ADJOINT BASED DESIGN OPTIMIZATION OF SUBSONIC AIRFOILS WITH A PANEL CODE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BERK SARIKAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
AEROSPACE ENGINEERING

JULY 2022

Approval of the thesis:

ADJOINT BASED DESIGN OPTIMIZATION OF SUBSONIC AIRFOILS WITH A PANEL CODE

submitted by **BERK SARIKAYA** in partial fulfillment of the requirements for the degree of **Master of Science in Aerospace Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Serkan Özgen
Head of Department, **Aerospace Engineering** _____

Prof. Dr. İsmail H. Tuncer
Supervisor, **Aerospace Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Yusuf Özyörük
Aerospace Engineering, METU _____

Prof. Dr. İsmail H. Tuncer
Aerospace Engineering, METU _____

Prof. Dr. Sinan Eyi
Aerospace Engineering, METU _____

Prof. Dr. Nafiz Alemdaroğlu
Aerospace Engineering, Atılım University _____

Assoc. Prof. Dr. Mustafa Kaya
Aerospace Engineering, Ankara Yıldırım Beyazıt University _____

Date: 25.07.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Berk Sarıkaya

Signature :

ABSTRACT

ADJOINT BASED DESIGN OPTIMIZATION OF SUBSONIC AIRFOILS WITH A PANEL CODE

Sarıkaya, Berk
M.S., Department of Aerospace Engineering
Supervisor: Prof. Dr. İsmail H. Tuncer

July 2022, 90 pages

An in-house panel code written in Fortran is automatically differentiated and developed into an adjoint-based aerodynamic shape optimization tool for airfoil profiles. The automatic differentiation tool, *FDOT*, is employed in reverse mode to obtain the discrete-adjoint solver. The adjoint-based sensitivity derivatives (i.e. gradient vector) are validated against the finite difference approach. The computed sensitivity derivatives are then employed for the gradient based aerodynamic shape optimization of subsonic airfoil profiles for given target lift and moment coefficients. In addition, an adverse pressure gradient minimization term is added to the objective function for milder stall characteristics. A multi-point design optimization method is also implemented for an improved off-design performance. Airfoils are parametrized by the Class Shape Transformation. Single and multi-point optimizations are driven by the open-source optimizer, *DAKOTA*, using a quasi-Newton method. Case studies for cambered airfoils at low angles of attack are presented for single and multi-point design optimizations. The surface pressure distributions and the aerodynamic loads for the optimum airfoil profiles are further verified with the open-source RANS solver, SU^2. It is shown that the adjoint based design optimization methodology developed

is efficient and robust.

Keywords: panel method, discrete adjoint, aerodynamic optimization, multi-point optimization, adverse pressure gradient

ÖZ

PANEL KODU İLE ADJOİNT TABANLI SESALTI KANAT PROFİLİ ENİYİLEMESİ

Sarıkaya, Berk
Yüksek Lisans, Havacılık ve Uzay Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. İsmail H. Tuncer

Temmuz 2022, 90 sayfa

Fortran ile yazılan bir panel kodu otomatik türev ile adjoint tabanlı bir aerodinamik kanat profili şekli eniyileme aracına dönüştürülmüştür. *FDOT* otomatik türev aracı ters mod ile ayrık-adjoint çözücü elde edilmesi için kullanılmıştır. Adjoint tabanlı hassasiyet türevleri (gradyan vektörü) sonlu farklar metodu ile doğrulanmıştır. Hesaplanan hassasiyet türevleri sesaltı kanat profillerinin hedef olarak verilen taşıma ve yunuslama katsayılarına göre gradyan bazlı aerodinamik tasarım eniyilemesinde kullanılmıştır. Ek olarak bir ters basınç gradyan terimi yumuşak perdövites karakteristiği için amaç fonksiyonuna eklenmiştir. Ek olarak, tasarım noktası dışındaki performansın daha iyi olması amacıyla çoklu nokta tasarım eniyilemesi yöntemi geliştirilmiştir. Kanat profilleri Sınıf Şekil Dönüşümü ile parametrize edilmiştir. Tekli ve çoklu nokta eniyilemeleri açık-kaynak kodlu bir eniyileme aracı olan *DAKOTA* ile quasi-Newton yöntemi kullanılarak gerçekleştirilmiştir. Kamburluklu kanat profilleri temel kabul edilerek düşük hücum açılarında yapılan tek ve çok-noktalı eniyileme sonuçları gösterilmiştir. Elde edilen aerodinamik yükler ve yüzey basınç katsayısı dağılımları bir açık-kaynak RANS çözücüsü olan SU^2 ile doğrulanmıştır. Son olarak, gösterilen ta-

