained. An important aspect to keep in mind while gathering those coordinates is that sections must be parallel to one of the global axis, with the $x-$axis being the more obvious one in this case, as this ensures correct linkage from the FFD points to the reference axis.



(a) FFD box generation scheme.

(b) FFD box originally generated by the script.

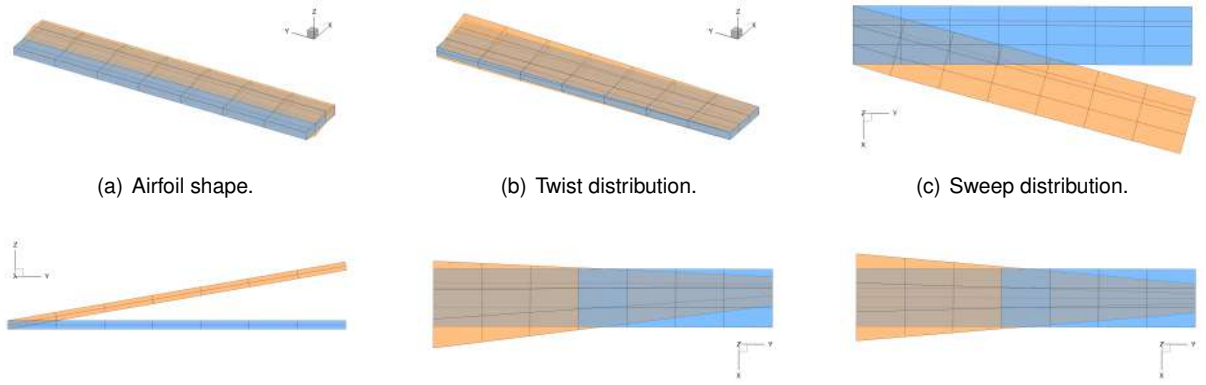
(c) Corrected FFD box.

Figure 5.7: FFD box around the winglet before and after changing the script.

5.4.3 Parameterizations

Within the MACH-Aero framework, deformative methods using FFD are used and the geometry manipulation is handled by a set of packages. pySpline [162] is used to generate and work with b-splines (curves, surfaces and volumes) and provide the derivatives of the created entities. The individual curves, surfaces and volumes are then joined together in pyGeo, a geometry manipulation tool that was specifically built for multidisciplinary optimization applications and allows the manipulation, parametrization and constraint handling of the geometric shape [143].

The nodes of the FFD box may be displaced individually in any spatial direction as local design variables. However, pyGeo also allows to create relations between node displacements in such a way that they are all affected by a smaller set of design variables, allowing the geometry parameterization with global design variables. This second approach is useful as the reduced number of design variables saves computational resources during the optimization whilst allowing for the creation of functions that can represent more intuitive design variables such as taper or twist. An example of a local design variable with vertical displacements is given in Figure 5.8 a) where the parameterization was set up such that all sections of the FFD box would mimic the same deformations.



(a) Airfoil shape. (b) Twist distribution. (c) Sweep distribution.

(d) Dihedral distribution. (e) Chord distribution (reference axis at 25%). (f) Chord distribution (reference axis at 50%).

Figure 5.8: Deformations to the FFD box associated with common design variables.

To create a global design variable using pyGeo, in addition to the FFD box, an axis is necessary. The axis can be created by specifying its direction and relative position in the FFD box using the ratio of two specified directions. For a typical wing optimization problem, the user should specify the direction that follows the wingspan, and the ration of the vertical and streamwise position of the axis, which will be, for most practical cases, 0.5 and 0.25 respectively, representing the quarter chord of the wing. It is important to note that the wing should be centered within the FFD box. The reference axis will be used to project all FFD nodes into it using an user defined direction for the projection. The points will then become rigidly linked to the reference axis, so any deformation on the FFD axis will have an effect on several FFD points at once [163]. Figure 5.9 shows the links between each FFD box point and the reference axis, represented as blue lines. The reference axis is represented in red, with its respective control points. A deformation on the FFD axis may be of three types: displacement, rotation and scalling.

The first type of deformation is the displacement of the points, which may be used to create dihedral

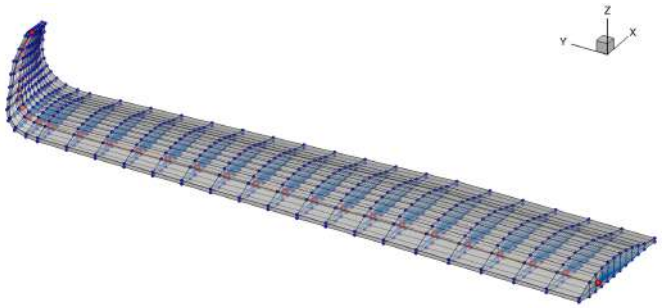


Figure 5.9: FFD box used to deform a wing surface mesh.

or sweep, for example, as Figures 5.8 d) and 5.8 c) show. pyGeo also allows rotations and scaling in any one of the three directions x, y and z. The most obvious application of a reference axis rotation would be to create a twist distribution, depicted in Figure 5.8 b), but it is also useful to keep the airfoil sections perpendicular to the reference axis and, thus, effectively keeping the same airfoil by applying rotations to the reference axis points across the axis of rotation. This effect is clearly visible in Figures 5.8 d) and 5.8 c) The scaling function can scale the FFD box points in a specified direction. Figures 5.8 e) and 5.8 f) show an example of scaling functions being applied to create a chord distribution. Besides the scaling in the streamwise direction, a scaling in the vertical direction was also applied to maintain the airfoil thickness to chord ratio. While Figure 5.8 e) shows a chord distribution for a reference axis at 25% of the FFD box, Figure 5.8 f) shows a chord distribution for a reference axis at 50%.

The design variables are not necessarily defined by a single number and can have as many degrees of freedom as the number of points in the reference axis. It is even possible to parametrize distributions of the variables with smaller sets of variables: a linear twist distribution can be parameterized with only the slope as the design variable, provided the twist at the wing root is fixed, for example.

As it was the case when generating the FFD box, parametrizations that are straightforward with mostly planar wings poses challenges when the local reference axis are not aligned with the global one. When dealing with local design variables, pyGeo offers a function to displace them using the local reference frame of each section instead of the global one. However, as global design variables are user-built with elementary operations (rotation, scalling and displacement), attention must be paid to the definition of global design variables. Figure 5.10, for instance, shows the result of applying twist using the global reference frame with the green geometry - a rotation was applied around the y-axis using `rot_y` and although the result is what would be expected in the planar region, in the winglet it is not the intended one. The orange geometry represents the undeformed FFD box.

To keep the rotation around the reference axis, the vector defining it at each point is calculated. At the wing root, it is the vector pointing from the first to the second point. At the wing tip, it is the vector pointing from the second to last to the last point, and at any arbitrary point i it is the vector between $i-1$ and $i+1$, eliminating the ambiguity of having two different slopes on each side as the reference axis is defined by straight lines between sections. This is represented in Figure 5.11 (a). With this vector calculated, it is possible to project the rotation around the global y-axis to a rotation around the vector itself. The result of this is shown in Figure 5.10 in blue.

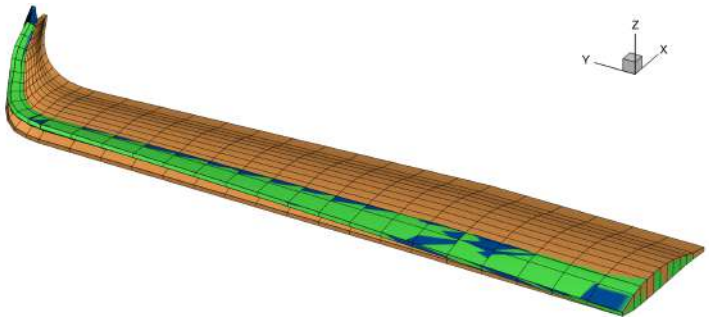
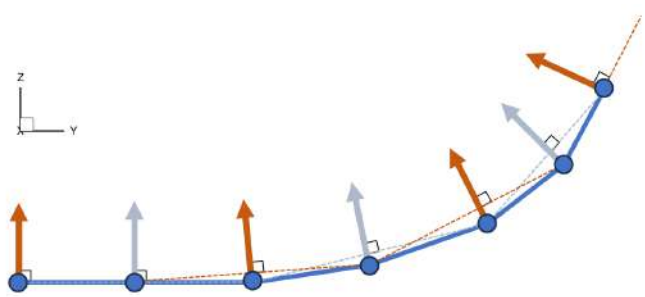
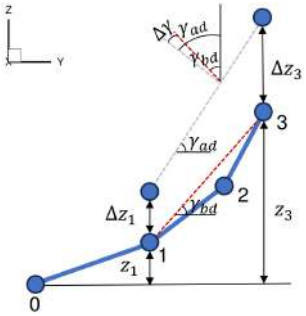


Figure 5.10: Comparison of different approaches to twist deformation on the FFD box.



(a) Method to compute the normals in each section.



(b) Section angles before and after applying the displacements.

Figure 5.11: Schemes detailing the parametrizations.

For dihedral and sweep, in addition to the displacements, there is the need to rotate the FFD sections to keep them perpendicular to the reference axis. Figure 5.11 (b) schematizes the process to compute the additional rotation needed for dihedral, $\Delta\gamma$: the orientation of the section before deflection, γ_{bd} is computed using the coordinates of points $i-1$ and $i+1$ and the required orientation after deflection, γ_{ad}, is computed considering the coordinates of the same points and the vertical displacements, Δz, applied to each of them through the design variables, with the rotation angle to be applied then being the difference between both, given by

$$\Delta\gamma = \underbrace{\text{arctg}\,\frac{z_{bd_{i+1}} + \Delta z_{i+1} - \left(z_{bd_{i-1}} + \Delta z_{i-1}\right)}{y_{i+1} - y_{i-1}}}_{\gamma_{ad}} - \underbrace{\text{arctg}\,\frac{\left(z_{bd_{i+1}} - z_{bd_{i-1}}\right)}{y_{i+1} - y_{i-1}}}_{\gamma_{bd}}, \tag{5.4}$$

where design variables are represented in red.

Finally, for chord distribution, the scale factor applied in the streamwise direction should also be applied in the direction of the section (represented with blue and red vectors in Figure 5.11 (a). For this, the orientation of the section, given by γ_{bd} is computed and the scale factor is projected onto it, ensuring a constant thickness-to-chord ratio. The code for all parameterizations is given on Appendix C.

5.4.4 Geometric constraints

Besides the parametrization discussed before, pyGeo can also be used to set geometric constraints. It is possible to add different types of constraints, being the most common: minimum thickness con-

straints, to ensure that there is enough room for structural components; minimum volume constraints, to guarantee enough internal volume to carry a specified amount of fuel; curvature constraint, typically used to ensure manufacturability; and LeTe constraints to avoid shearing twist at the leading and trailing edge when local design variables are present by ensuring that both top and bottom control points at those edges move in equal and opposite direction.

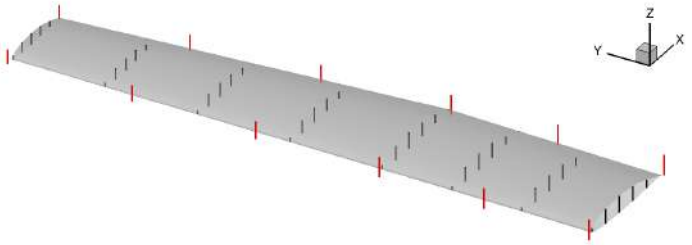


Figure 5.12: Minimum thickness and LeTe constraints defined using pyGeo.

Minimum thickness and volume constraints may be specified with either an absolute or a relative value to the baseline design. Figure 5.12 shows an example of the used minimum thickness (in black) and LeTe (in red) constraints. They are only enforced at certain discrete locations and are independent of the FFD nodes. To specify them, the user should provide a list of coordinates defining the leading and trailing edge, and the number of constraints to apply in each direction. As with the FFD box generation, a 2D grid is created between them and each point is projected to the upper and lower surface. When the geometry is deformed, the change of constraints location is handled automatically by pyGeo, which also takes care of the needed operations to use them as constraints in the optimization problem. In this work, only minimum thickness constraints and LeTe constraints will be used. Minimum volume constraints are not required, considering that the UAV in study does not store the fuel in its wings.

5.4.5 Optimization problem

With both constraints and design variables defined in pyGeo, they need to be passed to pyOptSparse for the optimization process. Variables from the aero problem, defined using the baseClasses can also be passed as optimization variables and a particularly important example of that is the angle of attack. When adding the angle of attack as a design variable, attention must be paid to its relation with twist, given that a geometric rotation of the root wing section has an equivalent effect to the angle of attack. For this reason, twist at the root is usually kept fixed and not added as a design variable. Dihedral and sweep were also fixed at the root, as they would not be meaningful for the isolated wing case. This also ensures that the shape of the wing mounted at the fuselage is conserved. The same principle may be applied to the other design variables.

Non-geometric constraints can also be created by user defined functions. The most important non-geometric constraint when optimizing for cruise flight is the lift constraint, which should equal a prescribed lift coefficient at cruise condition and ensure that it is possible to sustain flight. In a similar way,

it is possible to define the function or combination of functions to be used as the optimization objective. In this case, the optimization objective will be the drag coefficient minimization.

With the design variables, usually defined by pyGeo parametrizations, constraints from both the geometry and the AeroProblem and the objective function passed to the optimization problem, the optimization set-up is effectively complete and ready to run. Given that pyOptSparse sets up the optimization in a problem oriented manner, a problem statement in the standard form is easily translated to the optimization framework. The full optimization problem studied in this work is given by Table 5.4.

Table 5.4: Formulation of the full optimization problem in standard form.

		Quantity	Lower bound	Upper bound	Units
minimize	C_D	1	-	-	-
w.r.t.	α	1	0	15	$^\circ$
	γ (twist)	27	-15	15	$^\circ$
	c (chord)	22	15	150	%
	Λ (sweep)	27	0	10	$^\circ$
	Γ (dihedral)	27	-0.1	0.1	m
	airfoil shape	672	-0.05	0.05	m
subject to	$C_{L_{cruise}} = C_{L_{prescribed}}$	1	0.8932	0.8932	-
	$S_{proj} = S_{prescribed}$	1	2.1691	2.1691	m^2
	t (thickness constraint)	210	90	200	%

5.5 Flow visualization and post-processing

When it comes to post processing, Tecplot® and Matplotlib [164] were the main tools. Simulation data obtained from ADFlow is written into a CGNS file, which can be read by Tecplot®, and includes both surface and volume variables that can be visualized and analyzed, which include pressure, velocity in the three directions, magnitude of skin friction coefficient, dimensionless wall shear-stress components and $y+$ value of the cell center of the first cell were used, residual values and more. The full list of available surface and volume variables can be checked in the documentation [98].

It is important to note that specific plots that may be wanted by the user, such as lift distribution along the chord or pressure coefficient at different sections along the wingspan can also be obtained, but code for gathering data to that end should be added to the run script of the aerodynamic problem.

Concerning optimization, some aspects should be taken into account. Although lift distribution over the wing and pressure coefficient variation across specified sections can be obtained in Tecplot®, it is preferred to obtain them using ADFlow directly, as this ensures that the query locations will always be the same when the geometry is deformed. The result of this are text files that will be parsed to obtain the desired plots using developed Python scripts, allowing to quickly generate the hundreds of plots required due to all the optimization iterations. Another aspect is the conversion of the optimized geometry back to CAD, which is essential for manufacture. For this, after performing the optimization with the volume mesh, the CAD geometry in an IGES file is exported and embedded in the same FFD box used for the optimization. Then, the final design variables are applied to deform the FFD box and consequently, the CAD geometry.

Chapter 6

Baseline Wing Aerodynamic Analysis

In the previous section, the model to be simulated was defined. Here, it is intended to provide and comment the results obtained for the aerodynamic analysis of the current Tekever AR5 wing in order to assess its performance, to later compare it with the optimized solution.

6.1 Grid studies

6.1.1 Initial off-wall spacing

In Section 5.2.2, the initial off-wall spacing was estimated using the flat plate correlations to obtain an $y+ \approx 1$, required by the Spalart-Allmaras turbulence model and from that, a value of $1 \cdot 10^{-5}$ was used for the initial off-wall spacing of the hyperbolic volume mesh extrusion. However, the approximation needs to be verified. Figure 6.1 shows the obtained $y+$ on the surface of the wing and it can be seen that it is generally slightly smaller than one, thus satisfying the initial condition.

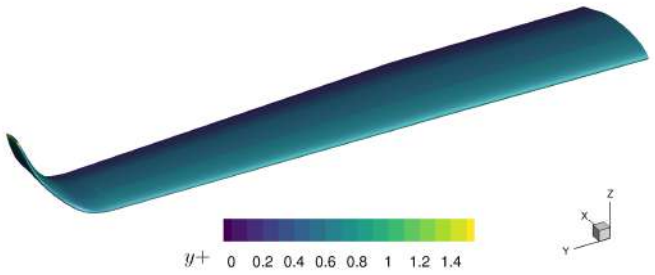
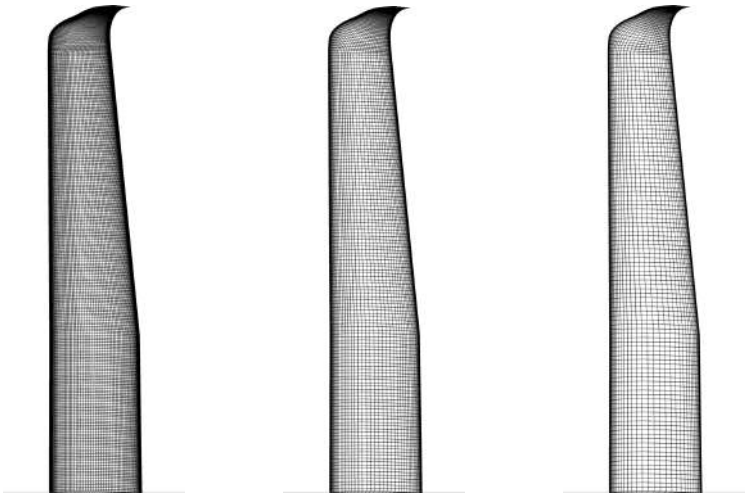


Figure 6.1: $y+$ on the Tekever AR5 wing.

6.1.2 Number of elements

A grid convergence study was performed by simulating the Tekever AR5 wing at an angle of attack of $1.5°$. Given the upper limit of RAM memory available (128 GB), the maximum number of elements that could be used was around $7 - 8$ million. Taking this limitation in mind, a grid with the maximum number

of elements was obtained (Figure 6.2 (a)) and the convergence study was performed by successively coarsening it with an approximate uniform ratio of 1.15^3 between grids, with some examples shown in Figures 6.2 (b) and (c). Sufficient grid refinement is important not only for analysis accuracy but also for optimization, namely chord distribution optimization, during which the mesh in the streamwise direction will get compressed or enlarged together with the geometry, effectively changing the elements local spacing. This effect will be particularly relevant at the wingtip, where induced drag is expected to be captured. All grids have a greater element clustering near the leading and trailing edge in the streamwise direction (with the spacing being 0.1% and 0.2% of the local chord respectively, as suggested in the literature [96]), as well as near the tip in the spanwise direction.



(a) Grid with 7,544,832 elements.

(b) Grid with 2,156,868 elements

(c) Grid with 644,112 elements

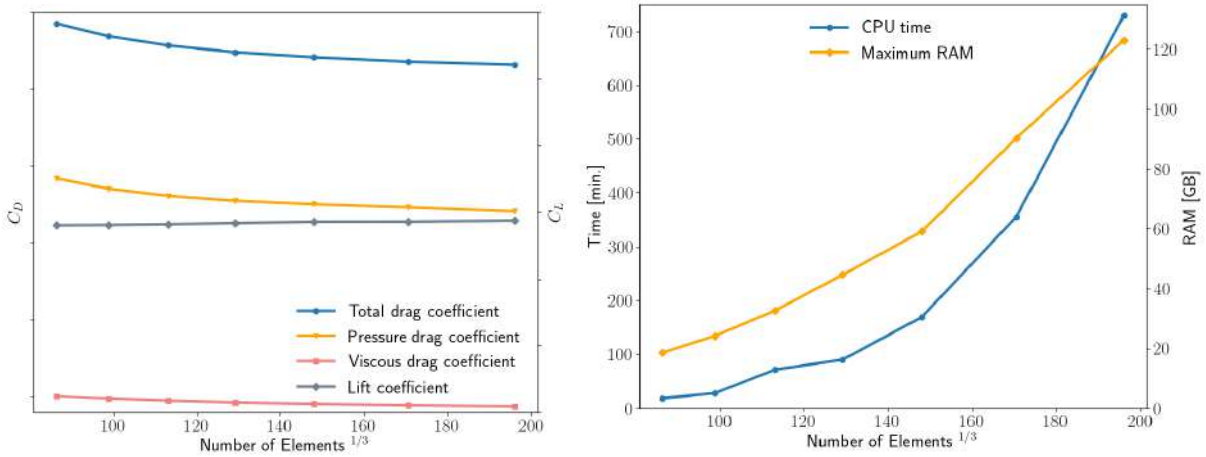
Figure 6.2: Surface discretization of some of the studied grids.

