POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Aeronautica

Aerodynamic Optimization based on a Discrete Adjoint Framework and Radial Basis Function Mesh Deformation in SU2

Master thesis of:
Luca ABERGO
Person code: 920826

Supervisor:
Full Prof. Alberto GUARDONE
Co-supervisor:
Dr. Myles MORELLI
Company tutor (Leonardo S.p.A.):
Ing. Riccardo GEMMA

Anno Accademico 2020/2021

Dedicata ai miei due nonni
siete qui con me

Ringraziamenti

Concluso questo percorso è giunto il momento dei ringraziamenti verso coloro che l'hanno reso possibile. Comporta guardarsi indietro e ripensare a questi cinque anni abbondanti al Politecnico, già sale il magone e il timore che questa parte diventi più lunga della tesi stessa.
In primis un doveroso ringraziamento a chi ha reso possibile questa tesi: il Prof. Alberto Guardone e il Dr. Myles Morelli. In questi mesi mi hanno dato fiducia, sfidato, stimolato, trasmesso una goccia delle loro competenze, facendomi sentire questo lavoro veramente mio e portandomi a raggiungere piccoli traguardi, come l'AIAA Aviation o la collaborazione con Leonardo Velivoli, che mai inizialmente avrei sperato. Un ringraziamento all'Ing. Gemma di Leonardo Velivoli per aver condiviso la sua esperienza nell'ambito della progettazione aerodinamica, fornendomi le geometrie di veri aerei progettati da Leonardo, che sono state per me fonte di stimolo, crescita e un po' di orgoglio.
Il più grande dei ringraziamenti va però a Mamma e Papà, per me vero riferimento e modello di vita che spero un giorno di emulare. L'educazione, i valori e la cultura del lavoro che mi avete trasmesso sono stati il fondamento di questo percorso e di ogni mio passo avanti. Mi avete dato tutto per poter affrontare il Poli in serenità, spianandomi la strada ma senza mai impormi quale fosse. Mi avete incoraggiato e sostenuto, anche nelle mie attività extra, anche se mi allontavano dai miei studi, facendomi però notare quando esageravo.
Un grazie importante a Chiara, mia sorella, coinquilina, talvolta migliore amica, talvolta... mamma mia quanto ti vorrei strozzare. Spesso non riesco a trasmettere quanto tu sia fondamentale per me e quanto ti ammiri, non c'è e non ci sarà momento importante in cui io non ti vorrò al mio affianco. Un grazie ai miei parenti, in particolare a zia Carla, zio Lino e nonna Carla, per l'affetto mai mancato, anche da lontato. Mi avete trasmetto i valori dell'umiltà, dell'altruismo, ad aprezzare le cose semplici e che la felicità non è per forza lontano da casa ma nelle persone di cui ci circondiamo.
Ai miei due nonni, entrambi mi hanno salutato durante questa tesi. Due persone molto diverse tra loro, ma entrambi perfezionisti e gran lavoratori. Ti trasmettevano quella sensazione di sicurezza in tutto ciò che facevano e ti trasmettevano davvero qualcosa ogni volta che parlavano. Questa tesi è vostra e sarete sempre con me.
In questi anni ho incontrato davvero troppe persone e stretto amicizie, alcune intense ma passeggere, altre forti e spero durature. Sono stato molto lontano da Genova, eppure di qualcuno proprio non sono riuscito a liberarmi. Tipo Michi, Eli, Pampo, Deffe (TheLucas), semplicemente ci siete costantemente, da sempre e spero per sempre. Possiamo non vederci per mesi ma resta tutto uguale, oppure vederci ogni giorno e la voglia di stare assieme non diminuisce mai. Non so quanti miei momenti di stress vi siete sopportati e ogni vostra visita a Milano rimarrà scritta negli annali. A Francesco, Stefano, Andrea, coach fede, al cnuc e alla vela in generale, son cresciuto con voi, diventato più forte di fisico e di testa, imparato ad apprezzare la tensione pre-regata, ad allenarsi sempre anche quando si gela o c'è tempesta, a non mollare fino alla linea dell'arrivo dell'ultima prova. Tantissime prime volte sono state con voi e di certo non possono comparire in una tesi ma sicuramente siamo cresciuti assieme durante i nostri lunghissimi viaggi per l'Europa con le barche dietro sul carrello. Un ringraziamento speciale va ad Aurora, sei stata per quasi tutto questo viaggio la mia marcia in più, hai reso felici tantissime giornate di puro studio, non sarei qui con questi risultati e probabilmente non sarei neanche questa persona senza te.
Poi ci sono le persone che sono nella mia vita relativamente da poco, vera eredità di questi anni a Milano.
Fra, Ale e Barte, non c'è esame o progetto che non abbiamo preparato assieme, in certi momenti senza dubbio le persone con cui ho trascorso più tempo e se il Poli è stato così bello è tanto tanto merito vostro.
Un grazie ad Adriana, Eleonora, Giampi, Giuse, Titti e al resto del gruppo, avete reso speciali, sop-

portabili e piene di risate le infinite giornate in aula, amicizie ben oltre il semplice studiare assieme che spero di non perdere e coltivare. Impossibile non citare Tami, Iva, Jordi, e tutti quelli che sono passati in viale Evaristo Stefini 2. Auguro a qualsiasi fuori sede di avere l'immensa fortuna che ho avuto io di trovare una casa del genere e dei coinquilini così pazzi Le parole noia e monotonia erano bandite eppure in quella continua aria di festa e via vai di gente ho trovato il perfetto mix di studio e svago.
Per ultimo, ma solo per ordine cronologico, un mega grazie a Tati, Ira, Giulia, Iacopo, Belen, Flavia, tutto Quarantasmus (eravamo troppi per nominarvi tutti) per aver condiviso con me questo folle Erasmus in piena pandemia, avrebbe potuto essere un disastro invece è stata un'avventura indimenticabile. Grazie a voi ho riscoperto la bellezza di vivere in modi diversi dal mio e la voglia di avvicinarmi a nuove culture. Con voi mi sono innamorato del Portogallo e di Lisbona, dove ho conosciuto la sorpresa più bella e inaspettata di questi mesi, che ha reso imprevedibili ed emozionanti questi mesi di tesi e che mi spinge a migliorarmi senza neanche accorgesene, Giorgia.

Abstract

Automatic shape optimization (ASO), based on gradient method and computational fluid dynamic (CFD), is becoming a powerful tool for aircraft design. This framework is turning to be increasingly popular due to its ability to improve the performance and the efficiency of lifting surfaces such as airfoils, wings and rotors. Discrete Adjoint is used to compute a high fidelity gradient of a selected objective function, usually an aerodynamic coefficient, with respect to some design variables that describe and control the body's geometry. In the open-source SU2 software, to contain the elevated computational cost of a design loop, the adjoint is implemented taking advantage of Automatic Differentiation with operator overloading and expression templates. The cost of computing the sensitivity does not scale with the number of design variables selected however with the number of targets and aerodynamic constraints. The research of the optimal shape, guided by the Sequential Least Square algorithm, can be obtained properly morphing at each step the original body to close the distance with the optimum. However, the result obtained is always going to be in the neighbour of the starting point and dependent on it, thus a local minimum. The deformation is guided by the sensitivity of an objective function with respect to some design variables. Usually, it is required to minimize the drag without decreasing the lift and not influence the momentum applied to the rigid body. The extension of the design space explored is strongly influenced by the type of parametrization of the body surface and the method used to update the computational grid. The number of design variables and constraints needed to obtain a proper improvement of the aerodynamic performance must be investigated for each case and the results are often counter-intuitive.

In this thesis a Radial Basis Function (RBF) mesh deformation is introduced into the discrete adjoint framework within the open-source toolkit SU2. The RBF mesh deformation technique allows to handle more complex geometries than the more standard elastic deformation approach, while enabling larger movements to expand the design space. Data reduction schemes including multilevel greedy algorithms are used to improve the computational efficiency of RBF mesh deformation on large data sets. Numerical experiments show a significant reduction of physical memory usage and cpu time over the linear elasticity analogy both for two-dimensional cases and for large, three-dimensional problems. Additionally, the mesh deformation process is differentiated by Automatic Differentiation within the discrete adjoint approach, resulting in method-dependent sensitivity of the design variables, thus allowing the Sequential Least Squares Programming optimizer to converge to a new local minimum by modifying the geometrical shape towards the final design.

The ASO implemented in SU2 is tested on a wide range of 2D and 3D test cases, subsonic and transonic. Some benchmark test cased are proposed to the CFD community by AIAA Aerodynamic Design Optimization Group (ADODG), including the the Common Research Model (CRM) wing and the RAE2822 airfoil. Furthermore, it is applied to a subsonic regional aircraft wing, which has been provided by Leonardo Aeronautics Division, the Italian leader company in aerospace field. The ultimate goal of optimizing a non-planar wing, specifically a wing-winglet configuration, is achieved proving the flexibility and robustness reached by the software.

Sommario

L'ottimizzazione automatica delle superfici di un corpo aerodinamico, basata su metodi al gradiente e sulla fluidodinamica computazionale, sta diventanto un potente e sempre più popolare strumento a disposizione dell'ingegnere durante la progettazione di un aereo. La sua popolarità è in costante aumento per la capacità di incrementare le performance e l'efficienza di superfici portanti quali profili, intere ali e pale rotanti. L'aggiunto discreto viene utilizzato per calcolare con elevata precisione il gradiente di una certa funzione obiettivo selezionata, rispetto a un vettore di variabili che controllano e descrivono la geometria del corpo. All'interno del software SU2 con licenza open, per ridurre l'elevato costo computazionale, l'aggiunto è implementato utilizzando la differenziazione automatica con tecniche quali "expression temple" e "operator overloading". Il costo legato al calcolo della sensitività non scala rispetto al numero di variabili di controllo ma con la quantità di obiettivi selezionati e vincoli aerodinamici imposti. La ricerca della miglior forma, eseguita usando SLSQP come algoritmo, è ottenuta deformando gradualmente ad ogni ciclo il corpo iniziale, avvicinandosi sempre di più all'ottimo. Tuttavia, il minimo raggiunto sarà sempre vicino al punto iniziale e dipendente da esso, individuando così un minimo locale. La dimensione dello spazio delle configurazioni possibili esplorate dipende fortemente dal tipo di parametrizzazione scelta per il corpo e dal metodo usato per deformare la griglia computazionale. Il numero di variabili di controllo e vincoli geometrici necessari per ottenere un adeguato miglioramento delle performance aerodinamiche va investigato per ogni applicazione e risulta spesso controintuitivo.

In questa tesi il metodo di deformazione della mesh chiamato "Radial Basis Function" (RBF) è integrato in SU2 con l'aggiunto discreto. RBF, rispetto al più comune metodo basato sull'analogia elastica, garantisce la capacità di gestire geometrie più complesse e applicare deformazioni maggiori aumentando il numero di forme investigabili. Per migliorare l'efficienza computazionale di RBF, soprattutto quando usato con mesh di grosse dimensioni, vengono applicati dei metodi di riduzione dei dati come "greedy algorithm" e riduzione dei nodi del dominio computazionale spostati. Le simulazione numeriche evidenziano una importante riduzione della ram usata e tempo cpu impiegato sia in casi bidimensionali ma soprattutto nel 3D. All'interno del calcolo delle variabili aggiunte, il processo di deformazione viene differenziato influenzando il valore della sensibilità ottenuto, questo può portare il metodo di ottimizzazione basato sul gradiente a convergere in un minimo locale differente.

Il processo di ottimizzazione implementato in SU2 viene testato su un ampio numero di casi 2D e 3D, subsonici e transonici. Alcuni, come l'ala CRM e il profilo RAE2822, sono stati proposti alla communità CFD dal gruppo ADODG dell'AIAA, in modo da avere dei casi di riferimento comune su cui confrontarsi. Inoltre è stata ottimizzata un'ala di un veivolo da trasporto regionale in regime subsonico, progettata e fornita dalla divisione veivoli di Leonardo, azienda aeronautica leader in Italia. L'obbiettivo ultimo di migliorare le prestazioni di un'ala non planare, cioè provvista di winglet, è stato raggiunto dimostrando la robustezza e la flessibilità del software.

Contents

List of Figures

*

List of Tables

*

Chapter 1

Introduction

Automatic Aerodynamic Optimization based on Computational Fluid Dynamics (CFD) is a powerful tool, able to improve the aero-performance, which has become a key step of a general aerodynamic design process. It is applied to a wide range of aerodynamic lifting surfaces including airfoils, wings and rotors. These aerodynamic lifting surfaces are highly sensitive to geometric modifications and even slight changes in the shape can have a significant influence on the final design's performance. The high number of scientific papers about this topic highlights the sharp and increasing interest of the CFD community and the aerospace industry. In fact, a recent review of the state of the art regarding shape optimization [2] identifies a total of 304 remarkable papers, which appeared in more than 120 conferences and journals. In the last forty years, researchers have focused efforts towards making shape optimization reliable and robust, although many improvements are possible and necessary, especially concerning multi-disciplinary design (MDO) and multi-objective optimization (MO).
As announced by the agenda of the Advisory Council for Aeronautics Research in Europe, the target is to reduce by 75% the CO_2 emissions per passenger kilometre and the noise by 65%. The airframe contribution is estimated around 20%, besides automatic shape optimization (ASO) could play a key role.
Optimization is intended as the search of the best configuration where, especially concerning gradient-based ASO, the shape obtained is not a global minimum although is clearly superior than the original one. The initial point, the type and the number of constraints, and the dimension of the design space strongly affect the final result. For instance, Chernukin and Zingg [9] generated 224 random initial shapes of a blended wing with a fixed number of DVs and they obtained 8 local optima. Every solution satisfies the optimal condition with a 5% range among the objective functions obtained. In contrast to what could be expected, increasing the Design Variables (DVs) or reducing the constraints does not always improve the performance of the optimizer [10].
The optimization chain is really a secession of multiple steps interconnected, the complexity is due to the fact that many different tools, sub-problems and mathematical aspects are involved, often they are an open and active field of research. Shahpar from Rolls Royce [11] identifies and describes seven modulus, each one has clear inputs, a process and outputs, and he calls them the "optimization seven pillar". They are geometry parametrization, mesh generation and deformation, flow solver, optimization algorithms, postprocessing, workflow management, and last IT issues. They are all connected and affect each others thus a unique and consistent strategy is needed. The developer must have an overall view of the context, from the mathematical aspects to programming challenges. This thesis briefly introduce each modulus and presents more in details the Discrete Adjoint and Mesh Deformation methods.

1.1 Sensitivity Analysis

In the early 1980's, the CFD community began to include in their codes the capability to perform a sensitivity analysis [12], which consist in the evaluation of the first derivatives of an aerodynamic quantity, which has a clear dependence both on geometry and the flow, with respect to some design variables. The DVs are the control parameters of functions that describes the surface of the body. Originally, the gradients of interest were calculated by Finite Differences, which is the simplest method. Firstly introduced by Reneaux and Thibert[13], it requires no modification of the solver itself, however the computational cost scales up with the number of design parameters. A more advanced and sophisticated method partially solving some issues linked to FD was proposed by Lyness and Morelli [14], named complex step FD. However, the turning point was the application of optimal control theory to incompressible flow equations, introduced by Pironneau [15]. Later, only with the famous article of Jameson [16] (1988) in which the continuous adjoint method for aerodynamics was introduced, shape optimization became attractive and affordable. For the first time it was possible to compute the objective sensitivity with respect to design variables at the cost scaling with the number of objective functions. The complexity of the applications progressed from 2D airfoil [17] to 3D wings [18] and finally the optimisation of a complete aircraft. The continuous adjoint approach theory derives from a natural extension of the linear algebraic duality theory to the partial differential equations (PDE). To the variables space containing the flow state the duality variables vector is added, named in this context as adjoint variables. It requires a deep manipulation of the flow equations and the boundary conditions, only at the end the equations are discretized to perform the numerical solution. The complexity of the mathematical formulation of the adjoint and the corresponding boundary conditions was partially overtaken by Shubin and Frank[19]. They created a discrete version of Jameson's adjoint method, that they named "implicit gradient approach" but, soon after, it was called "discrete adjoint method". In this latter approach, the control theory is applied directly to the set of discrete flow equations. The final system can be found also applying the duality theory, which seems more natural for some uses of the adjoint variables such as error analysis where constrains are not involved.
This was the beginning of an endless debate between who is in favor of the Jameson's method and those in favor of the discrete formulation. They represent two very different ways of reaching almost the same result. Many authors have dealt with the question which one is better by comparing different codes, notably are the studies of Giles and Pierce [1] and Nadarajah and Jameson [20].
If the discrete adjoint equation is solved exactly, then the resulting values of the Lagrange multipliers generate a perfect gradient of an inexact cost function and the derivatives are fully consistent with complex-step gradients independent of the mesh size. This guarantee that the optimisation process can converge. Meanwhile the accuracy of the continuous adjoint increases as the mesh is refined, although there is a slight inconsistency between the discrete objective functions and the computed gradient. As result, the optimisation process is more likely to get stuck in a local minimum. Indeed, the continuous approach yields to a discrete approximation of the gradient of the analytic objective function with respect to each of the DVs. It is not perfectly identical to the gradient of a discrete approximation of the objective function. The implementation of the continuous adjoint is simpler, quicker and requires way less virtual memory. There is total freedom on how to discretize the adjoint PDE, the choice is not dictated by the primal flow discretisation since the solution of the adjoint system is consecutive to the direct simulation and only the converged flow state is shared. In the context of the analytic formulation, while studying a transonic Euler flow, the shocks have to be treated as discontinuities across which the Rankine-Hugoniot shock jump relations are enforced. An additional boundary condition must be imposed along the shock, requiring an automatic identification of the shock position, which is still an hard task to accomplish. In practice, codes that use the continuous adjoint do not enforce this

extra condition since it has been proven that the quality of the result is not affected [21]. Last, it does not exist a continuous adjoint formulation for integral quantities defined inside the fluid domain since the term of the flow sensitivity can not be cancelled along the integration contour of the farfield.
Concerning the open-source multiphysics solver SU2 [22], the discrete adjoint code is implemented by Albring, Saugeman and Gauge [4]. The choice is driven by the capability to use Automatic Differentiation to calculate the numerous Jacobians involved taking advantage of an external library named CodiPack [23]. Even if the exact Jacobian, obtained by manually differentiate each terms were available, it is often ill-conditioned avoiding the convergence of the solution. Software structure is conceptually straightforward with the combination of automatic differentiation (AD) and discrete adjoint. However, it is significantly more complex from the coding point of view. One of the major advantages is the possibility to easily introduce different turbulence models and apply the adjoint also to multiphysics cases such as fluid-structure interaction (FSI) or noise reduction [24]. [25].

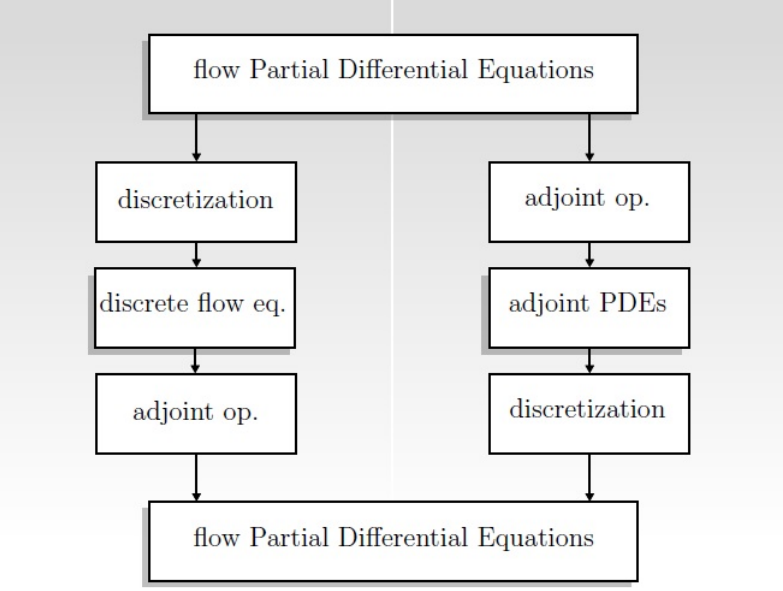

Figure 1.1: Differences Between Discrete and Continuous Adjoint Solvers, from [1]

1.2 Optimization Chain

The exploration of the design space generating different shapes is entrust to a gradient-based optimization algorithm based on the sensitivity computed with the adjoint. This natural choice has attached some limitations. It is only applicable when the value of the design variables is continuous and in the case that the objective function has multiple minimum, then the gradient approach will likely converge to the nearest local minimum, where “nearest” is intended respect to the starting point. This means that the software is not able to drastically change the shape of a body. It is a powerful tool in the last steps of a design process, when the general characteristics of the object are defined, the performances are already sufficiently close to the standard required, and it is necessary to further improve the aerodynamic coefficient in a way that is manually difficult to achieve. Therefore, not the entire designed space, just a limited area around the starting point is explored properly. An error in just one step of the optimization loop, due to a lost of mesh quality during the deformation process or a Jacobian ill-conditioned, would lead to a strong interruption of the optimisation chain. If a genetic algorithm would be applied, a flow

code failure would be treated as one member of the population not evaluated. Although, the choice of a genetic algorithm (GA) limits the number of design variables to less than ten, while regarding 3D bodies a sufficient accurate description of a surface requires hundreds of design variables. Moreover, the convergence of gradient based codes is significantly improved with respect to genetic algorithm codes. The open-source suite SU2 performs optimisation using the Sequential Least Squares Programming (SLSQP) [26], the process ends when the convergence criteria, the Karush-Kuhn-Tucker (KKT) conditions are satisfied [27] or the maximum number of design iterations is reached.
Once the surface sensitivity is obtained by the adjoint method, in order to explore the entire design field the software must have the capability to morph the body and update the mesh. Firstly, the wall boundaries need to be mathematically described. Shape parameterisation techniques can be divided into eight categories [28]: free-form deformation, partial differential equations, polynomials and splines, basis vector, domain element discrete or analytic, and computer-aided design (CAD). Only FFD and CAD among all these are efficient, compact and suitable for complex configuration. CAD is also able to describe large geometry deformation, commercial codes are based on FBSM functions to generate the parametric description. Existing feature-based solid modeling concept are not able to calculate sensitivity derivatives analytically so, up to now, they are not suitable for automatic shape optimisation. Instead, FFD is a subset of the Soft object animation codes adopted in computer graphics for morphing images. In this thesis it is extensively applied for its ability to handle arbitrary geometry and as it does not require the grid connectivity. FFD directly parametrize the nodes locations, meaning that it does not need the abstraction of the geometry, although is able to handle only small or medium geometry changes. The basic idea is to embed the body and the mesh inside a box of flexible plastic then, as the block is deformed, the grid is consistently shifted [29]. The box, which can have a cuboid, cylindrical or spherical shape, can be segmented in an arbitrary number of sub-modulus which vertices are shifted for beginning the deformation process.
The surface grid displacement is extracted with a local interpolation inside the FFD box, then the volume mesh is update to the new boundaries for the subsequent simulation. Concerning three-dimensional problems, remeshing the entire geometry has been adopted in the past as in the NERONE project [30]. Automatic mesh generation is time and CPU consuming moreover, a proper mathematical description of the object is required. Each surface has to be split in regions by segments of Bspline curves, for non-planar geometries this is still a hard task. In the case of 3D viscous flow, an hybrid mesh able to capture the boundary layer and respect the turbulence model requirement of $y+ < 1$ is needed, therefore the human intervention is currently unavoidable. Moreover, mapping the solution from the previous mesh to the new one consumes extra CPU effort and time. Consequently, remeshing is undesirable and the approach is not considered within this thesis. The mesh deformation option is for now more reliable and intensively investigated by the CFD community. Many methodologies have been developed, since mesh deformation is also used in the context of unsteady simulations with bodies moving in relative motion. They differentiate from each other for the robustness, the capability to obtain large deformation, to handle an arbitrary type of grid and the requirement of the mesh connectivity. Handling structured grid is way easier than unstructured and dealing with small deformation is simpler than large deformation. Talking about deformation, what is relevant is the entity of the local displacement, the relative movement between two adjacent nodes, not the entire body distortion such as the inflexion of the tip of a wing respect to the root. The techniques for modifying the fluid mesh matching the new solid body can be split into two main classes: physical analogy and interpolation technique [31]. The first one creates a similarity with a physical process that can be modeled using numerical methods, the spring analogy introduced by Batina [32] and linear elastic equations first presented by Baker and Cavallo [33] belong to this category. The most relevant drawback of physical analogy methods is that they generate a large system of equations, implying a higher computational cost. They also

are more difficult to parallelize. Mesh deformation using interpolation does not need the connectivity information, therefore can be apply to any kind of mesh that contain general polyhedral elements or hanging nodes [34]. The two main techniques are Inverse Distance Weighting methods (IDW) and Radial Basis Function (RBF). Memory requirement is clearly less compared to physical schemes, however the coding complexity is increased and an intrinsic error due to interpolation is introduced.

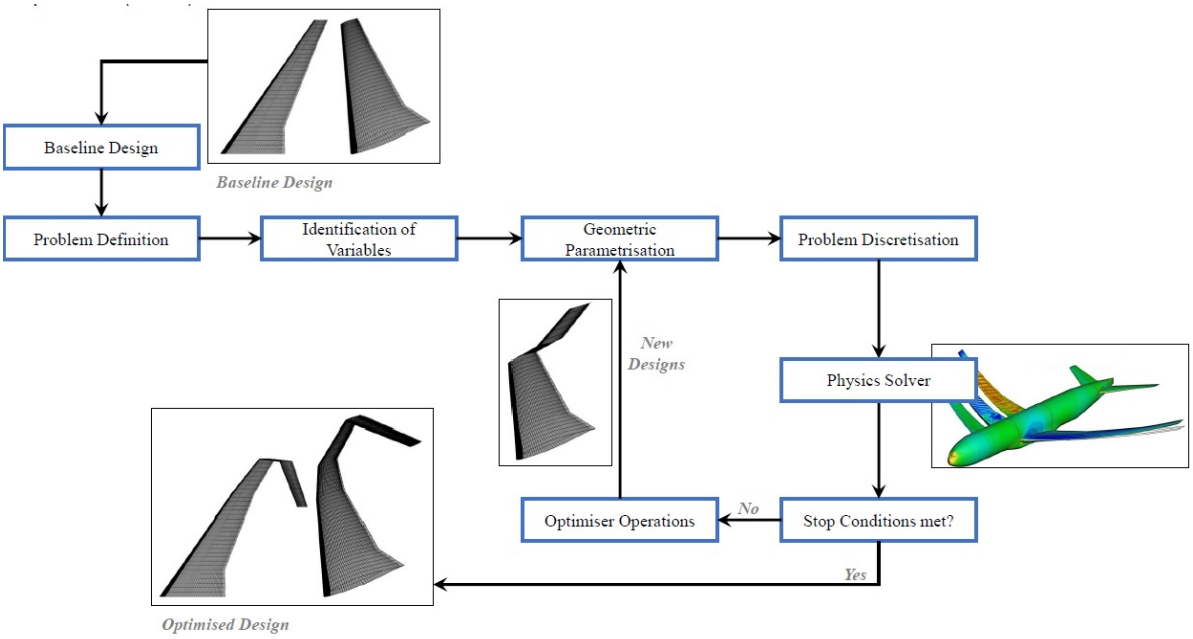

Figure 1.2: Automatic Shape Optimization ASO, from [2]

1.3 Research Objective and Contribution

Inside the optimization chain, the mesh deformation module has been identified by the author of this thesis a crucial and critical module. The routine to update the mesh is differentiated inside the adjoint computation accordingly it directly influences the computation of sensitivity of design variables. Besides, the possibility of the mesh deformation techniques of handling arbitrary geometry and apply significant displacements to the nodes strongly impacts the possibility of properly explore the design space. A loss of quality properties in the output grid could cause the divergence of the RANS or of the adjoint simulation. Moreover, the computational cost, especially the ram consumption of this step is not irrelevant, sometimes even higher than during the computation of the sensitivity.
The key contribution from this thesis is the introduction of Radial Basis Function mesh deformation into a gradient based discrete adjoint design optimization framework. Radial Basis Function mesh deformation has successfully been used in the past as part of design optimization frameworks and its has been already highlighted [35]. However the significant computational cost of standard Radial Basis Function mesh deformation has meant alternative mesh deformation techniques have been preferable for three-dimensional problems. Moreover Radial Basis Function mesh deformation is yet to be used alongside a discrete adjoint framework. This thesis seeks to address this and investigates the benefits of Radial Basis Function compared to standard mesh deformation techniques in a gradient based discrete adjoint framework. This work harnesses state-of-the-art data reduction schemes to address the major weakness of radial basis function mesh deformation for large-scale three-dimensional problems.

The robustness and high mesh quality properties of radial basis function mesh deformation allow for the design optimization of topologically complex geometries including transonic wings and winglets. Therefore the Radial Basis Function mesh deformation method implemented in SU2 [36], has been introduced into with the discrete adjoint framework. The target is to obtain a process of optimization that is able to handle more complex geometries, while enabling larger deformations to expand the design field automatically explored. This should make SU2 automatic shape chain easier and more attractive for industrial applications. To further increase the robustness it has been modified the way with the DVs sensitivity are scaled at each design loop, assuring that the maximum surface displacements is contained inside the range $[0.5mm; 1cm]$. Moreover, it is now possible to apply geometrical constraints also on planes with norm in z direction, which is essential for the optimization of a non-planar wing.
The new optimization chain is compared with a shape optimization process using a linear elastic analogy as mesh deformation method. The comparison is made with the 2D and 3d wing benchmark optimization problems proposed to the CFD community by AIAA Aerodynamic Design Optimization Group (ADODG). Every test case is performed both with RBF and ELA for updating the grid, the results are compared also in terms of virtual memory consumption. Thanks to Leonardo Aeronautics Division which provided the CAD of real subsonic wing of a regional airplane, the ASO robustness is evaluated also on an industrial case. The final goal is to optimize a non-planar wing. The optimization of a combined wing-winglet is primary executed using wind tunnel experiments [37], zero gradient method [38] or codes based on vortex lattice methods [39].

1.4 Thesis Outline

The thesis is structured as follows: Chap. 2 describes the numerical methods used for compute the sensitivity including the discrete adjoint approach and how to solve the system obtained. Chap. 3 explains how the automatic shape optimization is implemented in SU2 and how the required Jacobian are computed using Automatic Differentiation. Besides, the same chapter contains the gradient-based optimization algorithms description and the mathematical formulation of the geometrical parametrization method FFD. In the Chap. 4 interpolation technique and physical analogy for mesh deformation are exploited with a deep focus on RBF. Chap. 5 and Chap. 6 show the results of the 2D and 3D test cases. Finally, Chap. 7 discusses the main findings from the work done and the perspectives moving forward.

Chapter 2

Sensitivity Computation

With Automatic Shape Optimization by mean of Computational Fluid Dynamics, it indicates a process whose target is to increase the aero-performance of a generic solid modifying the external part of a body in contact with the fluid. First of all, a mathematical description of the wet surface is needed, generating some free and continuous parameters that directly manage the shape. The object is immersed in a flow volume that has to be discretized by a grid. The mesh itself has a clear dependence on the wall surface. The design parameters are collected in a vector denoted α with size n_α that belong to an abstract domain D_α which represents all the infinity possible shapes of the solid. The volume mesh is indicated with X_{vol} of size n_v and the surface grid X_{surf} of size n_s. The mutual connection can be underlined writing $X = X_{vol}(X_{surf}(\alpha))$ which is supposed to be at least continuously differentiable. Selecting a specific objective function J, such as an aerodynamic coefficient as C_d or torque moment, the minimization problem can be written as:

$$\begin{aligned} \min_{\alpha} \quad & J(U(\alpha), X(\alpha)) \\ \text{subject to} \quad & R(U(\alpha), X(\alpha)) = 0 \end{aligned} \tag{2.1}$$

The state variables U of a steady flow solution are computed using a finite volume scheme that discretize the compressible RANS equations. Selecting a specific numerical flux including Jameson-Schmidt-Turkel (JST)[40] or Roe approximate Riemann solver [41] and a slope limiter that enables second-order space integration, the flow residual $R(U)$ is obtained. The optimization problem has a steady state constraint that is expressed by equation 2.1. In every cell where the RANS calculation is performed, the flow is assumed to be exactly converged:

$$R(U_i, X_i) = 0 \quad \text{and} \quad det(\frac{\partial R}{\partial W})[U_i, X_i] \neq 0 \tag{2.2}$$

From this assumption derives the practical necessity to obtain a very small final residual during the iterative solution of the flow equations. The existence of an open set D_X containing X_i and C^1 unique function U is certificated by the implicit function theorem, such that:

$$U(X_i) = U_i \quad R(U(X), X) = 0 \quad \forall X \in D_X \tag{2.3}$$

U is assumed to be continuous and differentiable respect to the design parameters α all over D_α. Since the Navier-Stokes equations are solved, U can be split in two vectors, pure flow variables U_f and usually one or two terms linked to the turbulence model adopted, accordingly we have:

$$U := \begin{pmatrix} U_f \\ U_t \end{pmatrix} \quad R(U_f, U_T) = \begin{pmatrix} R_f(U_f, U_t) \\ R_t(U_f, U_t) \end{pmatrix} \tag{2.4}$$

The research inside D_α of the best shape is driven by a gradient-based algorithms, therefore the sensitivity of the functional respect to the design variables is required. In general J can be composed by n_f functions denoted as J_k, each one has to be defined and differentiated w.r.t the design parameters. The dependence among the objective functions, α and the mesh can be clarified by:

$$J_k(\alpha) = J_k(U(\alpha), X(\alpha)) \quad k \in \{1 \dots n_f\} \tag{2.5}$$

Originally, the goal was to calculate $\frac{\partial J_k(\alpha)}{\partial \alpha_i}$ which are $n_f \times n_\alpha$ derivatives. Instead, more recent codes compute the total derivative $\frac{dJ_k}{dX}$ with $k \in \{1 \dots n_f\}$ that incorporate the direct influence of the nodes positions and the indirect influence caused by a change in the flow field to reach steady state convergence. In this chapter different methods to compute the tangent are reported in chronological order, starting from Finite Difference in Sec. 2.1. A deep focus is posed on the discrete adjoint in Sec. **??**, which method is implemented in SU2 and used in this thesis inside the optimization chain. Moreover, the approximations introduced with the adjoint are underlined in Sec. 2.7 and an attempt to give a physical interpretation and a different use of the adjoint variables are described in Sec. 2.6 and Sec. 2.8.