sarım eniyileme metodunun fazlasıyla verimli ve gürbüz olduğu gösterilmiştir.

Anahtar Kelimeler: panel metodu, ayrık adjoint, aerodinamik eniyileme, çok noktalı eniyileme, ters basınç gradyanı

To my family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Prof. Dr. İsmail H. Tuncer for his encouragement, continuous support, and guidance throughout this study. His profound knowledge and experience in the field of aerodynamics and optimization has helped me immensely. I am grateful for the opportunity.

I would like to thank Dr. Reza Djeddi and Prof. Dr. Kivanc Ekici from Department of Mechanical, Aerospace and Biomedical Engineering at the University of Tennessee, Knoxville for providing the FDOT toolbox and their assistance on the implementation of FDOT.

I also appreciate the committee members, Prof. Dr. Yusuf Özyörük, Prof. Dr. Sinan Eyi, Prof. Dr. Nafiz Alemdaroğlu and Assoc. Prof. Dr. Mustafa Kaya and also Assoc. Prof. Nilay Sezer UZOL, for their time, constructive criticism and suggestions to further improve the study.

I am thankful to Dr. Halil Kaya from TAI with his help on the theory.

I would like to thank ASELSAN for the support and providing the necessary funds throughout this study. My thanks are extended to Tuğcan Selimhocaoğlu and Alper Sayıcı, to my other colleagues in the Flight Sciences, and the managers of the Aircraft Integration Engineering Department.

My thanks are extended to my dear friends Buğrahan Öztürk, Melikşah Koca (and others) from the department of Aerospace Engineering for their support and enjoyable discussions throughout the years. I also would like to wholeheartedly thank Ece Kanbur for her endless support and encouragement.

Finally, I would like to thank my beloved parents, Faik and Handan, who have always supported and loved me. My thanks are extended to my whole family.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimensional
AD	Automatic/Algorithmic Differentiation
CFD	Computational Fluid Dynamics
C_D	Drag Coefficient
C_L	Lift Coefficient
C_p	Pressure Coefficient
CST	Class Shape Transformation
DV	Design Variables
FD	Finite Difference
FDOT	Fast Automatic Differentiation Based on Operator-Overloading Technique
GA	Genetic Algorithm
NACA	National Advisory Committee for Aeronautics
C_M	Pitching Moment Coefficient
RANS	Reynolds-Averaged Navier Stokes
OO	Operator Overloading
SCT	Source Code Transformation

CHAPTER 1

INTRODUCTION

Computational Fluid Dynamics (CFD) have become key in aerodynamic design and shape optimization processes. Even though CFD cannot be separated from the wind tunnel tests, using CFD allows evaluation of new aerodynamic concepts faster and cheaper than that of a wind tunnel, generally. As the CFD tools developed and became accessible, these tools have seen widespread usage in the earlier design phases. With time, CFD branched out in fidelity, i.e. low and high-fidelity tools. The low fidelity tools are useful for rapid evaluation and exploration of new designs due to their lower computational cost whereas the high fidelity tools are useful for detailed design phases, where the design has matured. High-fidelity means that most of the physics features, if not all, have been properly captured and the solutions are more accurate than low-fidelity tools at the expense of computational resources. Solving the Reynolds-Averaged Navier-Stokes equations is an example of high-fidelity solution, whereas a panel method is an example of a low-fidelity model, which does not include effects like viscosity and flow separation. In this thesis, it is aimed to create an aerodynamic shape optimization tool for subsonic airfoils using a low-fidelity model with fast turnaround time.