Figure 6.3 (a) shows the evolution of the total drag coefficient, C_D, and the viscous (C_{D_v}) and pressure (C_{D_p}) components, as well as the lift coefficient, C_L. It is possible to observe that the drag coefficient is converging as the grids are refined, with both viscous and pressure components following a similar evolution. For the lift coefficient, the difference to the more refined grid was minimal throughout the tested range, with the relative difference being smaller than 0.4% between the finer and coarser grids. However, for the drag coefficient relative differences were higher, at 8.4% between the finer and coarser grid. To achieve a relative difference below 1% in relation to the finer grid, a mesh with 4.96 million elements would be required.

On the other hand, it can be seen in Figure 6.3 (b), where the required computational resources for the different grids tested are shown, that the increase in CPU time needed for the finer grids is very noticeable, with the more refined one taking more than 12 hours (CPU time) to converge running on four cores of a CPU with a clock speed of $4.5GHz$ and the one with 4.96 million elements taking almost six hours (CPU time), which was deemed far too expensive for optimization purposes. For this reason, the grid with 1.45 million elements was selected taking into account that although the drag error is 3.9% when compared to the finer grid, it converged in 20% of the time required for the grid with 4.96 million elements. The absolute drag values resulting from the optimization process are not expected to be accurate, but

as long as the grid is fine enough to capture relevant physical phenomena, it is expected that it will lead the optimization to the correct trend considering relative gains from the starting point. To ensure this, the final geometry is verified using a finer grid. Figure 6.3 also shows a linear increase in the required RAM at around $1.7GB$ per $100,000$ elements, highlighting the upper limit for the grid size.

The complete results in tabular form can be checked in Section B of Appendix B.



(a) Lift and drag coefficients variation with number of elements.

(b) Computational resources variation with number of elements.

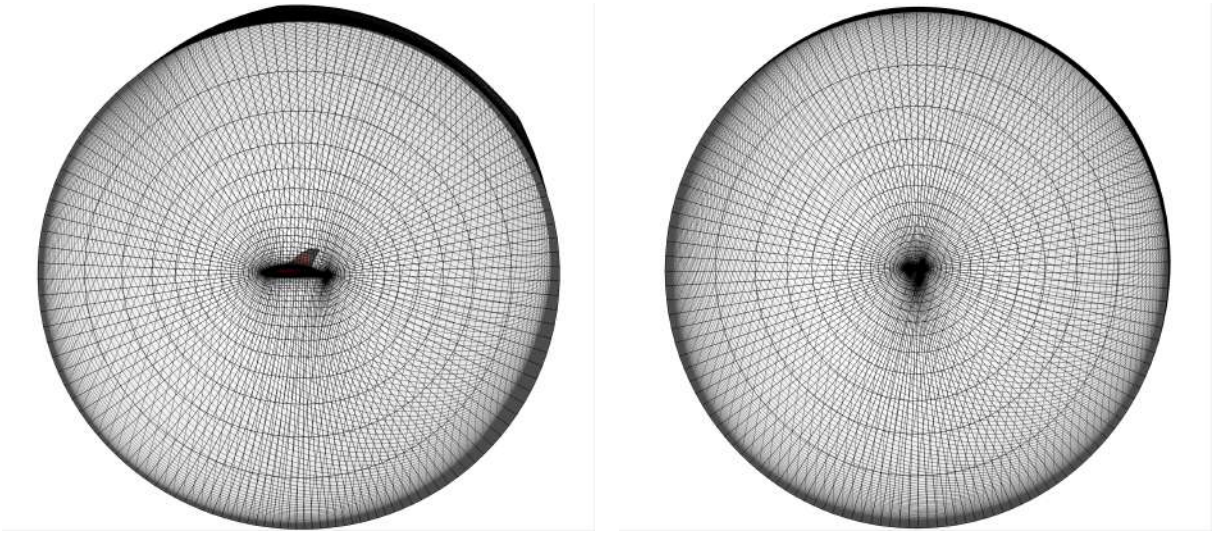
Figure 6.3: Grid convergence study.

6.1.3 Domain size

The influence of the domain size in the solution was also evaluated. For this, several grids with different extrusion distances from the surface normal were tested using the surface mesh chosen for the optimization process in Sub-Section 6.1.2. The volume mesh was extruded from the surface mesh using pyHyp, which uses a geometric progression to calculate the marching distance of each new layer of cells [153], given by

$$\Delta d_1(1 - q^{n_k - 1}) - d(1 - q) = 0\,, \tag{6.1}$$

in which Δd_1 corresponds to the marching distance of the first layer from the surface, q is the geometric progression ratio, n_k is the number of elements in the off-wall direction and d is the total marching distance. Here, the influence of d is studied. However, the geometric progression ratio needs to be kept constant to ensure that the results are not affected by the mesh refinement in the off-wall direction, so n_k is also changed, according to Equation (6.1) and q will be kept at around 1.25. As a result, the number of elements will increase at a much smaller rate than the size of the domain, with the far-field with 25 chords requiring just 14% more elements than the one with five chords. The domains are measured in mean aerodynamic chords and the extreme cases are presented in Figure 6.4. The obtained results are presented in Figure 6.5. The error between 25 chords and five chords is 3.26% and 7.45% for C_L and C_D respectively, and just 0.16% and 0.45% between 25 and 20 chords which is considered acceptable and



(a) 5 chords. (b) 25 chords.

Figure 6.4: Volume grid of some of the tested domains (not to scale).

thus, the domain with 20 chords was chosen for this work. It can also be observed that, as expected, the viscous drag coefficient is barely affected by the domain size, with the largest error of 0.22%. The complete results in tabular form can be checked in Section B of Appendix B.

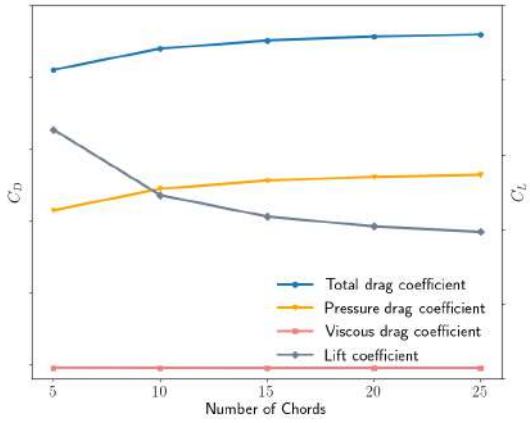
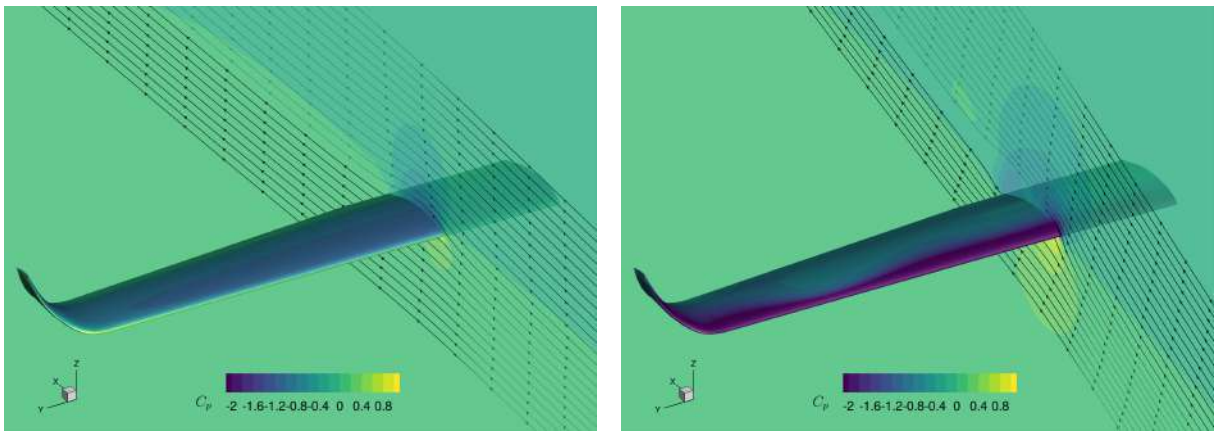


Figure 6.5: Domain size convergence study.

6.2 Cruise performance

The pressure distribution over the surface can be observed in Figure 6.6 (a) for an angle of attack of $1.9°$, which was found to be the one that produced the prescribed $C_{L_{wing}}$ for cruise conditions. It is possible to observe that on the upper surface, pressure is lower closer to the leading edge and in the winglet region and in this condition, the flow remains attached to the wing. Figure 6.6 (b), on the other hand, shows a flow distribution for a more extreme angle of attack, where it is clearly possible to see that the flow does not attach to most of the suction side of the wing. For comparison purposes, the pressure

coefficient (C_p) scale was truncated at $C_p = -2$, although for Figure 6.6 (b) the lowest verified value is $C_p = -10.2$.

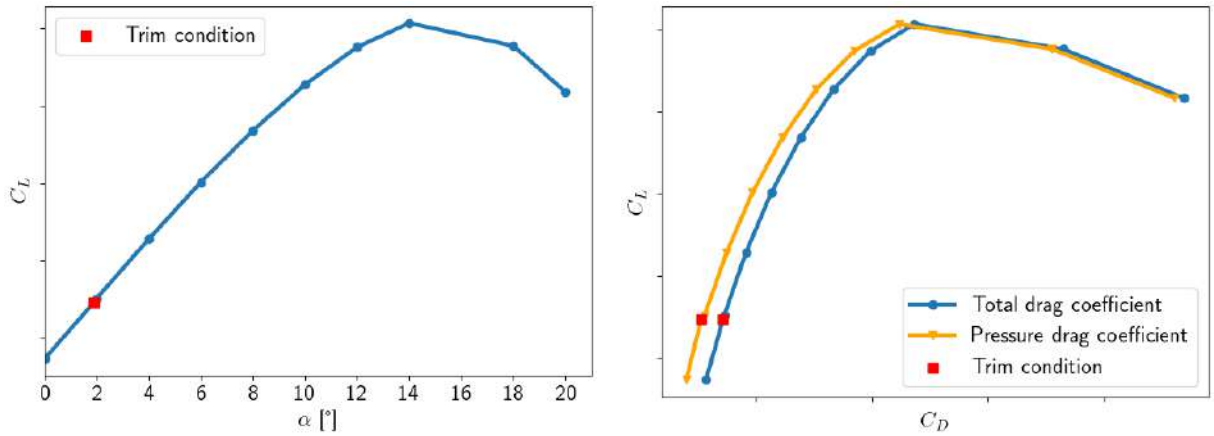


(a) $\alpha = 1.9°$.

(b) $\alpha = 20°$.

Figure 6.6: Pressure distribution and velocity streamlines for different angles of attack.

To observe the behavior of the wing with angle of attack, a study was performed and the results presented in Figure 6.7 a). It is possible to see that C_L and C_D increase continuously with angle of attack until around $\alpha = 14°$ and the variation is linear until around $\alpha = 8°$, after which boundary layer separation starts to occur on the suction side. In the linear region, $C_{L_\alpha} = 0.075/°$. A drag polar is also presented in Figure 6.7 (b), were efficiency can be analyzed. In this case, it would be higher for the smallest angle in the tested range, of $0°$, and the maximum would possibly be for negative angles of attack, outside the tested range, decreasing with increasingly higher angles of attack. The red square represents the trim operating condition, and it is possible to see that the wing is not operating at its maximum efficiency. In the same figure, C_{D_p} is also shown and as expected, its contribution to the overall drag increases with increasingly higher values of C_L, while the viscous contribution reduces continuously as flow separates from the surface.



(a) Lift coefficient curve.

(b) Drag polar.

Figure 6.7: Wing performance for angles of attack between $0°$ and $20°$.

Figure 6.8 shows with more detail the flow near the winglet at cruise condition, with the wingtip vortex being formed due to the pressure differential on both sides and the other vortices along the span merging

with it.

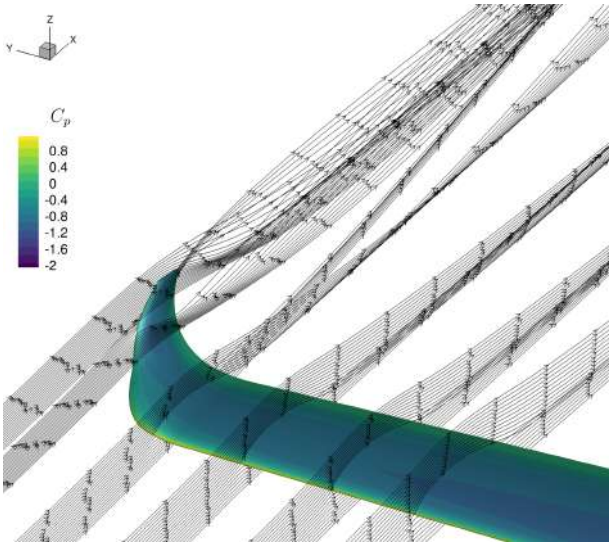


Figure 6.8: Wingtip vortex.

6.3 Influence of fuselage

The wing with fuselage was also analyzed. The trim angle of attack for the isolated wing ($\alpha = 1.9°$) was used here. Although C_L is expected to be lower, given that the isolated wing extended to the plane of symmetry, the manufacturer prescribed $C_{L_{wing}}$ already took this into account and was defined for an isolated wing. Figure 6.9 shows the pressure distribution over the wing and fuselage surfaces and some streamlines were plotted. It is possible to note that the flow over the wing is mostly undisturbed by the fuselage, but differences are verified near the wing root due to its interference, with a deflection of the streamlines in the negative y direction.

Figure 6.10 shows a comparison of the pressure coefficients on several slices across the wingspan for both the isolated wing and the wing with the fuselage. The sections are evenly distributed between $y = 0.21m$ and $y = 3.41m$ and the main differences are observed on the suction side of the first wing section, which was considerably decreased (around 30% when compared to the isolated wing) and pushed slightly backwards. On the pressure side, a smaller decrease in magnitude was also verified. From the third section onwards, differences are hardly noticeable.

The same effects can be seen from a different point of view in Figure 6.11 where the pressure distribution with the fuselage is also compared to the one with an isolated wing considering a frontal plane. The pressure coefficient under the wing is indeed similar for both cases, but perturbations can be observed on the upper side. As a result of the different pressure fields, the lift distribution along the wingspan is also changed between both cases, not only in the region where the fuselage covered the wing but also in the section closer to the root, as shown in the same figure, resulting in a total lift loss of approximately 5%. At 50% of the wingspan, the lift is just 1.5% smaller for the case with fuselage when compared to the isolated wing.

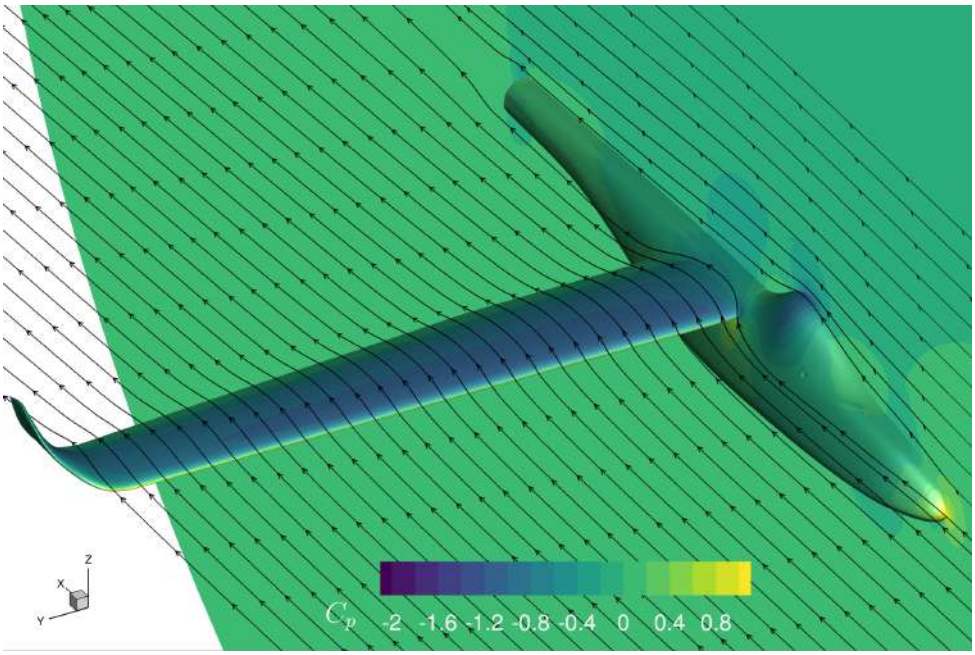


Figure 6.9: Pressure distribution over the wing and fuselage surfaces.

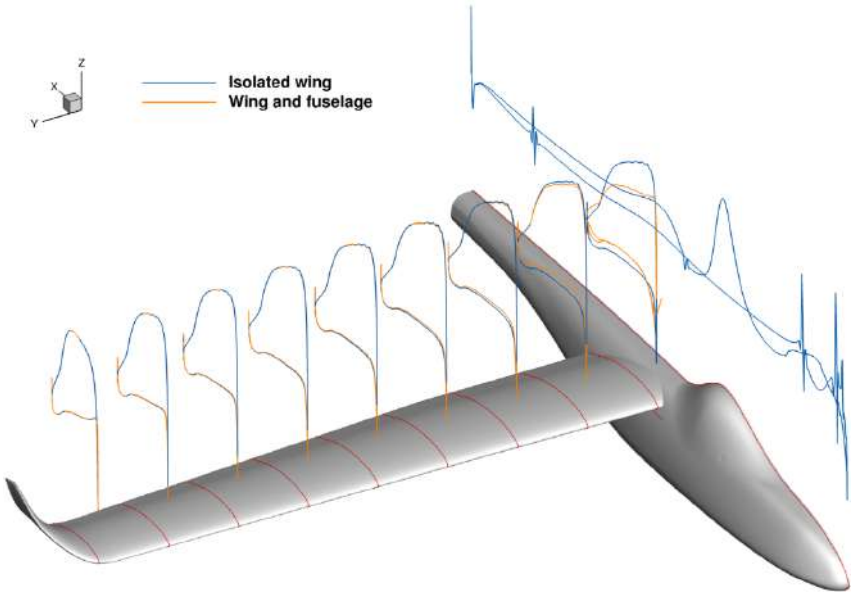


Figure 6.10: Pressure coefficient on several slices across the wingspan for both the isolated wing and wing and fuselage assembly.