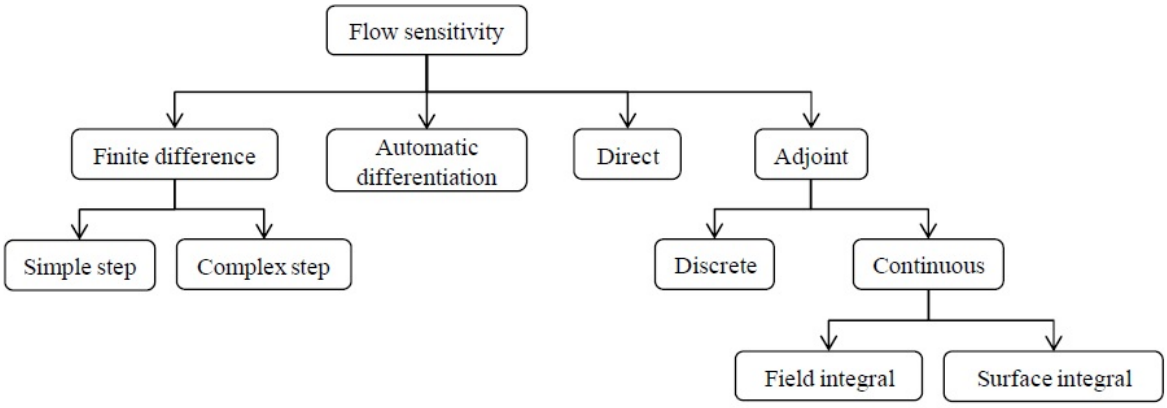

Figure 2.1: Framework for Computing the Surface Sensitivity

2.1 Finite Difference

Finite Difference is the simplest and oldest method for obtaining the object function gradient since it does not require any modification of the solver. It was used in the optimization scheme for airfoils by Hicks [42] and Vanderplaats [43]. The discrete flow solution has to be calculated around the foil defined by α and also for perturbed $\alpha + \delta\alpha$ and $\alpha - \delta\alpha$. Typically, a second-order finite difference is used. If the body is described using an FFD box, which method is described in Chap. 3, $\delta\alpha_l$ with $l \in (1 \dots n_\alpha)$ means a shift of a vertex, marked as design variable, in the direction l of a quantity δ_α. Two mesh deformations, $X(\alpha + \delta\alpha_l)$ and $X(\alpha - \delta\alpha_l)$, need to be performed and two flow solutions computed on the morphed grids, satisfying:

$$R(U(\alpha - \delta\alpha), X(\alpha - \delta\alpha)) = 0, \quad R(U(\alpha + \delta\alpha), X(\alpha + \delta\alpha)) = 0 \tag{2.6}$$

The method is impractical for a large number of design variables since the matrix $\frac{dJ(\alpha)}{d\alpha}$ is computed at the cost of $2 \times n_\alpha$ times the cost of a single flow solution. The single component of the matrix is the approximation of the derivative in the direction $\delta\alpha$ by a centered finite difference formula:

$$\frac{dJ_k(\alpha)}{d\alpha_l}\delta\alpha_l \approx \frac{1}{2}[J_k(U(\alpha + \delta\alpha), X(\alpha + \delta\alpha)) - J_k(U(\alpha - \delta\alpha), X(\alpha - \delta\alpha))] \quad \forall k \in \{1 \dots n_f\} \tag{2.7}$$

One critical point is how to set the step size $\| \delta\alpha \|$. The choice strongly influences the accuracy of the gradient computed. If the step is too small, the rounding error becomes important otherwise, if too large, the truncation of the higher orders in the Taylor expansion operated is not valid anymore. If ε is the machine error, which is the upper bound of the relative error due to rounding in floating point operations, then:

$$\| \delta\alpha \| >> \varepsilon \| \alpha \| \quad \| \delta\alpha \| >> \varepsilon \frac{|J(\alpha)|}{|\frac{dJ(\alpha)}{d\alpha}|} \tag{2.8}$$

The ideal step can be found only through a time-consuming parametric study. Since J has to be evaluated two times, the two simulations must converge in the same number of iterations on two very similar meshes with almost the same quality properties. In case of complicated configurations also, the presence of multiple solutions represents a serious issue [44]. To address this issue, a complex step finite difference method was introduced. Originally developed by Lyness and Moler in 1967 [14], they introduced a reliable way to compute the n^{th} derivative of an analytic function. The potential for shape optimization was fully understood by Anderson in 1999 [45]. This approach is now used as a method of validation for discrete adjoint approach [46]. Consider a function, $f = u + iv$, of complex variable $z = x + iy$. If f is differentiable in the complex plane, the Cauchy-Riemann equations apply:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \qquad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}. \tag{2.9}$$

Once a real h quantity is chosen, the right hand side of the first equation can be modified using the definition of the derivative.

$$\frac{\partial u}{\partial x} = \lim_{h \to 0} \frac{v(x + i(y+h)) - v(x+iy)}{h} \tag{2.10}$$

Since the function of interest is of real variables imposing $y = 0$, $v(x) = 0$, $u(x) = f$ and a small enough h the previous equation can be simplified in:

$$\frac{\partial f}{\partial x} \approx \frac{\mathrm{Im}[f(x+ih)]}{h} \tag{2.11}$$

This method is not subject to subtractive cancellation errors which is advantageous. Furthermore, h can be chosen close to ε machine [47].

2.2 Direct Linearization

Discrete adjoint equations can be obtained in several different ways. Two approaches widely used in literature are discussed in the following sections. The fundamental assumption for all the theories is that the computed flow state U^* is a fully converged solution of the discretized flow equations, given in residual form as:

$$R(U^*(\alpha), X(\alpha)) = 0 \tag{2.12}$$

$J = J(U^*(X(\alpha)), X(\alpha))$ is the objective function that has to be minimized. Design variables do not appear explicitly. However, it can be noticed that the flow residual and the objective function depend on the flow variables at the grid point position, which in turn are linked to the shape of the body, thus

to the vector, α. The gradient of J respect to α is obtained by straight forward differentiation. The superscript * indicating the converged flow variables has been dropped for clarity of the reader:

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = \frac{\partial J}{\partial X}\frac{\partial X}{\partial \alpha} + \frac{\partial J}{\partial U}\frac{\partial U}{\partial X}\frac{\partial X}{\partial \alpha} \tag{2.13}$$

The slope of the flow variables with respect to the grid coordinates can be extracted by differentiating Eq. 2.13, which yields:

$$\left[\frac{\partial R}{\partial U}\right]\frac{\partial U}{\partial X} = -\frac{\partial R}{\partial X} \tag{2.14}$$

Substituting Eq. 2.13 in the Eq. 2.14 it is obtained:

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = \left[\frac{\partial J}{\partial X} - \frac{\partial J}{\partial U}\left[\frac{\partial R}{\partial U}\right]^{-1}\frac{\partial R}{\partial X}\right]\frac{\partial X}{\partial \alpha} \tag{2.15}$$

Thus with a sequence of matrix-vector multiplications the sensitivity of the objective function with respect to the design variables is obtained. The Jacobian of the governing flow equations has to be separately computed and stored, further also the mesh sensitivity has to be available. The Jacobian matrix is in general sparse, instead the inverse matrix is dense and in general not feasible to be computed. Further, it is memory consuming. For each variable the product:

$$\left[\frac{\partial R}{\partial U}\right]^{-1}\frac{\partial R}{\partial X}\frac{\partial X}{\partial \alpha} \tag{2.16}$$

can be also produced by solving the system of equations:

$$\left[\frac{\partial R}{\partial U}\right]\frac{\partial U}{\partial \alpha} = -\frac{\partial R}{\partial \alpha} \tag{2.17}$$

The adjoint method solves the issue of evaluating numerous linearized flow solutions for a large number of design parameters by computing at the beginning the product:

$$\frac{\partial J}{\partial U}\left[\frac{\partial R}{\partial U}\right]^{-1} = \Lambda^T \tag{2.18}$$

and inserting Eq. 2.18 in Eq. 2.15. Finally, rearranging Eq. 2.18 the Adjoint Equation is obtained:

$$\left[\frac{\partial R}{\partial U}\right]^T \Lambda = \left[\frac{\partial J}{\partial U}\right]^T \tag{2.19}$$

How to properly compute the solution is explained in Sec. 2.5. The cost of the solution is the same of a single linearized flow equation problem. With the discrete adjoint it is possible to compute the tangent of a J_k respect to all the elements contained in α, at the cost of a flow and an adjoint simulation. The cost scales linearly with the number of objective functions selected and the aerodynamic constraints since they are treated as a target of the optimization. If n_k becomes comparable to n_α, considering the high Cpu cost of the adjoint and the complexity of the code involved, it is more appropriate to use the traditional Finite Difference approach. Although, this case is not typical in the aeronautical industry where the target is usually to minimize the drag or maximise the efficiency with a couple of aerodynamic constraints regarding the lift and torque moment. Meanwhile, the solid that needs to be investigated, especially in 3D, requires hundreds of design variables for a proper geometrical description. As a result, the discrete adjoint approach is computationally advantageous.

2.3 Discrete Adjoint: Lagrange Multipliers

The easier way to introduce the adjoint formulation is to use the Lagrange multipliers. Again, the optimization problem including a steady state constraint can be expressed as:

$$\begin{aligned} \min_{\alpha} \quad & J(U(\alpha),X(\alpha)) \\ \text{subject to} \quad & R(U(\alpha),X(\alpha)) = 0 \\ & X(\alpha) = M(\alpha)\,. \end{aligned} \tag{2.20}$$

In analogy with the work of Nielsen [48] the problem can be rewritten defining a Lagrangian function:

$$L(\alpha,U,X,\Lambda) = J(\alpha,U,X) + \Lambda^T R(\alpha,U,X) \tag{2.21}$$

The vector of Lagrange multipliers is denoted by Λ, every element is an unbounded free parameter. A change in value of a design variable means a modification of a certain physical boundary where a wall condition is applied. This would cause a deviation of the flow state U from the original value to satisfy the new boundary conditions and similarly a change of the surface and volume grid. It is always assumed to be dealing with smooth and continuous variations. Since Eq. **??** has to be respected, we are adding a null term, meaning that the Lagrangian function L has the same value of the objective function J, however the derivatives of L will reflect the fact that the constraint must be satisfied. Differentiating Eq. 2.21 with respect to the design parameters yields:

$$\frac{\mathrm{d}L}{\mathrm{d}\alpha} = \left\{ \frac{\partial J}{\partial \alpha} + \left[\frac{\partial X}{\partial \alpha}\right]^T \frac{\partial J}{\partial X} \right\} + \left[\frac{\partial U}{\partial \alpha}\right]^T \left\{ \frac{\partial J}{\partial U} + \left[\frac{\partial R}{\partial U}\right]^T \Lambda \right\} + \left\{ \left[\frac{\partial R}{\partial \alpha}\right]^T + \left[\frac{\partial X}{\partial \alpha}\right]^T \left[\frac{\partial R}{\partial X}\right]^T \right\} \Lambda \tag{2.22}$$

Since Λ is arbitrary it is possible to remove the terms multiplied by $\left[\frac{\partial U}{\partial \alpha}\right]$ solving the adjoint equation:

$$\left[\frac{\partial R}{\partial U}\right]^T \Lambda = \left[\frac{\partial J}{\partial U}\right]^T \tag{2.23}$$

Using a different approach, the same Eq. 2.19 has been obtained. Therefore, the unknown vector of Lagrange multipliers can be now computed, meanwhile the remaining terms of Eq. 2.22 can be used to evaluate the Lagrangian sensitivity:

$$\frac{\mathrm{d}L}{\mathrm{d}\alpha} = \left\{ \frac{\partial J}{\partial \alpha} + \left[\frac{\partial X}{\partial \alpha}\right]^T \frac{\partial J}{\partial X} \right\} + \left\{ \left[\frac{\partial R}{\partial \alpha}\right]^T + \left[\frac{\partial X}{\partial \alpha}\right]^T \left[\frac{\partial R}{\partial X}\right]^T \right\} \Lambda \tag{2.24}$$

The terms $\left[\frac{\partial R}{\partial U}\right]^T$ and $\left[\frac{\partial R}{\partial X}\right]^T$, which represent the derivative of the residual with respect to the flow variables and the grid nodes, for a structured grid they can be explicitly constructed and stored. The first term is closely linked to the discretization method used. The term $\left(\frac{\partial X}{\partial \alpha}\right)$ is the mesh sensitivity, it depends on the mesh movement strategy implemented. While considering the linear elasticity analogy technique for updating the mesh, the equations which require solving are:

$$\nabla^2 u + \frac{1}{1-2\nu}\frac{\partial \nabla V}{\partial x} = 0 \quad \nabla^2 v + \frac{1}{1-2\nu}\frac{\partial \nabla V}{\partial y} = 0 \tag{2.25}$$

where ν is the Poisson's ratio and V is a two component vector (u,v) which elements are the node displacements in the two directions. However the cost of solving the equations is significant. The

overall system can be written using the stiffness matrix K, which can be computed with a loop on each cell and than stored:

$$KX = X_{\text{surf}} \tag{2.26}$$

At the same cost of a mesh movement however for each design variable in α the mesh sensitivity can be computed from:

$$K\frac{\partial X}{\partial \alpha} = \left(\frac{\partial X}{\partial \alpha}\right)_{\text{surf}} \tag{2.27}$$

The CPU cost and the high virtual memory consumption linked to the calculation of the volume mesh sensitivity are however significant limitations for large three-dimensional problems.

2.4 Double adjoint

In summary, three type of sensitivity are required: flow, shape and grid sensitivity. The latter has a computational cost that strongly impact the optimization chain, especially concerning large unstructured meshes. "Double adjoint" or "adjoint mesh" is a technique that for the first time allows the discrete adjoint computation to be really free from any cost related to the number of design variables involved. In the mid 2000's it became clear that computing $\frac{\partial X}{\partial \alpha}$ was computationally restrictive limiting in practice the cost independence of the adjoint from the length of α vector. All costs scaling with the design parameters must be avoided. One elegant way to remove the volume mesh sensitivity from the discrete adjoint was proposed by Nielson and Park [49]. It must be highlighted that the surface sensitivity still depends on the grid deformation method selected even if the volume mesh sensitivity is eliminated. The total mesh can be considered as sum of two different components $X = X_{\text{vol}} + X_{\text{surf}} = M(\alpha)$ where $X_{\text{vol}} = X_{\text{vol}}(X_{\text{surf}})$. It means that the volume grid depends explicitly on the wall mesh, that depends in a smooth way on the design variables. No assumptions on the structure of M are considered, except that is differentiable. The solution process of Eq. **??** can be transformed into a fixed point equation $U^{n+1} =: G(U^n)$ which is discussed in the following chapter. The optimization problem acquires an additional constraint and now takes the form of:

$$\begin{aligned} \min_{\alpha} \quad & J(U(\alpha), X(\alpha)) \\ \text{subject to} \quad & U(\alpha) = G(U(\alpha), X(\alpha)) \\ & X(\alpha) = M(\alpha)\,. \end{aligned} \tag{2.28}$$

A new Lagrangian equation associated to this problem can be created:

$$L(\alpha, U, X, \Delta_f, \Lambda_g) = J(U, X, \alpha) + [G(U, X) - U]^T \Lambda_f + [M(\alpha) - X]^T \Lambda_g \tag{2.29}$$

where Λ_f represents the adjoint variables linked with the flow state, meanwhile Λ_g is a new set of adjoint variables multiplying the residual of the grid movement problem. Both vectors can assume arbitrary values. Introducing in Eq. 2.29 the shifted Lagrangian $N(U, \Lambda_f, X) := J(U, X) + G(U, X)^T \Lambda_f$ a more simple expression is obtained:

$$L(\alpha, U, X, \Lambda_f, \Lambda_g) = N(U, \Lambda_f, X) - U^T \Lambda_f + [M(\alpha) - X]\,\Lambda_g \tag{2.30}$$

As in the previous section L has to be differentiated with respect to the design variables using the chain rule. Taking advantage of the free value of the adjoint vectors, the terms $\frac{\partial U}{\partial \alpha}$ and $\frac{\partial X}{\partial \alpha}$ can be eliminated. This leads to two adjoint systems that need to be solved:

$$\Lambda_f = \frac{\partial N}{\partial U} = \frac{\partial J^T}{\partial U} + \frac{\partial G^T}{\partial U}\Lambda_f \tag{2.31}$$

$$\Lambda_g = \frac{\partial N}{\partial X} = \frac{\partial J^T}{\partial X} + \frac{\partial G^T}{\partial X}\Lambda_f \tag{2.32}$$

The total derivative of J is reduced to:

$$\frac{\mathrm{d}L}{\mathrm{d}\alpha}^T = \frac{\mathrm{d}J}{\mathrm{d}\alpha}^T = \frac{\mathrm{d}M(\alpha)^T}{\mathrm{d}\alpha}\Lambda_g. \tag{2.33}$$

For the sake of clarity, if the linear elasticity approximation is selected as grid deformation strategy, Eq. 2.26 provides the explicit relation between volume and surface mesh. Therefore, equations Eq. 2.29 and Eq. 2.33 can be specialized to this case:

$$L(\alpha, U, X, \Lambda_f, \Lambda_g) = J(U, X, \alpha) + [G(U, X) - U]^T \Lambda_f + [X_{\text{surf}} - KX]^T \Lambda_g \tag{2.34}$$

$$K^T \Lambda_g = -[\frac{\partial J}{\partial X} + \left(\frac{\partial R}{\partial X}\right)^T \Lambda_f] \tag{2.35}$$

The final form of the sensitivity vector turns out to be:

$$\frac{\mathrm{d}L}{\mathrm{d}\alpha} = \frac{\partial J}{\partial \alpha} + \Lambda_f^T \frac{\partial R}{\partial \alpha} - \Lambda_g^T \frac{\partial X_{surf}}{\partial \alpha} \tag{2.36}$$

With respect to the previous section, the term $\Lambda_g^T \frac{\partial X_{\text{surf}}}{\partial \alpha}$ is cheaper to be calculated and it only requires an explicit inner product. Its size is linked to the length of the α vector and to the number of surface nodes and not to the entire grid extension. $\frac{\partial X_{\text{surf}}}{\partial \alpha}$ depends on the shape parametrization selected, which creates the mathematical relation between the surface nodes and the DVs, it is usually linearized using finite difference. It is interesting to notice that if the design variable is a flow entity, such as the Mach number or the side slip angle, the terms $\frac{\partial R}{\partial X}$ and $\frac{\partial X}{\partial D}$ are null therefore the Eq. 2.36 becomes really fast and cheap to be computed.

2.5 Duality Preserving FPI and RPM

In this section it is explained how to solve the discrete flow and adjoint equations. Besides, a stabilization method is introduced. Once the finite volume method has been applied to the RANS equation, using an implicit Euler scheme with pseudo-time integration we end up with a linear system to be solved in n iterations:

$$\left(D^n + \frac{\partial R(U^n)}{\partial U^n}\right) \Delta U^n = -R(U^n) \tag{2.37}$$

$$\Delta U_i^n = U_i^{n+1} - U_i^n \quad (D^n)_{ij} = \frac{|\Omega_i|}{\Delta t_i^n}\delta_{ij} \tag{2.38}$$

where Ω_i is the volume of the cell i. The pseudo time associated has a different value in each cell owing to a local time-stepping technique. Eq. 2.37 can be rearranged into a Fixed Point Iteration (FPI) $U = G(U)$. The same feasible solution can be extracted from:

$$U^{n+1} = G(U^n) := U^n - P(U^n)R(U^n) \quad \textit{where} \quad P(U) := \left(D + \frac{\partial \tilde{R}(U)}{\partial U}\right)^{-1} \tag{2.39}$$

The tilde suggests that an approximated Jacobian can be used. G is stationary only at the feasible point. Once the converged solution U^* is available, we can solve the adjoint Eq. 2.31. The convergence of the adjoint equation is guaranteed under the assumption that G is contractive, than also $\frac{\partial N}{\partial U}$ is contractive.

The condition $\| \frac{\partial G}{\partial U} \| < 1$ should be reached at a certain level of convergence.
A more general fixed point iteration, including Runge-Kutta and implicit schemes can be expressed using a preconditioning matrix M like:

$$M(U^{n+1} - U^n) = -R(U^n) \tag{2.40}$$

Linearizing around the exact solution U^*, we obtain:

$$U^{n+1} = (I - M^{-1}A)U^n + g(U^*) + \| U^n - U^* \|^2 \tag{2.41}$$

where $A = \frac{\partial R}{\partial U}$ is the Jacobian evaluated at the exact solution and $g(-)$ is a general function independent of the iteration. A close relation with the linearized problem is evident. We consider the Fixed Point Iteration:

$$M(\Omega^{n+1} - \Omega^n) = \frac{\partial R}{\partial X}\frac{\partial X}{\partial \alpha} - A\Omega^n \tag{2.42}$$

$$\Omega^{n+1} = (I - M^{-1}A)\Omega^n + M^{-1}\left(\frac{\partial R}{\partial X}\frac{\partial X}{\partial \alpha}\right) \tag{2.43}$$

In these two iterations W^n and Ω^n are the same, therefore for a converged nonlinear iteration the errors contract at the same speed, therefore the convergence behaviour is the same. The rate of convergence is dictated by the greatest eigenvalue of $(I - M^{-1}A)$. Regarding the adjoint expression, it can be obtained a very similar FPI:

$$M^T(\Lambda^{n+1} - \Lambda^n) = \left[\frac{\partial J}{\partial U}\right]^T - A^T\Lambda^n \tag{2.44}$$

or rearranged into:

$$\Lambda^{n+1} - \Lambda^n = (I - M^{-T}A^T)M^{-T}\left[\frac{\partial J}{\partial U}\right]^T \tag{2.45}$$

Remembering that any real matrix and its transpose have identical eigenspectra, Eq. 2.45 will have the same convergence property of Eq. 2.43. A further convergence statement can be expressed requiring that $\Omega^0 = \Lambda^0 = 0$ and expanding the two equations to the N^{th} iteration:

$$\Omega^{n+1} = -\sum_{n=0}^{N}(I - M^{-1}A)^n M^{-1}\left[\frac{\partial R}{\partial X}\frac{\partial X}{\partial \alpha}\right] \tag{2.46}$$

$$\Lambda^{n+1} = -\sum_{n=0}^{N}(I - M^{-T}A^T)^n M^{-T}\left[\frac{\partial J}{\partial U}\right]^T \tag{2.47}$$

It quickly follows that:

$$\left[\Lambda^{n+1}\right]^T\left[\frac{\partial R}{\partial X}\frac{\partial X}{\partial \alpha}\right] = \left[\frac{\partial J}{\partial U}\right]^T \Omega^{n+1}. \tag{2.48}$$

Therefore the adjoint and the flow results for $\frac{\partial J}{\partial \alpha}$ are identical not only when the equations are fully converged, although also at every intermediate step.
Going back to the simpler formulation, in reality the condition $\| \frac{\partial G}{\partial U} \| < 1$ is not granted, the solver could end up in a limit cycle oscillation, where the residual cannot be reduced anymore. Minor unsteady physical phenomena are often present and they are reflected by the fixed point iteration behaviour due to the pseudo-time introduced, causing the stall of the convergence. Therefore, the lack of convergence is linked to the presence of m eigenvalues of the matrix $G(U^*)^T$ with modulus greater than one:

$$|\lambda_1| \geq \cdots \geq |\lambda_m| \geq 1. \tag{2.49}$$

From the eigvector corresponding to the greatest eigvalue in modulus, we define the unstable subspace:

$$\mathbb{P} := span\{v_1, v_2, \ldots, v_m\} \tag{2.50}$$

and its orthogonal complement $\mathbb{Q} = \mathbb{P}^{\perp}\mathbb{P}$. They both are linear sub-spaces and the sum creates $\mathbb{R}^N$, thus every $x \in \mathbb{R}^N$ can be expressed as:

$$x = x_p + x_q \quad x_p \in \mathbb{P}, \quad x_q \in \mathbb{Q} \tag{2.51}$$

If $V \in \mathbb{R}^{Nxm}$ is an orthogonal base of $\mathbb{P}$ and $VV^T = I$, two orthogonal projections P and Q can be defined as:

$$P := VV^T, \quad Q := I - VV^T \tag{2.52}$$

For the Recursive Projection Method (RPM), a more stable method is used on the projection of the iteration into the unstable subspace $\mathbb{P}$ while the usual iteration is applied on the stable complementary projection. This can be translated into:

$$\Lambda_q^{n+1} = QN_U(\Lambda_p^n + \Lambda_q^n) \tag{2.53}$$

$$\Lambda_p^{n+1} = PN_U(\Lambda_p^n + \Lambda_q^n) \tag{2.54}$$

$$\Lambda^{n+1} = \Lambda_p^{n+1} + \Lambda_q^{n+1} \tag{2.55}$$

Eq. 2.53 converges if Λ_p^n is fixed. However, Eq. 2.54 has poor convergence properties. It must be replaced with an implicit equation and by a linearization around Λ_p^n:

$$\Lambda_p^{n+1} = PN_U(\Lambda_p^{n+1} + \Lambda_q^n) + PG_U^T P(\Lambda_p^{n+1} - \Lambda_q^n) \tag{2.56}$$

$$\Delta\Lambda_p^n = (I - PG_U^T P)^{-1}\left[PN(\Lambda^n) - \Lambda_p^n\right] \quad so \quad \Lambda_p^{n+1} = \Lambda_p^n + \Delta\Lambda_p^n \tag{2.57}$$

where $\Delta\Lambda_p^n = \Lambda_p^{n+1} - \Lambda_p^n$. What is missing is how the basis V can be constructed. The original method was proposed by Shroff and Keller [50], called Krylov acceptance criterion. It is based on the idea that the vectors $\Delta\Lambda_q^n$ lie in the dominant eigenspace of $QG_U^T Q$ since they are power iterations applied to $\Delta\Lambda_q^0$. The problem is the fact that only the residual of the eigenspace of $QG_U^T Q$ is used and not its eigenvalues, so the dimension of the unstable subspace can be overestimated [51].

2.6 Error Estimation

This section shows one possible usage of the adjoint variables different from the optimization process. Discrete Adjoint is becoming widely spread in the field of mesh adaptation. Consider a coarse mesh Ω_H [52], dense enough to obtain a good convergence however at the same time affordable respect to the computational resources available. H is a characteristic length associated with the cell spacing in a finite volume scheme. The interest is always to compute an object function $J(U)$ increasing the accuracy. Since we are able to solve the Eq. 2.20 the discrete value $J_H(U_H^*)$ is computed and stored. A finer mesh Ω_h can be constructed from the coarser one, splitting each element into n self-similar sub-elements, where n is an integer number. The parameter n can be easily computed: $n = \left[\frac{H}{h}\right]^d$ where d is the spatial dimension of the problem. To estimate $J_h(U_h)$ without running a CFD simulation on the finer mesh, the object function has to be expanded in Taylor series:

$$J_h(U_h) = J_h(U_h^H) + \frac{\partial J_h}{\partial U_h}\Big|_{U_h^H}(U_h - U_h^H) + \ldots \tag{2.58}$$

The term $\frac{\partial J_h}{\partial U_h}\Big|_{U_h^H}$ is the linear sensitivity of the fine-mesh functional. U_h^H is a vector indicating the coarse mesh solution mapped into the fine mesh using a prolongation operator I_h^H:

$$U_h^H = I_h^H U_H \tag{2.59}$$

The order of the operator must be greater or at least equal to the order of the discretization. In addition, equation **??** needs to be linearized around the coarse-grid solution:

$$R_h(U_h) = R_h(U_h^H) + \frac{\partial R_h}{\partial U_h}\Big|_{U_h^H}(U_h - U_h^H) + dots \tag{2.60}$$

Dropping the higher order terms and imposing that the residual has to be null once the convergence is obtained, we can extract:

$$(U_h - U_h^H) = -\left[\frac{\partial R_h}{\partial U_h}\Big|_{U_h^H}\right]^{-1} R_h(U_h^H). \tag{2.61}$$

Inserting this expression into the estimation of the objective function become:

$$J_h(U_h) = J_h(U_h^H) - \frac{\partial J_h}{\partial U_h}\Big|_{U_h^H}\left[\frac{\partial R_h}{\partial U_h}\Big|_{U_h^H}\right]^{-1} R_h(U_h^H) \tag{2.62}$$

The term multiplying the residual looks familiar. It is the adjoint vector denoted Λ_h^T computed on the fine-space using the linearization around the injected coarse-space solution U_h^H. Finally, we obtain an error estimation of the functional:

$$J_h(U_h) - J_h(U_h^H) = -\Lambda_h^T R_h(U_h^H) \tag{2.63}$$

In summary, the adjoint variables can be used to underline the lack of accuracy in the space discretization for the calculation of the selected objective function J. It is opinion of the author of this thesis that goal-oriented mesh adaptation will soon be part of any automatic optimization chain. The introduction is certainly simplified by the already existing strong interconnection with the adjoint. The mesh required for obtaining the convergence of the adjoint is often significantly different to that of the direct simulation. Taking as example a transonic wing with a shock on the upper part, the operator has to think what part of the fluid domain is influencing the position of the shock. The refinement zone has to be placed from the leading edge to the farfield right up the flow. Mesh adaptation can be an efficient tool to identify where unexpected refinements are needed and how to combine the mesh's requirements of both direct and adjoint simulation obtaining a robust solution. Fig. **??** shows how starting from the same grid, different refinement zones are obtained and how they can be smoothed together in a unique mesh.

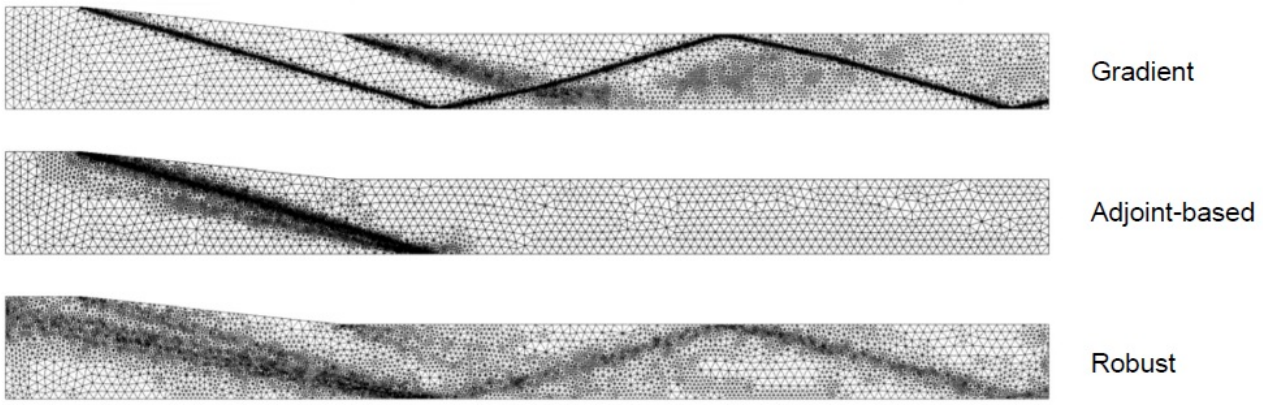

Figure 2.2: Object Oriented Mesh Adaptation, from [3]

2.7 Approximations

Due to the complexity of obtaining an accurate linearization of some Jacobian in the adjoint equation, certain assumptions have been introduced to simplify the problem. Many papers such as [45] examined the accuracy of the derivatives extracted with the adjoint approach simplified and estimate the amount of error generated.
In order to have a complete second-order accurate scheme, the linearization of the inviscid terms in the residual requires information from more mesh points than the ones immediately adjacent to the node. Thus, a large stencil is needed for the linearization making it complicated. The use of a first-order accurate scheme for the convective terms generates an easier linear system to solve and the bandwidth of the coefficient matrix is reduced significantly. However, the derivatives obtained have a discrepancy and often with an incorrect sign, this could affect the optimization process, particularly if the starting point is already near a local minimum.
The most widely used approximation is called "frozen turbulence" or "frozen eddy viscosity". The idea is to consider the variables linked to the turbulence model as constant, therefore the highly complex linearization of the turbulence model is not required. The complexity is strongly linked to the fact that the artificial viscosity term is dependent on the flow variables as well as on the distance from the wall. Nowadays only five turbulence models have been linearized by hand or using AD: the algebraic Balwin-Lomax, the one-equation model of Spalart-Allmaras [53] and the two equations models $k-\varepsilon$, $k-\omega SST$ [54] and the Wilcox $k-\omega$ [55]. The solution of the final linear system with full linearization can be exceptionally poorly conditioned [56]. The error introduced increases if the flow is largely separated, the magnitude of the error is always small although the sign of the derivatives is not always correct. This approximation is used within this thesis since it helps the convergence of the adjoint simulation.
The most extreme approximation was suggested by Mohammadi [57], all aerodynamic contributions to the gradient are neglected:

$$\frac{\mathrm{d}J(\alpha)}{\mathrm{d}\alpha} = \frac{\partial J}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}\alpha} + \frac{\partial J}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha} \simeq \frac{\partial J}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}\alpha}. \tag{2.64}$$

This truncation is based on the observation that the first term is often dominant in the case where J is an integral of boundary quantity and α deform directly that boundary. It gives good results near the leading edge also in the case of multi-elements configuration although it performs poorly at the trailing edge where the aero-gradients are stronger.

2.8 Adjoint Variables Interpretation

The adjoint variables can be treated as a pure mathematical object. It can be interpreted as the multiplier of the direct differentiation equation as in Sec. 2.2 which value is set to remove the flow sensitivity, or as in Sec. 2.3 interpreted as Lagrange multipliers. In both cases, the dual vector is related to the residual R and the mesh X, besides also to the numerical scheme adopted. Instead, Giles and Pierce firstly proposed a physical meaning [58]. The results presented were extracted using a continuous adjoint approach, the idea was transposed in the discrete adjoint context in Ref. [59]. In this section the residual R is a discretization of the Euler equations, the flux balance is not divided by the volume thus the discrete and continuous adjoint fields are similar. Four local perturbations δR are introduced in all cells of index m at the location ξ, the first one is a unit mass source injecting fluid with the local value of stagnation pressure and enthalpy:

$$\delta R_1 = [1, u, v, H] \tag{2.65}$$

The second source term is an applied force in the normal direction to the local flow:

$$\delta R_2 = [0, -\rho u, -\rho v, H] \tag{2.66}$$

The third perturbation is very similar to the previous one, it consists in a change of the total enthalpy at fixed static and total pressure:

$$\delta R_3 = \left[-\frac{1}{2H}, 0, 0, \frac{1}{2}\right] \tag{2.67}$$

The last one changes the total pressure at fixed total enthalpy and static pressure. The objective function taken into consideration is the integral over the airfoil of pressure times h, which involves the forces generated by the inviscid drag and lift. The variation of J due to the source term in cell $m, \left[\delta R_m^1, R_m^2, R_m^3, R_m^4\right]$ is calculated from:

$$(\delta J_m^1, \delta J_m^2, \delta J_m^3, \delta J_m^4) = (\Lambda_m^1, \Lambda_m^2, \Lambda_m^3, \Lambda_m^4) \begin{pmatrix} \delta R_1^1, \delta R_1^2, \delta R_1^3, \delta R_1^4 \\ \delta R_2^1, \delta R_2^2, \delta R_2^3, \delta R_2^4 \\ \delta R_3^1, \delta R_3^2, \delta R_3^3, \delta R_3^4 \\ \delta R_4^1, \delta R_4^2, \delta R_4^3, \delta R_4^4 \end{pmatrix} \tag{2.68}$$

The four perturbations of R are linearly independents. By inversion of Eq. 2.68 the adjoint vector can be obtained and analysed. For the first perturbation in a transonic flow, it appears that there is no singularity at the sonic line. This is in contrast with a previous results obtained for the same perturbation in a quasi 1D equation. The reason can lie in the fact that the flow in 2D is never perpendicular to the local streamlines. In addition the response to a punctual force (the second perturbation) is continuous across the stagnation streamline, the sonic line and any shocks. No perturbation of the pressure is produced by the third component, accordingly the associated linear functional is zero. Meanwhile, the last case highlights the existence of an inverse square-root singularity crossing the incoming stagnation streamline. There is no knowledge of work for the 3D case or for the complete RANS equations.

Chapter 3

SU2 Optimization Chain

The Stanford University Unstructured software [60] was created with the specific task of solving partial differential equations (PDE), multiphysics analysis and pde-constrained optimization problems on structured and unstructured grids. The core of the code is a RANS iterative solver able to simulate compressible, turbulent flows which are typical cases in the aerospace field. Multiple turbulence models are available, the one-equation Spalart Allmaras[53] typically used for airfoils with attached flow or very low separation, and the two-equation model Shear Stress Transport introduced in 1994 [54]. The usage of these turbulence models without wall functions impose the mesh requirement of having $y+ < 1$. The discretization of Navier-Stokes equations is performed using the Finite-volume method on vertex-based median-dual grid. Numerica schemes to solve the convective fluxes are implemented including JST, ROE, AUSM, moreover to enable a second-order space integration some slope limiters are available. The SU2 solver can also be used from multi-physics problems including Fluid-Structure interaction problems [61] and acoustics[24]. Additionally it can also be used for automatic shape optimization. The ability to compute the surface sensitivity using the discrete adjoint, described in the previous chapter, is only a small step of the complete chain. Chap. 2 introduced the mathematical theory of how to extract the adjoint equations and a modern method to solve them, what is lacking is how to compute the Jacobian that are inside of the adjoint equation. It is a specific and crucial choice of any single software, it strongly influences the overall computational cost and the quality of the final result. Formally, the exact linearization of the flow residual is demanded, although this requirement contrasts with the flow solver itself where an approximated Jacobian is sufficient to obtain convergence. To circumvent this problem in SU2 "Automatic Differentiation" is applied to the top-level routine of the solver, as explained in Sec. 3.2. The performances are still competitive, obtaining more flexibility and a more affordable extension to new turbulence models, transition models, or objective functions. Once the surface sensitivity is computed, it is then projected into the design space by a different C++ modulus of SU2 as explained in Sec. 3.1. How to mathematically describe the body with the design variables is another specific choice of the user, many options are available in 2D although only few for the three-dimensional case. This thesis takes advantage of the Free Form Deformation method that is more extensively described in Sec. 3.3. The following step consist in the update of the mesh around the body, since this part is the core of the thesis it is discussed in details in Chap. 4. Finally, the research of the minimum of the objective function is driven by a gradient-based optimization algorithm explained in Sec. 3.4.

3.1 Code Structure

The automatic optimization chain is performed by separated C++ modules, the sequential execution is driven by a Python script, in this case $shape-optimization.py$. Each C++ section reads its part of interest from the config file defined by the user and writes the state in a *.dat* file. Other outputs, such as *.vtu* files for the visualization of the results, must be explicitly demanded and do not influence the correct execution of the code. Every time that a C++ module is completely executed, the virtual memory is cleaned, and this is done to reduce the risk of accumulate useless data in the virtual memory. A baseline mesh is taken as input of the design cycle, along with an objective function, optimization constraints and a vector α of design variables. The configurations file contains the definition of the free-stream conditions, numerical methods, turbulence model, and convergence criteria. If the description of the body is made using FFD, the box vertices are written directly inside the *mesh.su2* file and the active design variables are specified in the $.cfg$ file. The principal modules called in the process are:

- SU2-CFD: performs flow and adjoint analysis.
- SU2-GEO: calculate the geometric characteristics of the airfoil or the wing to use them as constraints in the design cycle.
- SU2-DEF: deforms the grid surface and accordingly the surrounding volume mesh given the displacement of the design variables.
- SU2-DOT: calculate the gradient by projecting the surface sensitivity into the design space through a dot product.