The impact of better aerodynamic designs on aircraft performance cannot be emphasized enough. For example, Martins *et al.* [1] has shown that reduction of 1 drag count (d.c.) is worth 310 pounds of empty weight for a supersonic business jet. Similarly, the studies done on subsonic transport aircraft indicate that a reduction of 1 d.c. corresponds to a payload increase of roughly 200 lbs [2]. Also, while the governing bodies all around the world enforce strict emission targets, these could only be achieved by highly optimized aerodynamic configurations [3]. The aerodynamics of

the future aircraft is driven by the aviation's impact on the environment, such as the greenhouse gases, noise and so on. [3, 4].

The aerodynamic optimization process requires specialized solvers, algorithms and methods [5]. In the case of methods, there are two main kinds, which are the inverse methods and the numerical optimization methods, as shown in Fig. 1.1.

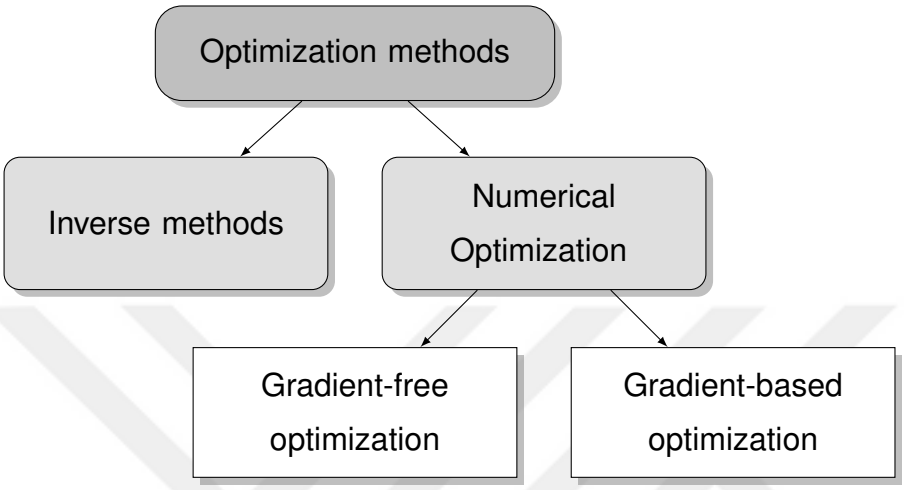


Figure 1.1: Optimization methods

Aerodynamic shape optimization studies, in fact, date as back as to 1680s. Newton [6] used calculus of variations to minimize drag over a solid body of revolution moving in a fluid with constant velocity. In 1935, Betz [7] proposed a way to find optimum airfoil shapes after applying predetermined pressure distributions by making use of conformal mapping, which may be considered as the earliest example of *inverse method*. Then, in 1945, Lighthill [8] obtained an exact solution for incompressible flows by conformally mapping the profile to a unit circle for the velocity over profile, with the help of an analytic mapping function.

Inverse methods allow prescribing the desired (target) pressure distribution directly at a constant angle of attack, which yields the optimal solution, i.e. the airfoil geometry. Inverse methods are useful for realizing target pressure distributions or flow velocities, for example. However, it is challenging to prescribe target distribution(s) that provide optimal performance over the range of operation. In fact, not knowing the distributions *a priori* is the essence of the problem. Moreover, inverse design using panel methods date as back as 1980s (Hawk and Bristow [9]) to solve problems like

matching target pressure coefficient distributions.