Having analyzed the overall flow, secondary flows can now be analyzed in greater detail. Figure 6.12 (a) shows a detailed view of the intersection between the wing and the fuselage. It is possible to observe the deflection of the streamlines to the symmetry plane and it can be seen that it is not only influenced by the intersection region but also by the presence of the canopy in front of the wing, which deflects the flow inwards. As a result, the streamwise velocity, V_x is reduced, decreasing the pressure peak as seen before. Some swirl can also be observed on the stremtraces closer to the intersection region. Figure 6.12 (b) shows a detailed view of the flow around the fuselage, and it is possible to see a clustering of the streamlines between the wing and the fuselage after the canopy, resulting in a higher streamwise

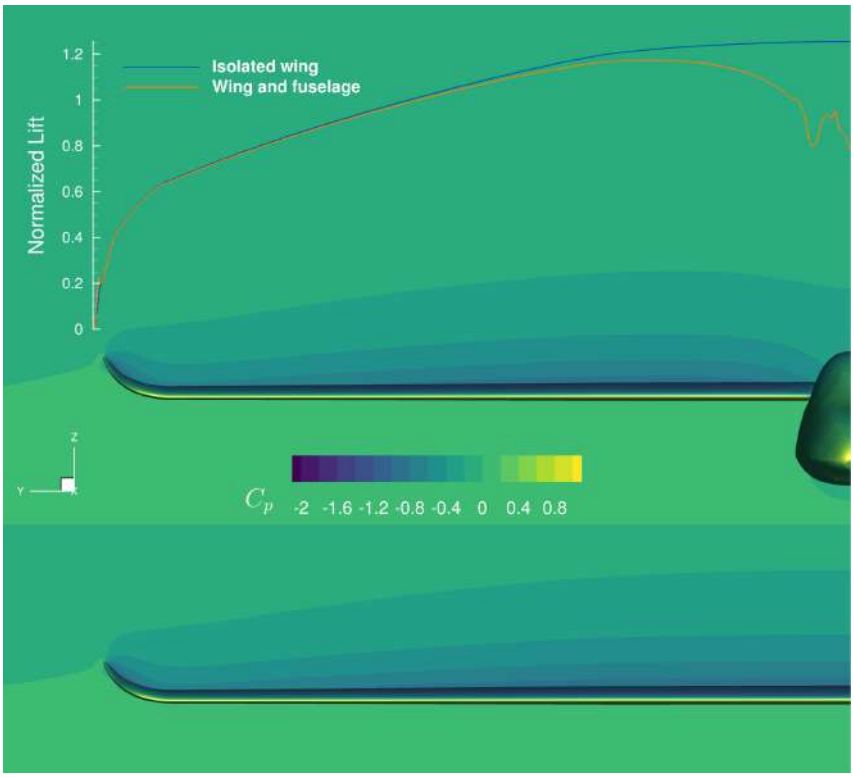
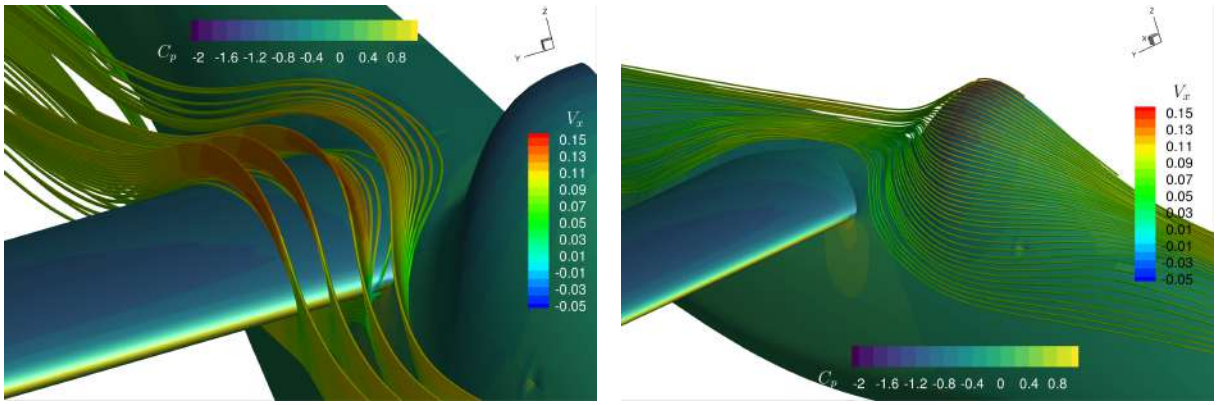


Figure 6.11: Pressure coefficient on a front plane and lift distribution comparison between the isolated wing and wing and fuselage assembly.

velocity and consequently explaining the suction peak in the upper surface of this region observed in Figure 6.10, followed by a smooth pressure recovery until the end of the fuselage, thus contributing to the creation of positive lift by the fuselage. When all these effects are considered, it is not expected that the optimized solution found with the isolated wing is the same as that when considering the full Tekever AR5 geometry near the wing-fuselage intersection. However, for the outer portion of the wing, similar results should be obtained.



(a) Streamlines near the wing-fuselage intersection.

(b) Streamlines around the canopy.

Figure 6.12: Flow details at the canopy and intersection.

Chapter 7

Wing Aerodynamic Optimization

Having analyzed the current Tekever AR5 wing in the previous section, results were obtained to benchmark the new optimized geometries. In this section, wing aerodynamic shape optimization is performed starting from 3 different initial geometries: a rectangular wing, a simplification of the current Tekever AR5 wing and the Tekever AR5 wing considering different sets of design variables.

7.1 Starting geometry

The optimization procedure requires the definition of an initial geometry. In addition to the current Tekever AR5 wing, characterized in Chapter 6, two simpler initial geometries were created. The first is a simple rectangular wing with the same projected wingspan and area as the Tekever AR5 wing, where the original custom airfoil was replaced with a symmetric NACA 4-series airfoil with the same thickness-to-chord ratio. The other, shown in Figure 5.6 (b), is a planification of the Tekever AR5 wing, with the winglet removed and subjected to three main modifications: wingspan extension to match the projected wingspan of the original wing; change in the trailing edge sweep of the outer section to keep the same projected area as the original wing; and removal of the existing twist. Figure 5.6 also shows the control points used with each geometry (blue circles) and the reference axis (shown in red). Both control boxes had dimensions $12 \times 8 \times 2$ (streamwise, spanwise and vertical direction, respectively).

The rectangular wing utilized a simple parallelepiped-shaped box, while the simplified Tekever AR5 wing employed a more complex FFD box that conforms to the surface geometry. This choice was made due to the non-symmetric nature of the airfoil, as discussed in Sub-Section 5.4.2. In all three cases, the reference axis is positioned at 25% and 50% in the streamwise and vertical directions, respectively.

A mesh with a similar refinement level as that obtained in Sub-Section 6.1.2 for the Tekever AR5 wing was used for the two simpler geometries. Given the geometric differences on the planform of all three wings, they are not expected to create the same lift at the same angle of attack. For this reason, an optimization considering only angle of attack as a design variable was performed. Given that the optimizer has only one degree of freedom and one constraint (a fixed lift coefficient), it is only able to satisfy the constraint, so no effective drag minimization is possible. This allows, however, to find the trim

angle of attack needed for both wings to maintain flight at the prescribed $C_{L_{wing}}$ and the corresponding C_D. The results obtained for a fixed $C_L = 0.8932 \pm 0.0001$ are shown in Table 7.1. It is important to note that, given the higher angles of attack required for the rectangular wing, its solution is not expected to be as accurate as the others. However, the rectangular geometry was mainly used to test and set-up the framework and so, a direct comparison with the other geometries is not the main goal.

Table 7.1: Drag coefficient and angle of attack required for all starting geometries.

Wing	α	C_D
AR5 Current	$3.73°$	Reference
AR5 Simplified	$4.03°$	$+2.20\%$
Rectangular	$10.25°$	$+11.63\%$

The parameterizations for the different variables are done as discussed in Sub-Section 5.4.3: Twist is associated with a rotation of each section around the reference axis in the spanwise direction, chord is parameterized with a scale relative to its initial value in the streamwise and normal to surface direction (with the same value for both) in relation to its initial value, sweep is parameterized with a displacement in the streamwise direction of every reference axis point followed by a rotation of each control point around the normal to surface direction, dihedral is parameterized with a displacement in the normal to surface direction followed by a rotation around the streamwise direction and airfoil shape is parameterized with displacements of the local FFD points in the vertical direction.

In general, a global design variable will have as much degrees of freedom as there are FFD sections in the spanwise direction. However, they can be condensed and represented by a smaller number of design variables by defining their variations through functions. For instance, the twist distribution may be forced to be linear and, in that case, only two design variables are required: The value at the origin and the slope. Root twist is kept at $0°$ as it can be effectively defined by the angle of attack, so it only has seven design variables. Root vertical and horizontal displacement, for dihedral and sweep, respectively, are also not allowed, as it is not meaningful for an isolated wing.

Most limits were defined in such a way that the optimizer would have enough freedom. However, in some cases, like the airfoil shape, the chord lower and the dihedral upper bounds tighter limits had to be imposed to avoid excessive deformations resulting in low quality or even unusable volume grids after volume mesh perturbation. The thickness constraints were set to span the whole wing and enforced at four spanwise points and six streamwise points for the rectangular and simplified initial geometries and at 35 spanwise locations for the Tekever AR5 wing geometry. They are important mostly for the airfoil shape optimization and, ideally, the wingbox for the required wing structure would be used to set them. As this work deals with the uncoupled aerodynamic shape optimization, it was arbitrated that, at each point, the thickness could not be less than 90% of the initial value to give some freedom to the optimizer. As the project will evolve into aerostructural optimization, the required thickness will be set automatically and in a coupled way by the optimizer in every iteration, eliminating the need to arbitrate this value in the future.

Regarding convergence criteria, a successful optimization is finished if either the convergence accu-

racy is smaller than $1 \cdot 10^{-6}$, defined by the Karush–Kuhn–Tucker (KKT) conditions (first derivative tests), or the number of iterations reaches 500. It was verified that, on successful optimizations, the desired convergence accuracy criterion was met well before the maximum number of iterations.

7.2 Results for the rectangular wing

The results obtained departing from the rectangular wing and considering different individual design variable optimization subproblems are presented in Table 7.2. and are briefly analyzed next.

Table 7.2: Optimization results for the rectangular wing as starting geometry.

Case		α	C_D	Iterations
Starting geometry		$10.25°$	Reference	-
Twist	Linear variation	$12.89°$	-2.60%	10
	3 FFD sections	$11.57°$	-3.06%	5
	7 FFD sections	$11.86°$	-3.21%	19
Chord	Linear variation	$10.05°$	-4.15%	14
	4 FFD sections	$10.02°$	-5.15%	20
	8 FFD sections	$10.03°$	-5.21%	41
Twist (7 sections) + Chord (8 sections) + Sweep		$10.03°$	-5.18%	10
Airfoil	Constant (24 control points)	$6.46°$	-6.17%	28
	Variable ($5 \times 12 \times 2$ control points)	$5.82°$	-8.55%	35

The table also shows the number of iterations that each optimization took. In general, it can be seen that for the same parameterization, an increase in the number of degrees of freedom increases the number of iterations required. Chord distribution always takes more iterations than the corresponding twist distribution.

7.2.1 Twist optimization

The drag coefficient reduces with the increased degrees of freedom provided to the optimizer, however, differences between three and seven FFD sections are very small. Furthermore, less FFD sections usually led to smoother geometries, which may be desirable. The linear twist variation also poses itself as an interesting option, given that the drag reduction difference between it and the best option is 0.61% and it may be easier to manufacture.

Figure 7.1 compares the lift distribution, normalized with the total lift produced, of the obtained solutions. It can be seen that the distributions tend to the elliptical one as the number of degrees of freedom increase, but do not match it, having a slightly lower variation of the twist angle across the wingspan. A reason for this is the fact that separation started being observed near the trailing edge for such high incidence angles, as shown by Figure 7.2 where the signal of the skin friction coefficient in the x direction is plotted and it is possible to note that it is negative at the trailing edge near the root. This is an effect that could not have been predicted with optimization considering inviscid flow, to which the elliptical lift distribution is optimal. Figure 7.3 shows the resultant lift distribution of a twist optimization using

Euler equations, and it can be seen that it is much closer to the elliptical distribution than what could be achieved considering RANS equations, thus illustrating the importance of using high-fidelity RANS models for optimization.

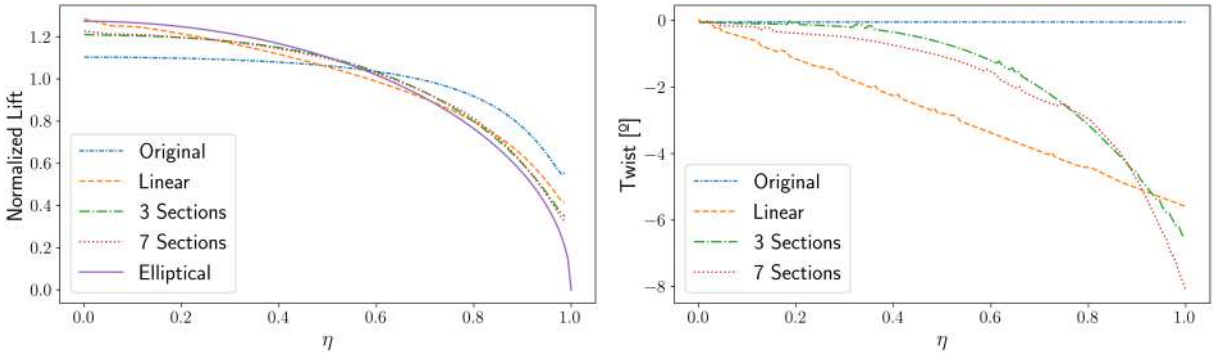


Figure 7.1: Lift distribution for the optimized twist distributions starting from the rectangular wing.

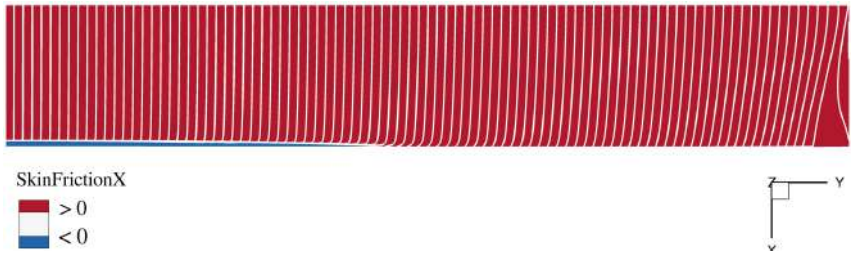


Figure 7.2: Friction coefficient along the x direction on the geometry with optimized twist distribution (top view).

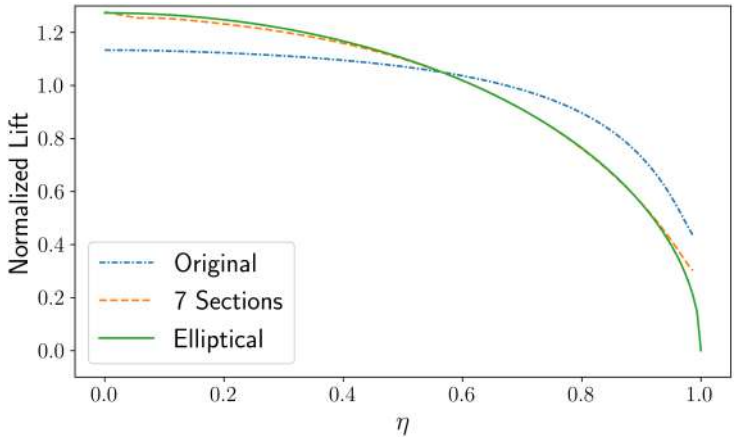


Figure 7.3: Lift distribution for the optimized twist distribution starting from the rectangular wing using inviscid models.

It is also possible to note that the optimized solutions for three and seven FFD sections are in fact very close, explaining the small differences in the drag coefficient observed. It should be noted that for the case with three FFD sections, the optimizer required only 18 function evaluations, whilst for the seven FFD sections case it required 79, representing a very significant difference in computational cost.

Compared to other variables, it is also possible to observe that the twist distribution let the angle of

attack be higher than the initial geometry. This is expected, given that all the obtained twist distributions applied a washout, reducing the incidence monotonically across the whole wingspan, thus reducing the angle of attack of those sections. The applied washout, besides its advantages in drag reduction, is also a desirable safety feature to ensure that the root section stalls before the outer section, were control surfaces are located, avoiding the loss of aileron authority.

7.2.2 Chord optimization

A similar pattern to the twist distribution was observed regarding the number of DoF, with the results improving with their increase but with the four and eight FFD section cases yielding very similar results. Here, the difference between the linear variation and the one with eight FFD sections is 1.06%.

The optimized wings feature lower drag coefficients than those obtained with twist distribution. One factor that may contribute to this is the additional degree of freedom that chord has at the root, despite having a constraint on the projected area to allow a fair comparison between different geometries. However, the main reason is that, while twist changed the lift distribution by changing the effective angle of attack at a section, increasing the possibility of flow separation near the root, chord distribution can change the lift distribution without suffering from the same issue. The linear chord variation is effectively equivalent to a trapezoidal wing with taper ratio $\lambda = 0.358$.

Lift and chord distributions for the three cases are shown in Figure 7.4, where it can be seen that the distribution is indeed closer to elliptical than that obtained with twist optimization. Regarding the four and eight FFD sections cases, similarly to the twist case, the lift distributions are very close to each other.

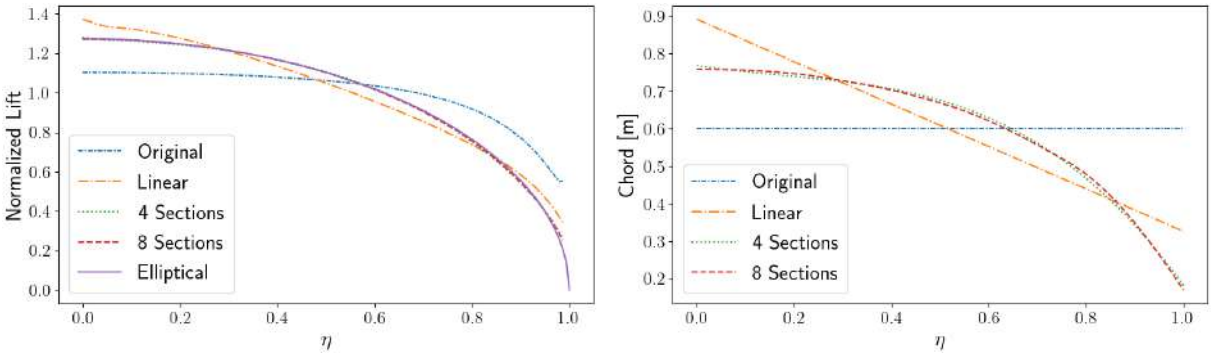


Figure 7.4: Lift distribution for the optimized chord distribution starting from the rectangular wing.

Figure 7.5 compares the suction side pressure distribution of the optimized wing with eight FFD sections (left) with the original one (right). The elliptical shape and lift distributions obtained are evident, with the most visible differences near the wingtip.

Chord optimization posed some initial challenges, as excessive expansions near the root would result in an unusable volume mesh, requiring the upper bound for said variable to be decreased. It was also verified that the optimizer was converging to non-optimal solutions with the initial conditions, so a monotonic constraint was applied to the chord distribution. Furthermore, it was also possible with this optimization to observe the importance of having an adequately refined grid. When using coarser

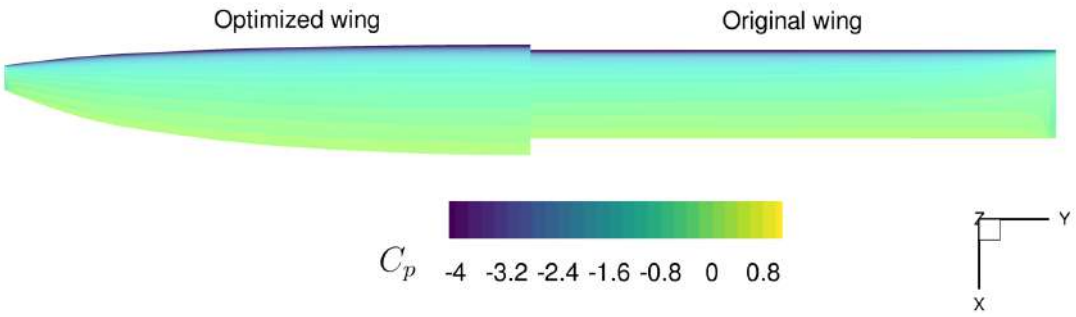


Figure 7.5: Pressure distribution in the suction side of the rectangular wing and wing with optimized chord.

grids, the flow solver would not capture the wingtip effects accurately and the optimizer would expand the chord at the tip instead of decreasing it, resulting in a lift distribution far from the ideal one.

7.2.3 Sweep optimization

Regarding sweep, the optimizer converged back successfully to a solution very close to the initial one, with zero degree sweep, although with a slightly higher drag coefficient. Given that sweep mainly intends to mitigate transonic effects and the Mach number is very low ($Ma = 0.082$) in our case, this result is to be expected and the optimization confirms that it is not beneficial to have a swept-back wing for the intended flight condition.