Once the gradient of J is available, the Sequential Least Squares Programming optimiser is employed to guide the search for the optimum design. The optimisation cycle is considered converged when the Karush-Kuhn-Tucker conditions are fulfilled or the number of design loops exceeds a declared value. First of all, a converged numerical solution U^* of the RANS equations is needed. As seen in the previous chapter, the request of null flow residual can be translated into a fixed point equation, from which we can compute a feasible flow solution and the coupled turbulent variables:

$$U^{n+1} = G(U^n) := U^n - P(U^n)R(U^n) \quad \text{where} \quad P(U) := \left(D + \frac{\partial \tilde{R}(U)}{\partial U}\right)^{-1} \tag{3.1}$$

The preconditioner P contains an approximation of the partial Jacobian of the residual. The coupling between the flow and turbulent equations is neglected, accordingly the extra diagonal elements in P are null. Moreover, the implicit terms are treated with a first-order approximation even though a higher order spatial discretization is applied to the rest. From the adjoint duality condition:

$$\frac{\partial J}{\partial U^*}\frac{\partial U^*}{\partial X} = -\Lambda_f \left[\frac{\partial R}{\partial X}\right]^T \tag{3.2}$$

which is preserved for each iteration Λ^n we obtain the adjoint equation:

$$\Lambda_f^n + 1 = P^{-T}\frac{\partial J}{\partial U^*} + \left(I - P^{-T}\left[\frac{\partial R}{\partial U^*}\right]\right)\Lambda_f^n \tag{3.3}$$

A dual adjoint approach is used in SU2 to not treat explicitly the volume mesh sensitivity with respect to α. Once U^* is calculated, an evaluation of the objective function and flow iteration routine is needed

in order to compute the gradient of any linear combination of G and J with respect to U^*. Starting with a proper initial guess, we can perform Eq. 3.3 until convergence is achieved. Now, with a single evaluation of J in Eq. 2.33, we can obtain the geometric adjoint vector Λ_g.

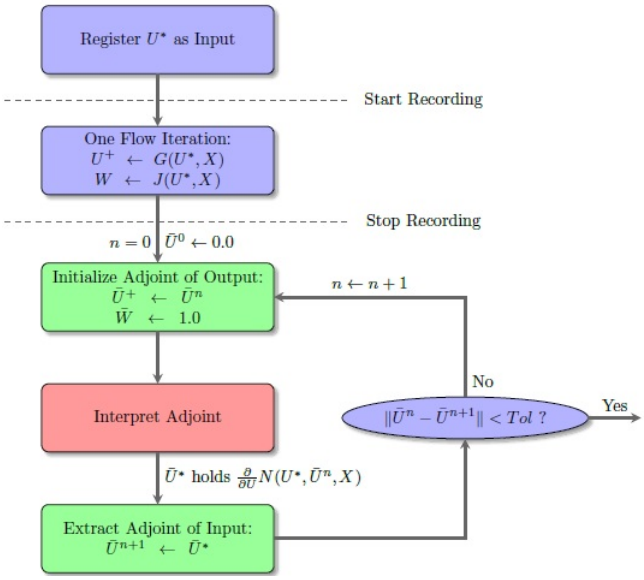

Figure 3.1: Top level approach of Adjoint solver in SU2, from [4]

The surface mesh sensitivity is calculated in the three Cartesian directions and then projected in the local normal direction, collecting the values for each surface cell inside the vector $\frac{\partial J}{\partial s}$. However, the shape of the object is parametrized with respect to the design variables and not respect to the wall mesh. Theoretically it would be possible to use directly the surface nodes to describe the model, however the computational cost would be excessive. Therefore, $\frac{\partial J}{\partial s}$ is projected into the space D_α through a dot product between the geometric and surface sensitivities, this operation is conducted by the $SU2-DOT$ module. The geometric sensitivity matrix indicates the influence of a change in a design variable's value on the location of a surface mesh node. It is computed using a finite difference method, which cost is almost negligible respect to the solution of a partial differential equation.

$$\begin{bmatrix} \frac{\partial J}{\partial \alpha_1} \\ \frac{\partial J}{\partial \alpha_2} \\ \vdots \\ \frac{\partial J}{\partial \alpha_{n\alpha}} \end{bmatrix} = \begin{bmatrix} \frac{\partial s_1}{\partial \alpha_1} & \cdots & \frac{\partial s_{ns}}{\partial \alpha_{n\alpha}} \\ \vdots & \ddots & \vdots \\ \frac{\partial s_1}{\partial x_{n\alpha}} & \cdots & \frac{\partial s_{ns}}{\partial x_{n\alpha}} \end{bmatrix} \begin{bmatrix} \frac{\partial J}{\partial s_1} \\ \frac{\partial J}{\partial s_2} \\ \vdots \\ \frac{\partial J}{\partial s_{ns}} \end{bmatrix} \tag{3.4}$$

The gradient just calculated can be amplified or reduced with a scale factor defined inside the $.cfg$ to have a movement of the FFD's vertex between 10^{-3}m and 10^{-2}m. Now the maximum displacement of the design variables is confronted with the upper and lower values declared in the configuration file that the program takes as input. In case the requirement is not fulfill, all values are scaled in order to be inside the admitted range. This part of the code has been updated and improved in the context of this thesis. $SU2-DEF$ has a double key function, it morphs the FFD box and consistently everything that is inside, including the surface grid. Lately the same C++ module deforms the entire grid updating the coordinates of each node. The RBF mesh deformation method which has been implemented inside the module $SU2-DEF$. An updated direct simulation can be now performed on the new grid until the criteria of convergence are matched.

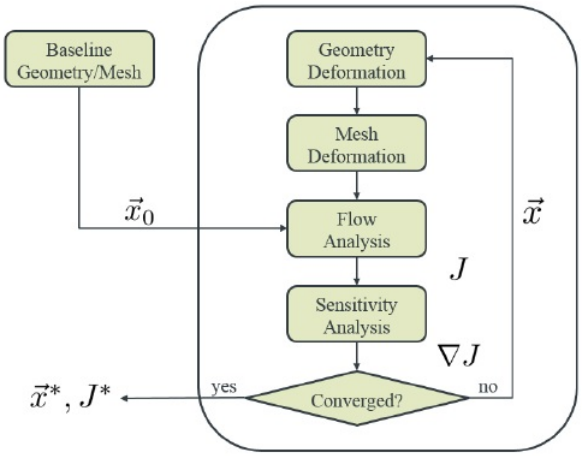

Figure 3.2: Optimization Design Chain, from [4]

3.2 Automatic Differentiation

At this point it should be more clear how to derive the adjoint set of equations and how to solve them efficiently. Although, before applying a fixed point iteration, a certain number of Jacobians contained in the system need to be computed. The accuracy and the method used to obtain the tangents is going to affect the quality of the results and the overall computational cost. Concerning a discrete approach implementation, five approaches are known: analytical method, finite-difference, complex-step method, AD and symbolic differentiation. The second and third options have been already exploited in the previous chapter. Instead, hand-written adjoint algorithms are really powerful, usually the cost is 3-4 times the original code [62], however they are extremely complex to extend, to debug and time consuming to conceive. The manual differentiation of turbulence models usually requires some short-cut assumptions whose physical meaning is not clear, besides how this approximation may impact on the value of the adjoint variables is difficult to understand and predict. Therefore, the analytic method, widely spread in the early codes, has largely fallen out in the last years with the development of other approaches. Symbolic differentiation has many similarities with the AD although in contrast it directly manipulates the complete mathematical expression of a function. It has to be expressed in a closed form, which is rare in CFD where instead iterative solutions are widely used and the mathematical expression created can be prohibitively long, therefore computationally expensive. Symbolic differentiation is still used in the FEM community [63]. Automatic Differentiation, also known under the name of Algorithmic Differentiation, is an attractive tool used in the context of SU2 to obtain the adjoint equation automatically from the original code. There are two approaches which can be used for implementing AD: source code transformation and operator overloading. Concerning source code transformation, it consist in a code modification adding new statements that computes the tangent of the original one. It is the most efficient way however it requires many interventions on the software. However, since SU2 is coded in C++, which is an object-oriented code, the implementation of the AD is based on the extensive use of Operator Overloading, which is the only option. The code is more virtual memory demanding although it is easier to implement and extend. The central idea is that every arithmetic operators and function are overloaded, following when called in the code, the original operation and the computation of the tangent are simultaneously done and stored. A benefit of AD is that it does not incur in any truncation error and yield derivatives with an approximation small as the epsilon machine. In similitude with symbolic differentiation, AD works applying the chain rule, therefore the full precise expression of the

derivative is not created. AD is often introduced starting from the consideration that the most complex mathematical code is in reality just a long concatenation of basic functions with at most two independent variables. This premise is intuitive however inefficient if applied in practice, since it requires the storage of every single operation. A more successful approach is to use AD at statement-level, where the information that need to be stored are independent from the number of operations internally involved [64]. A generic function f where $f : \mathbb{R}^n \to \mathbb{R}^m$ can be described as a succession of l statements $\varphi_i : \mathbb{R}^{n_i} \to \mathbb{R}$. Each one is a local evaluation arbitrary complex. In the following Tab. 3.1 it is shown how an arbitrary articulated expression is interpreted by the code and broken down into small and simple arithmetical operations.Where f takes as n inputs x and generates m outputs named y. Furthermore, $u_i := (v_j)_{j \prec i} \in \mathbb{R}^{n_i}$ exploits the relation between the intermediate variables v_i and v_j, where the first one depends directly on the second.

Table 3.1: Code Interpretation of a Generic Function

v_i	=	x_i	$i = 1 \dots n$
v_{i+n}	=	$\varphi_i(u_i)$	$i = 1 \dots l$
y_i	=	$v_{n+l-i+1}$	$i = 1 \dots m$

A more sophisticated representation of f can be given by the expression:

$$f(x) = Q_m * \Phi_l * \Phi_{l-1} * \cdots * \Phi_2 * \Phi_1 * P_n^T(x) \tag{3.5}$$

where the function $\Phi_i : V \to V$ with $V := \mathbb{R}^n \to \mathbb{R}^l$ sets v_{i+n} to $\varphi(v_j)_{j \prec i}$ although it does not change any other v_j. The final matrix Q has the rule of extract from a vector of $(n+l)$ components the last m elements.

3.2.1 Forward Mode

Taking advantage of the chain rule and applying it to each simple function present in Fig. 3.3, each statement can be augmented with its own derivative, which generates the forward AD. The name "forward" is linked to the fact that the values v_i are carried along at the same time with the derivatives $\dot{v}_i$.

Table 3.2: Evaluation procedure using forward AD

v_{i-n}	=	x_i	$i = 1 \dots n$
$\dot{v}_{i-n}$	=	$\dot{x}_i$	$i = 1 \dots n$
v_i	=	$\varphi_i(v_j)_{j \prec i}$	$i = 1 \dots l$
$\dot{v}_i$	=	$\sum_{j \prec i} \frac{\partial \varphi_i(u_i)}{\partial \dot{v}_j}$	$i = 1 \dots l$
y_i	=	v_{l-i}	$i = m-1 \dots 0$
$\dot{y}_i$	=	$\dot{v}_{l-1}$	$i = m-1 \dots 0$

The chain rule is the major difference from symbolic differentiation, where $\frac{\partial \varphi_i(u_i)}{\partial v_j}$ would be substituted with an algebraic expression and putting them together, it would create a derivative expression of

increasing complexity. Instead, with forward AD the memory used and the runtime are a priori limited, although an error is introduced due to the roundoff of the floating point values. Note that with the forward approach an input variable is selected and the first-order derivative of the intermediate steps needed to reach the output is calculated with respect to that input. Since the automatic differentiation is coded using Operator Overload, we want to stress that φ and $\dot{\varphi}$ are calculated at the same moment, sharing intermediate results. To do that a new notation is introduced, the tangent function can be written has:

$$\dot{y} = \dot{F}(x,\dot{x}) = F'(x)\dot{x} \quad where \quad \dot{F} : \mathbb{R}^{n+n} \rightarrow \mathbb{R}^m \tag{3.6}$$

that we can combine in a unique expression:

$$[y,\dot{y}] = \left[F(x),\dot{F}(x,\dot{x})\right] \tag{3.7}$$

This is consistent with the SU2 solver while using Automatic Differentiation. An external library named CoDiPack [23] will replace, where it is necessary, every *su2double* with a special structure called $ActiveReal < su2double, su2double >$. This couple of doubles are the computed value and the gradient of the operation. The entire general tangent procedure can be rewritten as:

Table 3.3: Compacted Evaluation procedure using forward AD

$[v_{i-n},\dot{v}_{i-n}]$	=	$[x_i,\dot{x}_i]$	$i = 1\ldots n$
$[v_i,\dot{v}_i]$	=	$[\varphi_i(u_i),\dot{\varphi}_i(u_i,\dot{u}_i)]$	$i = 1\ldots l$
$[y_{m-i},\dot{y}_{m-i}]$	=	$[v_{l-i},\dot{v}_{l-i}]$	$i = 1\ldots m$

3.2.2 Reverse Mode

Reverse mode is attractive for all that kind of problems where a single or a small number of objective functions depend on a large set of variables. Discrete Adjoint is a perfect example since we need the sensitivity of J with respect to the long vector α. In contrast to the forward modality, a single output is selected and the first-order derivative with respect to each of the intermediate variables and the input variables is calculated in an unique process. Reverse AD, coded using expression templates, is 2.7-4 times slower than a direct simulation, making it comparable to the hand-written Jacobians. However, reverse AD requires a large amount of physical memory, some techniques as local preaccumulation and the usage of checkpoints can help to contain the ram consumption [65]. First of all, to each v_i is associated a new variable $\bar{v}_i = \frac{\partial y}{\partial v_i}$ called "adjoint variable" and to the output selected an extra variable $\bar{y} = 1$. Now equation 3.5 has to be differentiated using the chain rule:

$$\dot{y} = Q_m A_l A_{l-1} \ldots A_2 A_1 P_n^T \dot{x} \quad where \quad A_i = \nabla\Phi_i \tag{3.8}$$

which is the tangent relation rewritten as an evaluation procedure. The Jacobian of the function can be immediately retrieved:

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = Q_m A_l A_{l-1} \ldots A_2 A_1 P_n^T \tag{3.9}$$

Recalling the duality identity $\bar{y}\dot{y} = \bar{x}\dot{x}$, the adjoint relation is obtained transposing the previous expression:

$$\bar{x} = P_n A_1^T A_2^T \ldots A_{l-1}^T A_l^T Q_m^T \bar{y} = \left(\frac{\mathrm{d}f}{\mathrm{d}x}\right)^T \bar{y} \tag{3.10}$$

This equation can be interpreted also as an evaluation procedure. Every product between matrix and vector is calculated for $i = l, l-1, \ldots, 1$. This is done backwards with respect to the sequence shown in Tab. 3.3. First, it is required to compute the value of each v_i values and temporary store it, this is called the forward sweep and it is due to the high physical memory consumption of this AD mode. Additionally, the sequence of the concatenation has to be registered, then it is inverted in the return sweep in order to have the tangent values. The reverse sweep is shown in the second part of the Tab. 3.4.

Table 3.4: Code interpretation of a generic function

v_{i-n}	=	x_i	$i =$	$1 \ldots n$
v_i	=	$\varphi_i(v_j)_{j \prec i}$	$i =$	$1 \ldots l$
y_{m-i}	=	v_{l-i}	$i =$	$m-1 \ldots 0$
v_{l-i}	=	y_{m-i}	$i =$	$1 \ldots m-1$
$\bar{v}_j$	=	$\bar{v}_j + \bar{v}_i \frac{\partial \varphi_i(u_i)_{j \prec i}}{\partial v_j}$	$i =$	$l \ldots 1$
$\bar{x}_i$	=	$\bar{v}_{l-n}$	$i =$	$n \ldots i$

3.2.3 Expression Template

Reverse mode technique for automatic differentiation becomes more attractive and computationally affordable if it is implemented using expression templates. This C++ technique drastically reduces the virtual memory's requirement. They were introduced to accelerate the evaluation of mathematical expressions based on operations or arrays. Nowadays, they are implemented in many C++ matrix libraries including Adept and CoDiPack. The central idea is that every operator has to be modified such that it does not return a value, however a new object that describes the type of operation that should be executed and registers the type of input expected. This object is called "Expression", every type of variables, arithmetical operators and functions are derived from one unique base class. The concept used in this kind of code is known as static polymorphism, an object of one type is hidden by another one, although the compiler is aware of this only when running the code. The use of virtual functions is avoided, which would have made the program way slower. Let us take as example the simple function:

$$f = \sin(a+b)(c-d)$$

each operator can be rewritten:

$$Expr_A \times Expr_B \rightarrow Expr_{A \times B}$$

Therefore, concatenating together all the expressions and creating just one operator for the whole statement, we obtain:

$$SIN < MULT < ADD < su2double, su2double >, SUB < su2double, su2double >>>$$

The information of the entire statement is available and no intermediate values are generated, just one operation with four arguments is stored. To be able to differentiate any expression applying the chain rule for every $f(a,b)$, such as a simple multiplication between two doubles, a member method has to be implemented. It takes $\frac{\partial f}{\partial a}$ and $\frac{\partial f}{\partial b}$and multiplies them with w, which is the value of the chain's

derivatives up to that point, and pass it to its arguments a and b. Every double is substitute with a *ADtype*, in the case of CodiPack is $ActiveReal < su2double, su2double >$, the first double is the primal value the second is the specific data for the tape.
If we recall the Reverse AD equation:

$$\bar{v}_j += \bar{v}_i \frac{\partial \varphi_i(u_i)}{\partial v_j}, j \prec i, i = l \dots 1 \tag{3.11}$$

For evaluating this expression, the value of $\bar{v}_i$ and u_i need to be known. Storing the gradient in an *ADtype*, as it is done in for forward mode, is not an option available since it would run out of scope when the direct evaluation loop is ended. Therefore it would be subsequently not available when the adjoint is computed. Instead, we use an identification technique. To every elementary operations φ_i that are inside f, an index is assigned. It goes from zero to l, which is the number of operations involved, the index is a global counter that increases every time that we store a statement. The zero index is used for passive variables as constants. The same index is also assigned to the intermediate variable on the left side of the assignment. Thus, for every elementary φ with n input, which are doubles (8 bytes), we need n indexes, another one for the output and one byte to store the number of arguments that can be at maximum 255. In total for each elementary statement $8 \times n + (n+1) \times 4 + 1$ bytes are used. Every statement is then evaluated during the reverse sweep but just in the opposite order. In SU2 the adjoint is entirely implemented using only the reverse method for computing the Jacobians.

3.3 Free Form Deformation

At this point, the displacement of each design variable is known. For the entire thesis the DVs used to describe the geometry of the body are the vertices of an FFD box. At the moment, FFD is the best option available for 3D simulation. A better representation of the shape could be obtained using feature-based solid modeling concepts, which are commonly adopted inside cad software. However, FBSM tools are not able to perform the sensitivity derivatives [66]. FFD is a remarkable versatile tool originally developed for computer graphics for deforming pics and morphing models. It can handle small and medium geometry changes, moreover the exact knowledge of the body is not necessary since only the grid's nodes are shifted. For this reason, the mesh connectivity never changes and the method is suitable for any type of grid and body. The central idea is straightforward, the FFD box is like a parallelpiped of deformable plastic in which an object that has to be morphed is immersed [5]. Also the body has to be considered flexible and it can change shape consistently with the box. The FFD container can also be cylindrical or spherical, both are implemented in SU2 however not used in this context since a simple parallelepiped can perfectly contain a wing or an airfoil.
Mathematically, FFD consists in a mapping from $\mathbb{R}^3$ to $\mathbb{R}^3$ through a tensor product Bernstein polynomial. First of all, a local coordinate system is set inside the delimited volume, any grid point X inside the control box has coordinates (s,t,u), also called lattice coordinates:

$$X = X_0 + sS + tT + uU$$

The box is divided into l x m x k sub-control volumes, which coordinates of the vertex (i,j,k) respect to a global reference system are contained in the matrix $P_{i,j,k}$. Some of these points or all of them can be used as design variable for the optimization process. The required displacement $\Delta P_{i,j,k}$ is obtained through the discrete adjoint and than the projection of the surface sensitivity into the design space. The movement of any point in local coordinates is calculated by:

$$x(s,t,u) + \Delta x(s,t,u) = \sum_{i=1}^{l} \sum_{j=1}^{m} \sum_{k=1}^{n} \left[B_{l-1}^{i-1}(s) B_{m-1}^{j-1}(t) B_{n-1}^{k-1}(u) \right] * \left[P_{i,j,k} + \Delta P_{i,j,k} \right] \tag{3.12}$$

where the Bernstein polynomial of degree l-1, also called the blending function, is determined as follows:

$$B_{l-1}^{i-1}(s) = \frac{(l-1)!}{(i-1)!(l-1)!} s^{i-1}(1-s)^{l-i} \tag{3.13}$$

Moreover, FFD can be easily formulated using B-spline or NURBS functions. FFD provides also the possibility to monitor the change of volume that the body undergoes. The Jacobian of the FFD indicates the rate of volume change imposed to each element. Imposing as constraint the Jacobian of F it is assured that the volume is preserved. Instead, if an inequality is used in the previous expression, it is required that the volume can only increase or reduce, besides a change of sign is forbidden.
Another interesting feature of FFD is the possibility to combine two or more control volumes adjacent and keep cross-boundary derivative continuity. It is considered to have two FFD boxes $X_1(s_1,t_1,u_1)$ and $X_2(s_2,t_2,u_2)$ which have in common a plane $s_1 = s_2 = 0$. The chain rule can be applied and find the tangent of the morphed surface:

$$\begin{aligned} \frac{\partial X_1(v,w)}{\partial v} &= \frac{\partial X_1}{\partial s}\frac{\partial s}{\partial v} + \frac{\partial X_1}{\partial t}\frac{\partial t}{\partial v} + \frac{\partial X_1}{\partial u}\frac{\partial u}{\partial v} \\ \frac{\partial X_1(v,w)}{\partial W} &= \frac{\partial X_1}{\partial s}\frac{\partial s}{\partial W} + \frac{\partial X_1}{\partial t}\frac{\partial t}{\partial W} + \frac{\partial X_1}{\partial u}\frac{\partial u}{\partial W} \end{aligned} \tag{3.14}$$

All second terms in the dot product are independent of the deformation. To impose a first derivative continuity it is necessary to impose:

$$\begin{aligned} \frac{\partial X_1(0,t,u)}{\partial s} &= \frac{\partial X_2(0,t,u)}{\partial s} \\ \frac{\partial X_1(0,t,u)}{\partial t} &= \frac{\partial X_2(0,t,u)}{\partial t} \\ \frac{\partial X_1(0,t,u)}{\partial u} &= \frac{\partial X_2(0,t,u)}{\partial u} \end{aligned} \tag{3.15}$$

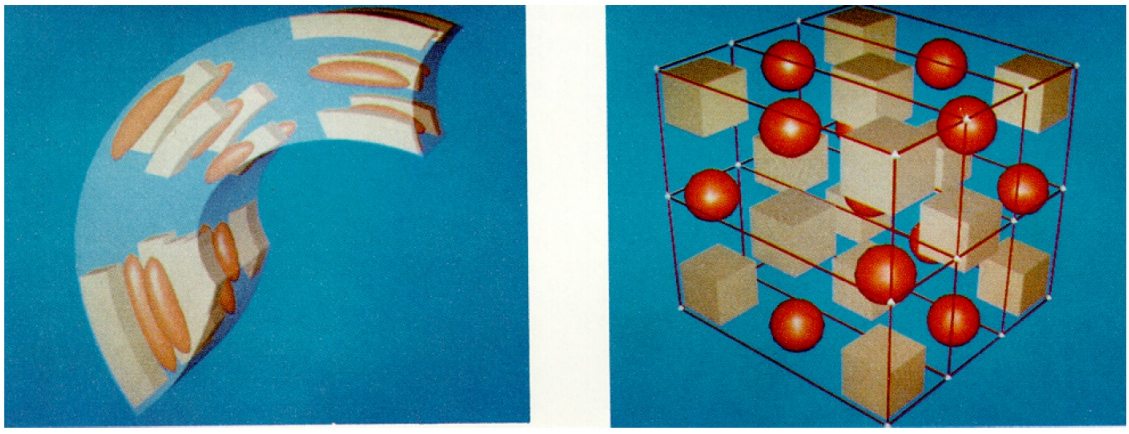

Figure 3.3: Example of FFD Box, from [5]

The usage of multiple FFD box helps to better describe and control complex bodies. However, in SU2 the common plane is block and the increasing number of continuity conditions reduced the possibility to morph the body. Moreover, the fact of having some slice of the body fixed drives the optimization far from the real optimum. This part of SU2 needs to be improved in the future.

3.4 Sequential Least SQuare Programming

Solving an optimization problem means to find at least one minimum of an objective function selected. A major difficulty for the algorithm, especially for the gradient-based, is to know if the minimum reached is a local or a global one. Concerning Su2, the process is driven by the gradient method SLSQP, which is provided by an external Python library Numpy/Scipy [67]. The best performance of this numerical scheme are obtained when the number of variables are between the number of constraints m and $50 * m$and a high degree of non-linearity is present, which is the typical case for aeronautical shape optimization [68]. No others algorithms have been tested inside the thesis, the choice of SLSQP is highly motivated by the results obtained in [69] where different methods, also genetic algorithms, were tested and SLSQP provide the best results in a relative limited number of iterations. It belongs to a more general family of algorithms called Sequential Quadratic Programming that are an extension of the Newton Methods for constrained nonlinear problem:

$$\begin{aligned} \min_{\mathrm{x}} \quad & f(x) \\ \text{subject to} \quad & g_j(x) = 0 \quad j = 1,\dots,m_e \\ & g_j(x) > 0 \quad j = m_e+1,\dots,m\,. \end{aligned} \tag{3.16}$$

In this section we will indicate as $A(x)$ the matrix which has as column the vectors $\nabla g_j^T(x)$ and $G_j(x) := \nabla^2_{xx} g_j(x)$. The non linear problem is solved iteratively from an initial guess vector x^0, after k loops we will have:

$$x^{k+1} = x^k + \alpha^k d^k \tag{3.17}$$

where d^k is called the search of direction and α^k is the step length. The two variables introduced are computed separately.
A quadratic programming subproblem QP is generated to find d. First, a classical Lagrangian function can be associated with the NLP and the constraints must be linearized:

$$L(x,\lambda) = f(x) - \sum_{j=1}^{m} \lambda_j g_j(x) \tag{3.18}$$

Generating in the standard way of quadratic programming the following problem:

$$\begin{aligned} \min_{\mathrm{d}} \quad & \frac{1}{2} d^T B^k d + \nabla f(x^k) d \\ \text{subject to} \quad & \nabla g_j(x^k) d + g_j(x^k) = 0 \quad j = 1,\dots,m_e \\ & \nabla g_j(x^k) d + g_j(x^k) \geq 0 \quad j = m_e+1,\dots,m\,. \end{aligned} \tag{3.19}$$

where $B := \nabla^2_{xx} L(x,\lambda)$ was introduced by Wilson [70] in 1963 .
Considering the step size, which has a strong analogy to the search of direction in the Newton's scheme, a one dimensional minimization has to be performed. More specifically, the following non-diffentiable penalty function has to be minimized:

$$\phi(x,e) := f(x) + \sum_{j=1}^{m_e} e_j |g_j(x)| + \sum_{j=m_e+1}^{m} e_j |g_j(x)| \tag{3.20}$$

It is also necessary a merit function $\varphi(\alpha) := \phi(x^k + \alpha d^k) \quad \varphi : R^1 \to R^1$, where x^k and d^k have been already computed and e_j are the penalty parameters. Many proposals are present in the literature to set these parameters [71] [72]. It is essential for practical applications to evaluate the Hesse-matrix of the Lagrange function B^k using only first-order information. Concerning unconstrained optimization, the BFGS algorithm is widely spread. Powell [71] developed an analogue formula for the constrained case, no deep explanation is reported here, just the strategy for updating the matrix:

$$B^{k+1} = B^k + \frac{q^k (q^k)^T}{(q^k)^T s^k} - \frac{B^k s^k (s^k)^T B^k}{(s^k)^T B^k s^k} \tag{3.21}$$

with $s^k := \alpha^k d^k$. The crucial point is that in the case of constrained optimization B^k could not remain positive definite for a positive definite initial estimate, although concerning the previous equation it can be avoided enforcing some conditions on the parameters involved.
The necessary conditions that have to be fulfilled for solving a nonlinear problem are called Karush-Kuhn-Tucker conditions (KKT) [73]. Each functions involved has Lipschitz continuous second derivatives and a feasible solution x^* of the NLP exist if:

$$\begin{aligned} \nabla_x L(x^*, \lambda^*) &= 0 & & \\ g_j(x^*) &= 0 & j &= 1, \ldots, m_e \\ g_j(x^*) &\geq 0 & j &= m_e + 1, \ldots, m \\ \lambda_j(x^*) &\geq 0 & j &= m_e + 1, \ldots, m \\ g_j(x^*)\lambda_j(x^*) &= 0 & j &= m_e + 1, \ldots, m\,. \end{aligned} \tag{3.22}$$

Many algorithms could be used to solve the QP such as a primal method, a dual quadratic programming or interior point method. In the context of SU2, NNLS is chosen, which was introduced by Lawson and Hanson [68]. The quadratic problem is transformed into an equivalent Linear Least Square formulation (LSEI) which is easier to be solved:

$$\begin{aligned} \min_{\mathrm{x}} \quad & \frac{1}{2}||Ex - f||^2 \\ \text{subject to} \quad & A^T x \geq b \\ & C^T x = d\,. \end{aligned} \tag{3.23}$$

SU2 is also able to run an optimization with multiple objective functions combined and multiplied by a specific weight imposed by the user. It can be required to minimize the drag and reduce the moment respect to a specific axis, the sideforces or the total pressure on a selected surface.

$$\min_x \quad f(x) = f_1(x) * w_1 + f_2(x) * w_2 + \cdots + f_j(x) * w_j \tag{3.24}$$

The user can also impose some aero-constraints such as requiring that the final lift has to be higher than a certain value or the pitch moment inside a certain range. An extra adjoint simulation is required for each of them and for each objective function required, therefore the computational cost and time required linearly increase. If the total number of aero constraints and objective functions is comparable or even higher than the length of the design variables vector, the adjoint approach is no more convenient. Geometrical constraints can be imposed without increasing significantly the cpu time, such as a maximum thickness can be required or the wing volume must not decrease. Each constraint is multiplied by a specific weight, the lower it is the more "strong" is the constraint: it has to be respected for each iteration and not only close to the minimum that we are looking for. The constraints are treated in

a hard way, therefore they must not be violated during the entire ASO, if a constraint is not respected the corresponding weight is increased. The sensitivity is computed only when the new point found is feasible and the objective function is better than the previous evaluation. Otherwise, SLSQP keeps morphing the grid and computes only the direct simulation and the geometrical description. With the same technique, SQP can handle multi-point optimization (MO), the same objective function is minimized for different free stream conditions such as different angle of attack, Reynolds, and Mach. In all these cases obtaining a feasible solution x^* that minimizes every function is unobtainable. It is possible to obtain a Pareto solution and the Pareto dominance concept is essential to compare two different points for a deeper knowledge [74].

Chapter 4

Mesh Deformation Methods

In this Chapter the different approaches for mesh deformation are discussed.Mesh deformation is required to update the node positions. Furthermore, it also influences the surface sensitivity for the design optimization. In SU2 the routine of the method is differentiated taking advantage of AD as suggested by Korivi [75], thus it is an active part in the computation of the adjoint vectors. This different result can drive the gradient base SLSQP to search a completely new point of equilibrium. Mesh deformation can be computationally prohibitive and require a large amount of virtual memory more than the adjoint simulation itself. Many methods are not robust and produce poor quality mesh. Mesh deformation is not only a prerogative of automatic optimization however it is required every time that a wall boundary undergoes a displacement. This is common for unsteady simulations where there is a relative movement between two parts or in Fluid-Structure interaction (FSI) problems, bio-mechanics and free surface flows. Theoretically, an entire mesh could be generated automatically from zero around the new boundaries as it is done by Sadrehaghighi[76] for airfoils. However, this would require a high level of knowledge of the body's geometry instead of just the old grid, which is complicated for 3D cases. Moreover, for full viscous simulations a hybrid mesh is often required for better capture the boundary layer and respecting the requirement of $y+ < 1$ imposed by the turbulence models. With a totally new mesh with different connectivity there is an extra effort to map the old flow state into the new nodes when it is necessary, instead with a deforming mesh an ALE approach can be used [77].
Two general classes of mesh deformation methods exist: physical analogy and interpolation. The first one model the process of deformation according to a physical process that can be solved applying a numerical method. Well known routine belonging to this category are introduced in Sec: 4.1 and Sec. 4.2. Usually they generate a slow mesh degradation in case of large deformations.
Instead, the second category transfers an imposed boundary's movement to the fluid mesh using an interpolation function. Interpolation schemes can be applied to any arbitrary grid types, including both structured and unstrustured meshes, since they do not require the mesh connectivity. Therefore, also the typical mesh with high aspect ratio cells near viscous surfaces used for high Reynolds flows can be treated, which is the typical case where physical analogy methods fail.
The two most widely used mesh deformation techniques include the linear elasticity analogy and radial basis function techniques. These mesh deformation techniques will now be discussed in further detail in Sec. 4.3 and Sec. 4.4. RBF was already implemented in SU2 and used for unsteady simulations and ice prediction [36]. In this thesis, RBF is incorporated inside the shape optimization chain making it compatible to work with CoDiPack for the automatic differentiation of the process. Radial Basis Function mesh deformation technique has been developed extensively throughout the literature and its potential for design optimization has been highlighted [35]. RBF guarantees an output mesh with high quality properties and it is able to manage larger displacements than the physical analogy approaches.

Concerning the design optimization process, RBF has been rarely used in the past and especially not coupled with the discrete adjoint approach. This is primarily due to the fact that the standard RBF mesh deformation technique is computational expensive. In the follow sections techniques to improve the efficiency of RBF mesh deformation are discussed. Consequently, RBF mesh deformation is only now becoming a viable option for the design optimization process on complex problems.

4.1 Spring Analogy

Tension Spring Analogy is the most widely used method for updating a mesh, it belongs to the first category and was originally proposed by Batina [32]. Two different kinds of spring analogy are presented: the vertex springs and the segment springs. Besides, a first development based on semi-torsional spring it is also introduced. Generally, non linear equations are avoided to keep it simple and cheap. Both methods consist in replacing the mesh edges with fictitious springs, the main difference lies on the equilibrium length chosen.
Regarding the vertex case the equilibrium length is imposed at zero. Since the springs are considered linear the force at each node i, applied by the connected node j, is determined taking advantage of the Hook's law:

$$\vec{F_i} = \sum_{j=1}^{v_i} \alpha_{ij}(x_j - x_i) \tag{4.1}$$

where α_{ij} is the stiffness of the edge between node i and j and v_i the number of edges connected to node i. It is required a solution of equilibrium, which is the one with lower energy accordingly the force at each node must be null. The iterative equation that need to be solved is:

$$\vec{x_i}^{k+1} = \frac{\sum_{j=1}^{v_i} \alpha_{ij} x_j^k}{\sum_{j=1}^{v_i} \alpha_{ij}} \tag{4.2}$$

The equation must be solved for each node, this time the number of iterations is contained since the model is used to retain mesh regularity. Dirichlet boundary conditions are applied, meaning that the movement of the boundary is strongly imposed. In practise, we are solving an elastic problem where the nodal position x_i is computed based on the weighted average of the near nodes. Mathematically, a linear system $[A](x) = (b)$ is generated, where the matrix contains the spring stiffness, the vector on the left represents the nodes positions and the right side is the non-homogeneous terms linked to the boundary movement. Every spring is under tension since the equilibrium is set for a null distance between nodes, therefore the mesh can be morphed even when the boundary is stationary.