Naturally, the inverse problems have been formulated using potential flows due to the simplicity of the governing equations such as Volpe and Melnik [10]. Similarly, hodograph techniques are used in Hassan *et al.* [11] and Hassan *et al.* [12] for subsonic and transonic cases, respectively. Conformal mappings are also used in the multi-point inverse design problems by Selig and Maughmer [13] and Selig [14]. Afterwards, Malone *et al.* [15] used user specified target pressures as expressed by flow velocities to design 2-D airfoils and 3-D asymmetric nacelle profiles using a full-potential code. As the computational power increased, Euler equations are used in the optimization process by Jameson and Reuther [16] for inverse design optimization of airfoils. Reuther and Jameson [17] extended the work to Euler equations and successfully applied the adjoint inverse design methodology to wing and wing/body cases. Euler equations may not yield realizable results due the absence of the viscous features like flow separation, therefore, the Navier-Stokes equations are used by Jameson *et al.* [18] and the design optimization of wings in transonic flow is successfully carried out. Recently, Morlando [19] used a panel method, which is also written in Fortran, with the adjoint method for sensitivity analysis of airfoils in inverse design where the objective function is automatically differentiated. Zhang *et al.* [20] proposed inverse design of airfoils by means of adverse pressure gradient distribution as the design target. The necessary derivatives are computed by adjoint method. As discussed above, the inverse design methods require obtaining feasible target distributions beforehand which need to obey the various constraints and objective functions (i.e. performance) defined by the designer. Obtaining such profiles is not a trivial task. Majority of the time may be spent determining the required pressure distributions rather than the actual optimization process.

As the computers developed and the computational power has increased over the years, the numerical optimization methods have become more feasible. It was in the early 1970s when the gradient-based optimization methods have surfaced [21]. Hicks *et al.* [22] designed low-drag, non-lifting transonic airfoils using an optimizer which uses the *method of feasible directions*. They solved the transonic small disturbance equation. Similarly, Hicks and Vanderplaats [23] obtained optimal designs for low-speed airfoils. In the study, lift maximization, pitching moment and adverse pressure

gradient minimization cases are shown. Both studies [22, 23] have used higher-order polynomial coefficients that govern the airfoil surface coordinates as the design variables. After that, Hicks and Henne optimized a three-dimensional wing [24] using Hicks-Henne bump functions as the design variables which control the wing twist and shape. The works cited above use finite-difference method to calculate the gradients of the objective function, such as the drag or the lift coefficient. The gradient vector can be obtained with small perturbations in design variables as shown in [23]. While this approach is the simplest one, the cost is highest among any method due to repeated flow solver calls. Additionally, finite differences are prone to truncation errors. There are other ways to compute the gradients to drive the optimization process, such as the adjoint method. Thus, to compute gradients much more efficiently, adjoint methods are used. Adjoint methods are widely used in the literature since their introduction by Pironneau in Stokes flow [25] and Euler equations [26]. Adjoint methods are extended by Jameson [27] to inviscid compressible flows. Afterwards, the method is applied to Euler equations by Reuther and Jameson [17] in a wing/body drag minimization case. Additionally, Baysal and Eleshaky [28] applied adjoint equations on sensitivity of the 2-D Euler equations on the optimum design of a simplified scramjet geometry. Eventually, the Navier-Stokes equations are used by Jameson *et al.* [18] in transonic wing optimization cases. Adjoint methods are also used by Kim *et al.* [29] for multi-element airfoil design in high-lift configuration with the viscous, continuous adjoint method. All the works cited above use structured grids, which may be burdensome to generate about complex geometries. The adjoint method is extended to the unstructured grids by Anderson and Venkatakrishnan [30] and Anderson and Bonhaus [31] for inviscid and viscous airfoil design cases, respectively.

Finally, there are gradient-free optimization techniques, where the gradient information with respect to design variables are not required unlike the methods discussed above. A few examples are genetic algorithms, direct search and random search methods [32]. Among these algorithms, genetic algorithms (GA) are one of the most popular ones. Marco *et al.* [33] successfully performed drag reduction over RAE2822 airfoil in transonic conditions using a genetic algorithm. Similarly, GA was used to optimize airfoils suitable for wind turbines using a multi-objective approach by He and Agarwal [34]. Gradient-free algorithms are useful when the problem considered

has multiple local minima (multimodal), discontinuous objective functions or discrete design variables, but the drawback is high number of flow solver calls [35] which automatically means high cost.