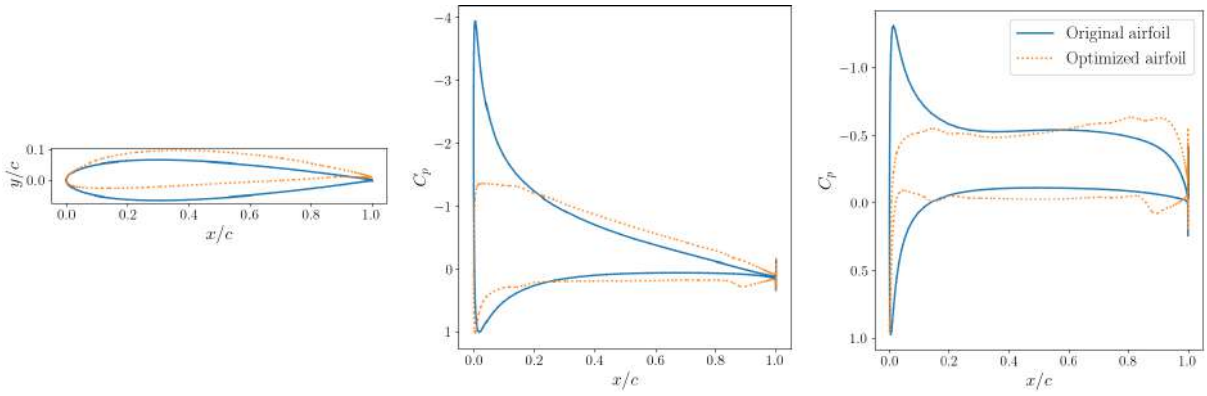
7.2.4 Twist + chord + sweep optimization

A combined case with all global design variables was also considered and a very similar drag coefficient to the chord optimization case with eight FFD sections was obtained, despite the fact that the optimized geometry had both a chord distribution and a twist distribution, with a negative twist of $1.89°$ at the tip, where the chord was scaled down to 28.8% of its original size (this figure was 23.6% for the chord optimization) and close to zero in the remaining wing, avoiding the separation observed before. The sweep was kept at zero again and the obtained lift distribution was very close to elliptical.

7.2.5 Airfoil optimization

The airfoil optimization considering a constant airfoil on the three-dimensional wing, allowing the capture of induced drag effects, yielded the second best result for drag reduction. Starting from a symmetric NACA 4-series airfoil, the optimizer took 113 function calls and 28 gradient calls to converge to a cambered airfoil, where thickness was mostly taken to the minimum value allowed by the constraint. The obtained airfoil profile is outlined in orange in Figure 7.6 (a). With this, the required angle of attack for the prescribed wing lift coefficient was reduced by $3.8°$. Another significant change was the distribution of pressure and viscous drag, with the viscous drag only representing 10.5% of the total drag of the optimized geometry, compared to 21.5% on the original wing.

Figures 7.6 (b) and 7.6 (c) compare the obtained pressure distribution for both airfoils at a section at the middle of the wing and near the tip, respectively. Considering the middle section, it can be seen that the pressure peak on the suction side was greatly decreased on the optimized airfoil, with a smoother recovery along the chord.



(a) Airfoil shape.

(b) Pressure distribution at 50% wingspan.

(c) Pressure distribution at 99% wingspan.

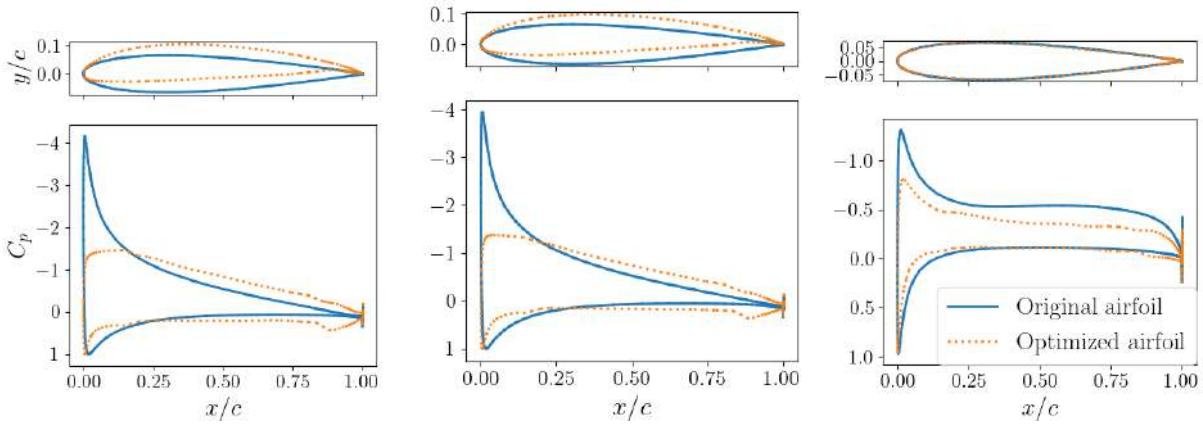
Figure 7.6: Pressure coefficient for the original and optimized constant airfoil starting from the rectangular wing.

The highest drag reduction was obtained for a variable airfoil across the wing. For this case, a different FFD box was used with only five sections in the wingspan direction to reduce the number of design variables and thus, the computational time needed. Figure 7.7 shows the airfoils obtained for three sections: near the root, at the middle of the wing and near the tip. As expected, the sections near the root and at the middle of the wing are similar, given that a symmetry boundary condition was applied at the root and wingtip effects are not felt yet at those regions, although it has a slightly higher camber near the root. The most notable difference occurs near the wing tip, where the airfoil closely matches the original one, with barely any camber. Although the airfoil profile is similar to the root one, the pressure distribution is different, as a result of the considerably smaller angle of attack required for trim. With this airfoil distribution, the optimizer was able to effectively control the lift distribution across the wingspan and approach it to elliptical without the use of global planform design variables. Despite representing a considerable improvement in aerodynamic performance, the use of cambered airfoils near the root and symmetric airfoils near the tip may result in poor stall characteristics of the wing so attention must be paid to the airfoil used in the control surfaces region.

To reach the optimized wing, 128 function calls and 35 gradient calls were needed, representing an increase of only 13% and 25% to the constant airfoil case. It should be noted, however, that the gradients naturally took longer to compute in the variable airfoil case, given the higher number of design variables.

7.3 Results for the simplified Tekever AR5 wing

The summarized results of the optimization using the simplified Tekever AR5 wing as starting geometry geometry are found in Table 7.3.



(a) 1% wingspan. (b) 50% wingspan. (c) 99% wingspan.

Figure 7.7: Pressure coefficient for the original and optimized variable airfoil starting from the rectangular geometry.

Table 7.3: Optimization results for the simplified Tekever AR5 wing as starting geometry.

Case		α	C_D	Iterations
Starting geometry		$4.03°$	Reference	-
Twist	Linear 2 sections	$3.96°$	-0.39%	10
	7 FFD sections	$4.56°$	-0.80%	48
Chord (Linear 2 sections)		$4.07°$	-0.11%	33
Twist (7 sections) + Chord (2 linear sections)		$4.09°$	-1.16%	55
Airfoil ($8 \times 12 \times 2$ Control points)		$4.59°$	-2.54%	39

The behavior regarding number of iterations is similar to that observed for the rectangular airfoil. Considering the only common parameterization, twist with 7 FFD sections, a significant increase in the number of iterations required is verified. However, for airfoil optimization, despite the fact that the simplified Tekever AR5 has more control points, the increase in the number of iterations is not as significant as expected. Linear chord with two sections, despite only having two control points, took a lot of iterations. This behavior is also verified for the rectangular wing and a possible explanation may be the need to satisfy the projected area constraint, which was not active in twist and airfoil optimizations.

7.3.1 Twist optimization

Two cases were defined: a linear variation and a variation with full freedom on the seven FFD sections. Given the nature of the original geometry that can be defined by two separate sections, the linear case was defined in two sections. To define them, the optimizer is able to change the twist angle at the junction between both sections and at the wingtip. From these values, two separate linear distributions are defined.

It is clear that the initial geometry is much closer to the elliptical lift distribution than the rectangular wing, so the drag reductions are much smaller. In both scenarios, whether we consider a linear twist distribution defined in two linear sections or seven FFD sections, the optimizer successfully approached an elliptical lift distribution. Notably, the case involving seven FFD sections resulted in nearly twice

the drag reduction compared to the linear twist distribution defined in two sections. This significant improvement can be observed in Figure 7.8, where the lift distribution of the seven FFD sections case is closer to elliptical as a result of the increase in degrees of freedom.

7.3.2 Chord optimization

The parameterization for the chord followed a similar approach to that of the twist. However, it was adjusted to ensure a constant taper at the inner section by enforcing the same scaling factor applied to the first two FFD sections, representing the first degree of freedom of the optimizer. The second determined the scaling factor at the wingtip and a linear distribution was then defined from the junction between sections to the wingtip. Due to the constraint imposed by the projected area, the optimizer only had one effective degree of freedom, which determined the taper ratio, λ, for the outer section of the wing. This case was the only configuration tested to maintain the original wing design philosophy. The optimizer produced a different geometry, with the final scale factors being 1.09 and 0.76, but the resulting drag reduction for this particular case was minimal. As Figure 7.8 shows, the optimized geometry approached the elliptical one near the root and near the tip, but moved away from it in the central section. This is a result of the 9% increase in chord in the inner region, resulting in more lift generated in that section and a decrease at the tip by reducing its chord.

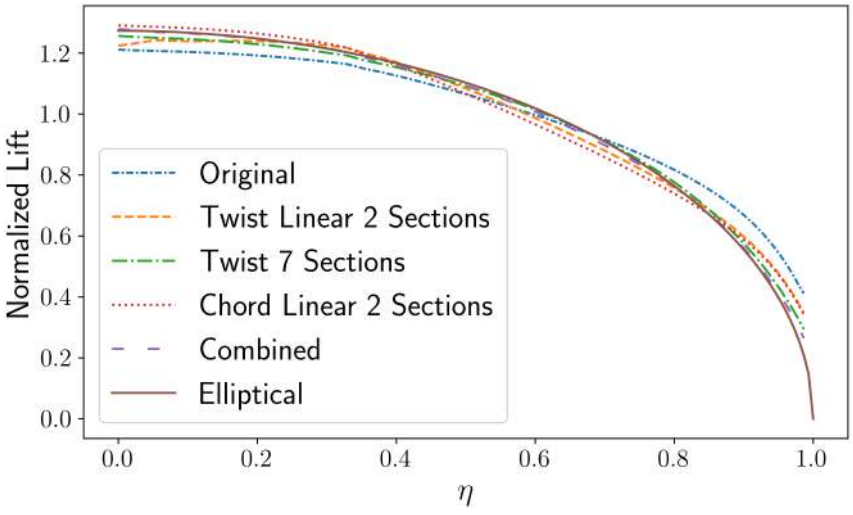


Figure 7.8: Lift distribution for the different optimized wings starting from the simplified Tekever AR5 wing.

7.3.3 Twist + chord optimization

A combined optimization case with full freedom to the seven twist design variables and the taper of the outer section was also considered and, unlike the rectangular wing, the results were better than any of the singular cases, obtaining a drag reduction of 1.16%. In Figure 7.9, it is possible to see the twist and chord distribution obtained for this and the other parameterizations tested, starting from the simplified Tekever AR5 wing geometry. The obtained scale factors for chord distribution were 1.13 and 0.67, and

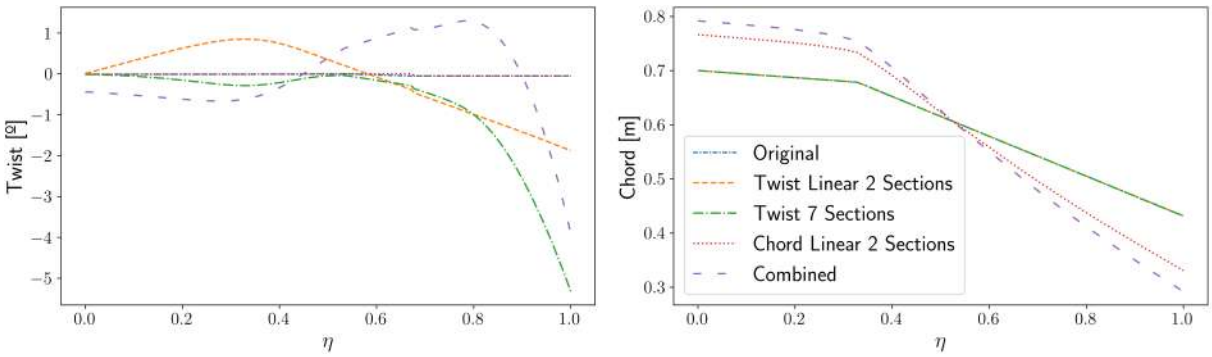


Figure 7.9: Chord and twist distribution for the different optimized wings starting from the simplified Tekever AR5 wing.

a twist distribution was also applied, with its lowest value, at the wingtip, being $-4.14°$, giving enough freedom for the wing to reach a near elliptical lift distribution, as Figure 7.8 shows. Figure 7.10 shows a comparison of the pressure distribution on the suction side between the original and the optimized wings.

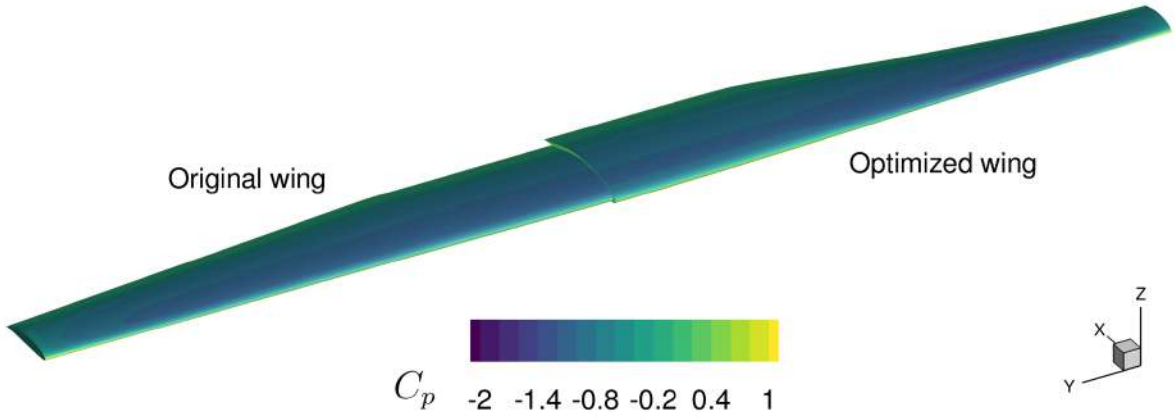
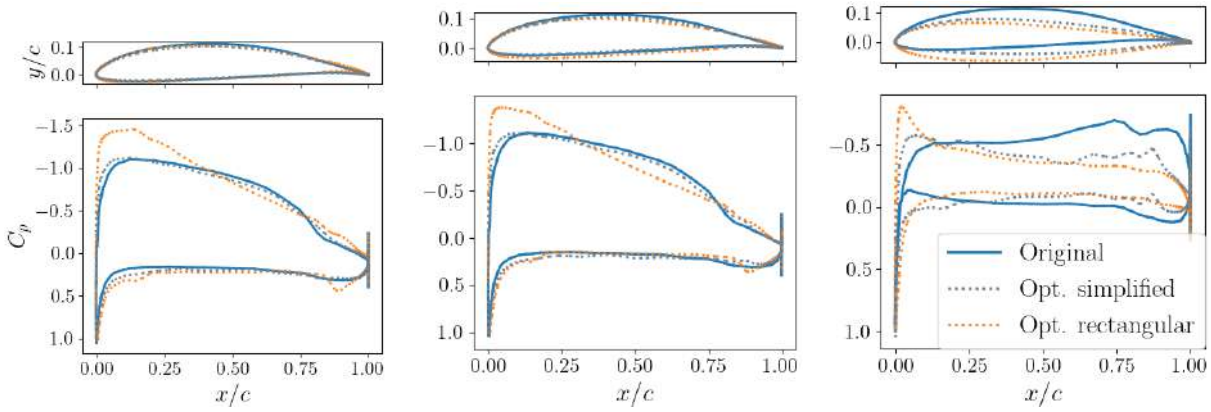


Figure 7.10: Pressure distribution for the original wing and the optimized wing considering twist and chord distributions starting from the simplified Tekever AR5 wing.

7.3.4 Airfoil optimization

Finally, a variation in airfoil shape along the wingspan was also optimized. Similar to the rectangular wing case, this optimization yielded the greatest reduction. However, the reduction achieved was significantly smaller than that of the rectangular wing, which was expected considering that the starting profile for this case was already optimized for the specified flight condition.

Figure 7.11 provides a comparison between the original airfoil and the optimized profiles at three different sections along the wingspan. The comparison reveals that the modifications in the initial sections mainly involved a decrease in thickness and a slight reduction in camber. At the wing tip, the optimizer converged to a symmetric airfoil, resembling the case of the rectangular wing with a symmetric airfoil



(a) 1% wingspan. (b) 50% wingspan. (c) 99% wingspan.

Figure 7.11: Pressure coefficient for the original and optimized variable airfoil for the simplified Tekever AR5 wing as a starting geometry.

geometry as a starting geometry, in order to minimize induced drag.

Furthermore, when comparing the obtained airfoil from this case directly with the rectangular one, it can be seen that, near the root, a very similar airfoil was obtained (also similar to the Tekever AR5 custom airfoil), although the pressure coefficient is not the same as a result of an higher angle of attack on the rectangular wing, indicating that this might be in fact the ideal airfoil for the specified flight condition. When analyzing the middle section, it is possible to note that the optimized airfoil for the simplified wing is closer to the original wing than the one obtained for the rectangular wing. When moving to the sections near the tip, it is possible to note that the airfoil obtained here as slightly more camber than that obtained for the rectangular wing, given that in this case the starting geometry was closer to the elliptical lift distribution, thus airfoil changes to approach that did not have to be so significant.

7.4 Results for the Tekever AR5 wing

Finally, the Tekever AR5 wing was optimized, with the obtained results presented in Table 7.4. Due to the complex nature of the initial geometry, the results interpretation for this case is not as straightforward as the previous ones. Here, a much more refined FFD box in the spanwise direction was required in order to correctly capture the winglet shape.

As for the number of iterations, the pattern between chord and lift distribution is verified once again, with more iterations needed when compared to the simplified Tekever AR5 geometry, which is to be expected not only due to the increased number of spanwise sections but also the increased complexity of the geometry. Dihedral converged in a surprisingly small number of iterations.

7.4.1 Twist optimization

As seen in Sub-Section 7.3, a linear twist distribution defined in two sections did not provide a significant improvement, so only the case with full freedom at the 28 sections was considered for the Tekever AR5 wing as a starting geometry. Even so, the obtained C_D reduction was just 0.42%. A concern

Table 7.4: Optimization results for the Tekever AR5 wing as starting geometry

Case	α	C_D	Iterations
Starting geometry	$1.92°$	Reference	-
Twist (28 FFD sections)	$0.24°$	-0.42%	49
Chord (28 FFD sections)	$1.98°$	-1.02%	89
Dihedral (28 FFD sections)	$1.88°$	-1.78%	11
Airfoil (28 x 12 x 2 control points)	$2.29°$	-3.90%	58 [a]
Twist (28 FFD sections) + Chord (22 FFD sections) + Dihedral (28 FFD sections) + Sweep (28 FFD sections) + Airfoil ($28 \times 12 \times 2$ control points)	$0.79°$	-4.52%	22 [b]

[a] Optimization stopped due to negative volumes. [b] Optimization stalled.

that arose during the set-up of the twist optimization was the fact that the Tekever AR5 wing already had a twist distribution, which could generate deformations on the airfoil when rotating the sections if the rotation axis of the FFD box was not the same as that used to impose the original twist distribution. However, no such effects were verified, as shown by Figure 7.12, on a section where the original wing had a twist angle of $0.72°$ and the optimized one had an angle of $2.92°$.

(a) Original and optimized airfoil.

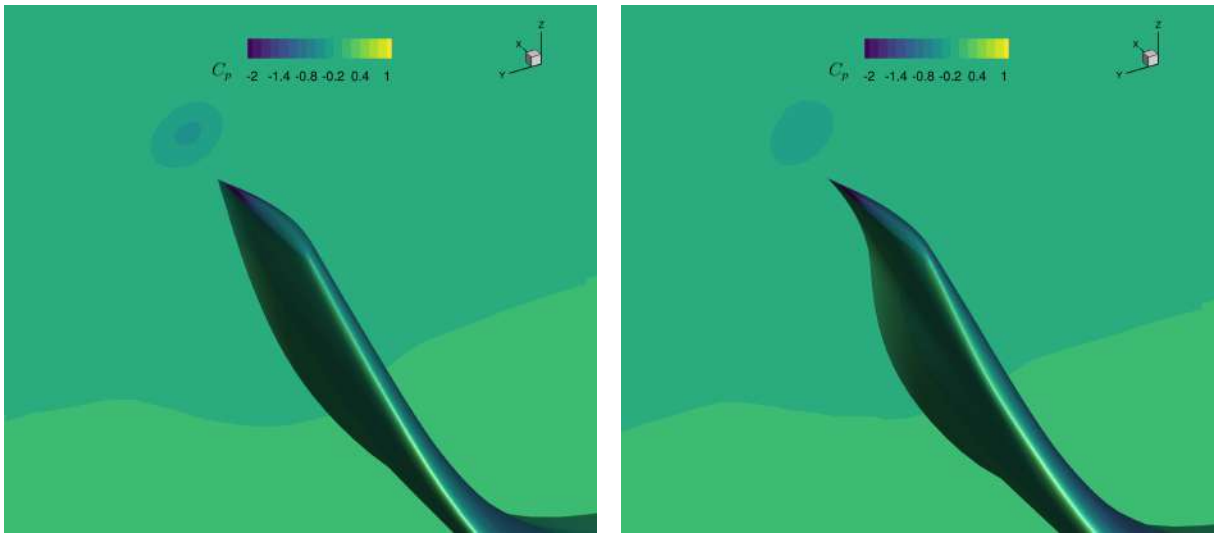
(b) Airfoil rotated to match local angle of attack.