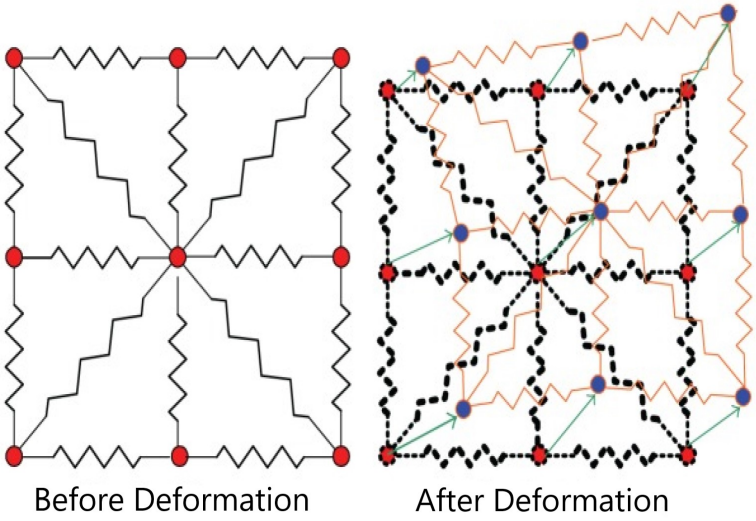

Figure 4.1: Mesh Deformation Using Linear Spring Analogy, from [6]

The second way to characterized the springs is the original one thought for a pitching airfoil, called segment spring analogy [7]. In the original position the springs are relaxed since the equilibrium length is set equal to the initial length of the segments. This time it is the displacement of the node δ_i that generates a force computed using the Hook's law:

$$\vec{F}_i = \sum_{j=1}^{v_i} \alpha_{ij}(\delta_j - \delta_i) \tag{4.3}$$

The static equilibrium of the system is reached when the total force at every node j is zero. This can be obtained solving the iterative equation:

$$\vec{\delta}_i^{k+1} = \frac{\sum_{j=1}^{v_i} \alpha_{ij}\delta_j^k}{\sum_{j=1}^{v_i} \alpha_{ij}} \tag{4.4}$$

The stiffness of the springs are proportional to the inverse of the distance between two nodes:

$$\alpha_{ij} = \frac{1}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}} \tag{4.5}$$

Node collision and element inversion, illustrated in Fig. **??**, are two of the main problem that could happen using spring analogy. Since node collision is more likely in a dense region, the second way of setting the stiffness helps to avoid this problem since mesh points that are closer to each other generates a more intense rejective force. However, the second method has the disadvantage to be more memory demanding since all the displacements must be kept in memory. In the end the updated node position is simply computed:

$$x_i^{\text{new}} = x_i^{\text{old}} + \delta_i^{k,\text{final}} \tag{4.6}$$

The torsional stiffness of each spring can be introduced to improve the robustness of the mesh deformation. Considering a two dimensional case, the target is to prevent adjacent triangles from interpenetrating each other. With only linear springs, the stiffness is not related to the area of the triangle or to the internal angles, therefore the element does not become more rigid when the area tends to be zero or

to have a bad aspect ratio.
As shown in Fig. **??** to each vertex i of the triangular element T^{ijk} a torsional spring is attached with stiffness:

$$C_i^{ijk} = \frac{1}{sin(\theta_i^{ijk})^2} \tag{4.7}$$

where θ_i^{ijk} is the angle included between the two edges ij and ik. This spring may prevent the vertices from crossing any edge and could prevent the triangle to become with a negative area since when $\theta_i^{ijk} \to \pi$ or $\theta_i^{ijk} \to 0$ the stiffness tends to infinity. Although this method does not totally eliminate the presence of elements with a bad aspect ratio. We can express everything as function of the edge lengths and the element area since:

$$sin(\theta_i^{ijk}) = \frac{2A_{ijk}}{l_{ij}l_{ik}} \tag{4.8}$$

$$C_i^{ijk} = \frac{l_{ij}^2 l_{ik}^2}{4A_{ijk}^2} \tag{4.9}$$

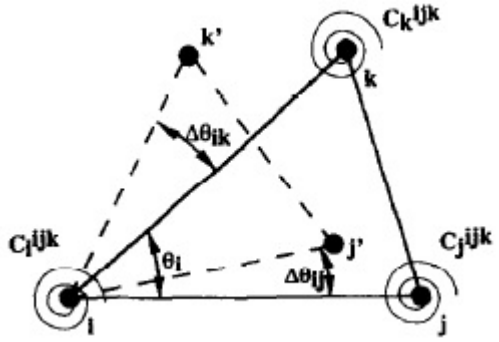

Figure 4.2: Spring Analogy Improved with Torsional Springs, from [7]

First, we need to determine the angular displacement, the entire process is well described in [44], the general 2D results is only reported here:

$$\Delta\theta_i^{ijk} = (b_{ik} - b_{jk})u_i + (a_{ij} - a_{ik})v_j + b_{ij}u_j - a_{ij}v_j - b_{ik}u_k + a_{ik}v_k \tag{4.10}$$

where (u_i, v_i) are the node i displacement in the two directions and

$$a_{ij} = \frac{x_{ij}}{l_{ij}^2} \quad b_{ij} = \frac{y_{ij}}{l_{ij}^2}$$

We can collect all the angular displacements in matrix form as:

$$\Delta\theta_i^{ijk} = R^{ijk} q^{ijk} \tag{4.11}$$

where:

$$\Delta\theta^{ijk} = \begin{bmatrix} \Delta\theta_i \\ \Delta\theta_j \\ \Delta\theta_k \end{bmatrix}$$

$$R^{ijk} = \begin{bmatrix} b_{ik} - bij & a_{ij} - aik & b_{ij} & -a_{ij} & -b_{ik} & a_{ik} \\ -b_{ji} & a_{ji} & b_{ji} - bjk & a_{jk} - aji & b_{jk} & -a_{jk} \\ b_{ki} & -a_{ki} & -b_{kj} & a_{kj} & b_{kj} - bki & a_{ki} - akj \end{bmatrix}$$

$$q^{ijk} = \begin{bmatrix} u_i \\ v_i \\ u_j \\ v_j \\ u_k \\ v_k \end{bmatrix}$$

Then, in order to determine the contribution of the added springs to the elastic forces acting on the mesh, we need to calculate the moments generated using a matrix formulation:

$$M^{ijk} = C^{ijk}\Delta\theta^{ijk} \tag{4.12}$$

It can be further adjusted explicating the nodal displacement dependence:

$$M^{ijk} = \left[C^{ijk}R^{ijk}\right] q^{ijk} \tag{4.13}$$

To easily combine the torsional effect with the linear springs and use the manipulation of the system already implemented, we convert the three moments into six forces F^{ijk}_{torsion} requiring that the work done by this two vectors is the same:

$$F^{ijk}_{\text{torsion}} q^{ijk} = M^{ijk^t}\Delta\theta^{ijk} \tag{4.14}$$

we can finally obtain that:

$$F^{ijk}_{\text{torsion}} = \left[R^{ijk^t}C^{ijk}R^{ijk}\right] q^{ijk} = K^{ijk}_{\text{torsion}} q^{ijk} \tag{4.15}$$

The superposition of the linear and torsional springs results in a dynamic mesh where the total fictitious force based on the edge length is described by:

$$F^{ij}_{\text{total}} = K^{ij}_{\text{linear}} q^{ij} + \sum \left[B^{ijk}_{ij} K^{ijk}_{\text{torsion}}\right] q^{ij} \tag{4.16}$$

where B is a Boolean operator that for each triangle it extracts a vector of sub-components associated with the edge ij.

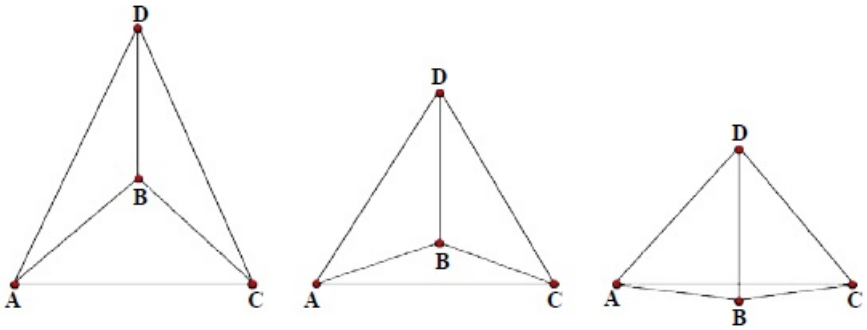

Figure 4.3: Negative Area Element Generated by Spring Analogy, from [7]

4.2 Linear Elasticity

Linear Elasticity is the most used method for mesh deformation in the context of shape optimization, especially in SU2 [78, 79, 80]. This method was implemented to avoid that a node crosses over an element face during the deformation process. The robustness is increased however at the same time it is computationally prohibitive and remains unable to handle large displacements [81, 82]. In this case the grid is described as an elastic continuum, the generation of a negative area is prevented by a natural mechanism. We could call this technique a FEM model which target is to compute the node displacement considering the mesh element as an homogeneous and isotropic material. Calling $u(x) = (u, v, w)$ a small displacement:

$$\nabla\sigma = f \quad on \quad \Omega \tag{4.17}$$

where the computational area is called Ω, f is some body force and σ is the stress tensor. Considering the strain tensor ε the constitutive law can be written as:

$$\sigma = \lambda Tr(\varepsilon)I + 2\mu\varepsilon \tag{4.18}$$

where λ and μ are properties of the elastic material known under the name of Lamé's constants, instead Tr indicates the trace of the tensor. These are usually expressed as function of E the Young's modulus and ν the Poisson's ratio:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)} \tag{4.19}$$

with $E > 0$ and $-1 < \nu < \frac{1}{2}$. A higher value of the Young modulus indicates rigidity. Besides, ν indicates, as consequence of an elongation in the axial vector, how much the material shrinks in the lateral direction. The kinematic law is:

$$\varepsilon = \frac{1}{2}(\nabla u + \nabla u^T) \tag{4.20}$$

The system is completed by imposing a Dirichlet boundary condition $u = g$ on $\partial\Omega$.
A Galerkian method based on trial and test space is applied to discretize the governing equation:

$$\begin{aligned} U^h &= \{u^h | u^h \in H^h(\omega)^n, u^h = g^h \quad on \quad \partial\Omega\} \\ \Phi^h &= \{\phi^h | \phi^h \in H^h(\omega)^n, \phi^h = 0 \quad on \quad \partial\Omega\} \end{aligned} \tag{4.21}$$

where H^h indicates a finite-dimensional function space on the computational domain of dimension n. We can formulate a finite element problem as:

$$\begin{aligned} find \quad u^h \in U^h \quad such \quad that \quad \forall\phi^h \in \Phi^h \\ \int_\Omega \varepsilon(\phi^h) : \sigma(u^h)\, d\Omega \quad = \int_\Omega \phi^h f\, d\Omega \end{aligned} \tag{4.22}$$

In order to simplify the implementation H^h is selected as the space of linear functions on the cells of the grid therefore $\varepsilon(\phi^h) : \sigma(u^h)$ is constant on the elements. The outcome linear system is solved with GMRES method.
The quality of the output mesh is strongly linked to how the two new variables E and ν are chosen, many proposals are presented in the literature. Tezduyar [83] for first suggested to link the stiffness to the dimension's elements. From this idea the POLIMI University [8] developed a code where the

Young modulus is proportional to the minimal vertex distance and an exponent β freely defined that can increase the stiffness of the smaller tetrahedra:

$$E_{el} = \frac{1}{min||x_j - x_i||^{\beta}} \tag{4.23}$$

The Poisson coefficient is modified inside the range $[0;0.45]$ to adjust the conditioning number of the matrix. Taking this choice, the mesh near the body where the element are typically smaller is rigid, therefore the most severe movement and deformations are relegated to the farfield's cells, which can absorb more strain. However, the specific code can treat only tetrahedrons. Therefore, an algorithm that splits a generic hexahedron into tetrahedra of good quality must be implemented. Concerning SU2, this issue is avoided since E is set to be $E = \frac{1}{V_i}$, where V_i is the element volume, thus the type of grid considered does not effect the result. Since the information needed to construct the matrix K are only geometrical, the mesh can be easily split into areas and each one assigned to a processor that is going to compute the properties of the cells.

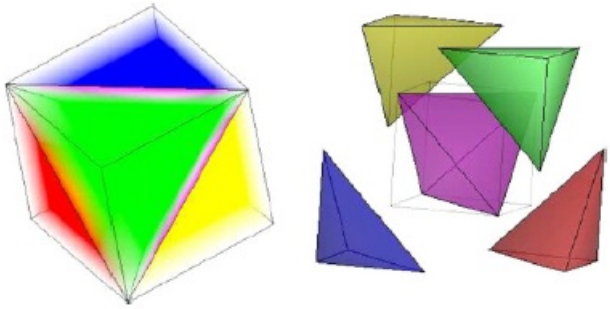

Figure 4.4: Subdivision Of Hexaedron Into Tetrahedra, from [8]

Another option is to set E as constant in the entire domain and tune ν such as the term $\frac{1}{1-2\nu}$ is comparable to the aspect ratio of the cell. In some articles, the Young modulus is chosen equal to the element condition number, also inversely proportional to the distance from the morphing boundary has produced good results [84]. It has also been noticed that increasing the value of E where the mesh has a poor quality makes the method more rigid. This variety of proposals pushed the community to find a more systematic and mathematical approach to obtain the optimal stiffness distribution, Yang and Mavriplis [85] proposed an adjoint based optimization with a gradient-based Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) [86] that computes the search direction. The solution of the final system to compute the new node position remains linear. E at each grid element is considered as design variable, a single objective function is defined L. We need to compute the gradient of L with respect to a very large number of DVs, millions for a typical three-dimensional mesh, so as seen in the previously chapters, the adjoint method perfectly suits. The objective function is defined as dependent on the cells' volume V:

$$L = L(V) \tag{4.24}$$

The global mesh quality must be described by L and the value has to increase where the element volume is vanishing. For multi-dimensional problems:

$$L = \sum_{i=1}^{N} L_i \quad \text{and} \quad L_i = e^{a(\xi_i - 1)^n} - 1 \tag{4.25}$$

where a and n are control variables and ξ_i is related to the change of volume $\xi_i = \frac{V_i}{V_{0i}}$, where V_{0i} is the initial volume and the numerator the final one. When ξ_i tends to zero, accordingly the cell i is vanishing, the value of L_i strongly increases. To compute the gradient, the volume V is considered function of the mesh initial position x_o and the displacement δx.

$$V = V(x_0 + \delta x) \tag{4.26}$$

The nodes movements depend on the starting grid configuration, on the prescribed movement of the boundary x_b and on the distribution of E_i that generates a different stiffness matrix $K(E)$. Therefore, $\delta x = \delta x(E, \delta x_b, x_0)$. Recalling the mesh motion equation:

$$K(E)\delta x = F(\delta x_b) \tag{4.27}$$

applying the chain rule, the derivative of L with respect to the design variables is expressed:

$$\frac{\mathrm{d}L}{\mathrm{d}E_i} = \frac{\mathrm{d}L}{\mathrm{d}V}\frac{\mathrm{d}V}{\mathrm{d}(\delta x)}\frac{\partial(\delta x)}{\partial E_i} \tag{4.28}$$

Besides, equation 4.27 has to be differentiated and inserted in the previous one:

$$\frac{\mathrm{d}K}{\mathrm{d}E_i} + K\frac{\partial \delta x}{\partial E_i} = 0 \tag{4.29}$$

$$\frac{\mathrm{d}L}{\mathrm{d}E_i} = -\frac{\mathrm{d}L}{\mathrm{d}V}\frac{\mathrm{d}V}{\mathrm{d}(\delta x)}K^{-1}\frac{\mathrm{d}K}{\mathrm{d}E_i} \tag{4.30}$$

Defining the transpose of a dual vector Λ equal to the first three terms, the adjoint equation to solve is:

$$K^T\Lambda = -\left[\frac{\mathrm{d}L}{\mathrm{d}V}\frac{\mathrm{d}V}{\mathrm{d}(\delta x)}\right]^T \tag{4.31}$$

The computation of the final sensitivity is obtained through a matrix-vector multiplication:

$$\frac{\mathrm{d}L}{\mathrm{d}E_i} = \Lambda^T\frac{\mathrm{d}K}{\mathrm{d}E_i} \tag{4.32}$$

In order to compute the sensitivity at each iteration of the optimization loop one mesh deformation problem must be solved and a mesh adjoint system. The computational cost increases with the complexity of the method, even if the accuracy of the result is improved. The physical memory required is linked only to the dimension of the grid. Essential for the robustness of this method is the quality of the input grid. Especially, the volume ratio among the cells strongly influences the final result, when the first cell of the boundary layer is really small (high Re) the progression to the larger cells of the farfield must be very smooth. The method will not fail in the deformation process, since regarding shape optimization the displacement are little. However, the consecutive RANS simulation on the deformed mesh will not converge interrupting the optimization process. This increases the number of cells of the grid accordingly the overall virtual memory consumption and cpu time.

4.3 Inverse Distance Weighting

This method, originally applied for creating the contour maps in meteorology and geography, belongs to the second class of mesh deformation technique: Interpolation Analogy. The connectivity is not required thus it can be applied to any type of mesh. With respect to the physical analogy, these schemes

are less ram demanding and easier to parallelize although as any interpolation process they are associated with an intrinsic error. In comparison with RBF, IDW does not require solving a linear system since the movement of each internal point is computed with a direct algebraic expression. The displacement is inversely proportional to the distance between the boundary element where a Dirichlet boundary condition is applied and the node considered inside the computational domain. This generates a distance decay effect, although the algebraic expression is globally applied, every node of the mesh has to be evaluated. If an interpolation surface $w(x)$ is considered, using n data collected in the vector $v = (v_1, v_2, \ldots, v_n)$ which can easily be associated with the displacement of the boundary nodes, the inverse distance weighting is given by:

$$w(x) = \frac{\sum_{i=1}^{n} v_i \Phi(r_i)}{\sum_{i=1}^{n} \Phi(r_i)} \tag{4.33}$$

where $\Phi(r) = r^{-c}$, and r is the Euclidean length between a data sample x and the fluid node position x_i, accordingly $r = ||x - x_i|| > 0$. The exponent c is crucial in this kind of technique, if $c < 1$ $w(x)$ is not differentiable, for $c > 1$ the displacement is smoothed. Best quality of the mesh is obtained for $c = 2$; if the value is further increased, the orthogonality of the elements does not improve.
A more sophisticated way to set the weights is presented in [34]. This different option should better preserve the orthogonality of the cells close to the morphing surface, to avoid that a more refined area of the grid does not augment its influence in the interpolation scheme, the surface cadet is included in the weighting function.

$$\Phi_i(r) = A_i * \{(\frac{L_{\text{def}}}{||r||})^a + (\frac{\alpha L_{\text{def}}}{||r||})^b\} \tag{4.34}$$

where A_i is the area weight concentrated to node i, L_{def} is a certain length of the region undergoing the deformation, α is a fraction of L_{def} indicating the size of the region near the body most influenced by the morphing process. Usually a is set to 3 and $b = 5$, this value are suggested by a bunch of testes performed. In comparison to RBF, Witteveen [87] underlines that with IDW is not necessary to solve a system of equations and comparing the two methods, IDW is computationally less demanding however it has poorer mesh quality. In both the cited publications RBF doesn't have the volume reduction method implemented here and it is not declared how many control points are selected and how many levels are required to the greedy algorithm, both factor strongly influences the CPU time. It is clear that both schemes are more robust than the physical analogy methods.

4.4 Radial Basis Functions

In the class of interpolation methods, RBF is one of the most attractive scheme. Initially thought for scattered data [88, 89], it was applied with great success since it is able to interpolate in the entire space a scalar function which the correct value is known only in some discrete points. In the context of mesh deformation RBF is used to transfer the known displacement of a certain boundary to the fluid grid. This scheme generates an output high-quality mesh with the orthogonality well preserved near the morphing profile. The extension of the method from the bi-dimensional cases to 3D grids is straightforward and it can be applied to any kind of mesh, both structural and unstructured hybrid mesh, since the connectivity is not required. Furthermore, the system of equations generated is linear, meaning that a large and well established amount of efficient schemes to solve it are available in literature. The size of the system is strongly linked to the mesh dimensions and if no expedient is coded the system matrix is strictly $N_s \times N_v$, where N_s are the surface nodes and N_v the volume points. Remarkable are the works of Rendall and Allen [90] and the study of Sheng [91]. They introduced the use of data reduction

schemes, in particular the multilevel greedy-algorithm for the selection of surface nodes described in Subsec. 4.4.2 and the volume reduction method for a large data set presented in Subsec. 4.4.3. Thanks to this, RBF is more efficient and less computationally expensive, therefore it can be applied to large problems such as wing-winglet optimization which is the ultimate target of this thesis. Since RBF from the fluid solver requires only the desired movement of a surface that is going to be treated as a Dirichlet boundary condition, the mesh deformation method can be treated and coded as a "black box" toll, which is ideal for unsteady simulations or shape optimization [92]. Regarding SU2, the RBF tool with the data reduction schemes has been implemented by Morelli and Bellosta [93] and used in the context of complex ice geometries.

4.4.1 Formulation

The general theory of RBF, extensively displayed by Wendland [94] and Buhmann [95] is based on a series of functions whose value is linked to the distance between the selected position and a supporting point named "control point" or "source points" [96]. The general formulation of a volume spline, consisting of a transformation $\mathbb{R} \rightarrow \mathbb{R}$ used for interpolation is:

$$\phi(r, r_i) = \phi(||r - r_i||) \tag{4.35}$$

where r_i is the radial basis centre and the distance is intended as an Euclidean distance, meaning the spatial length between two considered nodes. The displacement of a collection of nodes in the flow volume can be described by an interpolation function $F(r)$ which is a sum of basis functions multiplied by a scalar which is unknown. The interpolation firstly introduced in [97] can be expressed as:

$$F(r) = \sum_{i=1}^{N} \alpha_i \phi(||r - r_i||) \tag{4.36}$$

To compute the weight coefficients, an exact recovery of the assigned function values at the control points has to be performed. This method requires the knowledge of the desired displacement of the entire surface grid. Concerning the optimization process implemented in SU2, from the adjoint solution projected into the design space the movement of the vertices of the FFD box is computed. Subsequently, with the free form deformation routine, which is also in this case an interpolation inside the control volume, the surface displacement is obtained. The vector ΔX collect the surface nodes movement which is underlined by the subscript "s", it is described by:

$$\begin{aligned} \Delta X_s &= [\Delta x_{s_1}, \Delta x_{s_2}, \ldots, \Delta x_{N_s}] \\ \Delta Y_s &= [\Delta y_{s_1}, \Delta y_{s_2}, \ldots, \Delta y_{N_s}] \\ \Delta Z_s &= [\Delta z_{s_1}, \Delta z_{s_2}, \ldots, \Delta z_{N_s}] \end{aligned} \tag{4.37}$$

The three Cartesian directions can be combined in a more simplified formulation:

$$\Delta S = \Delta X_s \hat{x} + \Delta Y_s \hat{y} + \Delta Z_s \hat{z} \tag{4.38}$$

In analogy also the weight coefficients are collected in a vector:

$$\alpha_x = [\alpha_{x,s_1}, \alpha_{x,s_2}, \ldots, \alpha x, N_s]^T \tag{4.39}$$

The y and z coefficients are analogous. Following, the weights can be extracted by solving the linear system:

$$\Delta S = \Phi_{s,s} \alpha \tag{4.40}$$

where Φ is the universal basis matrix, it is generated with the radial basis function evaluated at each surface nodes, meaning that the matrix has size of N_s^2. The compact form of the universal basis function is expressed as:

$$\Phi_{s_j,s_i} = \phi||r_{s_i} - r_{s_j}|| \tag{4.41}$$

The next step is to compute the volume base matrix $\Phi_{v,s}$ of size $N_v \times N_s$, where "v" indicates a volume point. Finally, the volume displacement can be interpolated multiplying the above-mentioned matrix with the weights previously computed:

$$\Delta V = \Phi_{v,s}\alpha \tag{4.42}$$

The behaviour of the interpolation between points or outside the dataset (extrapolation) is linked to the kind of radial function selected. Many options are present in literature and can be grouped into three main categories: the global supported, local supported, and compact supported ones. The global support functions are everywhere non-zero and the value grows with increasing the distance from the source point. The second type are always non-zero however they get close to zero value with increasing distance from the control point. Instead, compact functions decay to exactly zero with increasing $||r||$. The choice of the interpolating base is crucial, indeed with a global supported RBF a fully populated linear system has to be solved, often with a ill-conditioned matrix. Instead with compact supported ones, there are some computational advantages, since the matrix is sparse, however at the cost of losing accuracy. In practice, the most used for complex applications are the Wendland [94] compact support functions listed in the next table:

Table 4.1: Wendland Compact Support Functions

Name	Definition
Wendland $C0$	$\varphi(\eta) = (1-\eta)^2$
Wendland $C2$	$\varphi(\eta) = (1-\eta)^4(4\eta+1)$
Wendland $C4$	$\varphi(\eta) = (1-\eta)^6(35\eta^2+18\eta+3)$
Wendland $C6$	$\varphi(\eta) = (1-\eta)^8(32\eta^3+25\eta^2+8\eta+1)$

Where $\eta = \frac{||r-r_i||}{d}$ and d is the supporting radius usually set to one. In this thesis, according to the papers of Randell [90] and Costin [98], the default option of SU2 is used: the Wendland $C2$. This compact base increase the smoothness and the quality of the final mesh with respect to the C0 option and meanwhile it gives computational advantages respect to higher order functions.

4.4.2 Greedy Algorithm

Up to now N_s surface nodes were used to generate the interpolation global basis matrix Φ, resulting in a cost of solving the linear system proportional to N_s^3 and for updating the entire volume grid of $N_s \times N_v$. For three dimensional large-scale problems, the number of surface nodes can be of hundreds of thousands. In such cases, a reduction scheme is essential to reduce the computational cost of RBF mesh deformations scheme. It consist in a process of selection of the surface nodes to obtain a subset set P^c with limited dimension. The algorithm is created to obtain a set of sample points according to the error generated by describing the entire surface displacement with a reduced RBF interpolation. The scheme start with a single point, the one used does not affect the final results therefore the first one in the mesh file is picked, then adding an extra surface node where the difference between the

interpolated value and the exact one is maximum. This loop is repeated until the interpolation meets a selected tolerance. The selected nodes are collected in a vector X_c of increasing size N_c where the subscript "c" denotes the control point. The error vector E is computed using:

$$E = \Delta S - \Phi_{s,c}\alpha \tag{4.43}$$

where the matrix $\Phi_{s,c}$ is now of size $N_s \times N_c$ and where E is always of size N_s. The prescribed tolerance ε is compared with the normalized largest error with respect to the maximum displacement:

$$\frac{E^{\max}}{\Delta S} < \varepsilon \tag{4.44}$$

Each time that a node is selected the linear system to obtain the weights must be solved. Therefore, the CPU cost of the greedy algorithm is of the order of N_c^4, where the final number of control points selected is the number of iterations of the process plus one. In the case of large displacement of complex geometry the computational cost of the simple greedy scheme becomes too large. The problem has been overtaken by introducing a multi-level subspace radial basis function interpolation, firstly introduced by Wang [99]. The object for the second level of interpolation is set equal to the error of the first step $E^{(0)}$. In a general form, it can be expressed as:

$$\Delta S_{l+1} = E^{(l)} \tag{4.45}$$

where the next step of the multi-level selection process is indicated by the subscript "$l+1$". The residual of Eq. 4.40 at the second level can be expressed as:

$$\Delta S^{(1)} = \Delta S^{(0)} - \Phi W^{(1)} = \Delta S - \Phi(\alpha^{(0)} + \alpha^{(1)}) \tag{4.46}$$

the size of the displacement is strongly reduced $\Delta S_{l+1} << \Delta S_l$. The computational cost for the multi-level greedy algorithm is now of order of $N_l \times N_c^4$ instead of $(N_l \times N_c)^4$ for the single step. The overall process can be summarised:

$$\begin{aligned} \Delta S &= \sum_{i=0}^{i=N_l-1} \Delta S^{(i)} = \sum_{i=0}^{i=N_l-1} \Phi_{s,c}^{(i)} \alpha^{(i)} \\ \Delta V &= \sum_{i=0}^{i=N_l-1} \Delta V^{(i)} = \sum_{i=0}^{i=N_l-1} \Phi_{v,c}^{(i)} \alpha^{(i)} \end{aligned} \tag{4.47}$$

4.4.3 Volume Point Reduction

The CPU cost of the volume interpolation, after the multi greedy point selection, is now of the order of $N_l \times N_c \times N_v$. In the case of large scale geometry, it is of interest to find how to decrease N_v. One method was proposed by Xie and Liu [100]. They introduced a function which value is based on the distance from the closest wall:

$$\psi = \psi\left(\frac{d(r)}{D}\right). \tag{4.48}$$

where $d(r)$ is the space distance and D a support value imposed. We define the ratio between the two distances as ξ. The function decays in value when the distance increase and is zero outside the supported distance, so it is a compact support function. It can be expressed as:

$$\psi(\xi) = \begin{cases} (1-\xi) & 0 \leq \xi < 1 \\ 0 & \xi \geq 1 \end{cases} \tag{4.49}$$

The value of support distance D depends on the maximum surface displacement multiplied for a volume reduction factor k imposed by the used. Mathematically it can be expressed as:

$$D = k(\Delta S_l)^{\max} \tag{4.50}$$

k in practice defines the range around the body inside which the flow nodes are going to be shifted, besides we are requiring that the elements outside this volume are not affected by the surface movement. It can be notice that is exactly the opposite behaviour respect to the linear elasticity analogy of the previous section where the higher stiffness of the grid's elements close to the wall boundary cases that the distortion is absorbed by the largest elements which are close to the farfield.
The interpolation Eq. 4.36 is modified in order to include the wall distance correction:

$$F(r) = \psi\left(\frac{d(r)}{D}\right)\sum_{i=1}^{N}\alpha_i\varphi(||r - r_i||) \tag{4.51}$$

When the volume points reduction method is combined with a multilevel greedy algorithm, as it is done in SU2, at the first level N_c is relative small meanwhile N_v remains high to absorb the large deformation. The volume point interpolation is then computed with:

$$\Delta V_l = \Phi_{v,c}\alpha \tag{4.52}$$

At each level the support distance is updated with the new maximum displacement. Since, as explained previously, $\Delta S_{l+1} << \Delta S_l$ the range of influence of the basis function is strongly reduced, at the same time also the number of volume points is decreased $N_{v,l+1} << N_{v,l}$. The computational cost descends with the number of steps.

Chapter 5

Results-2D

Several bi-dimensional benchmark test cases concerning airfoils have been performed to verify the reliability of the code on simple flows. Two classical airfoils are selected, the symmetric NACA0012 and the transonic RAE2822. The process of shape optimization is performed taking advantage of the discrete adjoint present in SU2 for computing the surface sensitivity. Furthermore, shape modification is conducted comparing RBF with ELA as strategy for the mesh deformation. It should be underlined that the comparison of the results is significant in the context of SU2 community. Regarding the physical analogy, the element stiffness is computed with an inverse volume criterion, accordingly the smaller elements are more rigid and the deformation is absorbed by the larger elements usually placed far from the profile. Concerning RBF, three levels are required in the greedy algorithm for the selection of the control points. The compact support function Wendland $C2$ is selected as the base of RBF. The support distance k, used for the reduction of the number of volume nodes (N_v) considered during the deformation, is ten times the maximum surface displacement. As previously anticipated the airfoil shape is described and parametrized using an FFD box, the displacements of the vertices are kept in a range between 10^{-3}m and 10^{-2}m. To obtain the best performance from the gradient based SLSQP, once $\frac{\mathrm{d}J}{\mathrm{d}\alpha}$ is computed then is scaled to have it at approximately 10^{-6}. The last benchmark test presented in this chapter is exactly the one proposed by AIAA Aerodynamic Design Optimization Group (ADODG) so the results are comparable with the other tests present in the literature [101, 102, 103]. The NACA0012 benchmark has the additional challenge of a non unique solution, instead the RAE2822 present the issue of a separated flow after the shock wave which can be challenging in case of frozen viscosity.

5.1 NACA0012 Drag Minimization

The optimization of the classical airfoil NACA0012 has been performed, differently from what suggested by ADODG the optimization is based on RANS equations instead of Euler, also in the viscid case a non unique solution is found. A hybrid mesh is generated in order to have high-quality mesh properties. To guarantee the convergence of the RANS simulation with SA as turbulence model [53] the $y+ < 1$ everywhere. The cells close to the airfoil are structured rectangles, the total height of this region have to contain the entire boundary layer. Meanwhile, the rest of the grid is unstructured. The free stream conditions selected are:

Table 5.1: Free Stream Conditions NACA0012

Mach	0.76
AoA	2°
Re	$6.04E6$
Temperature	$215.38K$

Experimental data are provided by NASA [104], and the mesh convergence is performed according to the method proposed by Roache [105]. The value of the drag coefficient estimated for an infinitely dense mesh is shown in Fig. 5.1, while the C_P convergence is displayed in Fig. 5.2:

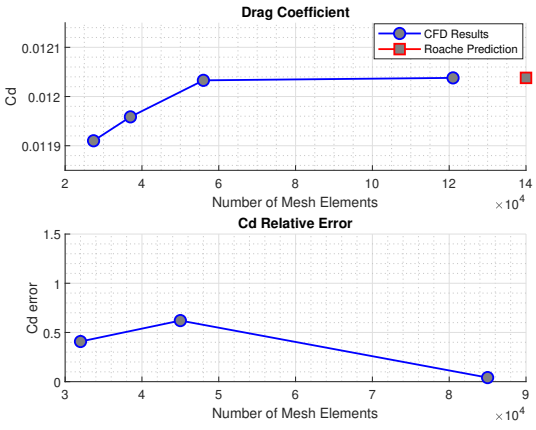

Figure 5.1: NACA0012: C_D Convergence

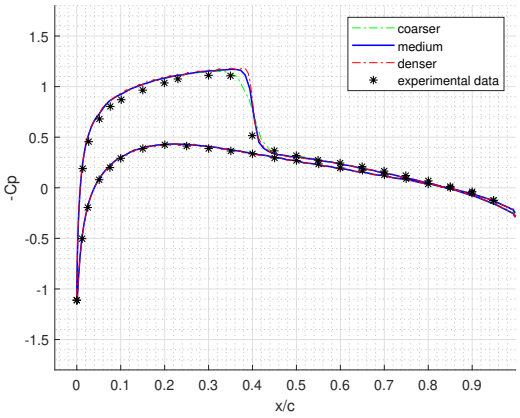

Figure 5.2: NACA0012: C_P Distribution

Selected the third mesh, also the influence of the farfield dimension on the drag and lift coefficients has been investigate. The same mesh although with different radius of the computational domain are tested and the relative error computed:

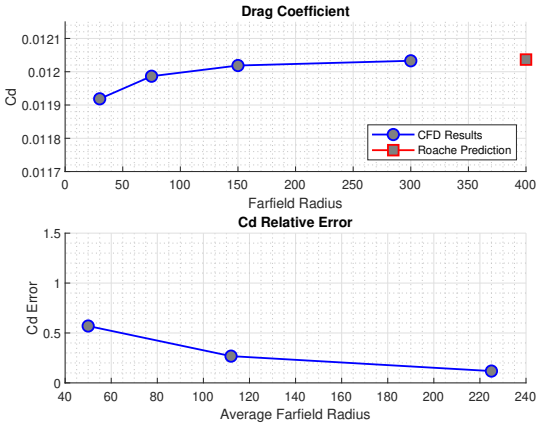

Figure 5.3: NACA0012: Farfield Investigation C_D

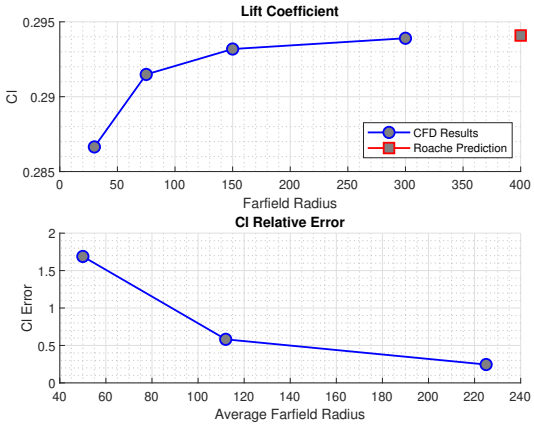

Figure 5.4: NACA0012: Farfield Investigation C_L

The goal of the optimization is to reduce the drag by modifying the shape without decreasing the maximum thickness of the airfoil. A constraint within the optimization is to maintain the C_l. Therefore,

two adjoint simulations are performed,one for the drag and the second regarding the lift sensitivity. The angle of attack is kept fixed. The NACA0012 airfoil is inserted inside an FFD box and split in a certain number of vertical rectangles, which vertices are selected as design variables. The approximation of frozen viscosity is used in this test case. Since the output of the optimization is strongly related to how is constructed the α vector, the first analysis shown in Fig. 5.5 is to find how many DVs are required to obtain the best performance. The optimization procedure is stopped after ten adjoint evaluations or if KKT [73] conditions are reached. While using sixteen design variables for the optimization process, the variation in the C_d with the number of deformations is shown in Fig. 5.6.