As stated before, the gradient-free methods perform well when the problem possesses multimodality as these methods are often aimed towards finding the global minimum [35]. The gradient-based methods may aim towards a local minimum value, rather than the global minimum value. Zingg *et al.* [36] has directly compared GA and gradient-based methods. They have shown that for single-point optimization cases, the computational cost of GA is 6 to 187 times that of adjoint method [36]. For the multi-point optimization cases, they have shown that the cost of GA is 24 to 200 times that of the gradient-based method. Moreover, they observed that both methods produce the same Pareto-fronts. Pulliam *et al.* [37] have shown that gradient-free algorithm requires many more flow solutions than adjoint method, but in the end both gradient-free and gradient-based optimization produce very similar designs. Chernukhin and Zingg [5] concluded that the airfoil design using Euler equations is unimodal, even though the wing design had multiple local optima in the absence of viscosity. Similarly, Holst and Pulliam [38] have indicated the unimodality (i.e. local minimum is the global minimum) for airfoil and wing optimization cases. Clearly, gradient-based methods reign supreme in aerodynamic shape optimization problems due to cost savings associated with obtaining the optimum, which is generally in the vicinity of the baseline profile. The airfoil design problem itself is also unimodal, which is beneficial for the gradient-based optimization approach. Finally, Martins [39] claim that the concern for aerodynamic shape optimization being multimodal is unwarranted. Additionally, it is impossible to prove whether an optimum is a local or global one and an optimum should be assumed to be global until proven otherwise [39].

Even though higher fidelity tools that solve the (Reynolds-Averaged) Navier-Stokes equations are used nowadays, the usage of the panel methods in aerodynamic optimization should also be addressed. The growing interest in design optimization with the lower-fidelity tools (which have fast turnaround times) is evident in today's data driven world. The availability of the panel methods and their ability to model configurations with ease, and their relative economy favors the use of these methods for many

applications [40]. A continuous-adjoint based method is used for shape optimization in free-surface potential flows in Ragab [41] for surface ships. Panel methods are used as a medium-fidelity tool in Kennedy *et al.* for aero-structural multi-disciplinary optimization (MDO) problems with a coupled-adjoint approach to obtain the aerodynamic sensitivity derivatives [42, 43, 44]. After that, Morlando [19] used a 2-D panel code, which is also written in Fortran, for sensitivity analysis of airfoils in inverse design using the continuous adjoint method. Maple is used to automate the continuous adjoint equation and its boundary conditions. An adjoint method based on a 3-D panel code is used in Conlan-Smith *et al.* [45] to evaluate the sensitivity derivatives of a wing for aerodynamic shape optimization. Finally, Sarıkaya and Tuncer [46] used *FDOT* to automatically differentiate an in-house panel code using the reverse mode with operator overloading to perform aerodynamic shape optimization studies using the adjoint method.

1.1 Adjoint Methods and Automatic Differentiation

The continuous and discrete adjoint methods differ in a way that in the continuous adjoint method the field equations are used to derive a new set of equations, which are discretized later. The continuous adjoint is also known as the *optimize-then-discretize* approach. In the continuous adjoint method, new boundary conditions must be derived for each objective function due to dependency of the objective function to boundary conditions [47]. This approach is used widely by Jameson and their peers with potential flows, Euler and N-S equations [27, 16, 17, 18, 48, 29]. In the continuous adjoint method, the exact sensitivity of the cost function with respect to the design variables are not obtained. The obtained sensitivities may be not be consistent with the discrete adjoint sensitivities or finite differences. Due to the preliminary work required just to derive the necessary equations, this method is harder to implement.