Figure 7.12: Comparison between airfoil sections at 80% wingspan.

The obtained twist distribution ranged from $2.5°$ to $1°$ at the start of the winglet (relative to the initial distribution), which increased the lift generation at the middle and outboard sections. At the winglet, the larger changes were at the last and second to last sections, where a twist of $15°$ and $-12°$ respectively was applied, resulting in the tip vortex dissipating slightly faster, as shown in Figure 7.13 on a vertical plane 7 cm behind the wingtip trailing edge. It should be noted that one of the winglet functions besides reducing induced drag is flow laminarization across the wing by reducing cross-flow, which is greatly impacted by winglet orientation with the flow, given by the twist angle in this case. However, the used flow model is not accurate for that application, so the full potential of the winglet is not explored.

7.4.2 Chord optimization

A chord optimization with complete freedom on all the 28 FFD sections was performed, reducing the drag coefficient by 1.02%, mainly achieved by a close approximation to the elliptical lift distribution, as shown in Figure 7.14. However, Figure 7.15 shows that this was obtained with a non-smooth chord variation along the wingspan with oscillations around the ideal elliptic shape, which is not desirable from a manufacturability point of view. Notably, the chord was significantly reduced at the winglet region, given that the applied constraint was to the projected area and changing chord at this region does not affect the projected area significantly, creating even more severe oscillations. To solve this, using a



(a) Original Tekever AR5 wing.

(b) Optimized Tekever AR5 wing.

Figure 7.13: Pressure distribution for the Tekever AR5 and the optimized wing for twist distribution on a vertical plane slightly behind the wingtip trailing edge.

tighter convergence for the optimizer could be an option, hopefully leading to a perfect elliptical chord distribution. Another alternative would be to reduce the number of wingspan sections, as this tends to lead to smoother distributions, as observed with the rectangular wing.

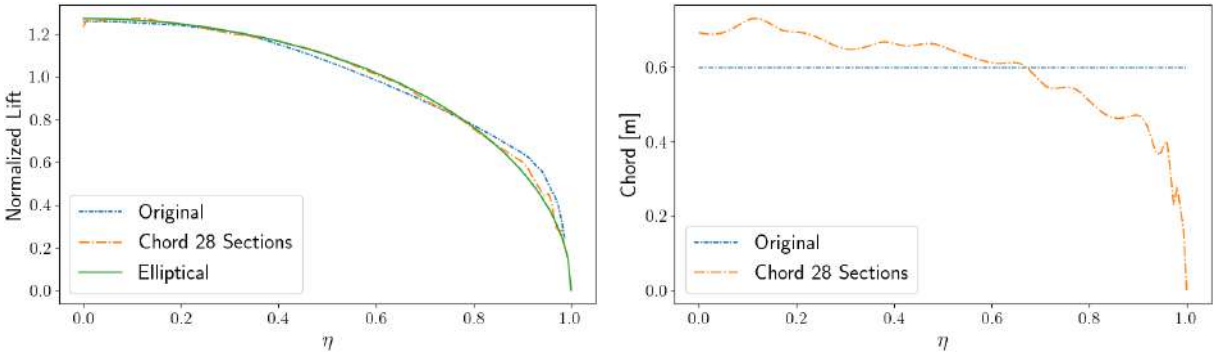


Figure 7.14: Lift distribution for the Tekever AR5 wing and optimized wing for chord distribution.

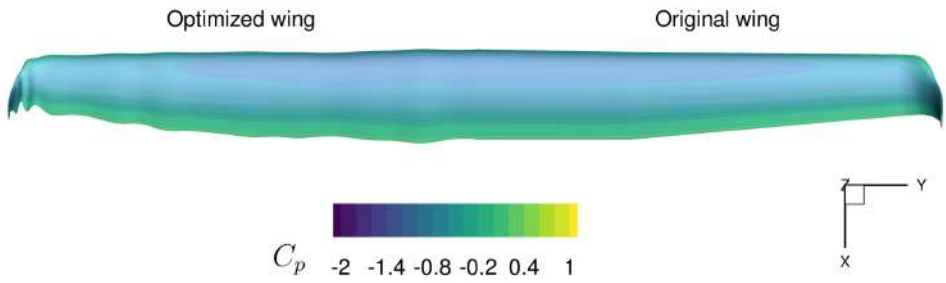


Figure 7.15: Pressure distribution in the suction side of the tekever AR5 wing and wing with optimized chord.

7.4.3 Dihedral optimization

In this case, the main differences are on the last two sections of the FFD box, where the optimizer increased the vertical displacement to the upper bound value of 10 cm. Other tests were performed where the upper bound was greater than that, up to 50 cm and it was verified that the optimizer tended to always maximize this value at the wingtip. However, resulting deformations often lead to unusable volume grids, preventing the optimization from finishing, forcing the imposition of a more modest 10 cm limit. The observed behavior is to be expected considering that increasing the height of the winglet tip effectively increases the total wingspan and reduces the magnitude of the pressure difference losses at the tip, while keeping the projected area constant, reducing the induced drag and making the wing more efficient. In Figure 7.17 (c), it is possible to see how the winglet keeps the pressure differential on a greater wingspan.

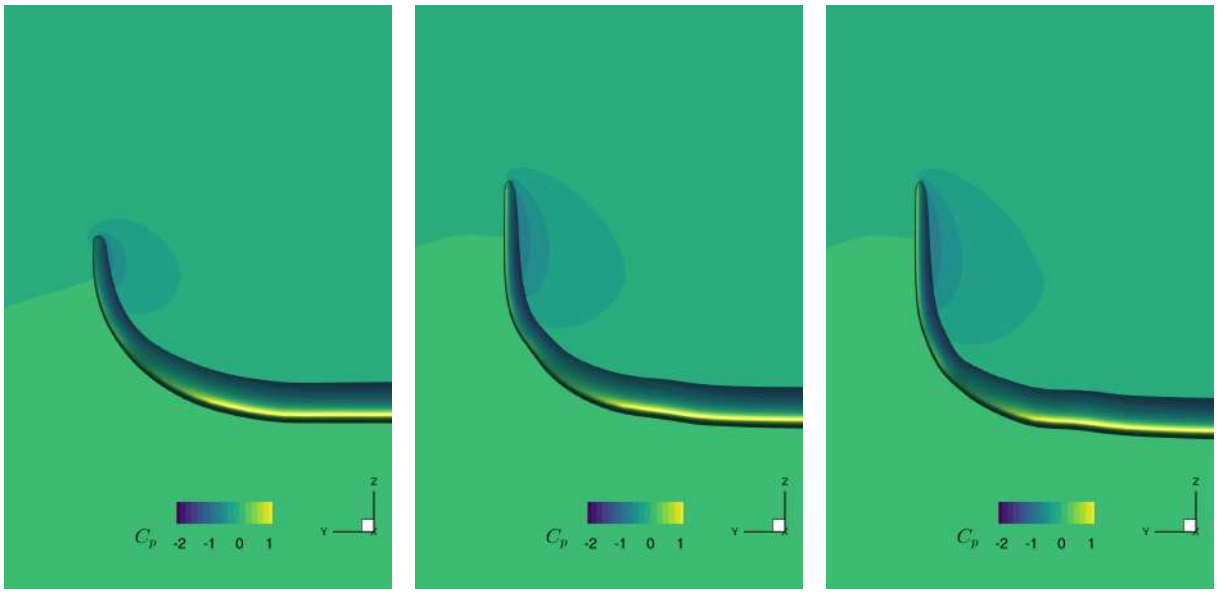
Figure 7.16: Comparison between the Tekever AR5 geometry (translucent blue) and the resulting geometry from dihedral optimization (orange).

In sections before the winglet, a negative vertical displacement was applied with values up to -5 cm, intended to increase even further the effects of the winglet height considering the upper bound of 10 cm. This might pose problems when integrating the wing structural components. Intermediate optimization results were found, as that shown in Figure 7.17 (b), where this downward deflection was less pronounced (less than 2 cm), reaching a drag reduction from the original wing of 1.64%, which is just 0.14% more than the final geometry.

7.4.4 Airfoil optimization

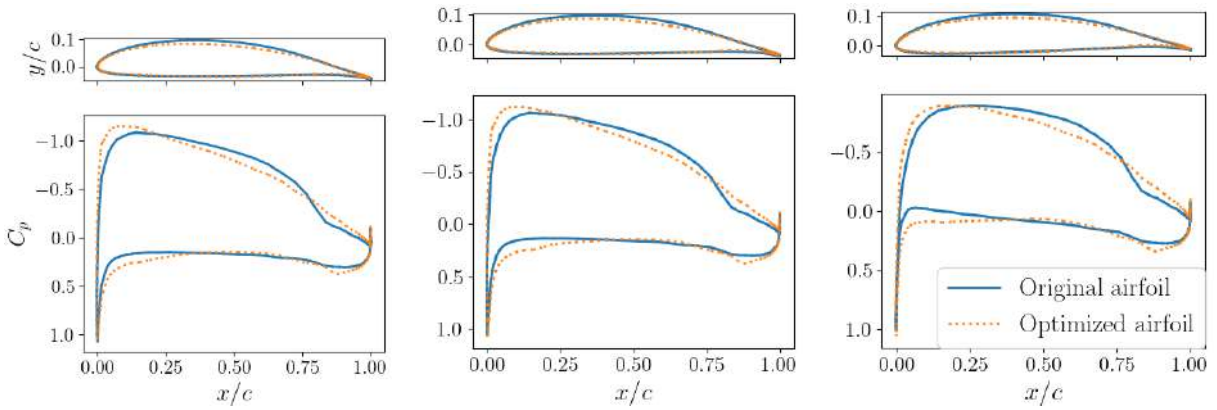
A variable airfoil optimization was also performed. Although the optimization faced problems with unusable volume grids after deformation, preventing it from finishing and reaching an optimum, a solution was obtained with significant drag reduction of 3.9% while verifying all constraints. Like with the optimization starting from the simplified wing, the main differences also resulted from a reduction of the airfoil thickness mostly to the minimum allowed. A slight change in the suction side curvature, near the trailing edge, also results in a significant change in the pressure distribution, with a much smoother pressure drop in that region, as shown in Figure 7.18.

Another interesting effect observed with this case was the effect of the constraint definition on the optimizer. As explained in Section 5.4.4, constraints are only enforced at discrete locations. As such, if the number of constraints in a certain direction is insufficient, as it is the case in Figure 7.19, the optimizer can exploit this and reduce the thickness where constraints are not enforced, with this effect being particularly notable when there is an FFD section between two sections of spanwise constraints, as the



(a) Initial geometry. (b) Intermediate geometry. (c) Final optimized geometry.

Figure 7.17: Pressure distribution on a frontal plane for the geometry optimized with dihedral starting from the Tekever AR5 wing.

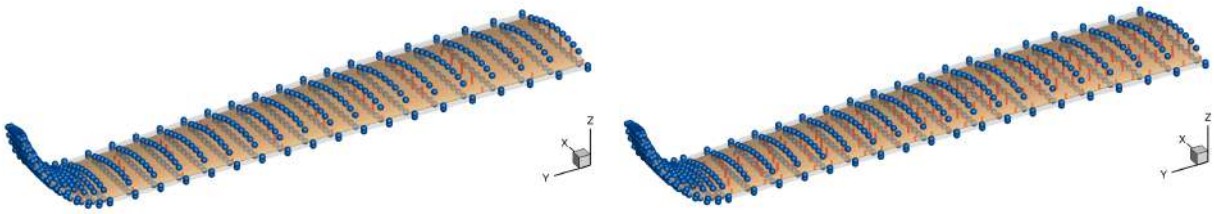


(a) 1% wingspan. (b) 45% wingspan. (c) 85% wingspan.

Figure 7.18: Pressure coefficient for the original and the optimized wing with variable airfoil for the Tekever AR5 wing as a starting geometry.

optimizer will have complete freedom to move that section. Figure 7.20 shows an intermediate result of an optimization with only 12 spanwise constraints, where the described effect is visible. Furthermore, the low density of constraints at the winglet, as they were distributed uniformly along the wingspan also allowed the optimizer to tend to a geometry with barely any thickness at that region. Although the geometry shown resulted in a drag reduction of 4.32%, it is not desirable from a structural and manufacturability standpoint. For this reason, the number, and consequently density, of constraints in the spanwise direction was increased to 35, eliminating the described problems.

Figure 7.21 shows the wing with the optimized airfoil, in orange, compared to the original wing, in translucent blue. Significant differences between the optimized and original wings were observed at the winglet region which was deflected outwards in the original Tekever AR5 wing due to the constant airfoil camber, and the optimizer deflected inwards, effectively reducing the airfoil camber at this region



(a) 12 spanwise constraints. (b) 35 spanwise constraints.

Figure 7.19: Different constraint definitions for the Tekever AR5 wing.

Figure 7.20: Intermediate solution obtained with 12 spanwise constraints.

as Figure 7.22 shows. A similar effect was observed with the two other starting geometries, where a symmetric airfoil was obtained at this region. However, in this case, the airfoil is not aligned with the lift force and so its contribution to the lift production is smaller, allowing more drastic changes.

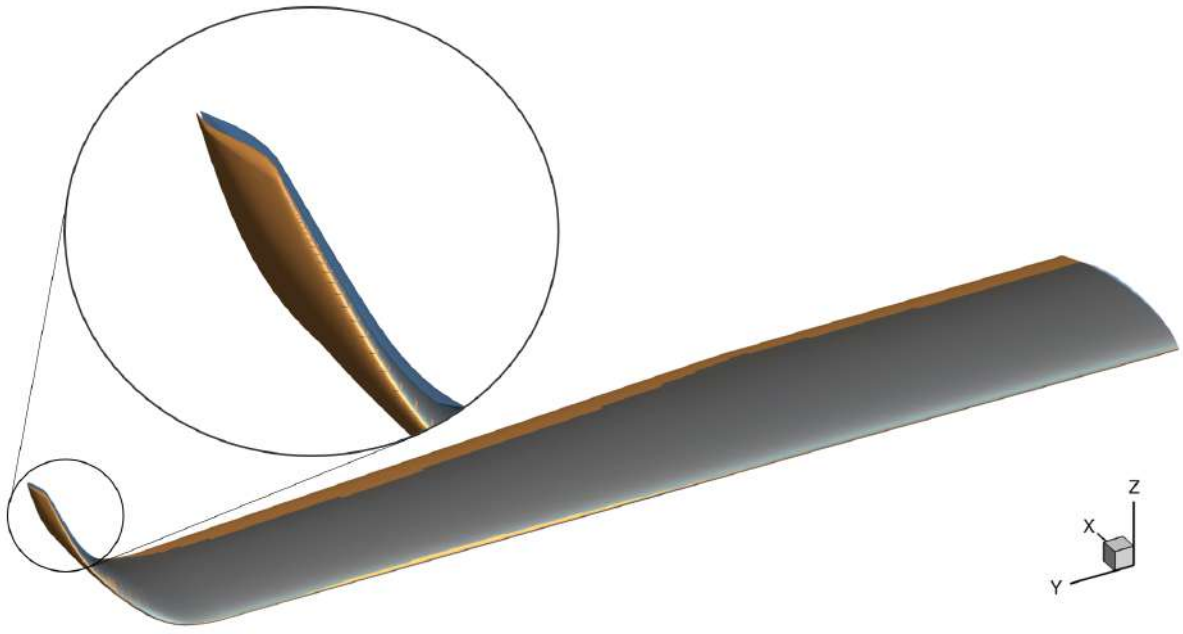
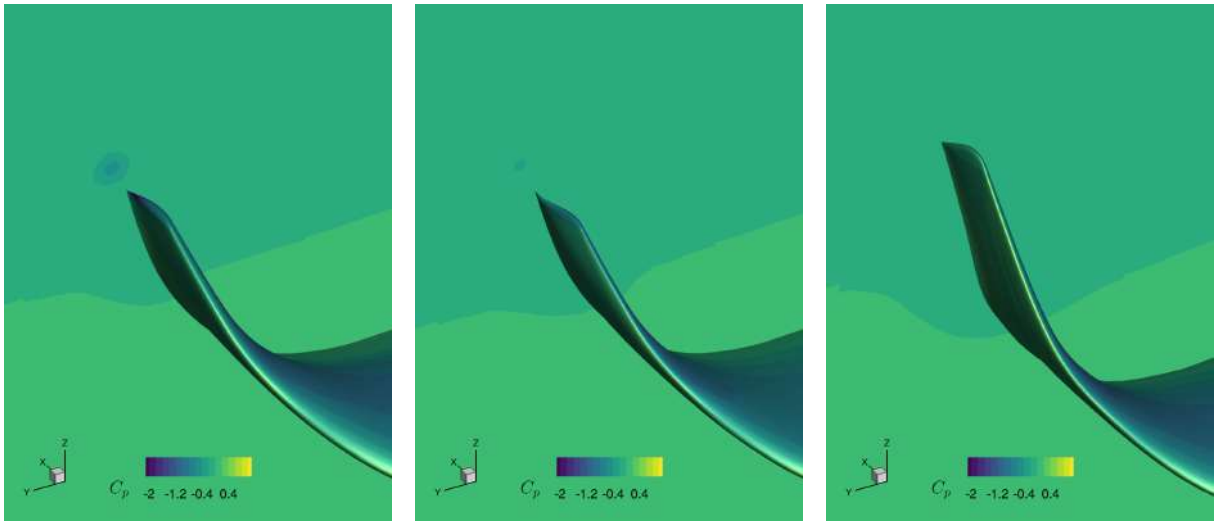


Figure 7.21: Comparison between the Tekever AR5 wing (translucent blue) and the resulting wing from airfoil optimization (orange).

As a result of this, a similar behavior as that obtained with twist distribution was observed, with the wingtip vortex greatly decreased, as shown by Figure 7.23, reducing the induced drag.

Figure 7.22: Optimized airfoil at the wingtip starting from the Tekever AR5 wing.



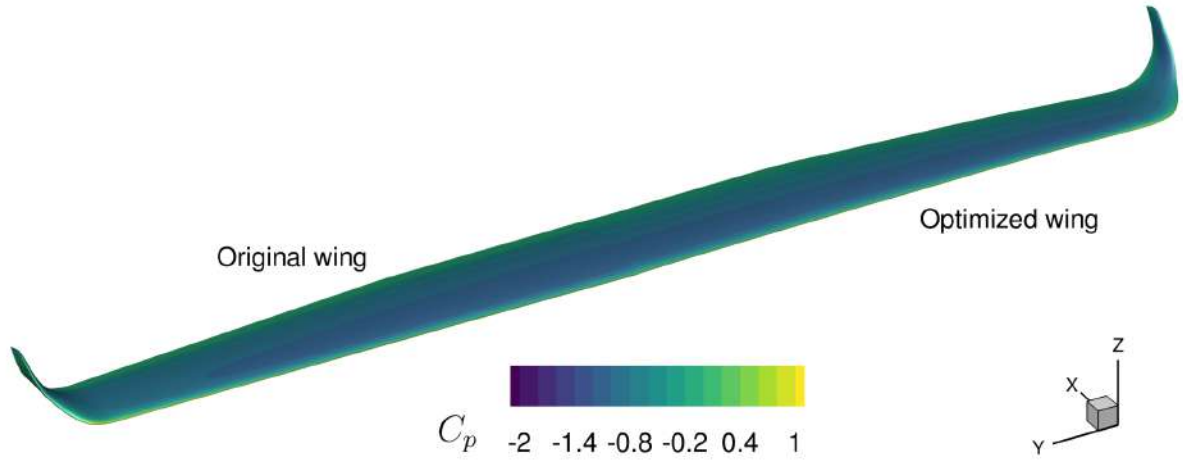
(a) Tekever AR5 wing. (b) Optimized airfoil. (c) Optimization with all design variables.