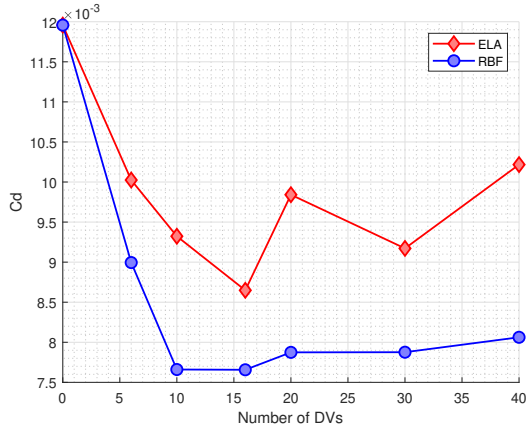

Figure 5.5: NACA0012: C_d Respect to DVs

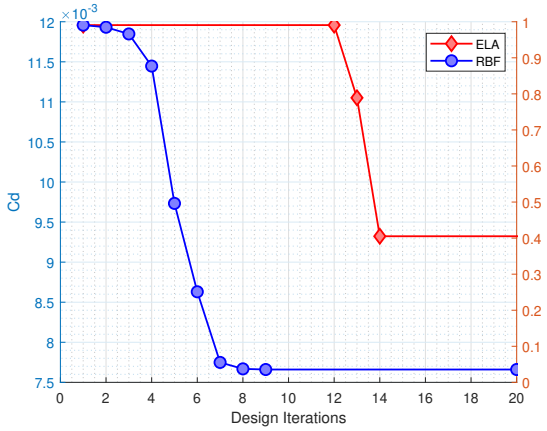

Figure 5.6: NACA0012: C_d V_s Design Loops

It is evident from Figs. 5.5 & 5.6 that the optimization process with RBF finds a deeper minimum of the C_d and it is also achieved quicker than with ELA. The C_d obtained using the RBF technique is 11.47% lower than the coefficient obtained with elastic analogy. Moreover, the reduction with respect to the original coefficient is 56%. In this specific case the large difference between the two methods is due to the computed surface sensitivities which are dissimilar in both sign and amplitude as shown by Fig. 5.7. Owing to this, the gradient-based optimization algorithm reaches two distant minimum, this highlights that at least one of the two is a local minimum and not a global one. Therefore, the shape of the profile undergoes to two disjointed movement, with ELA the upper part is thicker instead with RBF where most of the surface area is concentrated under the leading edge zone. The two sensitivity guide the optimization algorithm to obtain two different pressure distribution as it can be seen in Fig. 5.8

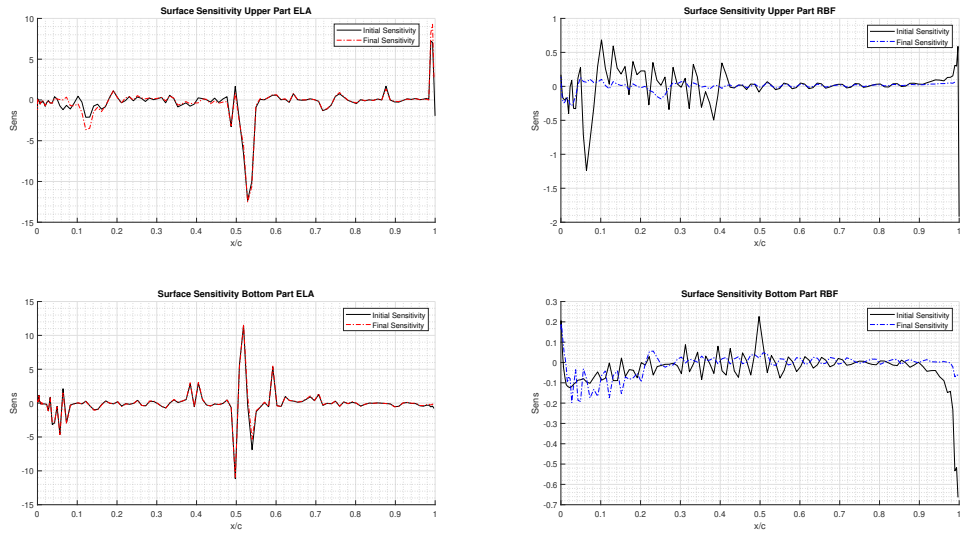

Figure 5.7: NACA0012: Initial and Final Surface Sensitivity

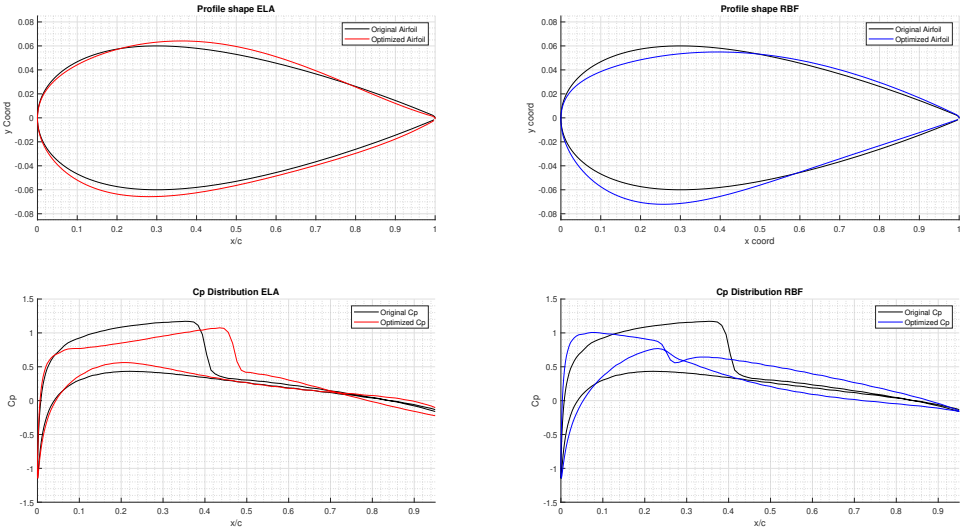

Figure 5.8: NACA0012: Airfoil Shape and C_p Distribution

The shock positioned on the upper surface of the airfoil in the second case is moved more forward, this means that the Mach one bubble has a shorter radius. The shock is subsequently less intense and the jump of pressure that impose is reduced.

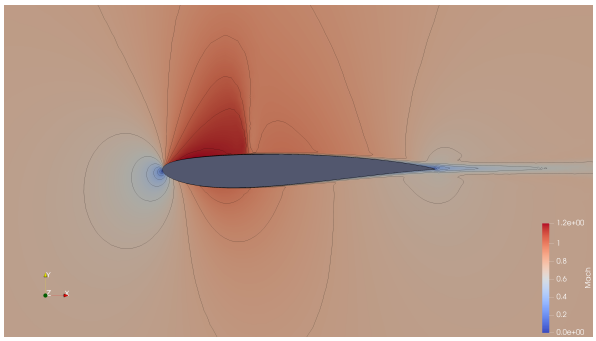

Figure 5.9: NACA0012: Final Mach Field with ELA

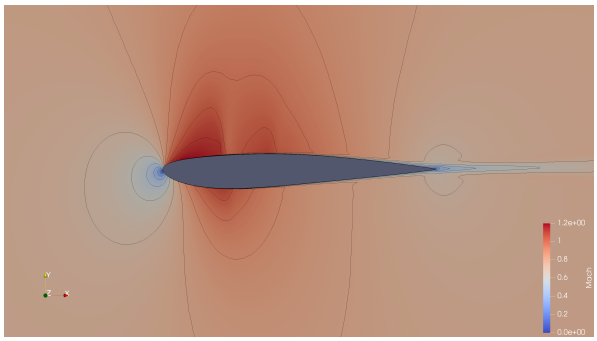

Figure 5.10: NACA0012: Final Mach Field with RBF

5.2 NACA0012 Efficiency Maximization

As in the previous section, a NACA0012 is optimized. The mesh is the same used for the drag minimization and the convergence of the aerodynamic coefficients with respect to the increasing number of cells is visible in Fig. 5.1. Moreover, the free stream conditions are always the one reported in Tab. 5.1. The optimization chain is repeated using first RBF and than ELA for the mesh deformation. Instead, the target of the optimization is different: the efficiency has to be maximize. Every maximization problem is easily transformed in a standard minimization changing the sign to the objective function. Also the constraints are untouched, lift and maximum thickness must not decrease. The hypothesis of frozen viscosity is assumed. This specific test case is not proposed by AIAA, although it is of interest to see the difference obtained between the two results touching just the objective function J.
First of all, it is investigated the number of design variables necessary to obtain the maximum Fig. 5.11, then the best case is selected and how the efficiency increase with the optimization loops is shown in Fig. 5.12.

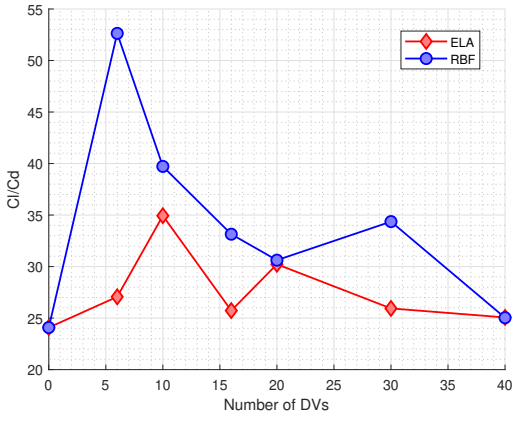

Figure 5.11: NACA0012: Efficiency Respect to DVs

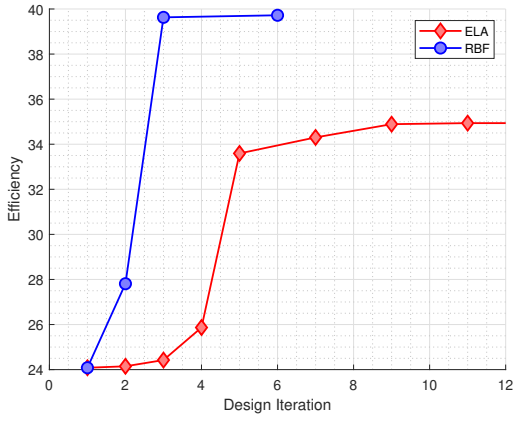

Figure 5.12: NACA0012: Efficiency w.r.t. Design Loops

From this two figure some differences and some analogies can be underlined comparing with the minimization of the drag of the previous section. The RBF method provides again the best results, besides

in this case less design variables are sufficient to achieve a great optimization and less deformations of the airfoil are computed before finding the maximum, six instead of nine. This can suggest that the starting shape of the NACA, therefore also the starting point of the optimization, is closer to the optimum. Comparing Fig. 5.13 with Fig. 5.21 it can be noticed that the surface sensitivity computed in this test case has an higher average value with remarkable picks. This means a greater gradient for the SLSQP algorithms, thus a quicker descent to the stabilization point.

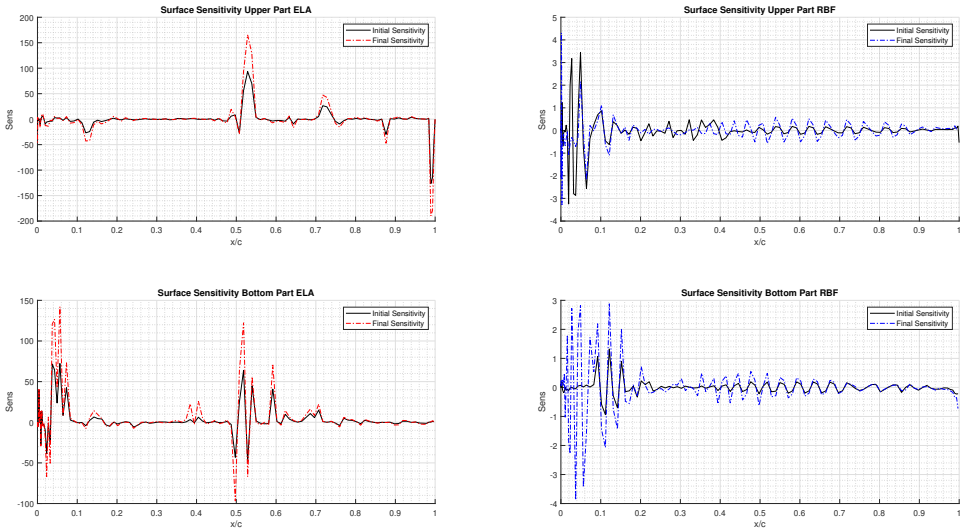

Figure 5.13: NACA0012: Initial and Final Surface Sensitivity for Efficiency

Using the RBF technique the efficiency is increased respect to the original airfoil of the 118%, the value reached with ELA is lower of the 48%. This is meanly linked to the fact that the shock wave on the upper part of the airfoil is strongly decreased by the optimization with RBF, the position is way closer to the leading point accordingly also the Mach one bubble radius is shorter. It is obtained generating a flatter upper part of the airfoil and most of the airfoil area is moved below the nose, see Fig. 5.14. It is interesting to notice from Fig. 5.16 that the change of shape is creating a shock wave in the lower part that was not present before.

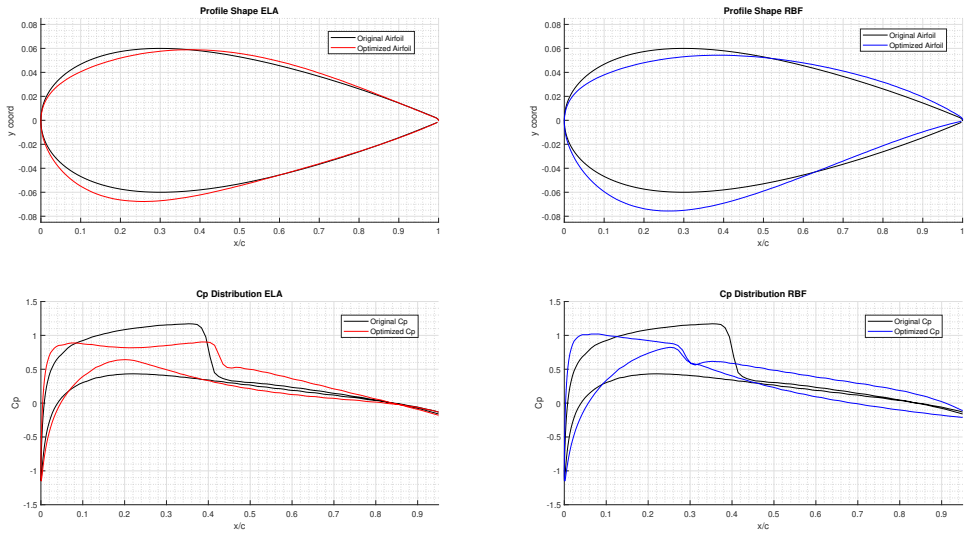

Figure 5.14: NACA0012: Profile Shape and C_p for Efficiency Maximization

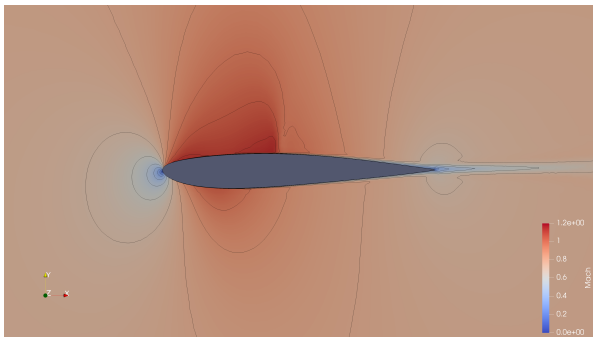

Figure 5.15: NACA0012: Final Mach Field with ELA for Maximum Efficiency

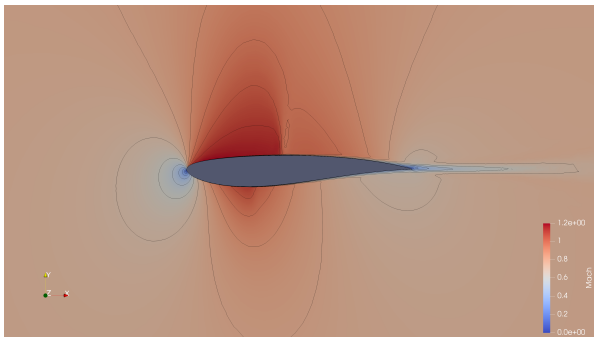

Figure 5.16: NACA0012: Final Mach Field with RBF for Maximum Efficiency

Since the mesh used, the free stream conditions and the constraints of the optimization are the same of the previous section, we can compare the results. Taking into account only the RBF data, the different final shape is shown in Fig. 5.17. In the second test case, the efficiency obtained is 30% higher, mainly since the lift is strongly increased.

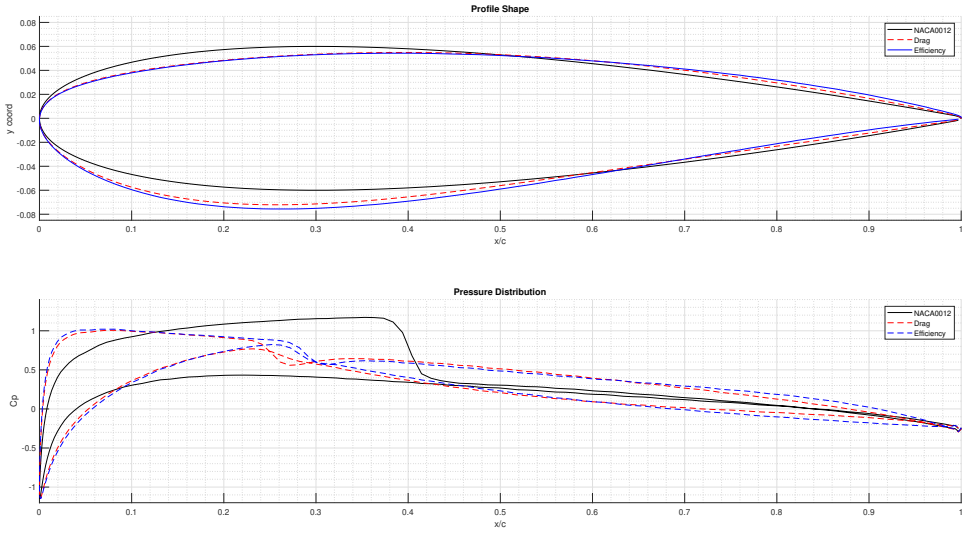

Figure 5.17: NACA0012: Airfoil Shape and C_p for Efficiency Maximization and Drag Minimization

5.3 RAE2822 Drag Minimization

The second benchmark test case is around the classical transonic airfoil RAE2822. At the chosen flow conditions, a strong shock is located on the airfoil upper surface as the Fig. 5.22 shows. Two different ways of minimizing the drag are analysed in this section, keeping fix the objective function and the geometrical constraints. Also the selected free stream conditions are slightly changed yet the results are strongly comparable. The second one is exactly the benchmark case proposed by AIAA group ADODG, which is faced introducing a multiphase optimization.

5.3.1 RAE2822 Double Adjoint

The goal is to reduce the C_d without decreasing the overall lift and the angle of attack. However, differently from the NACA, this time the area of the airfoil is fixed. The experimental data for the Cp comparison are provided by Cook [106]. The free stream conditions imposed are: Firstly, the mesh

Table 5.2: Free Stream Conditions RAE2822 1st Test

Mach	0.75
AoA	2.81°
Re	$6.2E6$
Temperature	$273.15K$

generated with Pointwise has to be reliable. Therefore, as previously done with the NACA0012 airfoil, a sequence of meshes with increasing number of segments on the profile is generated and tested. The aerodynamic coefficients must become stable in value as illustrated in the following figures:

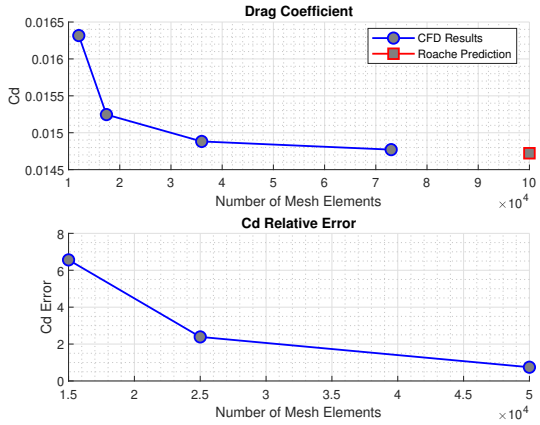

Figure 5.18: RAE2822: C_d Convergence

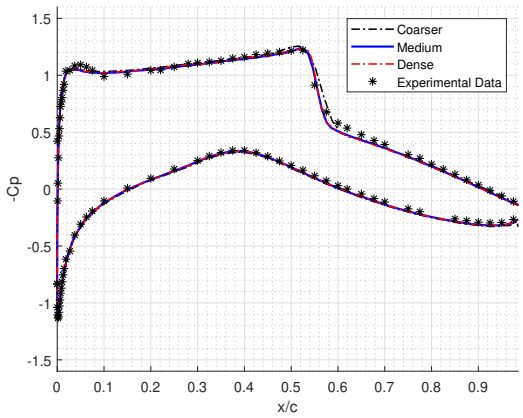

Figure 5.19: RAE2822: C_p Distribution

The same procedure as in the first benchmark test case is applied here. This time the two final forms of the airfoil are comparable and presented in the Fig. 5.24. The final shape of the FFD box is shown in Fig. 5.25 While using ten design variables the difference between the C_d obtained is less than the 1%. The gain of performance respect to the original airfoil for the selected flight conditions is further investigated in the next section.

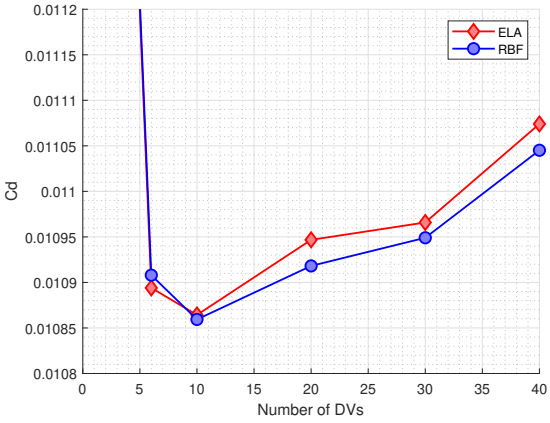

Figure 5.20: RAE2822: C_d Variation wrt DVs

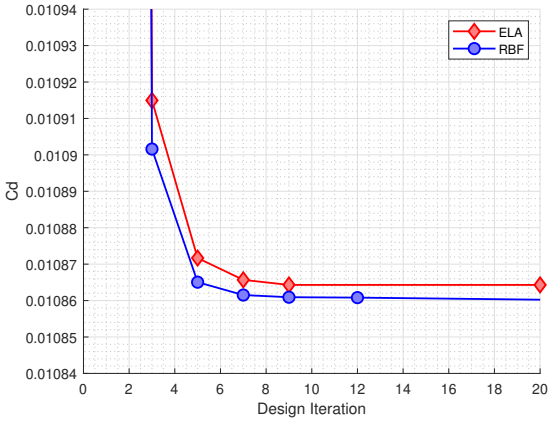

Figure 5.21: RAE2822: C_d Variation wrt Design Loops

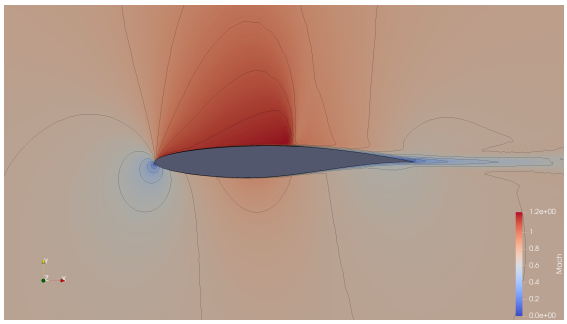

Figure 5.22: RAE2822: Original Mach Field

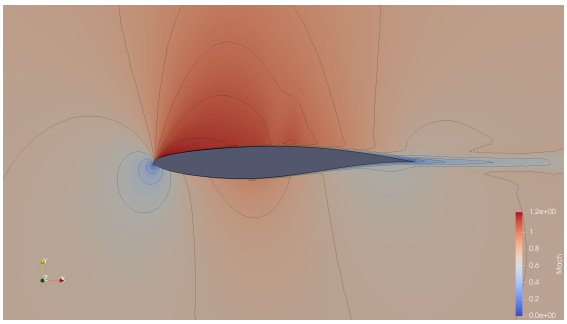

Figure 5.23: RAE2822: Final Mach Field

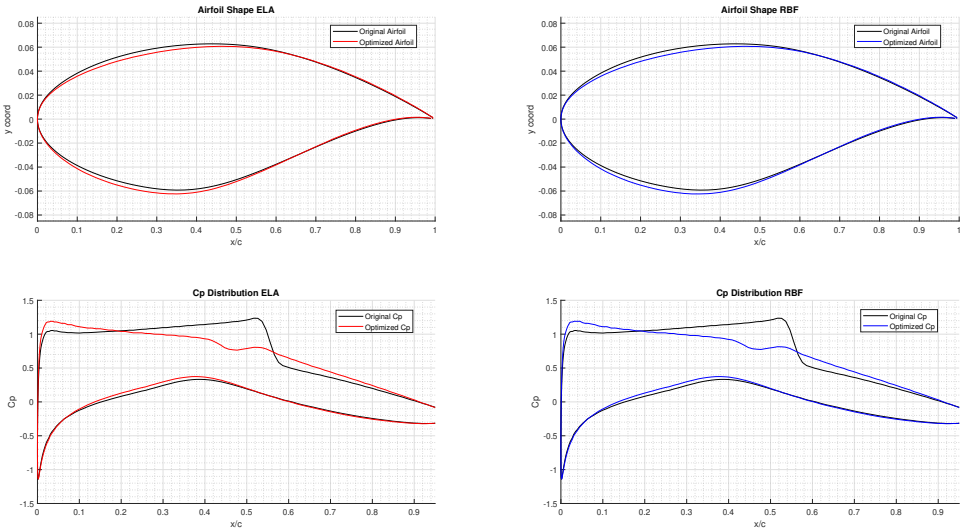

Figure 5.24: RAE2822: Profile Shape and C_p Distribution

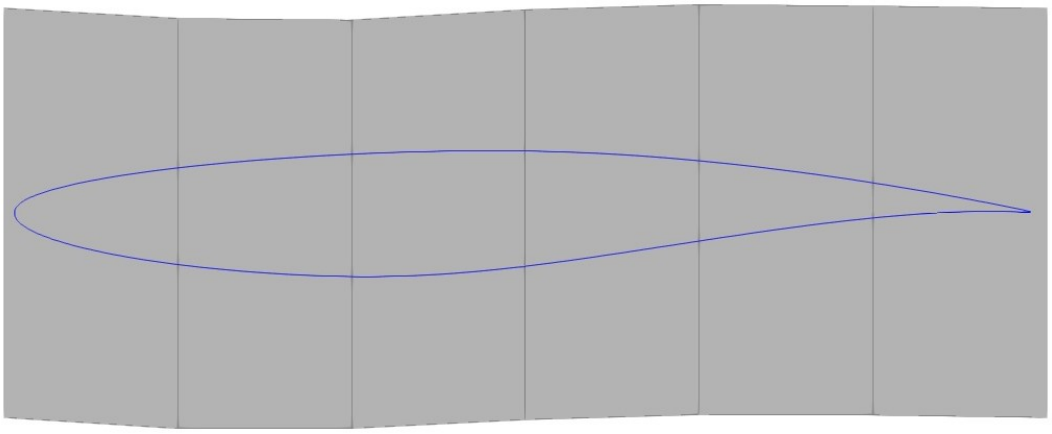

Figure 5.25: RAE2822: Final FFdbox

5.3.2 RAE2822 ADODG Benchmark Test

This time exactly the test case proposed by ADODG is followed. The target is always to decrease the drag keeping the lift fixed, although the angle of attack is free to change. Therefore, the direct simulation is conducted at fixed Cl. The area of the airfoil can not be reduced and the torque moment must no increase. The freestream conditions are reported in the following table:

Table 5.3: Free Stream Conditions RAE2822 2nd Test

Mach	0.729
Cl	0.829
Re	$6.5E6$
Temperature	$288.15K$

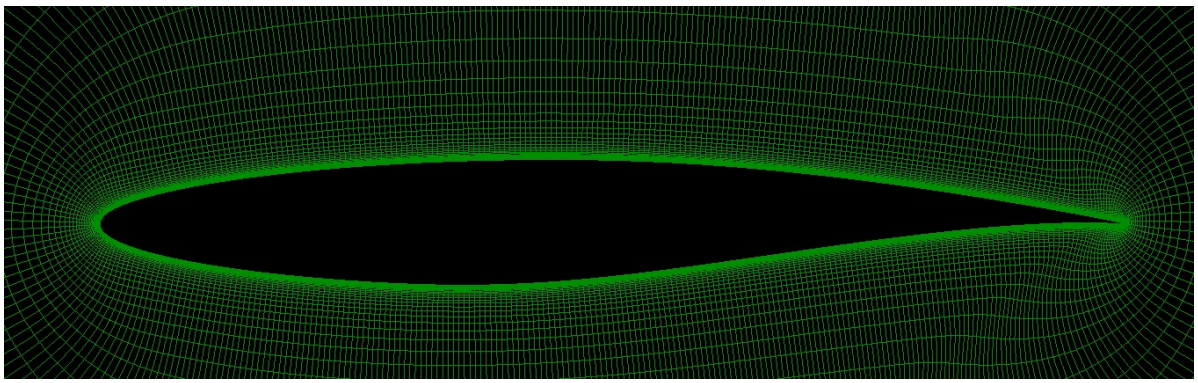

Figure 5.26: RAE2822: O Grid Mesh

An O-grid is generated around the same discretization of the airfoil selected in the previous case, the literature indicates that the best results are obtained for 252 wall nodes which is close to our segmen-

tation obtained after a mesh convergence very similar to the previous section therefore not reported again. A starting $y+ = 0.45$ is obtained, the first three cells have a constant height then a growth ratio of 1.2 is chosen, the farfield is of 150 chords. A preliminary study about the number of design variables is conducted, the complete optimization is performed using 16 DVs. A multi starting point strategy is adapted, the ASO is composed by three different stages. First the C_l is kept fixed during the direct simulation and the angle of attack is free to change. The initial process is stopped when the drag reduction between two consecutive adjoint evaluations is lower than 0.1 drag counts. Starting from the last deformed mesh a new ASO is performed, this time the AoA is kept fixed and the lift is conserved computing the corresponding sensitivity. The last phase is equal to the first one. Since the gradient-based algorithms are strongly influenced by the initial point and the minimum found are all locals, the scheme proposed helps to find a lower final value of $J(\alpha)$. The deformation of the mesh is executed applying RBF, set as in the previous section. It is interesting to notice that during the second stage the lift is increased even if the drag keeps diminishing, the convergence is obtained after a few loops, which underlines the proximity to the minimum point. The drag is reduced from 182.7 counts to 111.8, the shock wave is almost completely eliminated. In addition, ELA has been tested too, yet the results are not reported since the final drag value obtained is 10% higher.

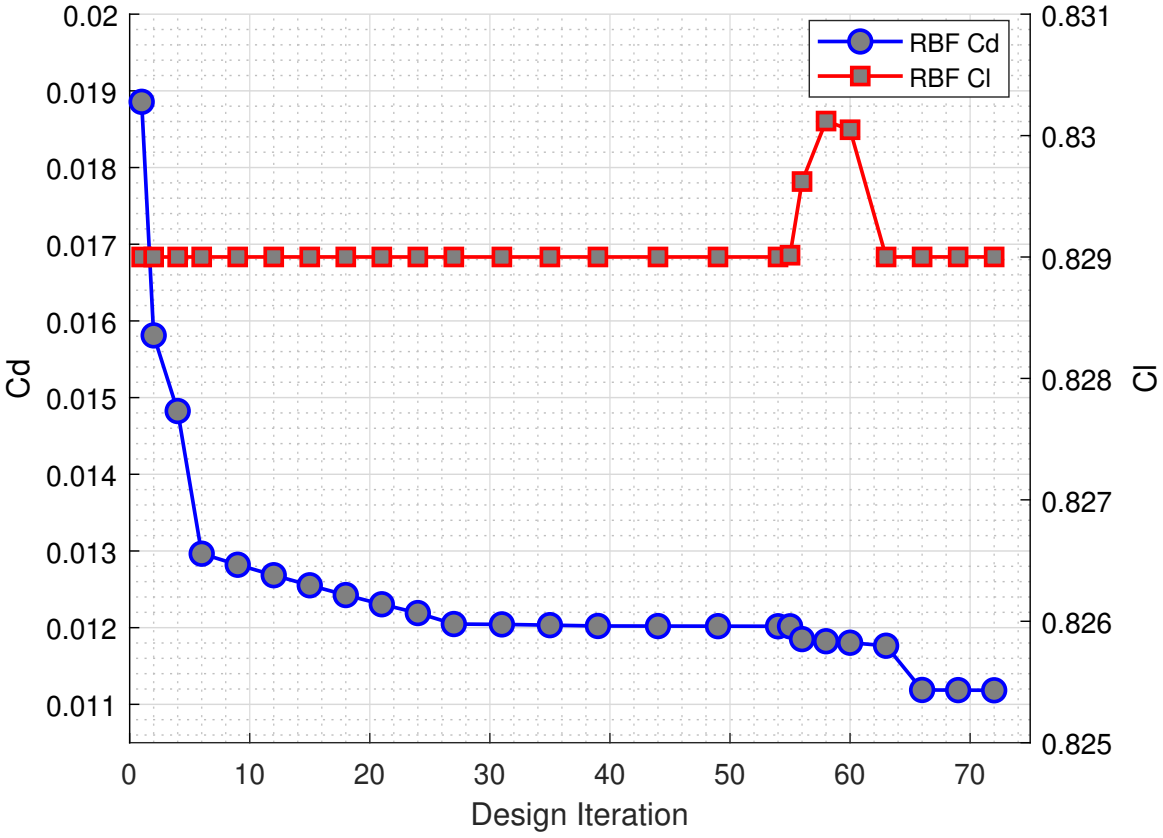

Figure 5.27: RAE2822 ADODG: Optimization Process

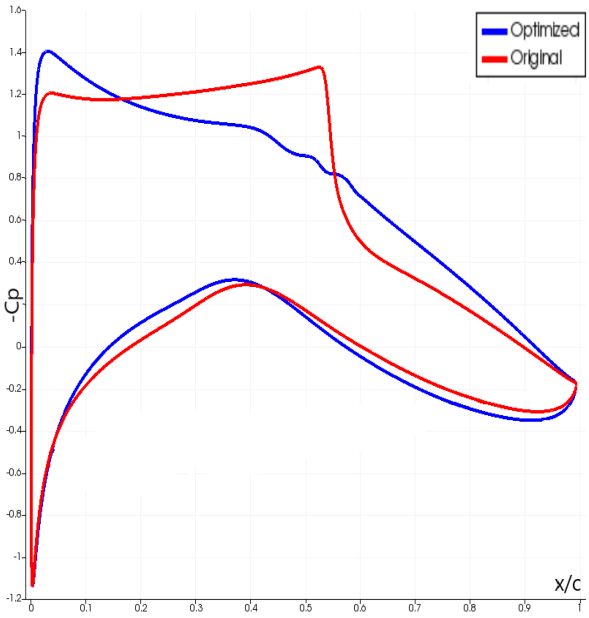

Figure 5.28: RAE2822 ADODG: Pressure Coefficient

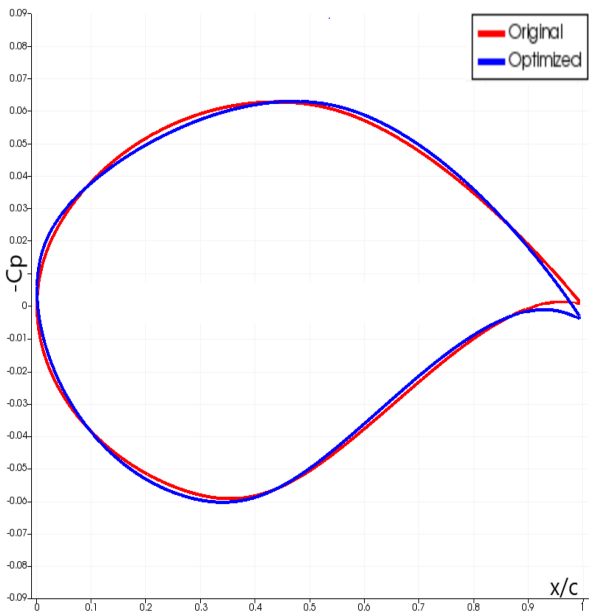

Figure 5.29: RAE2822 ADODG: Airfoil Shape

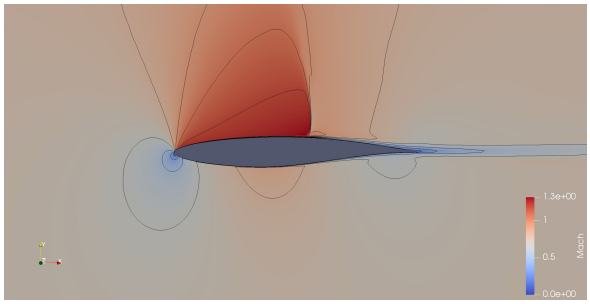

Figure 5.30: RAE2822 ADODG: Original Mach Field

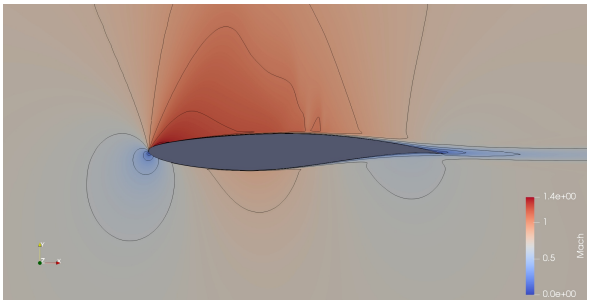

Figure 5.31: RAE2822 ADODG: Optimized Mach Field

He et al. [103] collected the results found in the literature about the RAE2822 benchmark test, our optimization did not achieve the best results however it is comparable with the others. The table is reported here, the values are in drag counts:

Table 5.4: RAE2822 Literature Results

Result	Cells	DVs	RAE 2822	Optimized	Reduction	%
He et al. [103]	131,071	40	194.4	108.9	-85.4	43.98%
Anderson et al. [107]	-	14	196.0	124.0	-72.0	36.73%
Bisson and Nadarajah [108]	3,264	16	177.8	102.3	-75.5	42.46%
Carrier et al. [10]	7,894,172	10	189.2	103.9	-85.3	45.08%
Gariepy et al. [109]	211,968	24	187.3	104.3	-83.0	44.31%
Lee et al. [79]	47,824	17	234.4	131.8	-102.6	43.77%
Poole et al. [110]	98,304	6	174.3	90.4	-83.9	48.13%
Zhang et al. [111]	165,888	18	194.0	103.62	-90.47	46.58%
Present work RBF	40,050	16	188.57	111.18	-77.52	41.08%

5.3.3 2D Mesh Deformation Performance

Taking advantage of the data obtained with the optimization of the RAE2822, certain aspects of the mesh deformation step are highlighted in this section. First of all, it is important that the quality of the mesh does not decrease due to the deformation. Some metrics are used to identify the quality such as volume ratio, skewness, orthogonality. In the context of the thesis, the orthogonality angle is used to evaluate the mesh quality. For bidimensional optimization, the displacement of the surface nodes is limited, both ELA and RBF methods preserve the quality of the mesh as shown in Fig. 5.32 and Fig. 5.33.