Figure 7.23: Pressure distribution for the Tekever AR5 wing and the wings with an optimized airfoil and with all design variables on a vertical plane slightly behind the wingtip trailing edge.

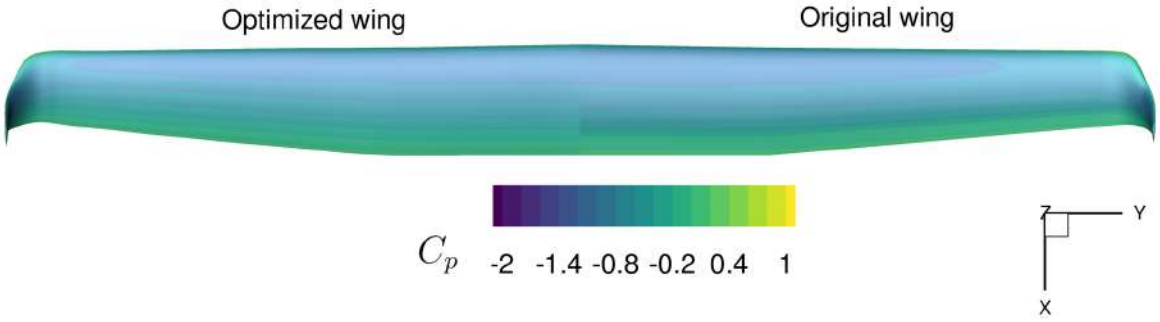
7.4.5 Complete optimization

A case with all the design variables was also performed. As the chord optimization, its finish was also forced, given that it stalled and started analyzing the same geometry on repeat without converging. Despite this, a significant improvement in drag reduction is verified. Twist, dihedral and airfoil parameterization were defined as before. A sweep design variable was also added, defined similarly to the dihedral but on the streamwise direction. For the chord distribution, to keep the wing root mount as similar as possible to the original, given that this wing is to be used in a growth version of the Tekever AR5, chord distribution was kept constant at the inner section. Furthermore, to avoid the oscilating distribution problems verified with chord optimization, a decreasing monotonic variation constraint was applied at the second wing section. At the winglet region, no special treatment was applied to the chord variation. The resulting geometry is shown in Figure 7.24. As before, dihedral was used to push the winglet upwards, but in this case the maximum value was not reached and the maximum vertical displacement was 9.7 cm. Some downwards deflection was also verified, although considerably smaller than the one observed for dihedral optimization. Chord scale factor varied between 1.02 and 0.94 at the middle section, where the monotonic constraint was applied and was used to smooth the junction between the inner and outer sections. As with the simplified wing, sweep variation was not significant, with the maximum streamwise displacement of 3.2 cm at the top of the winglet. Twist was kept under 1° on the first section and increased from there to a maximum value of 2.9° near the junction between the outer wing section and the winglet, being near zero at every section of the winglet.

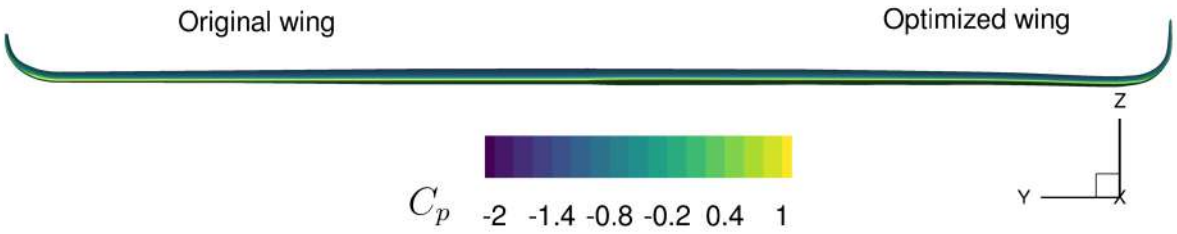
The airfoil is also compared to the optimization considering only airfoil and to the original geometry. It



(a) Side view.



(b) Front view.

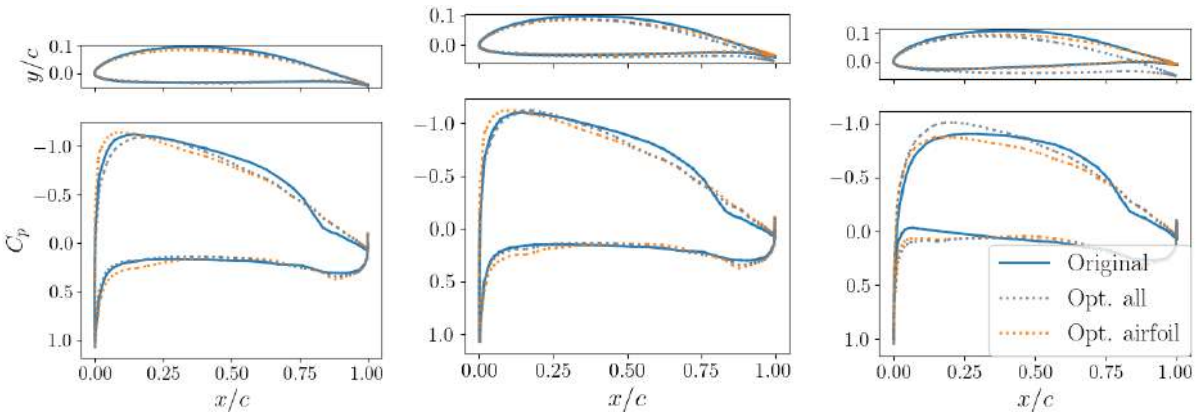


(c) Top view.

Figure 7.24: Pressure distribution comparison between the Tekever AR5 wing and the optimized wing considering all design variables.

is possible to observe that both optimized airfoils are similar and result in similar pressure distributions, particularly at the inner sections. At a section near the winglet (85% wingspan), the application of twist is also clear. The obtained drag reduction was greater than the one for airfoil optimization by 0.62%, which is noteworthy but may not be enough to justify the considerable changes in the overall wing design, particularly at the winglet with the dihedral application.

In Figure 7.23, it is possible to see that the vortex generated at the wingtip is not even visible with the used color scale, being even weaker than the one generated by the wing with the optimized airfoil, without the excessive airfoil deformation at the wingtip verified on the optimization with just the airfoil, as other design variables such as dihedral are available in this case to reduce induced drag.



(a) Pressure distribution at 1% wingspan. (b) Pressure distribution at 45% wingspan. (c) Pressure distribution at 85% wingspan.

Figure 7.25: Comparison between the original Tekever AR5 airfoil and the result from airfoil optimization and from optimization with all variables.

7.4.6 Verification with a finer grid

As discussed in Section 6.1, the grid used for optimization still presented a significant drag error compared to the finer grid tested. Thus, the optimized geometry considering airfoil shape was verified with the 7.5 million element grid and the trim angle of attack found to observe if the trend in drag change verified was similar to that obtained with the coarser grid. Comparing the optimized geometry to the initial one on this grid, a drag reduction of 2.56% was obtained, representing a difference of 1.34% to the coarser grid. However, it is important to note that the lift coefficient for the optimized geometry on the finer grid was also 0.6% higher than the one obtained in the coarser grid, and so the difference in aerodynamic efficiency obtained with both grids was just 0.8%. Figure 7.26 shows visually the variation observed in total, pressure and viscous drag with different grids for both the initial and optimized solution, and it can be seen that, although with different slopes, all the lines follow the same trend, validating the use of a coarser grid to speed up the optimization process, as long as it is refined enough to capture relevant flow features.

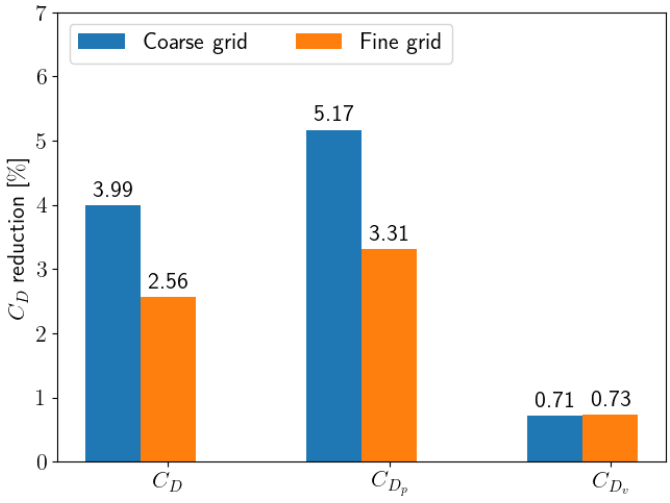


Figure 7.26: Drag coefficient trend with optimization for different grids.

7.4.7 Results for the Tekever AR5 wing with fuselage

The wing with an optimized airfoil was mounted in the fuselage again. For this, the methodology presented in Section 5.5 was used to deform the wing CAD file, from which a new collar grid was created. The new overset grid was then analyzed at the angle of attack found for the optimized wing and a drag reduction of 1.74% when compared to the initial geometry was obtained and, as expected, the reduction was smaller than the one found for the wing in free-stream, however it still represented an improvement because, as discussed before, most of the wingspan is not significantly affected by the presence of the fuselage, and so, the optimal wing aerodynamic shape at those regions is expected to be similar to the one obtained in free-stream. Furthermore, this figure is diluted by the fuselage drag, given that comparisons where performed with the total drag coefficient for the whole geometry. As discussed before, major changes were observed in the winglet region to reduce induced drag, where the influence of the fuselage is minimal. A 0.85% increase in produced lift from the initial geometry was also verified.

Figure 7.27 also shows the pressure coefficient on several sections along the wingspan for the optimized wing. For more outboard sections, the behavior is exactly as verified before. On the first wing section, there is an higher suction peak on the wing with optimized airfoil near the leading edge when compared with the original geometry, with the airfoil behaving as desired otherwise.

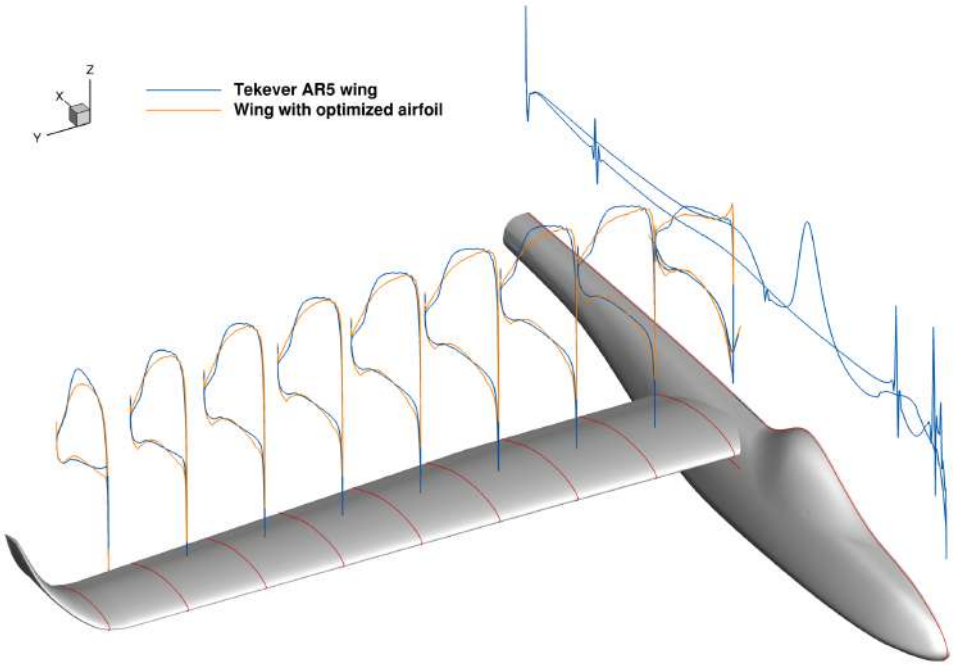


Figure 7.27: Pressure coefficient comparison between Tekever AR5 wing and wing with optimized airfoil on several sections along the wingspan.

Chapter 8

Conclusions and Future Work

8.1 Achievements

In this work, MACH-Aero, an aerodynamic shape optimization framework, was explored and adapted for the optimization of the Tekever AR5 wing. Furthermore, an integrated process was created around it, including pre- and post-processing. Modifications to the FFD-box generation function were performed in order to generalize it. Furthermore, new parameterizations to handle the deformation of complex initial geometries with meaningful design variables were defined and found to work as intended despite being piecewise approximations to the original geometry. The framework was then used with three different starting geometries - a rectangular wing with a symmetrical NACA airfoil, a simplification of the current Tekever AR5 wing and the Tekever AR5 wing.

For the initial rectangular wing case, a notable drag reduction of 8.55% was achieved with the optimization of a variable airfoil across the wingspan, while significant drag reductions were obtained for global design variables, whose distribution was mostly optimized to achieve an elliptical lift distribution, which was not always possible, highlighting the importance of using high-fidelity viscous models. Due to its simplicity, this initial geometry offered a much simpler set-up with straightforward FFD-box generation and parameterizations. The use of this naive starting geometry highlighted the capabilities of the framework for the creation of new designs from scratch considering a specified flight condition.

More modest drag reductions of 2.54% and 4.52% for the simplified Tekever AR5 wing and the Tekever AR5 wing respectively were obtained, mainly provided by airfoil optimization. Relatively small perturbations around the initial geometry were able to decrease drag significantly, posing itself as an interesting option for the development of a growth version of the Tekever AR5 UAV without requiring major changes in the design philosophy, which would necessarily be accompanied by major structural changes.

In the Tekever AR5 wing optimization, dihedral design variables were also found to yield significant drag reductions, although the optimal is clearly not ideal from a structures perspective. From the optimization with all design variables, major changes were observed at the winglet region, although this results should be analyzed carefully and taking into account the model limitations. With this, a second major application of the framework was demonstrated - the refinement of geometries resulting from

extensive design processes.

The use of a coarser grid for optimization also proved to be a good approach, as demonstrated by the verification of an optimized geometry. Although the reduction in drag coefficient was not as high as the coarser grid predicted, the correct trend was captured. Effects of the fuselage presence were also studied. It was concluded that fuselage interference only affected a small portion of the wing and as such, optimization with an isolated wing still yielded a drag reduction of 1.74% for the wing-fuselage assembly.

8.2 Future work

Considering the optimizations performed, the major limitation found was the mesh deformation algorithm used, which would often generate unusable volume grids when subjected to excessive deformations, severely limiting the optimization space and impeding optimizations from finishing properly and reach an optimum. Although some tuning of the mesh deformation parameters helped to mitigate this issue, it was never completely eliminated and so, more complex deformation algorithms may be required. Differentiable embedded boundary methods are a possibility for this, although still a research topic [165].

The flow model used also had some limitations related with the use of a compressible solver and to the transition modeling. In the future, the use of a transition model between laminar and turbulent flow would allow more accurate results for both analysis and optimization.

All optimizations performed in this work were for a clean wing at cruise condition, and the capabilities of the optimized wing were not verified at other conditions, which may limit the optimization space. To consider this, multi-point optimization can be used. Furthermore, attention was mainly focused on lift and drag coefficients. However, when designing a wing, other parameters should be verified, including moment coefficients and stall characteristics. Furthermore, manufacture constraints including, for example minimum curvature radius, can also be included and should be discussed directly with the manufacturer.

Analysis with the complete Tekever AR5 geometry, including not only the fuselage but also the tail and gimbal should be envisioned in the future, as that will determine the performance of the full UAV with the interaction between the different components. Furthermore, it would also allow a better validation of the obtained results, as the total lift produced could be directly compared with manufacturer data for the flight angle of attack of Tekever AR5 for different payload weights. Propulsion effects should also be added, which would certainly change the optimal wing shape since this UAV has the propellers directly in front of the wing.

Finally, this project should naturally evolve to an aerostructural optimization problem. As seen in some of the obtained geometries, if the wing structure was taken into account, they would look considerably different, as the only consideration here is a simplistic minimum thickness to accommodate structural components with an arbitrarily defined value, which would not be required if multidisciplinary analysis was performed. Furthermore, it is known that the wing will deflect under load due to aeroelasticity, resulting in a different flying shape from the designed one. The impact of this could be predicted with aerostructural optimization, where the wing would be optimized at its deformed flight shape.

Bibliography

[1] K. P. Valavanis and G. J. Vachtsevanos. *Handbook of Unmanned Aerial Vehicles*. Springer, 1^{st} edition, 2014. ISBN 9789048197064.

[2] J. F. Keane and S. S. Carr. A brief history of early unmanned aircraft. Technical Report 32-3, Johns Hopkins University Applied Physics Laboratory, December 2013.

[3] K. L. Cook. The silent force multiplier: The history and role of UAVs in warfare. In *2007 IEEE Aerospace Conference*, Big Sky, Montana, USA, March 2007. IEEE. doi:10.1109/AERO.2007.352737.

[4] D. W. Irvin. *History of Strategic Drone Operations*. Turner Publishing Company, 1^{st} edition, 2003. ISBN 9781563118913.

[5] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. *IEEE Access*, 7:48572–48634, 2019. doi:10.1109/ACCESS.2019.2909530.

[6] Drone Industry Insights. Global drone market report 2022-2030, 2022. URL `https://droneii.com/product/drone-market-report`. Accessed: 2022-10-28.

[7] Drone Industry Insights. Drone market environment database, 2019. URL `https://droneii.com/product/the-drone-market-environment-database-2019`. Accessed: 2022-10-28.

[8] K. Dalamagkidis, K. P. Valavanis, and L. A. Piegl. *On Integrating Unmanned Aircraft Systems into the National Airspace System: Issues, Challenges, Operational Restrictions, Certification, and Recommendations*. Springer, 2^{nd} edition, 2012. ISBN 9789400724785.

[9] J. R. Martins and A. B. Lambe. Multidisciplinary design optimization: a survey of architectures. *AIAA journal*, 51(9):2049–2075, 2013. doi:10.2514/1.J051895.

[10] J. R. Martins. Aerodynamic design optimization: Challenges and perspectives. *Computers & Fluids*, 239:105391, 2022. doi:10.1016/j.compfluid.2022.105391.

[11] X. He, J. Li, C. A. Mader, A. Yildirim, and J. R. R. A. Martins. Robust aerodynamic shape optimization—from a circle to an airfoil. *Aerospace Science and Technology*, 87:48–61, 2019. doi:10.1016/j.ast.2019.01.051.

[12] C. A. Mader, G. K. Kenway, A. Yildirim, and J. R. Martins. ADflow: An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization. *Journal of Aerospace Information Systems*, 17(9):508–527, 2020. doi:10.2514/1.I010796.

[13] G. K. W. Kenway, C. A. Mader, P. He, and J. R. R. A. Martins. Effective adjoint approaches for computational fluid dynamics. *Progress in Aerospace Sciences*, 110:100542, 2019. doi:10.1016/j.paerosci.2019.05.002.

[14] J. Roskam. *Lessons Learned in Aircraft Design*. DARcorporation, 2^{nd} edition, 2012. ISBN 9781884885587.

[15] D. P. Raymer. *Aircraft Design: A Conceptual Approach*. American Institute of Aeronautics and Astronautics, 6^{th} edition, 2018. ISBN 9781624104909.

[16] E. Torenbeek. *Advanced Aircraft Design*. Wiley, A. John and Sons, 1^{st} edition, 2013. ISBN 9781118568118.

[17] J. D. Anderson. *Aircraft Performance and Design*. WCB McGraw-Hill, 1^{st} edition, 1999. ISBN 9780070019713.

[18] S. Morton, R. D'Sa, and N. Papanikolopoulos. Solar powered UAV: Design and experiments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2460–2466, Hamburg, Germany, September 2015. IEEE. doi:10.1109/IROS.2015.7353711.

[19] T. Nietz and S. Baber. An innovative UAV design. In *AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, Chicago, Illinois, USA, September 2004. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2004-6380.

[20] E. Çetinsoy, S. Dikyar, C. Hançer, K. Oner, E. Sirimoglu, M. Unel, and M. Aksit. Design and construction of a novel quad tilt-wing UAV. *Mechatronics*, 22(6):723–745, 2012. doi:10.1016/j.mechatronics.2012.03.003.

[21] H. Karakas, E. Koyuncu, and G. Inalhan. ITU tailless UAV design. *Journal of Intelligent & Robotic Systems*, 69(1-4):131–146, 2013. doi:10.1007/s10846-012-9695-4.

[22] A. J. Keane, A. Sóbester, and J. P. Scanlan. *Small Unmanned Fixed-Wing Aircraft Design: A Practical Approach*. John Wiley & Sons, 1^{st} edition, 2017. ISBN 9781119406297.

[23] R. Austin. *Unmanned Aircraft Systems: UAVS Design, Development and Deployment*. John Wiley & Sons, 1^{st} edition, 2011. ISBN 9781119964261.

[24] D. Verstraete, J. L. Palmer, and M. Hornung. Preliminary sizing correlations for fixed-wing unmanned aerial vehicle characteristics. *Journal of Aircraft*, 55(2):715–726, 2018. doi:10.2514/1.C034199.

[25] L. Breguet. Aerodynamical efficiency and the reduction of air transport costs. *The Aeronautical Journal*, 26(140):307–320, 1922. doi:10.1017/S2398187300135807.

[26] R. Kress. Variable sweep wing design. In *The evolution of aircraft wing design; Proceedings of the Symposium*, Dayton, Ohio, USA, March 1980. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1980-3043.

[27] L. Demasi. Investigation on the conditions of minimum induced drag of closed wing systems and C-wings. *Journal of Aircraft*, 44(1):81–99, 2007. doi:10.2514/1.21884.

[28] S. Ameduri and A. Concilio. Morphing wings review: Aims, challenges, and current open issues of a technology. *Journal of Mechanical Engineering Science*, 237(18):4112–4130, 2020. doi:10.1177/0954406220944423.

[29] P. Okonkwo and H. Smith. Review of evolving trends in blended wing body aircraft design. *Progress in Aerospace Sciences*, 82, 2016. doi:10.1016/j.paerosci.2015.12.002.

[30] K. Krishnan, O. Bertram, and O. Seibel. Review of hybrid laminar flow control systems. *Progress in Aerospace Sciences*, 93:24–52, 2017. doi:10.1016/j.paerosci.2017.05.005.

[31] M. F. Platzer, K. D. Jones, J. Young, and J. C. Lai. Flapping wing aerodynamics: Progress and challenges. *AIAA journal*, 46(9):2136–2149, 2008. doi:10.2514/1.29263.

[32] J. S. Arora. *Introduction to optimum design*. Elsevier, 4^{th} edition, 2017. ISBN 9780128008065.

[33] T. Chau and D. W. Zingg. Aerodynamic design optimization of a transonic strut-braced-wing regional aircraft. *Journal of Aircraft*, 59(1):253–271, 2022. doi:10.2514/1.C036389.

[34] K. Telidetzki, L. Osusky, and D. W. Zingg. Application of jetstream to a suite of aerodynamic shape optimization problems. In *52nd Aerospace Sciences Meeting*, page 0571, National Harbor, Maryland, USA, January 2014. doi:https://doi.org/10.2514/6.2014-0571.

[35] D. Lovely and R. Haimes. Shock detection from computational fluid dynamics results. In *14th computational fluid dynamics conference*, page 3285, Norfolk, Virginia, USA, November 1999. doi:10.2514/6.1999-3285.

[36] S. Seraj and J. R. Martins. Aerodynamic shape optimization of a supersonic transport considering low-speed stability. In *AIAA SCITECH 2022 Forum*, San Diego, California, USA, January 2022. doi:10.2514/6.2022-2177.

[37] Y. Lu, T. Su, R. Chen, P. Li, and Y. Wang. A method for optimizing the aerodynamic layout of a helicopter that reduces the effects of aerodynamic interaction. *Aerospace Science and Technology*, 88:73–83, 2019. doi:https://doi.org/10.1016/j.ast.2019.03.005.

[38] S. Khosravi and D. W. Zingg. Aerostructural optimization of drooped wings. *Journal of Aircraft*, 55 (3):1261–1268, 2018. doi:10.2514/1.C034605.

[39] L. Zhang, M. Dongli, Y. Muqing, and W. Shaoqi. Optimization and analysis of winglet configuration for solar aircraft. *Chinese Journal of Aeronautics*, 33(12):3238–3252, 2020. doi:10.1016/j.cja.2020.04.008.

[40] Y. Denieul, J. Bordeneuve, D. Alazard, C. Toussaint, and G. Taquin. Multicontrol surface optimization for blended wing–body under handling quality constraints. *Journal of Aircraft*, 55(2):638–651, 2018. doi:10.2514/1.C034268.

[41] S. Nicolay, S. Karpuk, Y. Liu, and A. Elham. Conceptual design and optimization of a general aviation aircraft with fuel cells and hydrogen. *International Journal of Hydrogen Energy*, 46(64): 32676–32694, 2021. doi:10.1016/j.ijhydene.2021.07.127.

[42] E. M. Botero, A. Wendorff, T. MacDonald, A. Variyar, J. M. Vegh, T. W. Lukaczyk, J. J. Alonso, T. H. Orra, and C. Ilario da Silva. Suave: An open-source environment for conceptual vehicle design and optimization. In *54th AIAA aerospace sciences meeting*, page 1275, San Diego, California, USA, January 2016. doi:10.2514/6.2016-1275.

[43] J. M. M. Júnior, G. L. Halila, Y. Kim, T. Khamvilai, and K. G. Vamvoudakis. Intelligent data-driven aerodynamic analysis and optimization of morphing configurations. *Aerospace Science and Technology*, 121:107388, 2022. doi:10.1016/j.ast.2022.107388.

[44] M. Mani and A. J. Dorgan. A perspective on the state of aerospace computational fluid dynamics technology. *Annual Review of Fluid Mechanics*, 55:431–457, 2023. doi:10.1146/annurev-fluid-120720-124800.

[45] F. D. Witherden and A. Jameson. Future directions in computational fluid dynamics. In *23rd AIAA Computational Fluid Dynamics Conference*, Dayton, Ohio, USA, June 2017. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2017-3791.

[46] E. Tinoco. The role of computational fluid dynamics (CFD) in aircraft design. In *Aerospace Engineering Conference and Show*, Los Angeles, California, USA, February 1990. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1990-1801.

[47] W. Davis. The role of CFD applied to high performance aircraft. In *Flight Simulation Technologies Conference and Exhibit*, Dayton, Ohio, USA, September 1990. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1990-3071.

[48] J. Vos, A. Rizzi, D. Darracq, and E. Hirschel. Navier-Stokes solvers in european aircraft design. *Progress in Aerospace Sciences*, 38(8):601–697, 2002. doi:10.1016/S0376-0421(02)00050-7.

[49] M. Potsdam, M. Page, R. Liebeck, M. Potsdam, M. Page, and R. Liebeck. Blended wing body analysis and design. In *15th Applied Aerodynamics Conference*, Atlanta, Georgia, USA, June 1997. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1997-2317.

[50] P. Panagiotou, P. Kaparos, and K. Yakinthos. Winglet design and optimization for a MALE UAV using CFD. *Aerospace Science and Technology*, 39:190–205, 2014. doi:10.1016/j.ast.2014.09.006.

[51] G. M. Laskowski, J. Kopriva, V. Michelassi, S. Shankaran, U. Paliath, R. Bhaskaran, Q. Wang, C. Talnikar, Z. J. Wang, and F. Jia. Future directions of high fidelity CFD for aerothermal turboma-

chinery analysis and design. In *46th AIAA Fluid Dynamics Conference*, Washington, D.C., USA, June 2016. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2016-3322.

[52] S. Kalia, C. Vinay, and S. M. Hegde. CFD analysis of turboprop engine oil cooler duct for best rate of climb condition. In *IOP Conference Series: Materials Science and Engineering*, Bangalore, India, July 2016. IOP Science. doi:10.1088/1757-899X/149/1/012196.

[53] Z. H. Jia and S. Lee. High-fidelity computational analysis on the noise of a side-by-side hybrid vtol aircraft. *Journal of the American Helicopter Society*, 67(2), 2022. doi:10.4050/JAHS.67.022005.

[54] G. Djahid, K. Elena, P. Maxim, K. Alexander, S. Popov, and M. Sinha. Experimental and cfd investigation of directional stability of a box-wing aircraft concept. *Fluids*, 7(11):340, 2022. doi:10.3390/fluids7110340.

[55] A. Bertram, N. Hoffmann, S. Goertz, R. Gebbink, and S. R. Janssen. An alternative wind tunnel data correction based on CFD and experimental data in the transonic flow regime. In *AIAA Aviation 2021 Forum*, Virtual event, August 2021. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2021-2982.

[56] R. Ansorge and T. Sonar. *Mathematical Models of Fluid Dynamics: Modelling, Theory, Basic Numerical Facts - An Introduction*. John Wiley & Sons, 2^{nd} edition, 2009. ISBN 9783527627967.

[57] C. Hirsch. *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics*. Elsevier, 2^{nd} edition, 2007. ISBN 978-0-7506-6594-0.

[58] P. Rubbert and G. Saaris. Review and evaluation of a three-dimensional lifting potential flow computational method for arbitrary configurations. In *10th Aerospace Sciences Meeting*, San Diego, California, USA, January 1972. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1972-188.

[59] L. Fornasier. Linearized potential flow analysis of complex aircraft configurations by HISSS, a higher order panel method. In *23rd Aerospace Sciences Meeting*, Reno, Nevada, USA, January 1985. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1985-281.

[60] R. Carmichael and L. Erickson. PAN AIR-A higher order panel method for predicting subsonic or supersonic linear potential flows about arbitrary configurations. In *AIAA 14th Fluid and Plasma Dynamics Conference*, Palo Alto, California, USA, June 1981. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1981-1255.

[61] L. Euler. Principes généraux du mouvement des fluides. *Mémoires de l'académie des sciences de Berlin*, 11:274–315, 1757.

[62] H. Gagnon and D. W. Zingg. Euler-equation-based drag minimization of unconventional aircraft configurations. *Journal of Aircraft*, 53(5):1361–1371, 2016. doi:10.2514/1.C033591.

[63] M. H. Straathof, G. Carpentieri, and M. J. van Tooren. Aerodynamic shape optimization using the adjoint euler equations. *Engineering Computations*, 30(4):469–493, 2013. doi:10.1108/02644401311329334.

[64] Y. Kaneda and T. Ishihara. High-resolution direct numerical simulation of turbulence. *Journal of Turbulence*, 7(20), 2006. doi:10.1080/14685240500256099.

[65] G. Alfonsi. On direct numerical simulation of turbulent flows. *Applied Mechanics Reviews*, 64(2): 020802, 2011. doi:10.1115/1.4005282.

[66] S. V. Poroseva, J. D. Colmenares F, and S. M. Murman. On the accuracy of RANS simulations with DNS data. *Physics of Fluids*, 28(11), 2016. doi:10.1063/1.4966639.

[67] S. Nishiki, T. Hasegawa, R. Borghi, and R. Himeno. Modelling of turbulent scalar flux in turbulent premixed flames based on DNS databases. *Combustion Theory and Modelling*, 10(1):39–55, 2006. doi:10.1080/13647830500307477.

[68] M. Germano. Turbulence: the filtering approach. *Journal of Fluid Mechanics*, 238:325–336, 1992. doi:10.1017/S0022112092001733.

[69] O. Reynolds. On the dynamical theory of incompressible viscous fluid and the determination of the criterion. *Philosophical Transactions of the Royal Society of London*, 174(1883):935–982, 1894.

[70] G. Alfonsi. Reynolds Averaged Navier–Stokes equations for turbulence modeling. *Applied Mechanics Review*, 62(4), 2009. doi:10.1115/1.3124648.

[71] W. Rodi. Comparison of LES and RANS calculations of the flow around bluff bodies. *Journal of Wind Engineering and Industrial Aerodynamics*, 69:55–75, 1997. doi:10.1016/S0167-6105(97)00147-5.

[72] S. Heinz. A review of hybrid RANS–LES methods for turbulent flows: Concepts and applications. *Progress in Aerospace Sciences*, 114:100597, 2020. doi:10.1016/j.paerosci.2019.100597.

[73] F. Menter, A. Hüppe, A. Matyushenko, and D. Kolmogorov. An overview of hybrid RANS–LES models developed for industrial CFD. *Applied Sciences*, 11(6):2459, 2021. doi:10.3390/app11062459.

[74] D. C. Wilcox et al. *Turbulence modeling for CFD*. DCW industries, 3^{rd} edition, 2006. ISBN 9781928729082.

[75] P. Durbin and T. Shih. An overview of turbulence modeling. *WIT Transactions on State of the Art in Science and Engineering*, 16, 2005. doi:10.2495/978-1-85312-956-8/01.

[76] E. Tulapurkara. Turbulence models for the computation of flow past airplanes. *Progress in Aerospace Sciences*, 33(1), 1997. doi:10.1016/S0376-0421(96)00002-4.

[77] J. Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 3^{rd} edition, 2015. ISBN 9780080999951.

[78] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, USA, January 1992. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1992-439.

[79] D. Wilcox. A half century historical review of the k-omega model. In *29th Aerospace Sciences Meeting*, Reno, Nevada, USA, January 1991. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1991-615.

[80] T. Sivagnanam, A. Ridha, and G. Charles. Application of a new k-tau model to near wall turbulent flows. *AIAA Journal*, 30(2), 1992. doi:10.2514/3.10952.

[81] F. R. Menter, M. Kuntz, R. Langtry, et al. Ten years of industrial experience with the sst turbulence model. *Turbulence, heat and mass transfer*, 4:625–632, 2003.

[82] D. Laurence, J. Uribe, and S. Utyuzhnikov. A robust formulation of the v2f model. *Flow, Turbulence and Combustion*, 73:169–185, 2005. doi:10.1007/s10494-005-1974-8.

[83] M. Z. M. Shah, B. Basuno, and A. Abdullah. Comparative study on several type of turbulence model available in Ansys-Fluent software for Onera M6 wing aerodynamic analysis. *Journal of Advanced Mechanical Engineering Applications*, 1(1):9–19, 2020. doi:10.30880/jamea.2020.01.01.000.

[84] P. Eliasson. Influence of turbulence modelling and grid resolution in computations of the DPW7 CRM configuration. In *AIAA Aviation 2023 Forum*, San Diego, California, USA, June 2023. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2023-4091.

[85] L. Kral. Recent experience with different turbulence models applied to the calculation of flow over aircraft components. *Progress in Aerospace Sciences*, 34:481–541, 1998. doi:10.1016/S0376-0421(98)00009-8.

[86] J. Bardina, P. Huang, and T. Coakley. Turbulence modeling validation. In *28th Fluid Dynamics Conference*, Snowmass Village, Colorado, USA, July 1997. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1997-2121.

[87] J. G. Coder, T. H. Pulliam, D. Hue, G. K. Kenway, and A. J. Sclafani. Contributions to the 6th AIAA CFD drag prediction workshop using structured grid methods. In *55th AIAA Aerospace Sciences Meeting*, Grapevine, Texas, USA, January 2017. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2017-0960.

[88] C.-W. Shu. High-order finite difference and finite volume WENO schemes and discontinuous galerkin methods for CFD. *International Journal of Computational Fluid Dynamics*, 17(2):107–118, 2003. doi:10.1080/1061856031000104851.

[89] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 1^{st} edition, 2016. ISBN 9783319168739.

[90] S. F. Olatoyinbo. Verification of a high-order FEM-based CFD code using the method of manufactured solutions. *INCAS Bulletin*, 15(2):75–89, 2023. doi:10.13140/RG.2.2.29289.62567.

[91] J. N. Reddy and D. K. Gartling. *The Finite Element Method in Heat Transfer and Fluid Dynamics*. CRC press, 3^{rd} edition, 2010. ISBN 9781420085983.

[92] W. D. Henshaw. Automatic grid generation. *Acta numerica*, 5:121–148, 1996. doi:10.1017/S0962492900002634.

[93] I. Sadrehaghighi. Mesh generation in CFD. Technical Report 1.86.7, CFD Open Series, Annapolis, Maryland, January 2020.

[94] T. J. Baker. Mesh generation: Art or science? *Progress in Aerospace Sciences*, 41(1):29–63, 2005. doi:10.1016/j.paerosci.2005.02.002.

[95] S. Pandya, W. Chan, and J. Kless. Automation of structured overset mesh generation for rocket geometries. In *19th AIAA computational fluid dynamics*, San Antonio, Texas, USA, June 2009. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2009-3993.

[96] W. Chan, R. Gomez, S. Rogers, and P. Buning. Best practices in overset grid generation. In *32nd AIAA Fluid Dynamics Conference and Exhibit*, St. Louis, Missouri, USA, June 2002. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2002-3191.

[97] R. V. Garimella and M. S. Shephard. Boundary layer mesh generation for viscous flow simulations. *International Journal for Numerical Methods in Engineering*, 49(1-2):193–218, 2000. doi:10.1002/1097-0207(20000910/20)49:1/2<193::AID-NME929>3.0.CO;2-R.

[98] MDO Lab. *ADFlow Documentation*. MDO Lab, University of Michigan, USA, 2022. URL `https://mdolab-adflow.readthedocs-hosted.com`. Accessed: 2022-11-07.

[99] P. Wesseling. *An introduction to multigrid methods*. John Wiley & Sons, 1^{st} edition, 1992. ISBN 9780471930839.

[100] F. Witherden, A. Jameson, and D. Zingg. The design of steady state schemes for computational aerodynamics. *Handbook of Numerical Analysis*, 18:303–349, 2017. doi:10.1016/bs.hna.2016.11.006.

[101] E. L. Allgower and K. Georg. *Introduction to numerical continuation methods*. SIAM, 2^{nd} edition, 2003. ISBN 9780898715446.

[102] J. E. Hicken and D. W. Zingg. Parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms. *AIAA Journal*, 46(11):2773–2786, 2008. doi:10.2514/1.34810.

[103] M. Osusky and D. W. Zingg. Parallel Newton–Krylov–Schur flow solver for the Navier–Stokes equations. *AIAA Journal*, 51(12), 2013. doi:10.2514/1.J052487.

[104] A. Yildirim, G. K. W. Kenway, C. A. Mader, and J. R. R. A. Martins. A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations. *Journal of Computational Physics*, 397: 108741, 2019. doi:10.1016/j.jcp.2019.06.018.

[105] A. Jameson, W. Schmidt, and E. Turkel. Numerical solution of the euler equations by finite volume methods using runge kutta time stepping schemes. In *14th fluid and plasma dynamics conference*, Palo Alto, California, USA, 1981. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1981-1259.

[106] C. Xin, W. Gang, Y. Z. Yin, and W. Xiaojun. A review of uncertainty quantification methods for computational fluid dynamics. *Acta Aerodynamica Sinica*, 39(4), 2021. doi:10.7638/kqdlxxb-2021.0012.

[107] L. Eça. *Fundamentos de aerodinâmica incompressível: Exercícios*. IST Press, 2^{nd} edition, 2020. ISBN 9789898481337.

[108] S. H. Zanakis and J. R. Evans. Heuristic "optimization": Why, when, and how to use it. *Interfaces*, 11(5):84–91, 1981. doi:10.1287/inte.11.5.84.

[109] S. Katoch, S. S. Chauhan, and V. Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80:8091–8126, 2021. doi:10.1007/s11042-020-10139-6.

[110] J. Kennedy and R. Eberh. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, Australia, November 1995. IEEE. doi:10.1109/ICNN.1995.488968.

[111] D. Bertsimas and J. Tsitsiklis. Simulated annealing. *Statistical science*, 8(1), 1993. doi:10.1214/ss/1177011077.

[112] M. Gilli and P. Winker. A review of heuristic optimization methods in econometrics. Technical Report 08-12, Swiss Finance Institute, June 2008.

[113] S. Darvishpoor, A. Darvishpour, M. Escarcega, and M. Hassanalian. Nature-inspired algorithms from oceans to space: A comprehensive review of heuristic and meta-heuristic optimization algorithms and their potential applications in drones. *Drones*, 7(7), 2023. doi:10.3390/drones7070427.

[114] J. R. R. A. Martins and A. Ning. *Engineering Design Optimization*. Cambridge University Press, 1^{st} edition, 2021. ISBN 9781108833417.

[115] M. Cavazzuti. *Optimization Methods: From Theory to Design Scientific and Technological Aspects in Mechanics*. Springer, 1^{st} edition, 2012. ISBN 9783642311864.

[116] D. Jones, C. Perttunen, and B. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79:157–181, 1993. doi:10.1007/BF00941892.

[117] D. M. Olsson and L. S. Nelson. The Nelder-Mead simplex procedure for function minimization. *Technometrics*, 17(1):45–51, 1975. doi:10.1080/00401706.1975.10489269.

[118] J. Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218 (3):587–601, 2012. doi:10.1016/j.ejor.2011.09.017.