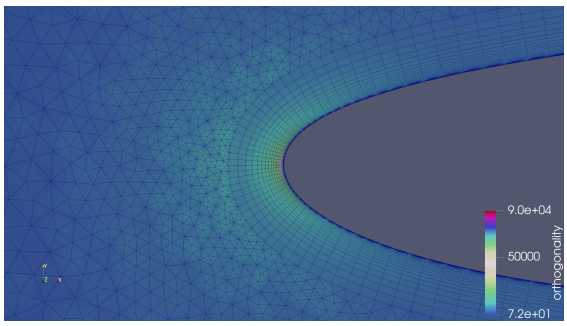

Figure 5.32: RAE2822: Mesh Orthogonality with ELA

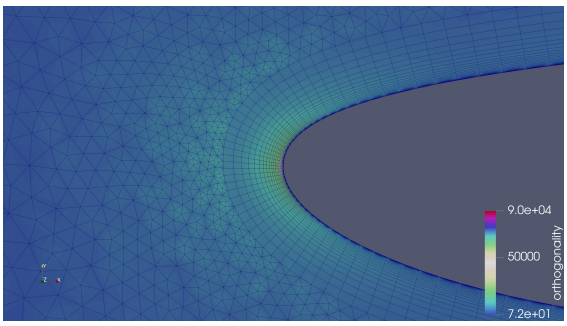

Figure 5.33: RAE2822: Mesh Orthogonality with RBF

More interesting is the comparison between the two methods of the ram consumption. Regarding RBF, the ram usage is related to the number of levels performed by the greedy algorithms. This can influence the number of control points selected on the surface. The second parameter is the volume reduction factor k, which in 2D has been noticed to be almost irrelevant thus it is kept fixed to ten. The deformation of the mesh with RBF has been repeated three times increasing the number of levels. The control points used are shown in the next table and figures.

Table 5.5: Mesh Deformation RAM Consumption

Type	Control Points	RAM
RBF one level	11	91Mb
RBF two level	43	93 Mb
RBF three level	253	108 Mb
Linear elasticity	all	680 Mb

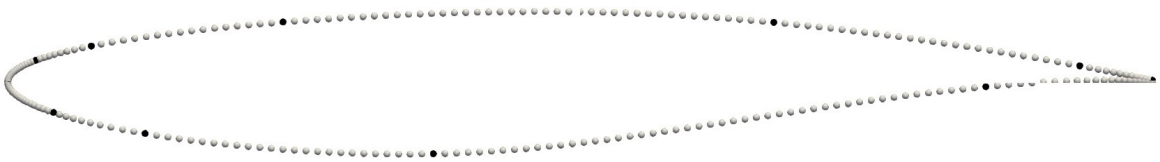

Figure 5.34: RAE2822: Control Points 1 Level

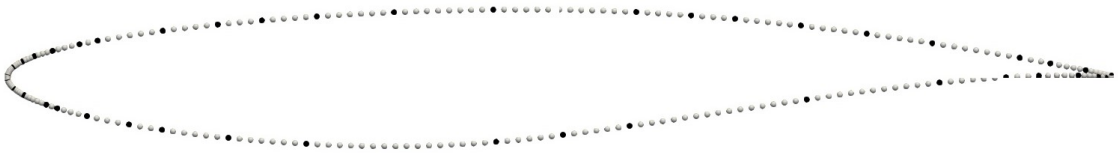

Figure 5.35: RAE2822: Control Points 2 Levels

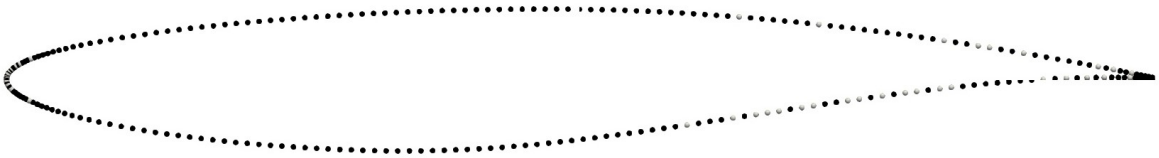

Figure 5.36: RAE2822: Control Points 3 Levels

The evolution in time of the allocated virtual memory during in the deformation process is shown in Fig. **??** for RBF with different level of selection of the control points, instead in Fig. 5.40 for the ELA method. Almost a factor of six is evident between the two. If a single level greedy algorithm is used in RBF, also the wall time is slightly lower, instead with two levels the two methods become comparable concerning the time.

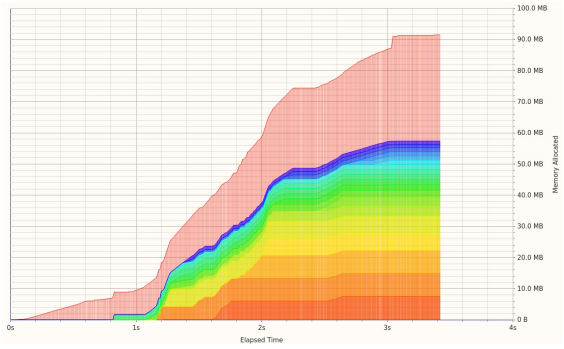

Figure 5.37: RAE2822: RAM Usage RBF 1 Level

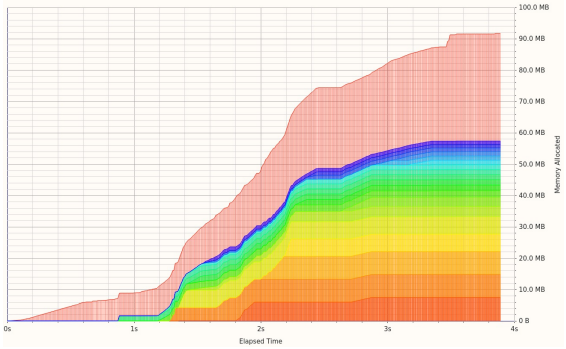

Figure 5.38: RAE2822: RAM Usage RBF 2 Levels

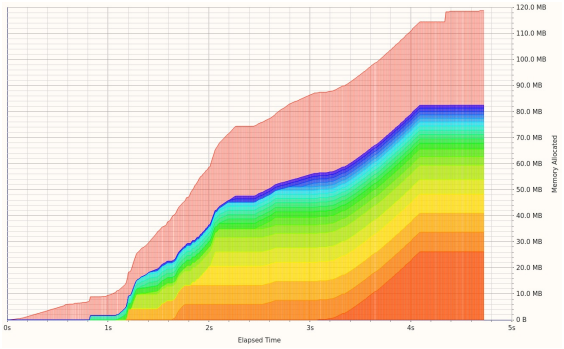

Figure 5.39: RAE2822: RAM Usage RBF 1 Level

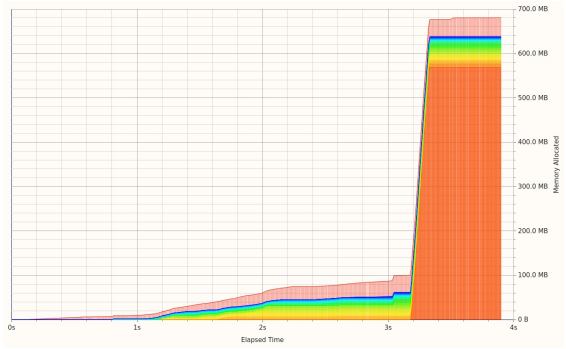

Figure 5.40: RAE2822: RAM Usage ELA

Chapter 6

3D Results

6.1 Onera M6 Drag Minimization

The ONERA M6 can be described as a swept, semi-span wing with no twist. The symmetric ONERA D section is used as an airfoil. It is a typical test case for turbulence flow over a transonic wing, widely adopted for CFD validation. Experimental data for the comparison of the pressure distribution are provided in [112]. The flight conditions are chosen to deal with a strong shock on the upper part of the wing collocated close to the 25% of the chord.

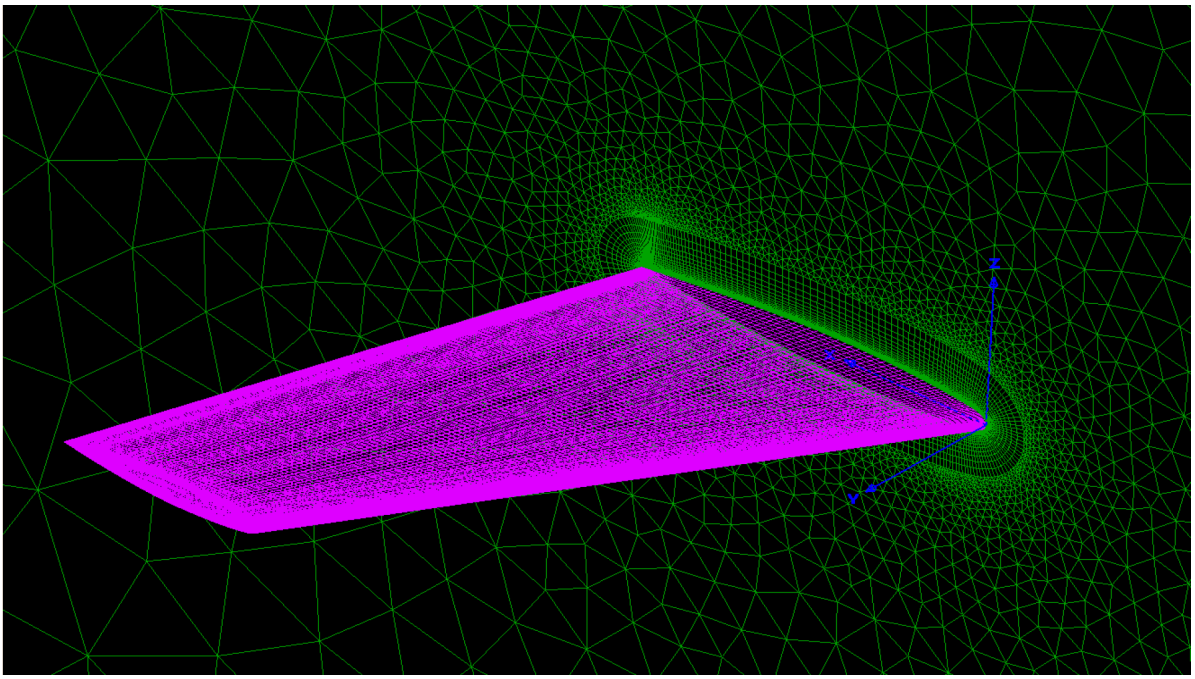

Figure 6.1: Onera M6: Medium Mesh

Table 6.1: Free Stream Conditions Onera M6

Mach	0.84
AoA	3.06
Re	$14.6E6$
Temperature	$300K$

Firstly, mesh convergence has been performed, four hybrid meshes are generated. The smallest one, even if it is able to properly predict the intensity of the shock and the pressure distribution, it is then used for optimization. This choice is dictated by the computational power available. The first three layers of the mesh have constant height than the growth ratio decrease from the coarse to the finer grid. The number of layers of the structured part has an opposite behaviour. Since the surface grid is almost everywhere structured, the number of points in the x,y direction are easily multiplied by a factor of 1.5 for the convergence. The following figures report the aerodynamic coefficients obtained and the relative error, in accordance to the bi-dimensional case, also the predicted value for an infinity dense mesh is marked. Moreover, the C_p distribution at four different stations is monitored.

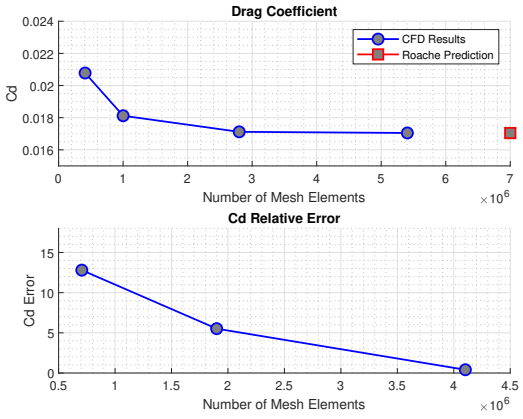

Figure 6.2: Onera M6: C_d Convergence

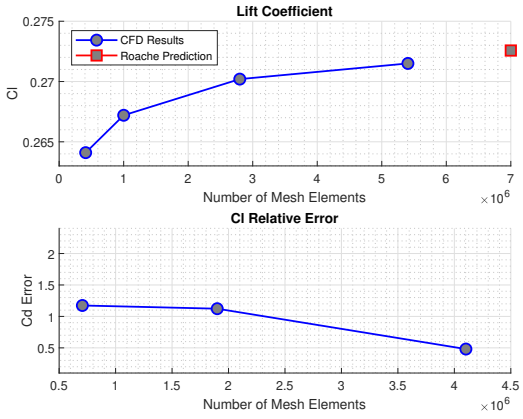

Figure 6.3: Onera M6: C_l Convergence

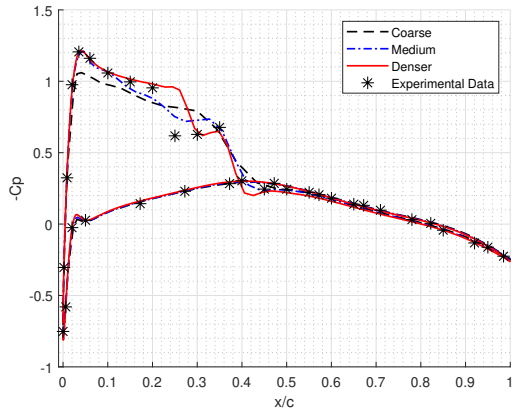

Figure 6.4: Onera M6: C_p Convergence y=80%b

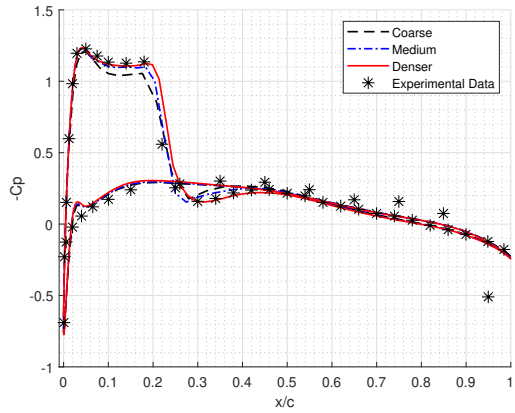

Figure 6.5: Onera M6: C_p Convergence y=95%b

The goal of the optimization is to reduce the drag as much as possible without decreasing the lift and the volume occupied by the body. The angle of attack is free to change, the variation of the torque is only monitored. Only six geometrical constraints are imposed: the final maximum thickness at different span stations cannot be lower than the 75% of the initial value. The gradient based optimization using the SLSQP algorithm is performed once with RBF as mesh deformation method, then with ELA. Regarding RBF, the maximum number of control points selectable is the 10% of the surface's nodes, the volume reduction factor k is set at 5 and the Wendland $C0$ is selected as the interpolation function. This configuration makes RBF really robust, fast, and computationally cheap. Instead, considering ELA a final residual of $10^{(-10)}$ is required for the solution of the linear system and the stiffness of the cells is computed inversely with respect to their volume. Free Form deformation volume approach is used to parametrize the wing surface, no study was conducted to see the effect of different parametrization, luckily [113] compared FFD with B-splines. The drag reduction was very similar suggesting to choose FFD in this thesis since it makes easier to manipulate complex geometry. Results are reported in the following figures:

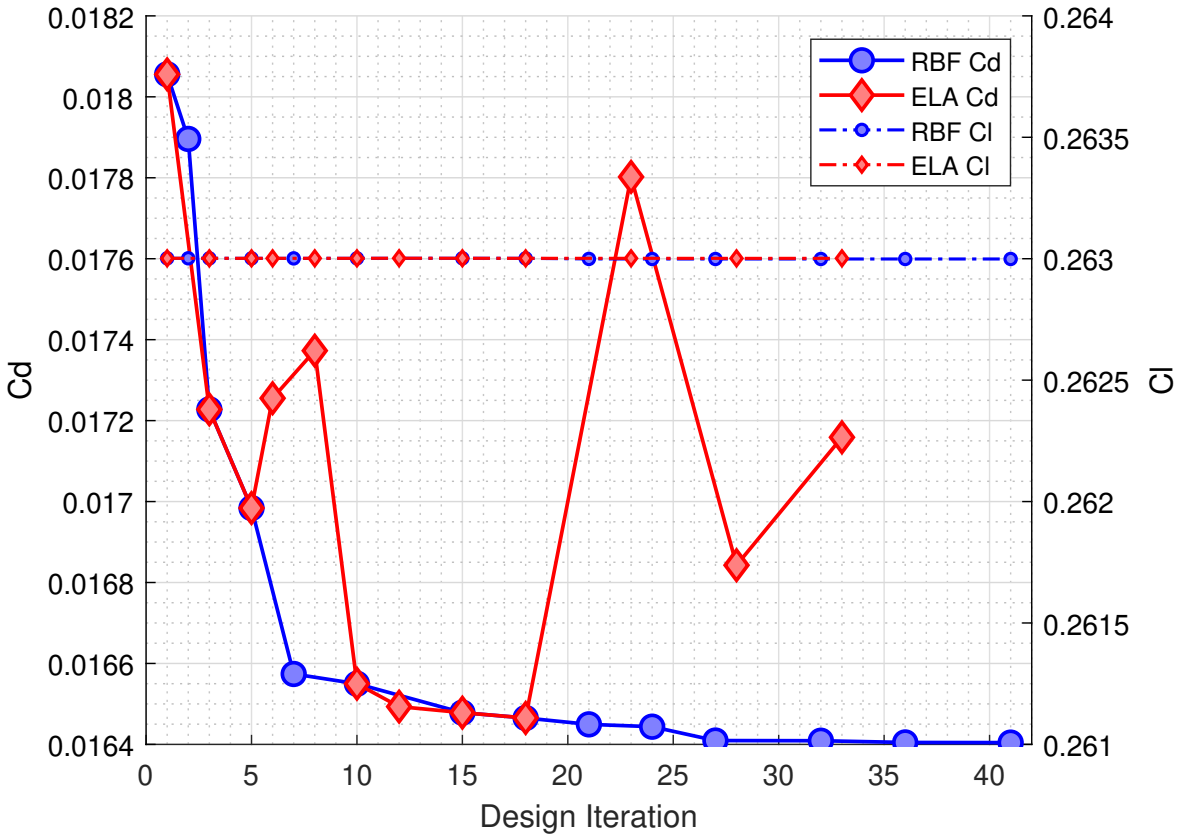

Figure 6.6: Onera M6: C_d Variation

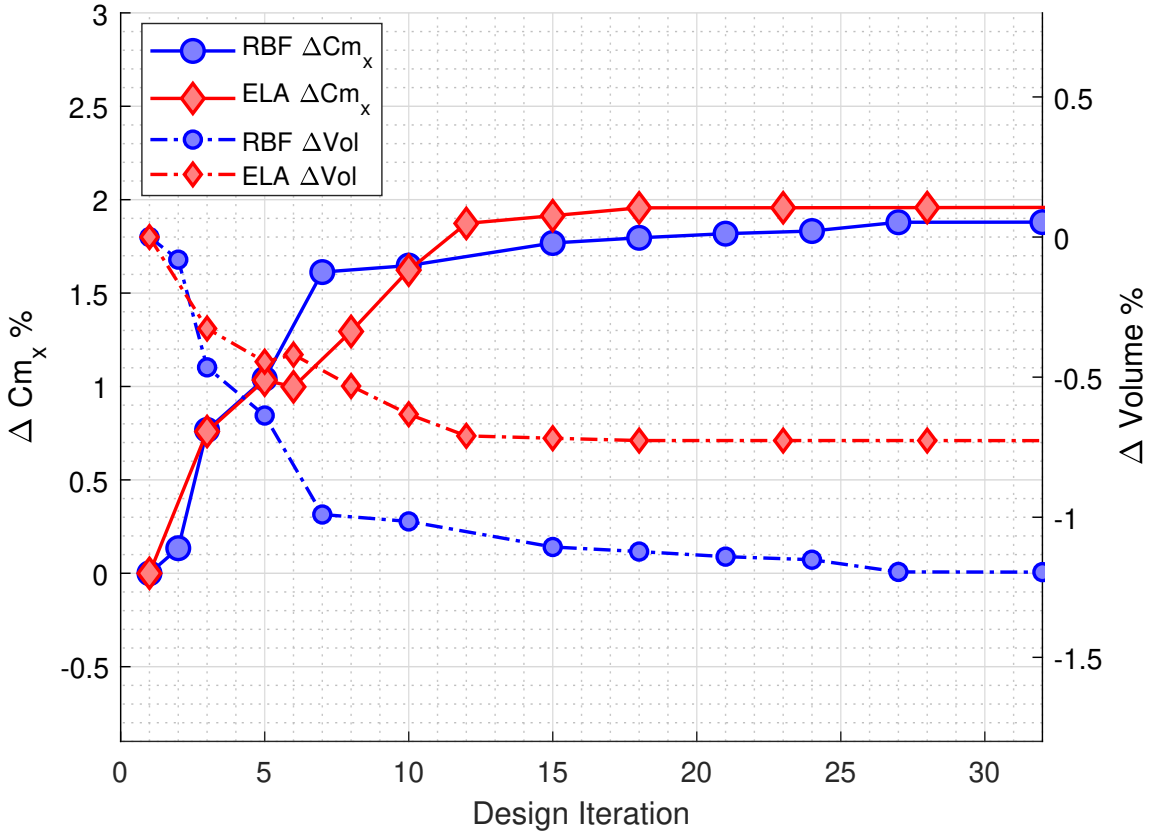

Figure 6.7: Onera M6: volume and torque variation

Fig 6.6 clearly shows that both methods are able to conserve the lift and reduce the drag, however the optimization process seems to be way more robust using RBF with a more clear convergence. The process is interrupted after 72 hours of computation. Regarding RBF, the overall drag reduction is around 9.15%, particularly the drag reduction between the last two loops is insignificant underlying the achievement of a minimum. Instead, ELA obtains a slightly smaller reduction of the C_d around 8.57%. The evolution of the drag with respect to the deformation loops is oscillating, this behaviour has been observed also by other researchers [114]. Further, the optimum is obtained at the $18th$ iteration than a big jump happens and this seems to be typical of SU2 [115]. Both method increase the torque of the 2% and slightly reduce the volume even if a clear geometrical constraint is imposed. How the shape of the sections and thus the pressure distribution are modified after the optimization is shown in the next pics. The sections close to the root are more morphed, instead the tip of the wing is just more twisted. Especially close to the wing's tip, the peak of suction is decreased and this results in a less intense shock wave, accordingly a lower jump of pressure Fig. 6.10. The position of the shock does not change as we found for the 2d test cases.

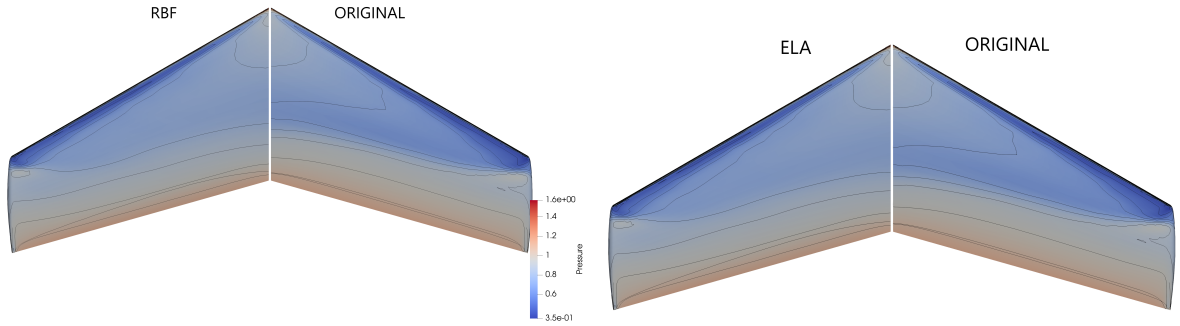

Figure 6.8: Onera M6: Isopressure Lines Upper Surface

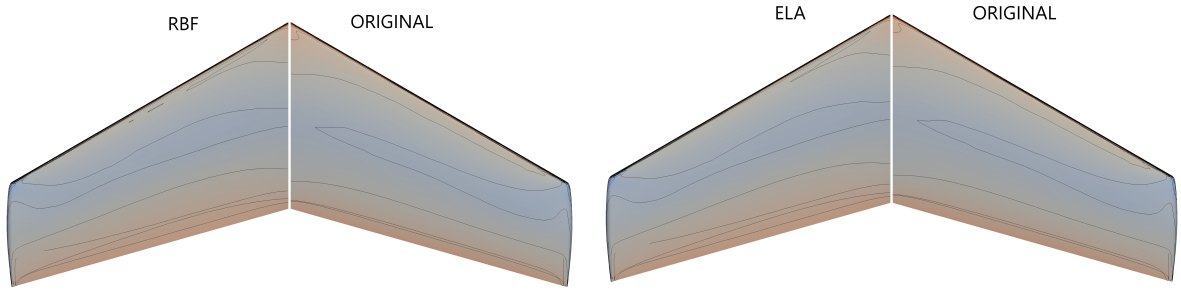

Figure 6.9: Onera M6: Isopressure Lines Bottom Surface

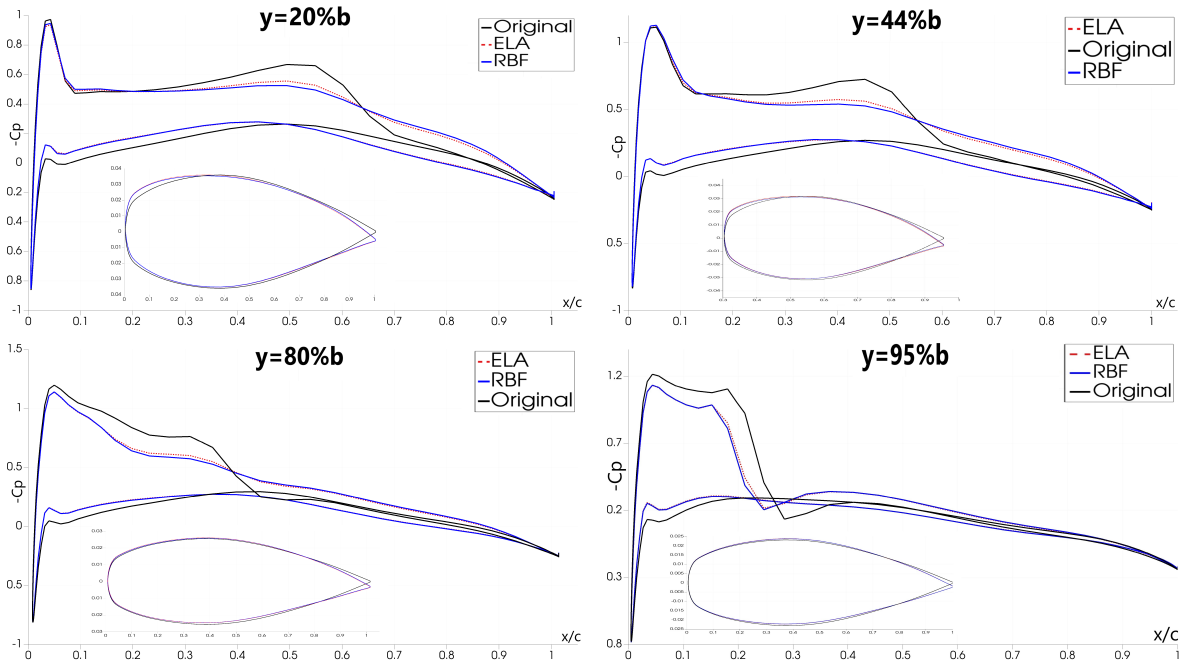

Figure 6.10: Onera M6: C_p Optimization

Moreover, Fig. 6.11 shows that the control points selected on the upper part of the wing, the distribution is strictly linked to the shape of the deformation wanted. In this case, the movement was the result of the first loop of optimization and it is evident that the tip is not touched instead the leading edge and the central part of the wing are more affected. The region of greater displacement should correspond to a zone of the surface with higher sensitive. It is confirmed by Fig. 6.12 where is shown the value of the adjoint variables for the last equation of the adjoint system and the magnitude of the three momentum equations combined.

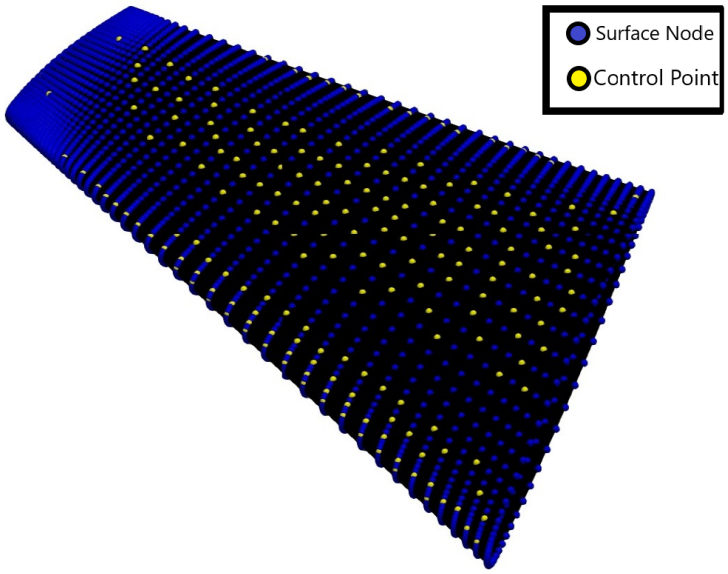

Figure 6.11: Onera M6: RBF Case 1 Control Points

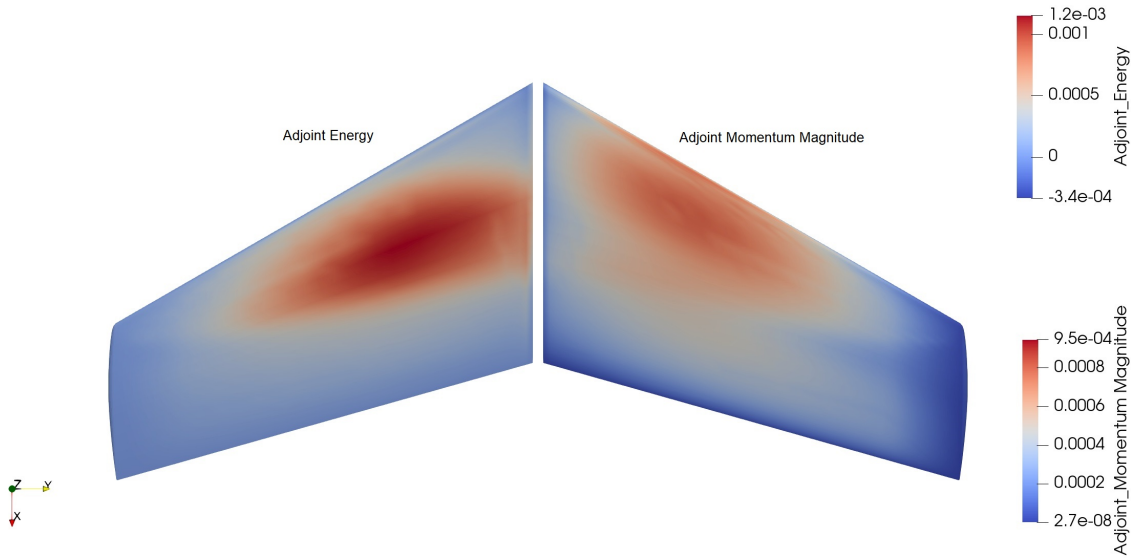

Figure 6.12: Onera M6: Adjoint Variable

6.1.1 Mesh Deformation Performance

The Onera M6 test case has been used to compare the wall time and the ram consumption of the two mesh deformation considered in this text. The grid considered is hybrid, with 7319 surface nodes and 446685 cells. The performances of RBF are measured considering different settings: interpolation function, k, number of levels, amount of control points. The mesh deformation processes are executed on a single core of an AMD EPYC^{TM} of 2.4 GHz. The latter are imposed as a percentage of the total surface points. The next table reports the combination of the parameters tested and the results obtained.

Table 6.2: RBF Parameters

N.	Wendland	Control Points	K	N. Levels
0	$C0$	5%	5	one
1	$C0$	10%	5	one
2	$C0$	15%	5	one
3	$C0$	10%	10	one
4	$C0$	10%	5	two
5	$C2$	10%	5	one

Three outputs are of our interest: the maximum virtual memory allocated, the wall time excluded the deallocation of the data and the error due to the surface interpolation. Regarding ELA, as said in the previous section, the stiffness of the cells is set with an inverse volume logical. The final linear system has to be solved with a final residual of $10^{(-10)}$ in maximum 800 iterations which are largely sufficient, this is the classical setting for the linear elasticity analogy proposed in the SU2 tutorials. The results are reported in the next table:

Table 6.3: Performance Results

N.	RAM (G_b)	Interpolation Error	Cpu Time (min)
RBF.0	1.94	2.18%	1.81
RBF.1	2.55	1.07%	3.33
RBF.2	5.41	0.51%	7.89
RBF.3	2.55	0.98%	3.4
RBF.4	3.68	0.47%	6.46
RBF.5	2.74	0.15%	6.53
ELA	14.8	0%	13.66

The comparison between ELA and the RBF number one, which is used for the optimization, shows remarkable results. The RAM usage is almost six times lower and the time employed is one quarter. Considering only RBF, in order to have a very low interpolation error maintaining excellent performance, the comparison shows that is better to increase the order of the interpolation function instead of the number of levels of the greedy algorithm. The maximum number of control points selectable should be the 10% of surface nodes. In every 3D optimization ELA has shown a better ability to conserve the volume of the wing, it could be linked to the interpolation error introduced using RBF. This underline

the necessity to contain as much as possible the approximation introduced. The following figures show the allocation of the data in the virtual memory with respect to the cpu time. It can be noticed that the behavior of RBF and ELA are completely different, RBF is progressive, instead ELA quickly allocate all necessary information, then the ram consumption remains constant until the linear system is solved.

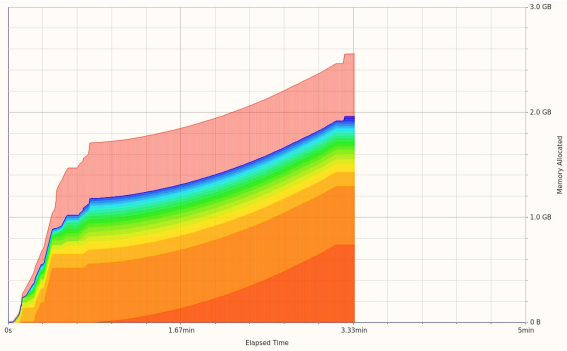

Figure 6.13: Onera M6: Performance Case 1 Rbf

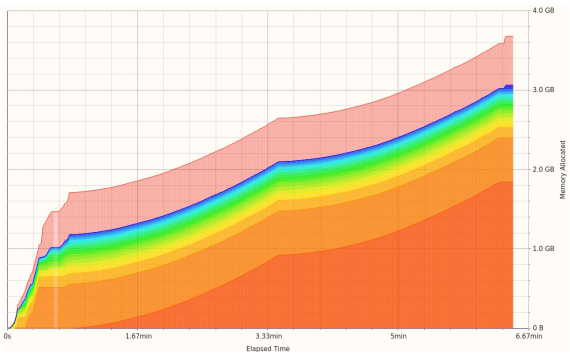

Figure 6.14: Onera M6: Performance Case 4 Rbf

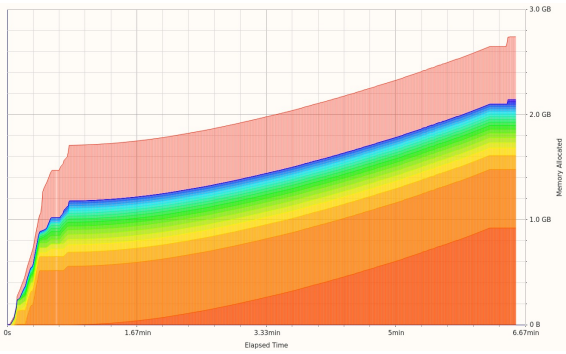

Figure 6.15: Onera M6: Performance Case 5 Rbf

Figure 6.16: Onera M6: Performance Ela

6.2 CRM Wing Drag Minimization

The NASA CRM geometry was thought for validation studies of CFD codes [116]. AIAA Aerodynamic Design Optimization Discussion Group proposes to optimise only the wing extracted from the complete CRM cad where also the body, the nacelle and the tail are present. The wing is developed for transonic conditions, with a blunt trailing edge and a strong similarity with the Boeing 777-200. Respect to the Boeing wing it has 10.3% less wet area and in order to make it more attractive for research purposes a strong pressure recovery is introduced in the outboard wing. The wing is optimised for working in combination with the fuselage, accordingly its performance is degraded when the body and the tail are removed. Furthermore, the root is positioned on the symmetry plane and all the coordinates are divided with the mean aerodynamic chord obtaining a reference length of one meter. Since the reference area is mesh dependent, the assigned value of $3.407014m$ is fixed. Many information about how to properly generate a mesh for this geometry are present in literature since CRM is used by AIAA for the CFD drag prediction Workshop [117]. However, the computational resources available for this thesis are limited, especially the bottleneck is the maximum RAM of $64Gb$. To not exceed this value during the adjoint simulation maximum a mesh of 0.8 million of cells can be used, therefore the drag is not properly captured. The main part of the wing's surface is discretized with a structured grid, at the trailing edge the first cell length is 0.075% of the local chord instead the value of 0.1% is

chosen for the leading edge . The trailing edge i vertically cut and dived in 11 cells. Instead, for the tip an unstructured surface mesh is applied. Regarding the volume grid, the first cell size guarantee a maximum $y+ < 0.43$, the first three layer have constant thickness that a growth rate of 1.3 is used. The farfield is fixed at 100 times the half span of the wing which is $3.76m$.
A mesh convergence is performed at fixed C_l, the relative error between the different meshes is small and the starting drag not far from the one obtained in the reference paper thus the optimization will provide reliable results. The second mesh is selected for the ASO which represents a good balance between the convergence of the aerodynamic coefficients and the computational power available.

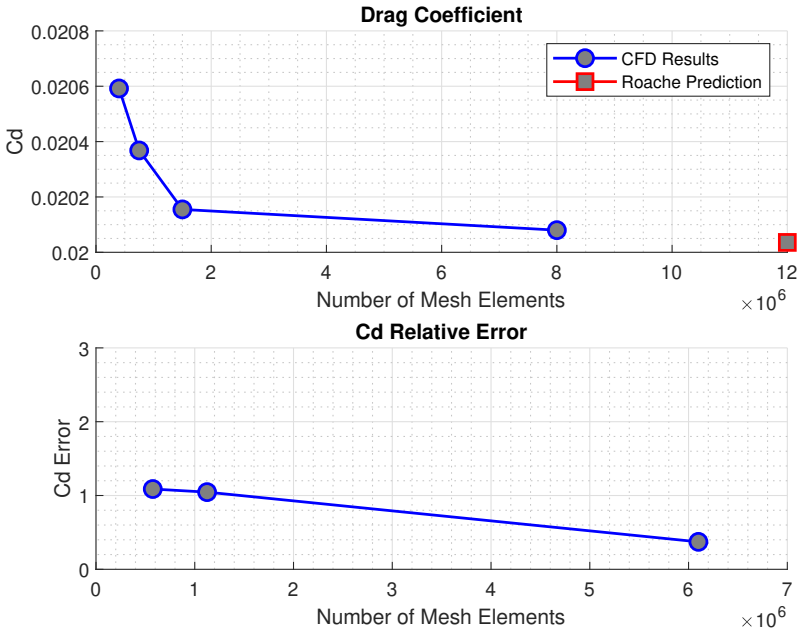

Figure 6.17: CRM Mesh Convergence

The optimization's target is reducing the drag coefficient without decreasing the lift and increasing the pitching moment. No constraint is imposed on the moment coefficients, they are just monitored. The free stream conditions selected are:

Table 6.4: Free Stream Conditions CRM Wing

Mach	0.85
Cl	0.5
Re	$5E6$
Temperature	$288.15K$

The angle of attack is free to change to adjust the lift and the internal volume is forced to be equal or grater to the baseline one. ADODG proposes to impose a maximum thickness of $\frac{1}{4}$ with respect to the original at sections A : 2.35%, B:26.7%, C : 55.7%, D : 69.7%, E : 82.8% and F : 94.4% of the span. These values are unrealistic, however it helps to guide the optimization process to achieve a reasonable final shape. The results of reference belongs to the MDO laboratory research group of

Prof. Martins. Working on adjoint optimization since 2002, they produced extraordinary results on the CRM contained in the publication of the 2018 [114], with a reduction of the drag of 8.5%. However, the root of the wing is free to change and, as it can be observe in the pictures of that paper, the final thickness of the root strongly increased cancelling the shock wave in that zone. This is not possible in SU2 where the root has to be constrained, in my opinion this is even more meaningful since the wing is extracted from a complete aircraft and the optimization of the root should be done combined with the fuselage. Therefore, to be able to compare our values with them we decided to replicate the test case proposed in section 4.2 of [114]. The only difference between the ADODG benchmarks is the geometrical constraints are imposed at 30 spanwise sections and the root cannot twist. They obtained that the drag is cut from 206.7 to 196.6 counts, which corresponds to a reduction of 4.83%. The type of grid that they are using is very different, an o-grid with high growth ratio. They implemented a multi stage optimization method that we cannot use due computational resource constraints. They start with a very coarse mesh, when the optimization slows down they generate a denser and more accurate mesh on the morphed surface and proceed with the ASO. The initial grid is composed by half million cells to ends with 28 million volumes which is simple not affordable for us.

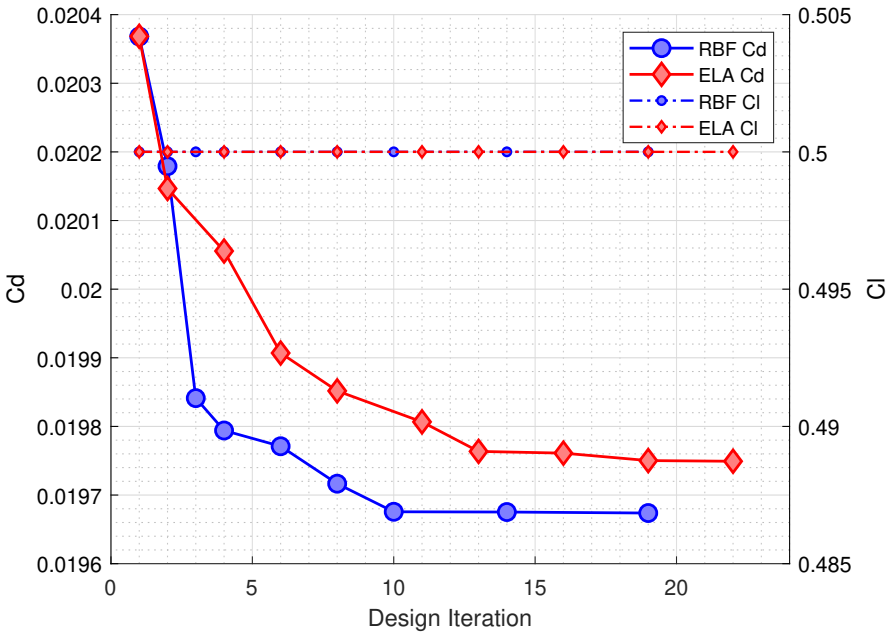

Figure 6.18: CRM: Cd Optimization

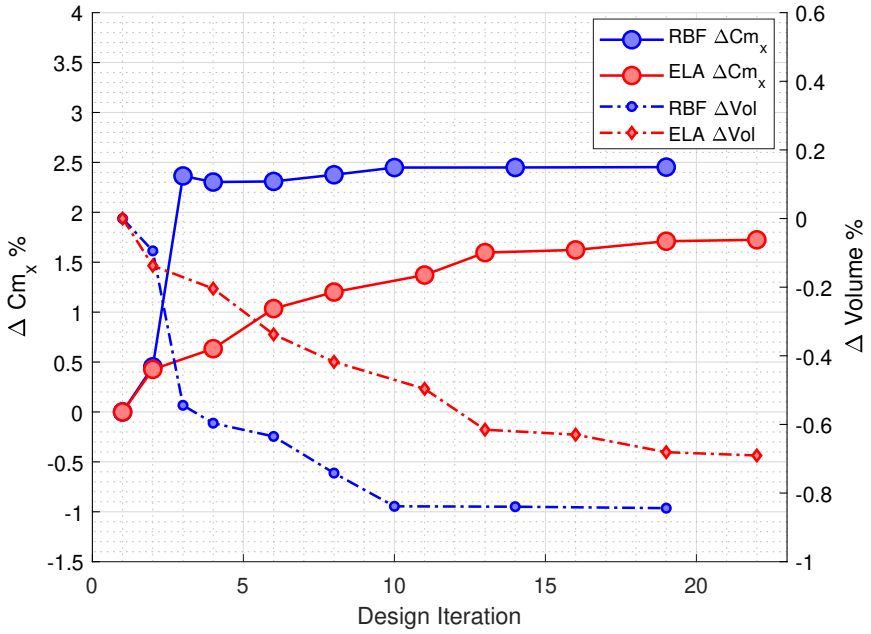

Figure 6.19: CRM: Volume and Cm_y Variation

The wing is entirely immersed into a single FFD parallelepiped. In this reference case 128 DVs are used, equally distributed on 8 spanwise planes. There is no direct control of the section rotation, the twist optimization is performed by the z-direction movement of the FFD vertices. Regarding the mesh deformation, RBF can select as control points maximum 10% of the surface nodes and the more accurate Wendland $C2$ is used as a base for the interpolation. A single level greedy point is applied, the intrinsic error is for the entire ASO lower than 0.6%. ELA is set as in the other sections. The improvement of the C_p plot and the shape deformation are reported about sections B,C,D and F. The history of the optimization reported in Fig. 6.19between the two methods seems to be identical. Once more the optimization obtained with RBF provides a better result. The volume is reduced of 0.8%, the other constraints are perfectly respected. The central portion of the wing is strongly morphed, the wing is less twisted, the lower part has more gentle curvature, instead the upper part is more rounded. The drag is reduced from the starting 203.68 0f 196.74 drag counts, which corresponds to a percentage of 3.41%. The shock wave is reduced especially close to the tip which is more free to deform. Fig. 6.22 and 6.23 highlight how the average value of the surface sensitivity at the end of the optimization is decreased, which indicates that the optimization is working and converging. The orthogonality of the mesh is reported for the starting grid and the final one, ELA shows a progressive decay of the grid quality:

Table 6.5: CRM Mesh Orthogonality

Deformation Method	Initial	Final
RBF	27.12	27.06
ELA	27.12	-29.80

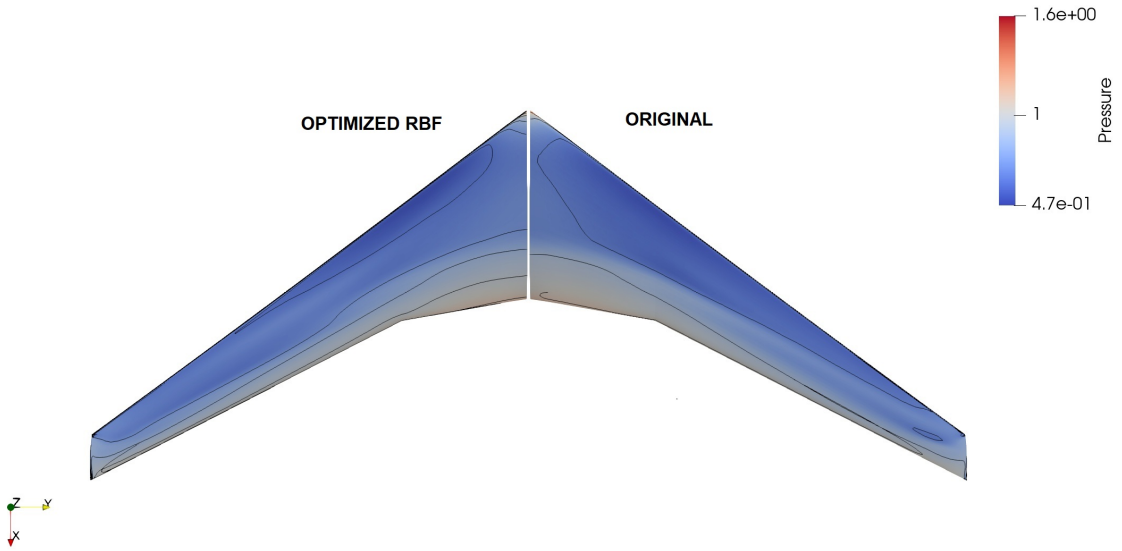

Figure 6.20: CRM: Pressure Isolines Upper Part

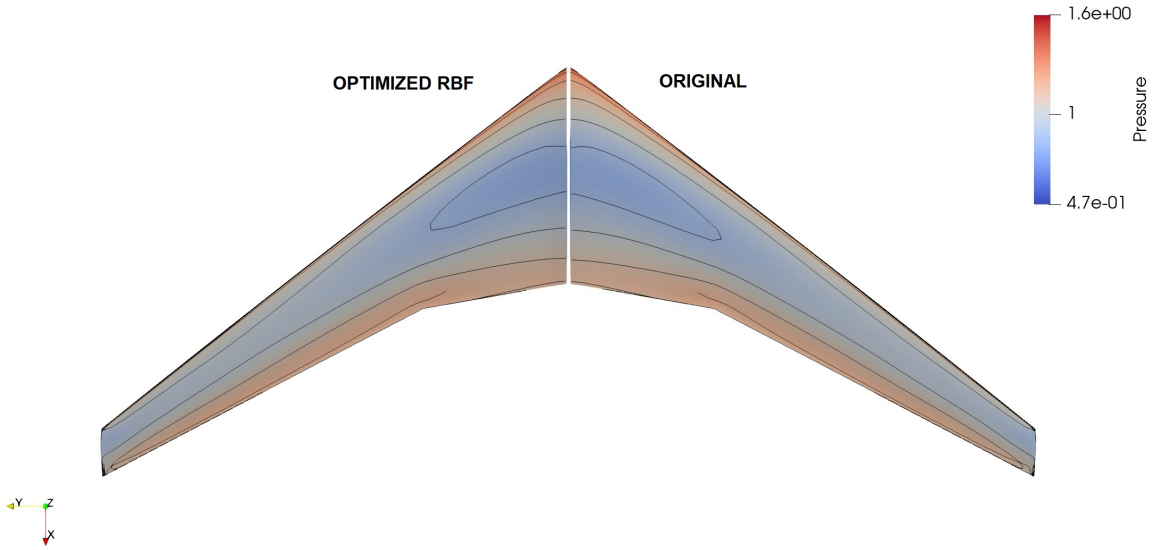

Figure 6.21: CRM: Pressure Isolines Lower Part

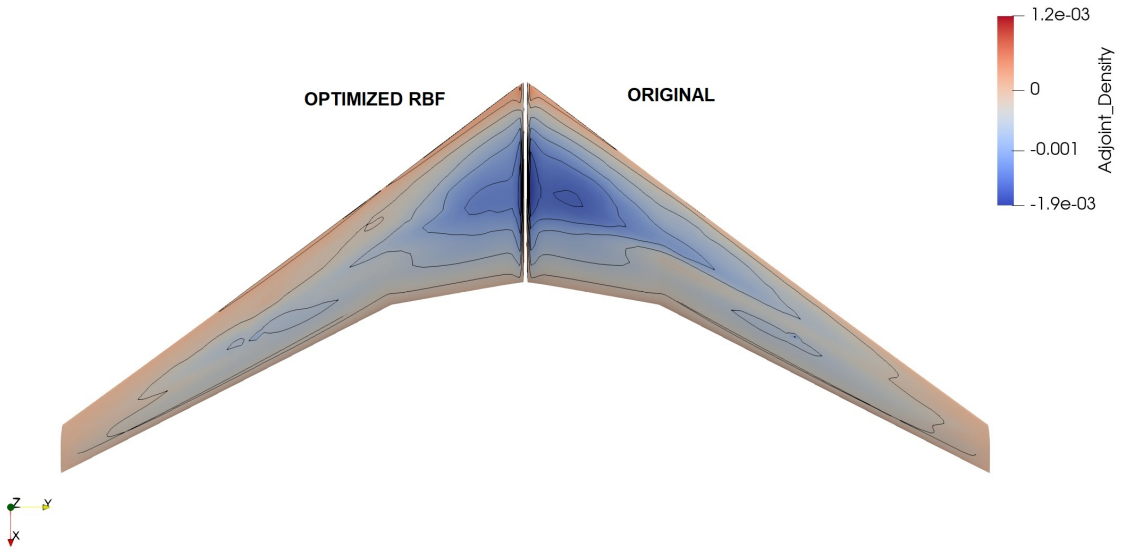

Figure 6.22: CRM: Adjoint Density

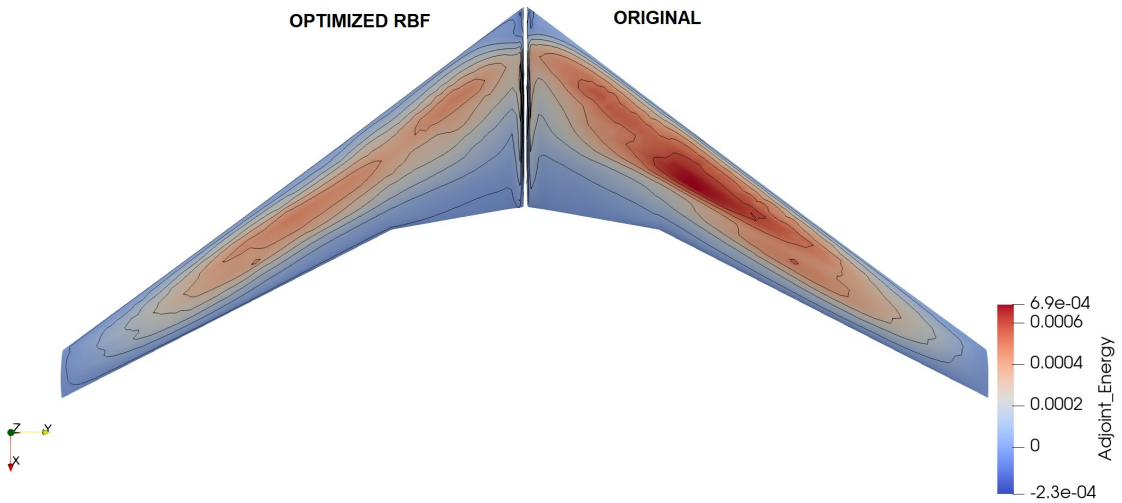

Figure 6.23: CRM: Adjoint Energy

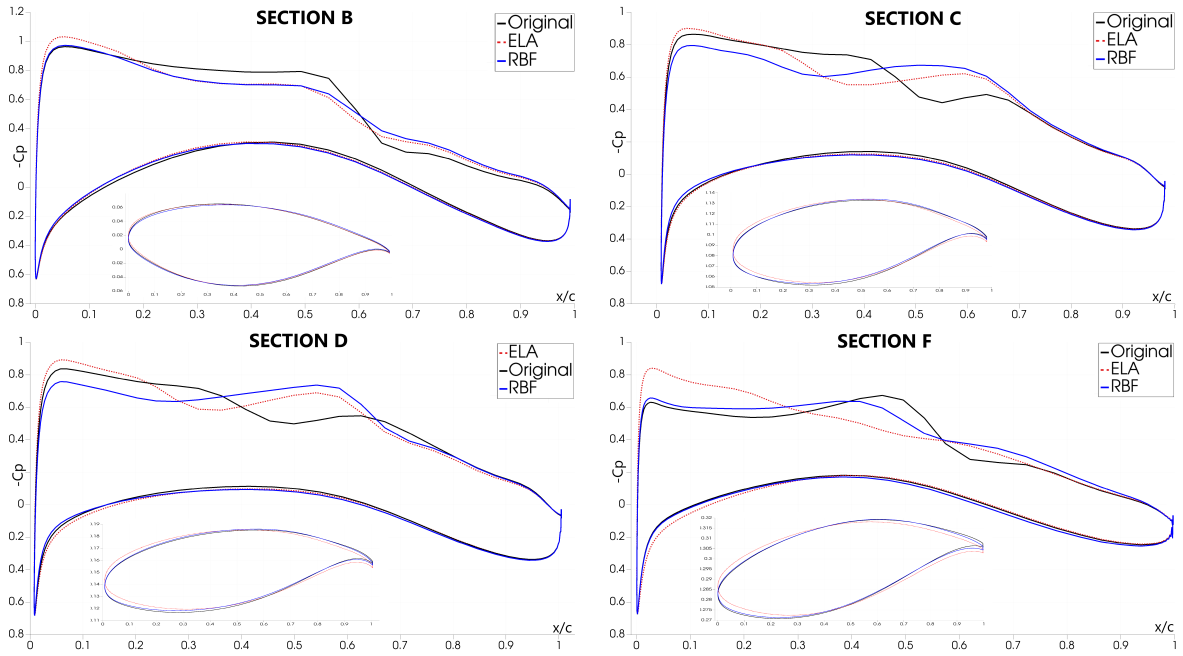

Figure 6.24: CRM: C_p Optimization

The polar of the wing has been computed both on the original wing and on the optimized one. Mach, Re and free stream temperature are fixed and equal to the previously selected. The results show an unexpected improvements of the performance also far from the design condition selected. The same value of lift is always shifted to a lower angle of attack Fig. 6.26. Looking the data at a fixed C_l the efficiency is increased in the entire range selected $[0°;6°]$ Fig. 6.27. The maximum efficiency is slightly shifted to a lower value of the lift which is not the one around which the CRM has been optimized. These results are unexpected since only a single point optimization is conducted, probably the original wing was far from the optimum point, the strong shock is present in the entire range, therefore the optimization decreased the intensity also in off-point conditions.

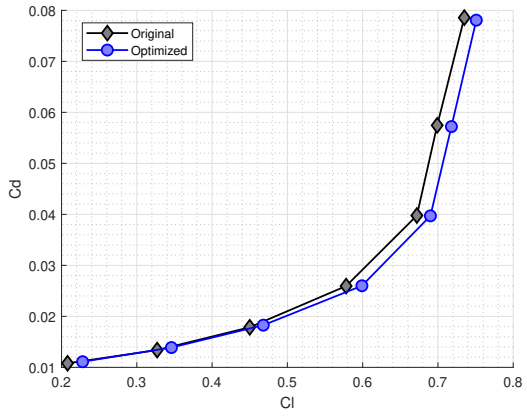

Figure 6.25: CRM: Original and Optimized Polar

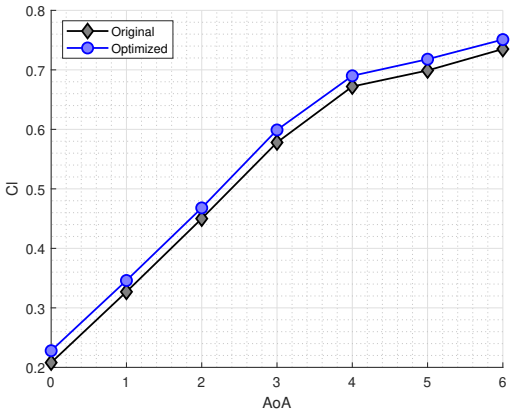

Figure 6.26: CRM: C_l Respect To AoA

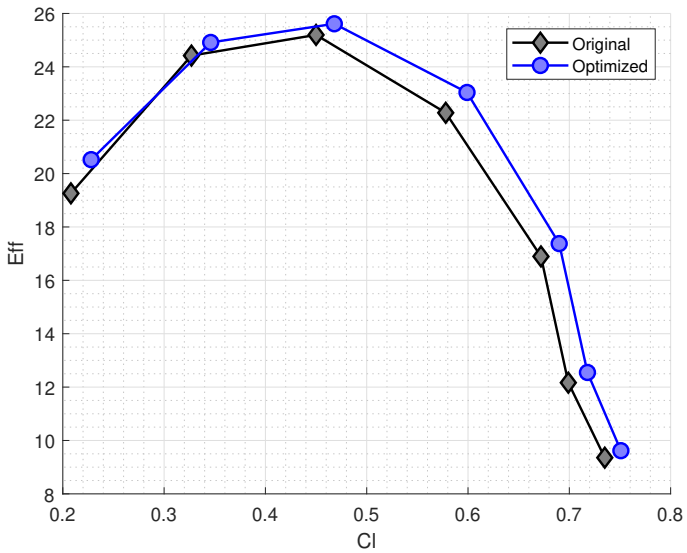

Figure 6.27: CRM: Efficiency Respect to C_l

6.3 Wing-Winglet Optimization

The ultimate goal of this thesis is to optimized a complex geometry including a non-planar wing. Regarding SU2, the task has been already faced by prof. Palacio in the project NERONE[30]. However, in that case the mesh is not deformed although entirely generated from scratch for each iteration and an inviscid optimization is performed. A more high fidelity ASO is presented in this section, where RANS equations with SA turbulence model are resolved and RBF is applied to deform the hybrid mesh. The Onera M6 wing is used, to which a winglet has been added. The winglet is drawn at the CAD software generating a loft by a gradual contraction of the tip's airfoil until it halves the initial chord. The free stream conditions are the same used in Onera section, the target C_l is recomputed for the new geometry:

Table 6.6: Free Stream Conditions Onera M6 with Winglet

Mach	0.84
AoA	3.06
Re	$14.6E6$
Temperature	$300K$
Target C_l	0.263

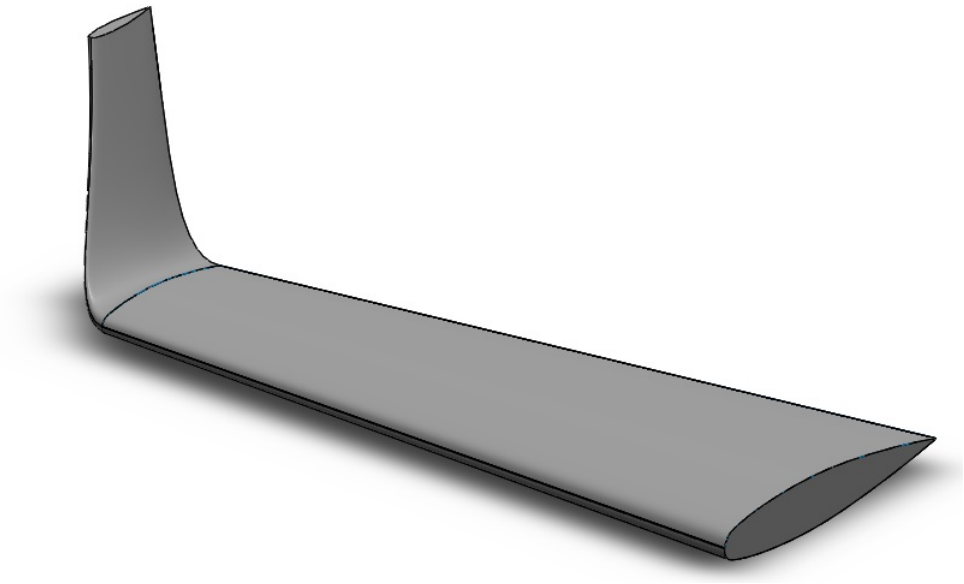

Figure 6.28: Onera M6 With Winglet Attached CAD

The convergence of the mesh is conducted at fixed C_l, since the surface grid is structured the number of points in the three directions is multiplied for 1.5. The boundary layer structured mesh progressively has a lower growth ratio and an higher number of first cells with constant height. Although only the coarser grid is used for the optimization to respect the virtual memory available. Therefore, the optimization starts from an inaccurate value of the drag which is still highly mesh dependent. The results should be considered as a demonstration of the capability to improve the aero-performace of a non planar wing and not as a benchmark to match.

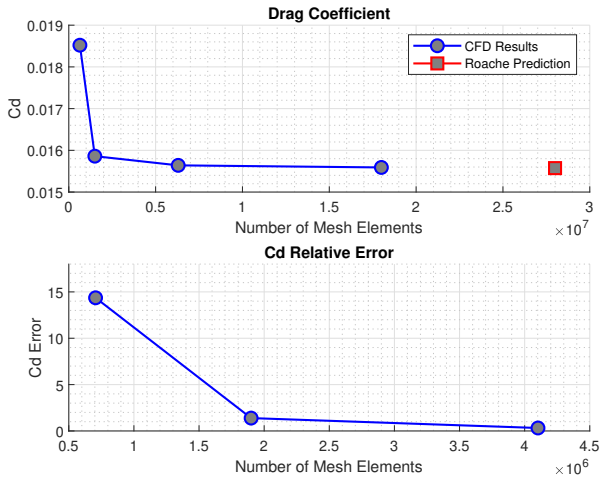

Figure 6.29: Onera M6 With Winglet Mesh Convergence

Regarding the optimization, the drag is selected as the objective function, the lift and the volume must not decrease. The angle of attack is free to change, the y-momentum is just monitored, while the root is fixed. The area of five sections on the wing and three on the winglet cannot become lower than the 75%

of the initial value. A hybrid mesh of 0.7 million cells is generated, with the first two layers of constant height proving a starting $y+ < 0.9$. A single level RBF is used with Wendland $C0$ as the basis function and maximum 10% od the surface nodes selectable as control points. This configuration provides for all the design loops a maximum error of the displacement lower than 0.5%. The optimization process is split in two stages. First, two FFD boxes are used, one containing the wing with 144 DVs that can translate only in z-direction and a second around the winglet with 70 DVs free to shift in y,x direction. The sensitivity computed for wing DVs is one order of magnitude higher, accordingly the nodes of the horizontal part of the body are more shifted than the rest of the wing. When the drag reduction starts to be too low, the ASO restarts from the last output grid, this time with only FFD on the winglet. The number of DVs is increased to 160, this way the winglet is proper morphed and the difference is visible also with the naked eye.

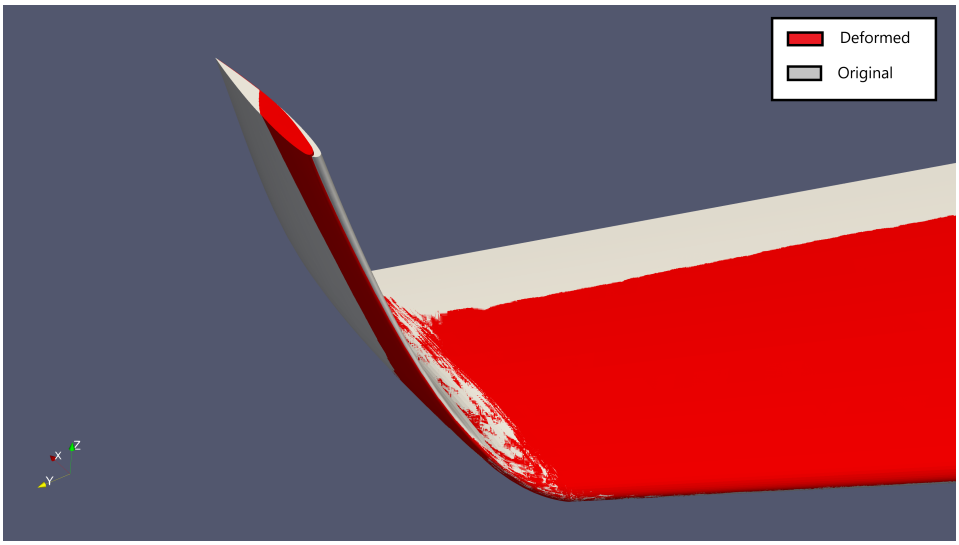

Figure 6.30: Onera M6 Original and Deformed Winglet

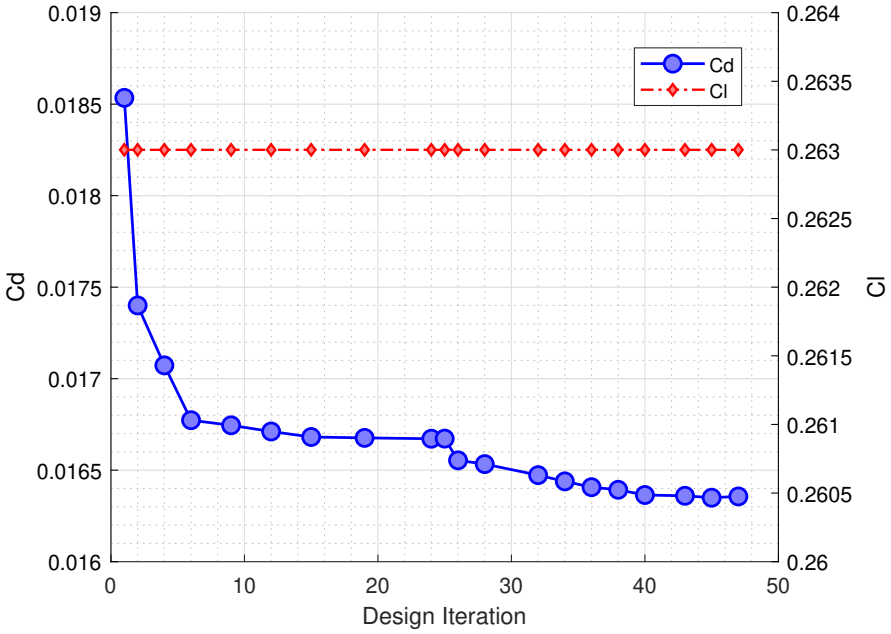

Figure 6.31: Onera M6 Winglet C_d Optimization

As can be seen in the previous figure, the drag is reduced from 185.34 counts to 163.43 which correspond to a decrease of 11.82%. The volume is perfectly preserved, however the y-momentum, which is only monitored in this case, remarkably increases from -0.112 to -0.148. The shape of the section and the C_p distribution are reported for four positions, two related to the winglet and two to the horizontal part.The optimization has been tested also with an elastic analogy method for the mesh deformation. The orthogonality of the mesh after the first design loop is chosen as indicator of the grid quality. Concerning ELA, if the stiffness is set with an inverse volume criteria the following direct simulation on the updated grid diverges. It must be underlined that it does not happen in case of constant stiffness or wall distance method for creating the matrix of the system although the minimum orthogonality is a bit lower than the one obtained with RBF suggesting a progressive lost of quality.