[119] G. R. Wood. The bisection method in higher dimensions. *Mathematical programming*, 55:319–337, 1992. doi:doi.org/10.1007/BF01581205.

[120] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta numerica*, 4, 1995. doi:10.1017/S0962492900002518.

[121] Y.-x. Yuan. A review of trust region algorithms for optimization. In *ICM99: Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, Edinburgh, Scotland, September 1999. International Council for Industrial and Applied Mathematics.

[122] A. Marta. Lecture notes on aircraft optimal design, September 2021. Instituto Superior Técnico.

[123] S. N. Skinner and H. Zare-Behtash. State-of-the-art in aerodynamic shape optimisation methods. *Applied Soft Computing*, 62:933–962, 2018. doi:10.1016/j.asoc.2017.09.030.

[124] D. W. Zingg, M. Nemec, and T. H. Pulliam. A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization. *European Journal of Computational Mechanics*, 17(1-2):103–126, 2008-01-01. doi:10.3166/remn.17.103-126.

[125] Z. Lyu, Z. Xu, and J. R. R. A. Martins. Benchmarking optimization algorithms for wing aerodynamic design optimization. In *Proceedings of the 8th International Conference on Computational Fluid Dynamics*, Chengdu, Sichuan, China, July 2014. ICCFD.

[126] H. R. Karbasian and B. C. Vermeire. Gradient-free aerodynamic shape optimization using large eddy simulation. *Computers & Fluids*, 232:105185, 2022. doi:10.1016/j.compfluid.2021.105185.

[127] D. Rajnarayan, A. Haas, and I. Kroo. A multifidelity gradient-free optimization method and application to aerodynamic design. In *12th AIAA/ISSMO multidisciplinary analysis and optimization conference*, Victoria, Canada, September 2008. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2008-6020.

[128] D. Sasaki, M. Morikawa, S. Obayashi, and K. Nakahashi. Aerodynamic shape optimization of supersonic wings by adaptive range multiobjective genetic algorithms. In *First International Conference on Evolutionary Multi-Criterion Optimization Proceedings*, pages 639–652, Zurich, Switzerland, March 2001. Springer. doi:10.1007/3-540-44719-9_45.

[129] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002. doi:10.1109/4235.996017.

[130] N. Bons. *High-fidelity Wing Design Exploration with Gradient-based Optimization*. PhD thesis, University of Michigan, USA, 2020.

[131] H. Koyuncuoglu and P. He. CFD based multi-component aerodynamic optimization for wing propeller coupling. In *AIAA SciTech 2023 Forum and Exposition*, page 1844, National Harbor, Maryland, USA, January 2023. doi:10.2514/6.2023-1844.

[132] G. Yang and A. Da Ronch. Aerodynamic shape optimisation of benchmark problems using SU2. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0412, Kissimmee, Florida, USA, January 2018. doi:10.2514/6.2018-0412.

[133] Y. Yu, Z. Lyu, Z. Xu, and J. R. Martins. On the influence of optimization algorithm and initial design on wing aerodynamic shape optimization. *Aerospace Science and Technology*, 75:183–199, 2018. doi:10.1016/j.ast.2018.01.016.

[134] D. Kraft. A software package for sequential quadratic programming. Technical Report 88-28, Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt, 1988.

[135] J. E. Peter and R. P. Dwight. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, 39(3):373–391, 2010. doi:10.1016/j.compfluid.2009.09.013.

[136] J. E. V. Peter and R. P. Dwight. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, 39(3):373–391, 2010. doi:10.1016/j.compfluid.2009.09.013.

[137] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, 2003. doi:10.1145/838250.838251.

[138] A. H. Gebremedhin and A. Walther. An introduction to algorithmic differentiation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(1), 2020. doi:10.1002/widm.1334.

[139] L. Hascoët and V. Pascual. The tapenade automatic differentiation tool: Principles, model, and specification. *ACM Transactions on Mathematical Software*, 39, 2013. doi:10.1145/2450153.2450158.

[140] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3:233–260, 1988. doi:10.1007/BF01061285.

[141] J.-D. Müller and P. Cusdin. On the performance of discrete adjoint CFD codes using automatic differentiation. *International Journal for Numerical Methods in Fluids*, 47(8-9):939–945, 2005. doi:10.1002/fld.885.

[142] S. Nadarajah. Adjoint-based aerodynamic optimization of benchmark problems. In *53rd AIAA aerospace sciences meeting*, page 1948, Kissimmee, Florida, USA, January 2015. doi:10.2514/6.2015-1948.

[143] G. Kenway, G. Kennedy, and J. R. Martins. A CAD-free approach to high-fidelity aerostructural optimization. In *13th AIAA/ISSMO multidisciplinary analysis optimization conference*, Fort Worth, Texas, USA, September 2010. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2010-9231.

[144] G. Anderson, M. Aftosmis, and M. Nemec. Constraint-based shape parameterization for aerodynamic design. In *7th International Conference on Computational Fluid Dynamics*, Big Island, Hawaii, USA, July 2012. ICCFD.

[145] H. M. Hajdik, A. Yildirim, and J. R. R. A. Martins. Aerodynamic shape optimization with CAD-based geometric parameterization. In *AIAA SciTech Forum*, National Harbor, Maryland, USA, January 2023. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2023-0726.

[146] R. Haimes and J. Dannenhoffer. The engineering sketch pad: A solid-modeling, feature-based, web-enabled system for building parametric geometry. In *21st AIAA Computational Fluid Dynamics Conference*, San Diego, California, USA, June 2013. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2013-3073.

[147] D. Agarwal, S. Marques, and T. T. Robinson. Aerodynamic shape optimisation using parametric CAD and discrete adjoint. *Aerospace*, 9(12):743, 2022. doi:10.3390/aerospace9120743.

[148] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. SU2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, 54(3):828–846, 2016. doi:10.2514/1.J053813.

[149] G. Anderson, M. Aftosmis, and M. Nemec. Parametric deformation of discrete geometry for aerodynamic shape design. In *50th AIAA Aerospace Sciences Meeting*, Nashville, Tennessee, USA, January 2012. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2012-965.

[150] L. Li, J. Jiao, S. Sun, Z. Zhao, and J. Kang. Aerodynamic shape optimization of a single turbine stage based on parameterized free-form deformation with mapping design parameters. *Energy*, 169:444–455, 2019. doi:10.1016/j.energy.2018.12.031.

[151] R. A. McDonald and J. R. Gloudemans. Open Vehicle Sketch Pad: An open source parametric geometry and analysis tool for conceptual aircraft design. In *AIAA SciTech 2022 Forum and Exposition*, San Diego, USA, January 2022. doi:10.2514/6.2022-0004.

[152] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11): 1309–1331, 2009. doi:10.1002/nme.2579.

[153] N. Secco, G. K. W. Kenway, P. He, C. A. Mader, and J. R. R. A. Martins. Efficient mesh generation and deformation for aerodynamic shape optimization. *AIAA Journal*, 59(4), 2021. doi:10.2514/1.J059491.

[154] MDO Lab. *pyHyp Documentation*. MDO Lab, University of Michigan, USA, 2022. URL `https://mdolab-pyhyp.readthedocs-hosted.com`. Accessed: 2022-10-26.

[155] L. Eça, F. Pereira, and G. Vaz. Viscous flow simulations at high reynolds numbers without wall functions: Is $y^+ \simeq 1$ enough for the near-wall cells? *Computers & Fluids*, 170:157–175, 2018. doi:10.1016/j.compfluid.2018.04.035.

[156] F. M. White. *Fluid mechanics*. McGraw Hill, 7^{th} edition, 2011. ISBN 9780073529349.

[157] ICAO. U.S. Standard Atmosphere, 1976. Technical report, National Oceanic and Atmospheric Administration, National Aeronautics and Space Administration, United States Air Force, 1976.

[158] Tekever. Tekever>Meet Our UAS Models>AR5, 2022. URL `https://http://www.tekever.com/models/ar5/`. Accessed: 2022-10-30.

[159] S. Seraj, A. Yildirim, J. L. Anibal, and J. Martins. Improving the performance of a compressible RANS solver for low and high mach number flows. In *11th International Conference on Computational Fluid Dynamics*, Maui, Hawaii, USA, July 2022.

[160] MDO Lab. *MACH-Aero Documentation*. MDO Lab, University of Michigan, USA, 2022. URL `https://mdolab-mach-aero.readthedocs-hosted.com`. Accessed: 2023-07-12.

[161] N. Wu, G. Kenway, C. A. Mader, J. Jasa, and J. R. Martins. pyOptSparse: A Python framework for large-scale constrained nonlinear optimization of sparse systems. *Journal of Open Source Software*, 5(54):2564, 2020. doi:10.21105/joss.02564.

[162] MDO Lab. *pySpline Documentation*. MDO Lab, University of Michigan, USA, 2022. URL `https://mdolab-pyspline.readthedocs-hosted.com`. Accessed: 2023-06-05.

[163] MDO Lab. *pyGeo Documentation*. MDO Lab, University of Michigan, USA, 2022. URL `https://mdolab-pygeo.readthedocs-hosted.com`. Accessed: 2023-02-15.

[164] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi:10.1109/MCSE.2007.55.

[165] J. Ho and C. Farhat. Aerodynamic optimization with large shape and topology changes using a differentiable embedded boundary method. *Journal of Computational Physics*, 488:112191, 2023. doi:10.1016/j.jcp.2023.112191.

Appendix A

UAV Market Overview

Table A.1: Survey of some drones currently on the market for ISR applications

Manufacturer	Model	Endurance [h]	Comms Range [km]	MTOW [kg]	Payload [kg]	Wingspan [m]	Ceiling [m]	Cruise speed [km/h]
UKRSpecsystems	PD-1	7	100	40	10	4	3000	95
UKRSpecsystems	PD-2	8	200	55	11	5	4500	100
Tekever	AR3	16	100	25	4	4	3600	85
Tekever	AR4	2	20	4	1	2	4500	54
Tekever	AR5	12	∞	180	50	7	4500	100
UAVision	Ogassa	10	100	38	5	4	2438	95
UAVFactory	Penguin C	20	100	23	2	3	5000	79
AeroVironment	T-20	24	185	102	22,7	6	4600	111
AeroVironment	Jump 20	14	185	97,5	13,6	6	4572	93
AeroVironment	Puma LE	6,5	60	12,2	2,5	5	4600	46
Textron	Shadow	8	125	195	43	6	5486	148
Textron	Aerosonde	14	140	36,4	9,1	4	4572	83
Aeronautics Group	Dominator	28	∞	1610	410	14	9100	140
Airbus	Sirtap	20	∞	750	150	12	6400	90
Aisheng	ASN-229	10	200	800	100	11	5000	140
ADCOM Systems	United 40	120	∞	1500	980	20	7000	145
Bluebird Aero	Spylite	4	50	9,5	2	3	1000	100
Boeing	RQ-21	24	2400	61	18	5	4572	100
Boeing	ScanEagle	20	100	26,5	8,6	3	5950	80
Northrop Grumman	RQ-4	34	22800	14628	1400	40	18000	573
Elbit	Hermes 900	36	2500	1180	350	15	9145	112
EMT	Aladin	1	15	4	1	1	4500	45
IAI	Searcher II	20	350	450	68	9	6100	150
General Atomics	MQ-9 Reaper	42	1900	4760	1700	20	7500	313
General Atomics	Predator XP	35	4000	1160	386	17	7620	200
Denel Dynamics	Bateleur	24	750	1000	200	15	8000	220
Zala	421-08	1,5	15	2	1	1	3600	65
Leonardo	Crex-B	1,2	10	2	1	2	3100	36
Leonardo	AWHERO	6	100	200	60	4	4268	90
Leonardo	Falco EVO	20	200	650	100	13	6000	216
Leonardo	Falco Explorer	24	∞	1300	350	14	3000	216
Lockheed Martin	Stalker	8	222	22	2,5	5	3660	90

Appendix B

Grid Refinement Studies

Table B.1: Results of the grid refinement study

			Error rel. to the grid before [%]				Error rel. to the finer grid [%]				Comp. resources	
Elements	h	r	C_L	C_D	C_{D_v}	C_{D_p}	C_L	C_D	C_{D_v}	C_{D_p}	RAM [GB]	CPU time [s]
7,544,832	0.0051	-	-	-	-	-	-	-	-	-	123.2	43740
4,961,740	0.0059	1.15	-0.09	0.62	0.74	1.20	-0.09	0.62	0.74	1.20	90.3	21343
3,245,472	0.0068	1.15	-0.01	0.89	0.91	0.89	-0.10	1.52	1.65	2.10	59.1	10158
2,156,868	0.0077	1.15	-0.12	1.00	1.01	1.00	-0.22	2.54	2.68	3.12	44.6	5407
1,446,940	0.0088	1.14	-0.11	1.35	1.26	1.38	-0.33	3.92	3.96	4.55	32.6	4299
965,472	0.0101	1.14	-0.06	1.82	1.40	1.99	-0.39	5.81	5.42	6.63	24.2	1673
644,112	0.0116	1.14	-0.03	2.45	1.57	2.80	-0.41	8.40	7.08	9.62	18.6	1114

Table B.2: Results of the domain size study

		Error rel. to the domain after (%)				Error rel. to the larger domain (%)			
Chords	Elements	C_L	C_D	C_{D_v}	C_{D_p}	C_L	C_D	C_{D_v}	C_{D_p}
5	1,289,092	2.07	-4.61	0.17	-6.71	3.26	-7.46	0.22	-10.68
10	1,368,016	0.68	-1.75	0.09	-2.53	1.17	-2.98	0.05	-4.26
15	1,420,632	0.32	-0.81	-0.05	-1.14	0.49	-1.26	-0.04	-1.77
20	1,446,940	0.17	-0.45	0.01	-0.64	0.17	-0.45	0.01	-0.64
25	1,473,248	-	-	-	-	-	-	-	-

Appendix C

Code Implementation

```
def twist(val, geo):
"""This function parameterizes the twist DV
Inputs: val - Design variable array
geo - DVGeo object
"""

  C = geo.extractCoef("quarter_chord")    # Get the coordinates of the ref. axis points

  for i in range(1, nRefAxPts):
    # Get the vector to apply rotation on each section
    if i == 0:  # Wing root (generally fixed)
      rotVec = C[i + 1, :] - C[i, :]      # Get the vector from point i to point i+1
    elif i == nRefAxPts - 1:  # Wing tip
      rotVec = C[i, :] - C[i - 1, :]      # Get the vector from point i-1 to point i
    else:  # Wing middle section
      rotVec = C[i + 1, :] - C[i - 1, :]  # Get the vector from point i-1 to i+1

    unit_rotVec = rotVec / np.linalg.norm(rotVec)  # Transform it to a unit vector

    geo.rot_x["quarter_chord"].coef[i] = val[i - 1] * unit_rotVec[0]   # Twist around
  global x-axis
    geo.rot_y["quarter_chord"].coef[i] = val[i - 1] * unit_rotVec[1]   # Twist around
  global y-axis
    geo.rot_z["quarter_chord"].coef[i] = val[i - 1] * unit_rotVec[2]   # Twist aroung
  global z-axis
```

Listing C.1: Twist parameterization

```
def dihedral(val, geo):
"""This function parameterizes the dihedral DV
Inputs: val - Design variable array
geo - DVGeo object
"""

  C = geo.extractCoef("quarter_chord")    # Get the coordinates of the ref. axis points
  C_orig = copy.deepcopy(C)               # Store the original values

  for i in range(1, nRefAxPts):
    # Compute the angle to rotate the sections
    if i == 0:  # Wing root (generally fixed)
      alpha_0 = np.arctan((C_orig[i + 1, 2] - C_orig[i - 1, 2])/((C_orig[i + 1, 1] - C_orig[i - 1, 1])))            # Initial section orientation
      alpha_1 = np.arctan((C_orig[i + 1, 2] + val[i] - C_orig[i, 2] - 0)/((C_orig[i + 1, 1] - C_orig[i - 1, 1])))   # Section orientation after deformation
    elif i == nRefAxPts - 1:  # Wing tip
      alpha_0 = np.arctan((C_orig[i, 2] - C_orig[i - 1, 2])/((C_orig[i, 1] - C_orig[i - 1, 1])))
      alpha_1 = np.arctan((C_orig[i, 2] + val[i - 1] - C_orig[i - 1, 2] - val[i - 2])/((C_orig[i, 1] - C_orig[i - 1, 1])))
    else:  # Wing middle section
      alpha_0 = np.arctan((C_orig[i + 1, 2] - C_orig[i - 1, 2])/((C_orig[i + 1, 1] - C_orig[i - 1, 1])))
      alpha_1 = np.arctan((C_orig[i + 1, 2] + val[i] - C_orig[i - 1, 2] - val[i - 2])/((C_orig[i + 1, 1] - C_orig[i - 1, 1])))

    alpha_d = (alpha_1 - alpha_0) * 180/np.pi      # Additional rotation needed to go from the initial orientation to the new orientation

    C[i, 2] += val[i - 1]                          # Vertical displacement of the reference axis points
    geo.rot_x["quarter_chord"].coef[i] = alpha_d   # Rotate the section to the correct orientation

  geo.restoreCoef(C, "quarter_chord")
```

Listing C.2: Dihedral parameterization

```
def sweep(val, geo):
"""This function parameterizes the dihedral DV
Inputs: val - Design variable array
geo - DVGeo object
"""

  C = geo.extractCoef("quarter_chord")    # Get the coordinates of the ref. axis points
  C_orig = copy.deepcopy(C)               # Store the original values

  for i in range(1, nRefAxPts):
    # Compute the angle to rotate the sections
    if i == 0:  # Wing root (generally fixed)
      beta_0 = np.arctan((C_orig[i + 1, 0] - C_orig[i - 1, 0])/((C_orig[i + 1, 1] -
    C_orig[i - 1, 1])))             # Initial section orientation
      beta_1 = np.arctan((C_orig[i + 1, 0] + val[i] - C_orig[i, 0] - 0)/((C_orig[i + 1,
    1] - C_orig[i - 1, 1])))    # Section orientation after deformation
    elif i == nRefAxPts - 1:  # Wing tip
      beta_0 = np.arctan((C_orig[i, 0] - C_orig[i - 1, 0])/((C_orig[i, 1] - C_orig[i - 1,
     1])))
      beta_1 = np.arctan((C_orig[i, 0] + val[i - 1] - C_orig[i - 1, 0] - val[i - 2])/((
    C_orig[i, 1] - C_orig[i - 1, 1])))
    else:  # Wing middle section
      beta_0 = np.arctan((C_orig[i + 1, 0] - C_orig[i - 1, 0])/((C_orig[i + 1, 1] -
    C_orig[i - 1, 1])))
      beta_1 = np.arctan((C_orig[i + 1, 0] + val[i] - C_orig[i - 1, 0] - val[i - 2])/((
    C_orig[i + 1, 1] - C_orig[i - 1, 1])))

    beta_d = (beta_1 - beta_0) * 180/np.pi         # Additional rotation needed to go
    from the initial orientation to the new orientation

    C[i, 0] += val[i - 1]                          # Streamwise displacement of the
    reference axis points
    geo.rot_z["quarter_chord"].coef[i] = - beta_d  # Rotate the section  to the correct
    orientation

  geo.restoreCoef(C, "quarter_chord")
```

Listing C.3: Sweep parameterization

```
def chord(val, geo):
"""This function parameterizes the chord DV
Inputs: val - Design variable array
geo - DVGeo object
"""

  C = geo.extractCoef("quarter_chord")    # Get the coordinates of the ref. axis points

  for i in range(0, nRefAxPts):
    # Compute the initial section orientation
    if i == 0:  # Wing root
      alpha_0 = np.arctan((C[i + 1, 2] - C[i, 2])/((C[i + 1, 1] - C[i, 1])))   # Initial
    section orientation
    elif i == nRefAxPts - 1:  # Wing tip
      alpha_0 = np.arctan((C[i, 2] - C[i - 1, 2])/((C[i, 1] - C[i - 1, 1])))
    else:  # Wing middle section
      alpha_0 = np.arctan((C[i + 1, 2] - C[i - 1, 2])/((C[i + 1, 1] - C[i - 1, 1])))

  geo.scale_x["quarter_chord"].coef[i] = val[i]                    # Scale on global x-
    axis direction
  geo.scale_y["quarter_chord"].coef[i] = val[i] * np.sin(alpha_0)  # Scale on global y-
    axis direction
  geo.scale_z["quarter_chord"].coef[i] = val[i] * np.cos(alpha_0)  # Scale on global z-
    axis direction
```

Listing C.4: Chord parameterization