Table 6.7: Mesh Orthogonality

Method	N_c	N_v	Min Orthogonality
RBF C0	1100	392619	6.32
ELA inv volume	12676	645590	-43.92
ELA constant	12676	645590	6.21
ELA wall distance	12676	645590	6.28

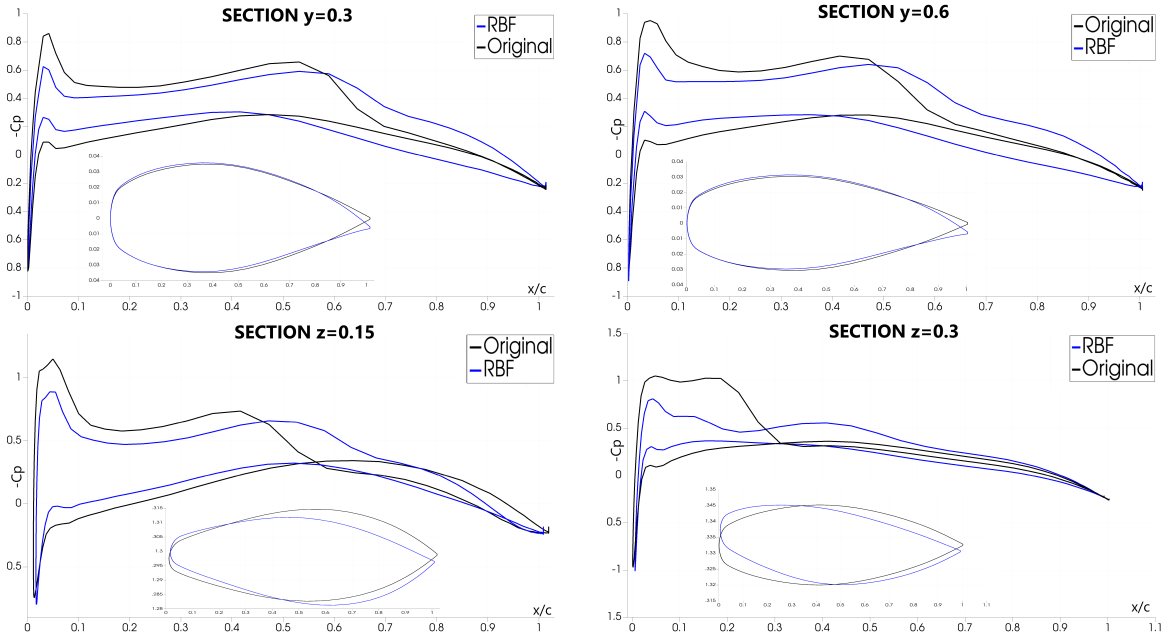

Figure 6.32: Onera Winglet: C_p Optimization

6.4 Subsonic Wing Multipoint Optimization

This section is the result of a collaboration with the Leonardo SPA, which is the leading Italian aerospace company. The goal is to increase the performance of an isolated planar wing extracted from a regional aircraft which is able to operate from the most rudimentary airstrips and in extreme environmental conditions. The CAD file is provided directly by the Leonardo Aeronautical Division that produces the aircraft. The RANS optimization is conducted in subsonic regime $M = 0.405$. The company provided the polar data for the desired flight conditions, the coefficients are matched through a mesh convergence process than to be compatible with the computational power available the grid selected is strongly relaxed bringing the ASO to start from an inexact aero-coefficients. The results obtained should be interpreted as the prof that SU2 is usable also for real industrial applications although not as reference values for the project. The optimization should be repeated exactly with the same criteria on a more proper and refined mesh. As in the rest of the thesis it is required to reduce the drag without changing the lift and the wing volume. The angle of attack is free to change, the moments respect to the three axes are only monitored for reducing the computational cost. At 28 stations spanwise the maximum thickness of the section must not become lower than 75% of the initial value. The wing is really straight with almost no arrow angle and tapering. Only RBF is used for deforming the grid at each design loop. The optimization is first conducted selecting one C_l value obtained for an $AoA = 2°$. Just one FFD box is used for containing the body, different number and distributions of DVs are tried. Only the best result is reported, which is obtained for 240 vertices equally allocated on 20 sections of the wing . Besides, a multipoint optimization is conducted around three lift value initially computed for $AoA = 0°, 2°, 4°$. In both cases the polar of the final morphed wing is computed and compared with the initial distribution.

Table 6.8: High Mach Multipoint Optimization

C_l	$\Delta\alpha$	ΔC_d	ΔCm_y
0.332	$-0.292°$	-0.65%	$+4.7\%$
0.526	$-0.29°$	-1.14%	$+3.57\%$
0.717	$-0.28°$	-1.36%	$+2.89\%$

The multipoint and single point optimization provide practically the same results, the drag reduction is limited since we re subsonic, no shock wave is present, the wing is already optimized and there is no separation of the flow to control. What can be notice from the following images is that the same lift is obtained for a lower angle of attack and the efficiency is slightly increased for the entire range of angle considered. This is happening for both the free stream conditions selected.The maximum efficiency is shifted to a little higher value of lift.

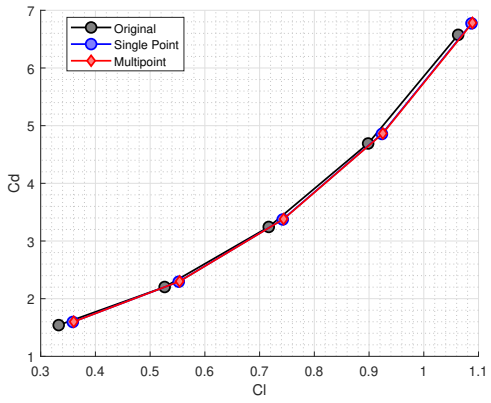

Figure 6.33: Polar M=0.405

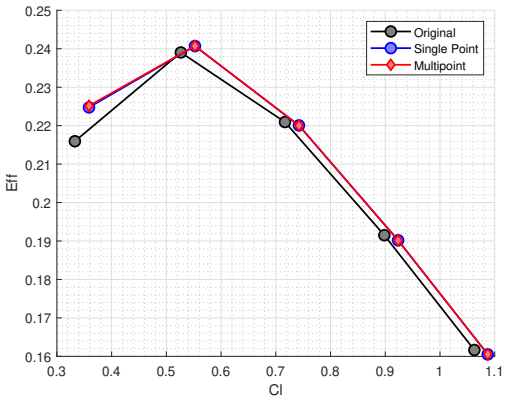

Figure 6.34: Efficiency vs Lift M=0.405

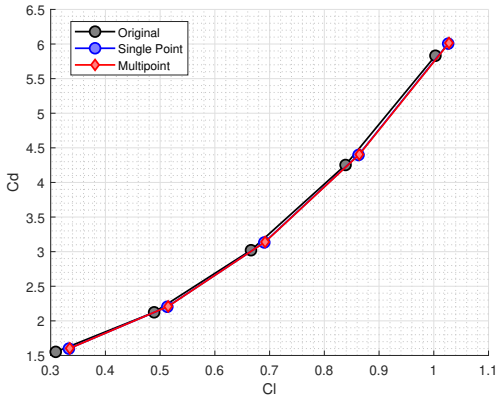

Figure 6.35: Polar M=0.2073

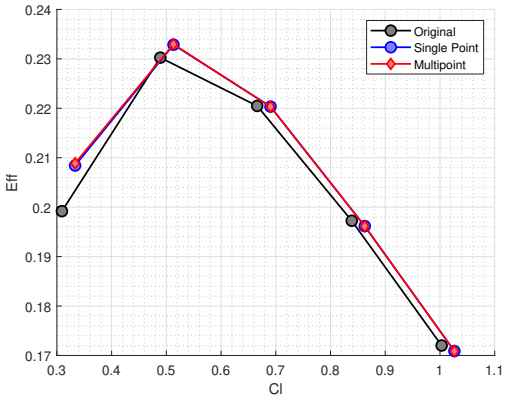

Figure 6.36: Efficiency vs Lift M=0.2073

To confirm the results the original and optimized surface mesh has been extracted, each elements is split in two cells, a more accurate boundary layer structured mesh is generated and a wake refinement is added, with the first 5 cells of constant height and than a growth ratio of 1.18. The two grid are tested in the case of high Mach and two target C_l obtained around 0° and 2° as angle of attack. The converged RANS confirm a little reduction of 0.55 drag counts and the angle of attack for obtaining the same C_L is lower of 0.25°. However, the increase of the y-momentum is confirmed around 4% for the first angle of attack and 3.3% for the second. The C_p distribution and the section shape are reported at $A = 24.5\%, B = 48.8\%, C = 73.29\%, D = 96.23\%$ of the span for the lower lift value. The peak of suction near the leading edge is reduced and the backward part of the wing is generating more lift causing the increment of the y momentum. Moreover, due to the change of the twist distribution more lift is generated close to the tip.

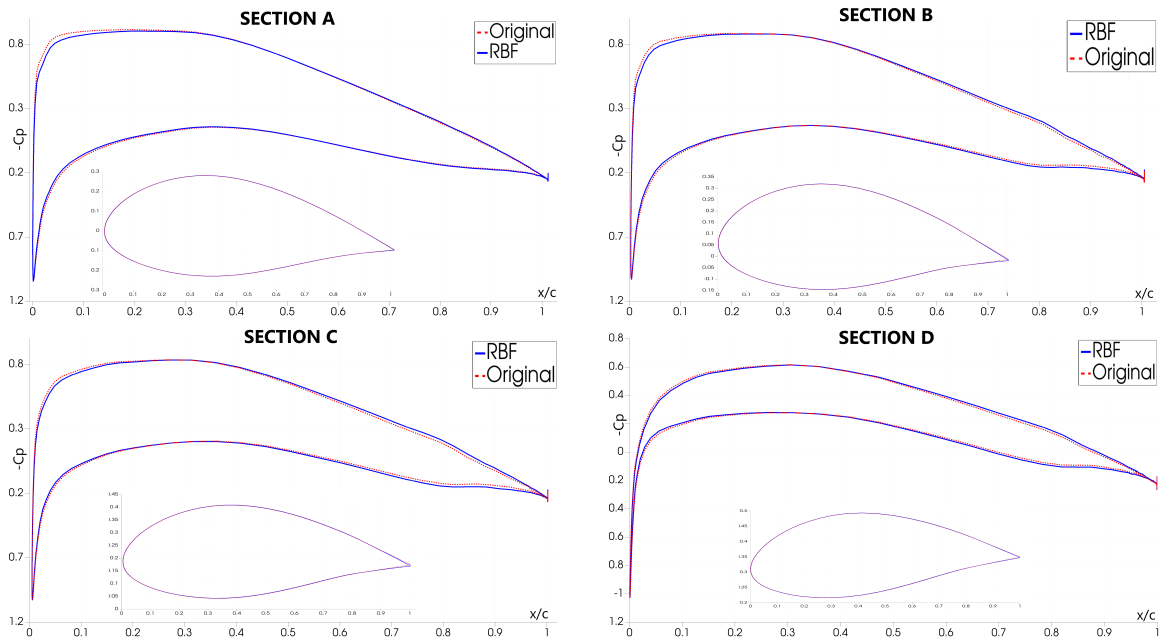

Figure 6.37: Multipoint Optimization C_p

Chapter 7

Conclusions and Perspectives

The aim of this thesis was to increase the robustness and the possibility of the gradient based aerodynamic optimization chain implemented inside the open-source SU2. Considering only steady RANS simulations, the software has to be able to improve the aero-performance of a 3D rigid body in supersonic or subsonic conditions and regarding Su2 for the first time of a wing-winglet configuration. All the targets have been achieved combining the discrete adjoint, for computing the surface sensitivity, and RBF for deforming the computational grid. As optimization in this context it is intended the research of the body shape that provides a lower value of the objective function selected.

Both C++ modulus were already coded inside the software, however they were not interconnected and RBF needed to be adapted to be compatible with the Automatic Differentiation. The displacements of the design variables are directly affected by the method selected for morphing the grid. This could drive the gradient based optimizer SLSQP to find a different minimum, as is evident in the NACA0012 test case. The version of RBF implemented in SU2 includes two data reduction schemes: a multilevel greedy algorithms and a volume reduction system. Selecting the maximum number of the control points permitted, the levels and the base function it is possible to regulate the computational time and the virtual memory usage. This is extremely useful in case of scarce computational resource available. Concerning the computational efficiency, the benchmark testes highlight a remarkable improvement respect to the elastic analogy method ELA, widely used not only in SU2, for updating the grid. In the 3D Onera M6 case the ram usage is reduced of the 80% and the cpu time of the 75%, since the mesh is deformed at each design loop it means also a quicker optimization. Two test cases proposed by ADODG has been performed, the transonic RAE2822 and the CRM wing. This way the results obtained with SU2 are compared with the global CFD community showing that our ASO is inline with the more advanced methods implemented. To increase the robustness of the multidimensional optimization it has been modified the method to control the maximum surface displacement and eventually scale all the values, obtaining at each design loop a movement of the order of millimeters. Regarding the non planar case, it has been necessary to add the ability to impose geometrical constraints also on planes with z norm. To further test SU2 with more difficult tasks, a multipoint subsonic optimization of the wing of Leonardo's regional aircraft has been performed. The reduction of drag is achieved for a considerable range of angle of attack and for two different free-stream conditions. Concerning SU2, for the first time it has been successfully optimized a wing-winglet geometry with a 11.82% drag reduction at the cost of an increasing y-momentum. It has been accomplished thank to RBF which is able to morph more complex grids and manage larger displacements, in fact with ELA this type of geometry is not well handled and the ASO easily fails.

This thesis, thanks to the collaboration with Leonardo Aeronautics Division which provided the CAD, demonstrates that the optimization chain implemented in SU2 with the discrete adjoint and RBF can be applied to real industrial geometries, achieving remarkable improvements in the aero coefficients. Since it is gradient based, it will always find the closest local minimum respect to the starting point therefore it is a tool perfect for the last steps of a design project yet not for define the body from scratch. Even if now the ASO is more robust and flexible, many improvements are still necessary and possible. Some ideas strictly related to the possible advancement in the SU2 software are here proposed:

- The direct simulation and the adjoint solutions requires sometimes different refinement zone in the grid. Moreover the deformation of the body could strongly impact the flow solution making the original distribution of cells not optimal to capture the flow behaviour. One way to overcome this issue is to introduce inside the optimization chain a goal oriented mesh adaptive scheme that also smooth the direct and adjoint necessity. Besides, SU2 already is compatible with INRIA AMG Library [118] making this solution easy implementable.

- FFD is really limiting for the deformation process. It is difficult to proper contain a complex geometry with only one box. If more boxes are used, the faces in common cannot move limiting the design space explorable. This constraint is too hard and not strictly necessary, it should be relaxed and also the continuity conditions can be handled in a more flexible way.

- The position and number of the design variables should become adaptive. It could help the optimization and slightly reduce the computational cost to be able to add a vertices of control above that region of the surface where the sensitivity is higher. With the same concept some DVs which prescribed displacement is irrelevant could be switch off in the successive design loop.

- To generate a proper tool for designing a wing since the first steps of the project, the gradient based optimization should be combined with a GA algorithm. The general shape of the wing could be described by less than ten parameters such as the span, root chord, tip chords, arrow angle. First, the GA generating a certain population could identify a sub-optimal shape. Secondly, the ASO used in this thesis could work on the best body previously found further reducing the C_d. This solution, not implemented in SU2, is known in literature under the name of Hybrid Optimization.

———-

Bibliography

[1] Giles, M. B. and Pierce, N. A., "An introduction to the adjoint approach to design," *Flow, turbulence and combustion*, Vol. 65, No. 3-4, 2000, pp. 393–415.

[2] Skinner, S. N. and Zare-Behtash, H., "State-of-the-art in aerodynamic shape optimisation methods," *Applied Soft Computing*, Vol. 62, 2018, pp. 933–962.

[3] Juan J. Alonso, Thomas D. Economon, V. M., "Mesh Adaptation for SU2 with INRIA AMG Library," *1st ANNUAL SU2 DEVELOPERS MEETING, Tu Delft, Netherlands*, Set.2016.

[4] Albring, T. A., Sagebaum, M., and Gauger, N. R., "Efficient aerodynamic design using the discrete adjoint method in SU2," *17th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2016, p. 3518.

[5] Sederberg, T. W. and Parry, S. R., "Free-form deformation of solid geometric models," *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986, pp. 151–160.

[6] Lin, X., Ruan, S., Qiu, T., and Guo, D., "Nonrigid medical image registration based on mesh deformation constraints," *Computational and mathematical methods in medicine*, Vol. 2013, 2013.

[7] Blom, F. J., "Considerations on the spring analogy," *International journal for numerical methods in fluids*, Vol. 32, No. 6, 2000, pp. 647–668.

[8] Cavagna, L., Quaranta, G., and Mantegazza, P., "Application of Navier–Stokes simulations for aeroelastic stability assessment in transonic regime," *Computers & Structures*, Vol. 85, No. 11-14, 2007, pp. 818–832.

[9] Chernukhin, O. and Zingg, D. W., "Multimodality and global optimization in aerodynamic design," *AIAA journal*, Vol. 51, No. 6, 2013, pp. 1342–1354.

[10] Carrier, G., Destarac, D., Dumont, A., Meheut, M., Salah El Din, I., Peter, J., Ben Khelil, S., Brezillon, J., and Pestana, M., "Gradient-based aerodynamic optimization with the elsA software," *52nd Aerospace Sciences Meeting*, 2014, p. 0568.

[11] Shahpar, S., "Challenges to overcome for routine usage of automatic optimisation in the propulsion industry," *Aeronautical Journal*, Vol. 115, No. 1172, 2011, pp. 615.

[12] Sobieszczanski-Sobieski, J., "The case for aerodynamic sensitivity analysis," *Sensitivity Analysis in Engineering, compiled by H.M. Adelman and R. T. Haftka, NASA CP 2457*, Feb 1987, pp. 77–96.

[13] Reneaux, J. and Thibert, J.-J., "The use of numerical optimization for airfoil design," *3rd Applied Aerodynamics Conference, Colorado Springs*, 1985, p. 5026.

[14] Lyness, J. N. and Moler, C. B., "Numerical differentiation of analytic functions," *SIAM Journal on Numerical Analysis*, Vol. 4, No. 2, 1967, pp. 202–210.

[15] Pironneau, O., "On optimum profiles in Stokes flow," *Journal of Fluid Mechanics*, Vol. 59, No. 1, 1973, pp. 117–128.

[16] Jameson, A., "Aerodynamic design via control theory," *Journal of scientific computing*, Vol. 3, No. 3, 1988, pp. 233–260.

[17] Jameson, A., "Re-engineering the design process through computation," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 36–50.

[18] Reuther, J. J., Jameson, A., Alonso, J. J., Rimllnger, M. J., and Saunders, D., "Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 2," *Journal of aircraft*, Vol. 36, No. 1, 1999, pp. 61–74.

[19] Shubin, G. R., "Obtaining\cheap" optimization gradients from computational aerodynamics codes," *Applied Mathematics and Statistics Technical Report AMS, Boeing Computer Services*, 1991.

[20] Nadarajah, S. and Jameson, A., *Studies of the continuous and discrete adjoint approaches to viscous automatic aerodynamic shape optimization*.

[21] Giles, M., "Analysis of the accuracy of shock-capturing in the steady quasi-1D Euler equations," 1995.

[22] Palacios, F., Colonno, M. R., Aranake, A. C., Campos, A., Copeland, S. R., Economon, T. D., Lonkar, A. K., Lukaczyk, T. W., Taylor, T. W., and Alonso, J. J., "Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design," *AIAA paper*, Vol. 287, 2013, pp. 2013.

[23] Sagebaum, M., Albring, T., and Gauger, N. R., "High-performance derivative computations using codipack," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 45, No. 4, 2019, pp. 1–26.

[24] Zhou, B. Y., Albring, T. A., Gauger, N. R., Economon, T. D., Palacios, F., and Alonso, J. J., "A discrete adjoint framework for unsteady aerodynamic and aeroacoustic optimization," *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2015, p. 3355.

[25] Zhou, B., Albring, T. A., Gauger, N. R., Ilario, C., Economon, T. D., and Alonso, J. J., "Reduction of airframe noise components using a discrete adjoint approach," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017, p. 3658.

[26] Johansen, T. A., Fossen, T. I., and Berge, S. P., "Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming," *IEEE Transactions on Control Systems Technology*, Vol. 12, No. 1, 2004, pp. 211–216.

[27] Kuhn, H. W. and Tucker, A. W., "Nonlinear Programming," *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, Calif., 1951, pp. 481–492.

[28] Samareh, J. A., "Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization," *AIAA journal*, Vol. 39, No. 5, 2001, pp. 877–884.

[29] Samareh, J., "Aerodynamic shape optimization based on free-form deformation," *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2004, p. 4630.

[30] Pustina, L., Cavallaro, R., and Bernardini, G., "NERONE: An Open-Source Based Tool for Aerodynamic Transonic Optimization of Nonplanar Wings," *Aerotecnica Missili & Spazio*, Vol. 98, No. 1, 2019, pp. 85–104.

[31] Selim, M., Koomullil, R., et al., "Mesh deformation approaches–a survey," *Journal of Physical Mathematics*, Vol. 7, No. 2, 2016.

[32] Batina, J. T., "Unsteady Euler airfoil solutions using unstructured dynamic meshes," *AIAA journal*, Vol. 28, No. 8, 1990, pp. 1381–1388.

[33] Baker, T. and Cavallo, P., "Dynamic adaptation for deforming tetrahedral meshes," *14th Computational Fluid Dynamics Conference*, 1999, p. 3253.

[34] Luke, E., Collins, E., and Blades, E., "A fast mesh deformation method using explicit interpolation," *Journal of Computational Physics*, Vol. 231, No. 2, 2012, pp. 586–601.

[35] Jakobsson, S. and Amoignon, O., "Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization," *Computers & Fluids*, Vol. 36, No. 6, 2007, pp. 1119–1136.

[36] Morelli, M., Bellosta, T., and Guardone, A., "Efficient radial basis function mesh deformation methods for aircraft icing," *Journal of Computational and Applied Mathematics*, 2021, pp. 113492.

[37] Belferhat, S., Meftah, S., Yahiaoui, T., and Imine, B., "Aerodynamic Optimization of a Winglet Design," *EPJ Web of Conferences*, Vol. 45, EDP Sciences, 2013, p. 01010.

[38] Cella, U. and Romano, D., "Assessment of Optimization Algorithms for Winglet Design," *EnginSoft-Newsletter Year 7nl*, 2010.

[39] Weierman, J. and Jacob, J., "Winglet design and optimization for UAVs," *28th AIAA Applied Aerodynamics Conference*, 2010, p. 4224.

[40] Swanson, R. C. and Turkel, E., "On central-difference and upwind schemes," *Journal of computational physics*, Vol. 101, No. 2, 1992, pp. 292–306.

[41] Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of computational physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[42] Hicks, R. M., Murman, E. M., and Vanderplaats, G. N., "An assessment of airfoil design by numerical optimization," 1974.

[43] Vanderplaats, G. N. and Hicks, R. M., "Numerical airfoil optimization using a reduced number of design coordinates," 1976.

[44] Haftka, R. T., "Sensitivity calculations for iteratively solved problems," *International Journal for Numerical Methods in Engineering*, Vol. 21, No. 8, 1985, pp. 1535–1546.

[45] Nielsen, E. J. and Anderson, W. K., "Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations," *AIAA journal*, Vol. 37, No. 11, 1999, pp. 1411–1419.

[46] Gao, Y., Wu, Y., and Xia, J., "Automatic differentiation based discrete adjoint method for aerodynamic design optimization on unstructured meshes," *Chinese Journal of Aeronautics*, Vol. 30, No. 2, 2017, pp. 611–627.

[47] Martins, J. R., Sturdza, P., and Alonso, J. J., "The complex-step derivative approximation," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 29, No. 3, 2003, pp. 245–262.

[48] Nielsen, E. J., *Aerodynamic design sensitivities on an unstructured mesh using the Navier-Stokes equations and a discrete adjoint formulation*, Ph.D. thesis, Virginia Tech, 1998.

[49] Nielsen, E. J. and Park, M. A., "Using an adjoint approach to eliminate mesh sensitivities in computational design," *AIAA journal*, Vol. 44, No. 5, 2006, pp. 948–953.

[50] Shroff, G. M. and Keller, H. B., "Stabilization of unstable procedures: the recursive projection method," *SIAM Journal on numerical analysis*, Vol. 30, No. 4, 1993, pp. 1099–1120.

[51] Peter, J. E. and Dwight, R. P., "Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches," *Computers & Fluids*, Vol. 39, No. 3, 2010, pp. 373–391.

[52] Venditti, D. A. and Darmofal, D. L., "Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow," *Journal of Computational Physics*, Vol. 164, No. 1, 2000, pp. 204–227.

[53] Spalart, P. and Allmaras, S., "A one-equation turbulence model for aerodynamic flows," *30th aerospace sciences meeting and exhibit*, 1992, p. 439.

[54] Menter, F., "Zonal two equation kw turbulence models for aerodynamic flows," *23rd fluid dynamics, plasmadynamics, and lasers conference*, 1993, p. 2906.

[55] WILCOX, D., "A half century historical review of the k-omega model," *29th Aerospace Sciences Meeting*, 1991, p. 615.

[56] Dwight, R. P. and Brezillon, J., "Adjoint algorithms for the optimization of 3d turbulent configurations," *New Results in Numerical and Experimental Fluid Mechanics VI*, Springer, 2007, pp. 194–201.

[57] Mohammadi, B., "Dynamical approaches and incomplete gradients for shape optimization and flow control," *14th Computational Fluid Dynamics Conference*, 1999, p. 3374.

[58] Giles, M., Pierce, N., Giles, M., and Pierce, N., "Adjoint equations in CFD-Duality, boundary conditions and solution behaviour," *13th Computational Fluid Dynamics Conference*, 1997, p. 1850.

[59] Peter, J., *Contributions to discrete adjoint method in aerodynamics for shape optimization and goal-oriented mesh-adaptation*, Ph.D. thesis, UNIVERSITE DE NANTES, 2020.

[60] Palacios, F. and Economon, T., "Stanford University Unstructured (SU2): Open-Source Analysis and Design Technology for Turbulent Flows, 52nd AIAA Aerospace Sciences," *National Harbor, Madrid*, 2014.

[61] Sanchez, R., Kline, H., Thomas, D., Variyar, A., Righi, M., Economon, T. D., Alonso, J. J., Palacios, R., Dimitriadis, G., and Terrapon, V., "Assessment of the fluid-structure interaction capabilities for aeronautical applications of the open-source solver SU2," 2016.

[62] Hogan, R. J., "Fast reverse-mode automatic differentiation using expression templates in C++," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 40, No. 4, 2014, pp. 1–16.

[63] Grabmeier, J. and Kaltofen, E., *Computer Algebra Handbook: Foundations, Applications, Systems;[with CD-ROM]*, Springer Science & Business Media, 2003.

[64] Albring, T. A., Sagebaum, M., and Gauger, N. R., "Development of a consistent discrete adjoint solver in an evolving aerodynamic design framework," *16th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2015, p. 3240.

[65] Griewank, A. and Walther, A., *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM, 2008.

[66] Yamazaki, W., Mouton, S., and Carrier, G., "Efficient design optimization by physics-based direct manipulation free-form deformation," *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2008, p. 5953.

[67] Bauckhage, C., "NumPy/SciPy Recipes for Data Science: Ordinary Least Squares Optimization," *researchgate. net, Mar*, 2015.

[68] Boggs, P. T. and Tolle, J. W., "Sequential quadratic programming for large-scale nonlinear optimization," *Journal of computational and applied mathematics*, Vol. 124, No. 1-2, 2000, pp. 123–137.

[69] Lyu, Z., Xu, Z., and Martins, J., "Benchmarking optimization algorithms for wing aerodynamic design optimization," *Proceedings of the 8th International Conference on Computational Fluid Dynamics, Chengdu, Sichuan, China*, Vol. 11, Citeseer, 2014.

[70] Wilson, R. B., "A simplicial algorithm for concave programming," *Ph. D. Dissertation, Graduate School of Bussiness Administration*, 1963.

[71] Powell, M. J., "A fast algorithm for nonlinearly constrained optimization calculations," *Numerical analysis*, Springer, 1978, pp. 144–157.

[72] Hock, W. and Schittkowski, K., "Test examples for nonlinear programming codes," *Journal of optimization theory and applications*, Vol. 30, No. 1, 1980, pp. 127–129.

[73] Kuhn, H. W., "Nonlinear programming: a historical view," *Traces and Emergence of Nonlinear Programming*, Springer, 2014, pp. 393–414.

[74] Fliege, J. and Vaz, A. I. F., "A SQP type method for constrained multiobjective optimization," *Proc. Optim. Online*, 2015, pp. 1–35.

[75] Korivi, V. M., Newman, P. A., and Taylor III, A. C., "Aerodynamic optimization using sensitivity derivatives from a three-dimensional supersonic Euler code," *Journal of aircraft*, Vol. 35, No. 3, 1998, pp. 405–411.

[76] Sadrehaghighi, I., Smith, R. E., and Tiwari, S. N., "Grid sensitivity and aerodynamic optimization of generic airfoils," *Journal of aircraft*, Vol. 32, No. 6, 1995, pp. 1234–1239.

[77] Lomtev, I., Kirby, R., and Karniadakis, G., "A discontinuous Galerkin ALE method for compressible viscous flows in moving domains," *Journal of Computational Physics*, Vol. 155, No. 1, 1999, pp. 128–159.

[78] Truong, A. H., Oldfield, C. A., and Zingg, D. W., "Mesh movement for a discrete-adjoint Newton-Krylov algorithm for aerodynamic optimization," *AIAA journal*, Vol. 46, No. 7, 2008, pp. 1695–1704.

[79] Lee, C., Koo, D., Telidetzki, K., Buckley, H., Gagnon, H., and Zingg, D. W., "Aerodynamic shape optimization of benchmark problems using jetstream," *53rd AIAA Aerospace Sciences Meeting*, 2015, p. 0262.

[80] Yang, G. and Da Ronch, A., "Aerodynamic shape optimisation of benchmark problems using SU2," *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018, p. 0412.

[81] Nielsen, E. J. and Anderson, W. K., "Recent improvements in aerodynamic design optimization on unstructured meshes," *AIAA journal*, Vol. 40, No. 6, 2002, pp. 1155–1163.

[82] Dwight, R. P., "Robust mesh deformation using the linear elasticity equations," *Computational fluid dynamics 2006*, Springer, 2009, pp. 401–406.

[83] Tezduyar, T. E., Behr, M., Mittal, S., and Johnson, A., "Computation of unsteady incompressible flows with the stabilized finite element methods: Space-time formulations, iterative strategies and massively parallel implementations," *ASME PRESSURE VESSELS PIPING DIV PUBL PVP, ASME, NEW YORK, NY(USA), 1992,*, Vol. 246, 1992, pp. 7–24.

[84] Yang, Z. and Mavriplis, D. J., "Higher-order time integration schemes for aeroelastic applications on unstructured meshes," *AIAA journal*, Vol. 45, No. 1, 2007, pp. 138–150.

[85] Yang, Z. and Mavriplis, D. J., "Mesh deformation strategy optimized by the adjoint method on unstructured meshes," *AIAA journal*, Vol. 45, No. 12, 2007, pp. 2885–2896.

[86] Nocedal, J. and Wright, S., *Numerical optimization*, Springer Science & Business Media, 2006.

[87] Witteveen, J. and Bijl, H., "Explicit mesh deformation using inverse distance weighting interpolation," *19th AIAA Computational Fluid Dynamics*, 2009, p. 3996.

[88] Hardy, R. L., "Multiquadric equations of topography and other irregular surfaces," *Journal of geophysical research*, Vol. 76, No. 8, 1971, pp. 1905–1915.

[89] Hardy, R. L., "Theory and applications of the multiquadric-biharmonic method 20 years of discovery 1968–1988," *Computers & Mathematics with Applications*, Vol. 19, No. 8-9, 1990, pp. 163–208.

[90] Rendall, T. C. and Allen, C. B., "Efficient mesh motion using radial basis functions with data reduction algorithms," *Journal of Computational Physics*, Vol. 228, No. 17, 2009, pp. 6231–6249.

[91] Sheng, C. and Allen, C. B., "Efficient mesh deformation using radial basis functions on unstructured meshes," *AIAA journal*, Vol. 51, No. 3, 2013, pp. 707–720.

[92] Morris, A., Allen, C., and Rendall, T., "Domain element paramterisation for CFD-based optimisation of aerofoils using deformation by radial basis functions," *International Journal for Numerical Methods in Fluids*, Vol. 58, No. 8, 2008, pp. 827–860.

[93] Morelli, M., Bellosta, T., and Guardone, A., "Efficient Radial Basis Function Mesh Deformation Methods for Aircraft Icing," *In the 7th European Seminar on Computing*, Pilsen, Czech Republic, June 8-12, 2020.

[94] Buhmann, M. D., *Radial basis functions: theory and implementations*, Vol. 12, Cambridge university press, 2003.

[95] Wendland, H., *Scattered data approximation*, Vol. 17, Cambridge university press, 2004.

[96] Biancolini, M. E., *Fast radial basis functions for engineering applications*, Springer, 2017.

[97] Rendall, T. C. and Allen, C. B., "Unified fluid–structure interpolation and mesh motion using radial basis functions," *International journal for numerical methods in engineering*, Vol. 74, No. 10, 2008, pp. 1519–1559.

[98] Costin, W. and Allen, C., "Numerical study of radial basis function interpolation for data transfer across discontinuous mesh interfaces," *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 10, 2013, pp. 1076–1095.

[99] Wang, G., Mian, H. H., Ye, Z.-Y., and Lee, J.-D., "Improved point selection method for hybrid-unstructured mesh deformation using radial basis functions," *AIAA Journal*, Vol. 53, No. 4, 2015, pp. 1016–1025.

[100] Xie, L. and Liu, H., "Efficient mesh motion using radial basis functions with volume grid points reduction algorithm," *Journal of Computational Physics*, Vol. 348, 2017, pp. 401–415.

[101] Li, D. and Hartmann, R., "Adjoint-based airfoil optimization with discretization error control," *International Journal for Numerical Methods in Fluids*, Vol. 77, No. 1, 2015, pp. 1–17.

[102] Shi, Y., Mader, C. A., He, S., Halila, G. L., and Martins, J. R., "Natural Laminar-Flow Airfoil Optimization Design Using a Discrete Adjoint Approach," *AIAA Journal*, Vol. 58, No. 11, 2020, pp. 4702–4722.

[103] He, X., Li, J., Mader, C. A., Yildirim, A., and Martins, J. R., "Robust aerodynamic shape optimization—from a circle to an airfoil," *Aerospace Science and Technology*, Vol. 87, 2019, pp. 48–61.

[104] Ladson, C. L., Hill, A. S., and Johnson, W., "Pressure distributions from high Reynolds number transonic tests of an NACA 0012 airfoil in the Langley 0.3-meter transonic cryogenic tunnel," 1987.

[105] Celik, I. B., Ghia, U., Roache, P. J., and Freitas, C. J., "Procedure for estimation and reporting of uncertainty due to discretization in CFD applications," *Journal of fluids Engineering-Transactions of the ASME*, Vol. 130, No. 7, 2008.

[106] Cook, P., McDonald, M., and Firmin, M., "Aerofoil RAE 2822: Pressure Distribution and Boundary Layer and Wake Measurements. AGARD AR 138," *Research and Technology Organisation, Neuilly-sur-Seine*, 1979.

[107] Anderson, G. R., Nemec, M., and Aftosmis, M. J., "Aerodynamic shape optimization benchmarks with error control and automatic parameterization," *53rd AIAA Aerospace Sciences Meeting*, 2015, p. 1719.

[108] Bisson, F. and Nadarajah, S., "Adjoint-based aerodynamic optimization framework," *52nd Aerospace Sciences Meeting*, 2014, p. 0412.

[109] Gariepy, M., Trepanier, J.-Y., Petro, E., Malouin, B., Audet, C., LeDigabel, S., and Tribes, C., "Direct search airfoil optimization using far-field drag decomposition results," *53rd AIAA Aerospace Sciences Meeting*, 2015, p. 1720.

[110] Poole, D. J., Allen, C. B., and Rendall, T., "Control point-based aerodynamic shape optimization applied to AIAA ADODG test cases," *53rd AIAA Aerospace Sciences Meeting*, 2015, p. 1947.

[111] Zhang, Y., Han, Z.-H., Shi, L., and Song, W.-P., "Multi-round surrogate-based optimization for benchmark aerodynamic design problems," *54th AIAA Aerospace Sciences Meeting*, 2016, p. 1545.

[112] Schmitt, V., "Pressure distributions on the ONERA M6-wing at transonic mach numbers, experimental data base for computer program assessment," *AGARD AR-138*, 1979.

[113] Lee, C., Koo, D., and Zingg, D. W., "Comparison of B-spline surface and free-form deformation geometry control for aerodynamic optimization," *AIAA Journal*, Vol. 55, No. 1, 2017, pp. 228–240.

[114] Yu, Y., Lyu, Z., Xu, Z., and Martins, J. R., "On the influence of optimization algorithm and initial design on wing aerodynamic shape optimization," *Aerospace Science and Technology*, Vol. 75, 2018, pp. 183–199.

[115] Palacios, F., Economon, T. D., and Alonso, J. J., "Large-scale aircraft design using SU2," *53rd AIAA aerospace sciences meeting*, 2015, p. 1946.

[116] Vassberg, J., Dehaan, M., Rivers, M., and Wahls, R., "Development of a common research model for applied CFD validation studies," *26th AIAA Applied Aerodynamics Conference*, 2008, p. 6919.

[117] Tinoco, E. N., Brodersen, O. P., Keye, S., Laflin, K. R., Feltrop, E., Vassberg, J. C., Mani, M., Rider, B., Wahls, R. A., Morrison, J. H., et al., "Summary data from the sixth AIAA CFD drag prediction workshop: CRM cases," *Journal of Aircraft*, Vol. 55, No. 4, 2018, pp. 1352–1379.

[118] Loseille, A., "Unstructured mesh generation and adaptation," *Handbook of Numerical Analysis*, Vol. 18, Elsevier, 2017, pp. 263–302.