The discrete adjoint method is known as the *discretize-then-optimize* approach. In this method, the governing equations are discretized first and then optimized later. The exact sensitivity of the objective function with respect to the design variables are obtained, which are consistent with the discrete form of the objective function computed by the discrete form of the governing equations. These properties make

this method appealing for gradient-based optimization studies [49]. Development of a discrete-adjoint solver is easier than a continuous-adjoint one, thanks to automatic differentiation (AD) techniques. The automatic (or algorithmic) differentiation technique allows to users to differentiate a code line-by-line [50]. This technique may be used to develop a discrete-adjoint solver. The discrete-adjoint solvers for CFD can be generated via two methods, operator-overloading and source-code transformation. In the former, the whole code is automatically differentiated as given by Djeddi [51], for example. But the downside is that automatically differentiating the whole code requires substantial amount of memory since every operation is stored in a file called *tape*. It is clear that storing every operation adds overhead compared to storing the useful parts. The issue is alleviated by fixed-point iterations and checkpointing features. *FDOT*, which is used in this study as the AD tool, adresses the high memory footprint issue with the help of checkpointing [51]. In the source-code transformation approach, the code is selectively differentiated by an AD tool as shown in Kaya [49]. The necessary lines are added by the AD tool to compute derivatives after solving the linear system. This way, flux Jacobian matrices are computed with less memory footprint and computational cost than operator-overloading. However, since source-code transformation requires adding to the code with a tool, this step requires more work than operator overloading approach. Both of these methods will be discussed in detail in the Method part.

Nevertheless, it is clear that both methods are useful when used appropriately. The adjoint method, since its introduction, is widely used in the literature due to advantages it provides when computing sensitivities. Moreover, aerodynamic shape optimization using adjoint methods is still a hot topic, as shown in the preceding part. Additionally, the adjoint methods are also used in transition modeling and uncertainty quantification problems [52], which indicates the applicability of the adjoint method in the other aerodynamics related areas.

1.2 Objectives of the study

This study aims at developing a discrete adjoint based aerodynamic design optimization tool for subsonic airfoil profiles. The efficiency and accuracy of panel codes over

the RANS solvers for attached flows is exploited in this study. The developed tool is expected to be effectively used in preliminary and advanced design stages. The discrete adjoint solver for an in-house panel code written in Fortran is obtained by using the automatic differentiation. The main objectives of this study are

- to employ the automatic differentiation tool, *FDOT*, to develop a discrete adjoint solver based on the in-house panel code
- to use the reverse mode of *FDOT* to obtain the adjoint solver
- to validate the adjoint solver and the sensitivity derivatives
- to use Class Shape Transformation (CST) to represent airfoil profiles
- to use an open-source tool, *DAKOTA*, as the optimizer
- to perform aerodynamic shape optimization of subsonic airfoil profiles for target lift and moment coefficients
- to incorporate a multi-point design optimization algorithm
- to enhance the design objective for a reduced adverse pressure gradient over the upper airfoil surface
- to verify the optimum designs against RANS solutions

1.3 The Outline of the Thesis

In the introduction part, the importance of the aerodynamics optimization on the aircraft performance is mentioned. Advantages and disadvantages of various numerical optimization techniques and the related works making use of these techniques are mentioned. Continuous and discrete adjoint methods are discussed. In the Chapter 2, the necessary prerequisites to construct an aerodynamic shape optimization framework are discussed: panel method and the gradient based optimization method is discussed. The automatic differentiation techniques, and shape parametrization techniques are explained in the following subparts. Then, in Chapter 3, validation and verification studies are first presented. The in-house panel code and the open-source RANS solver, SU^2, is validated using the experimental pressure coefficient data from a wind tunnel for various angles of attack. Grid convergence studies for RANS solver

are shown. The sensitivity derivatives are validated by using the adjoint solver and the results are compared with finite differences. The results of the optimization studies are presented and further discussed. The single-point and the multi-point optimization cases are presented with increasing complexity in objective function. In the end, the conclusions drawn are given in the Chapter 4.

CHAPTER 2

OPTIMIZATION METHODOLOGY

In this study an in-house panel code is used for the shape optimization of subsonic airfoil profiles. A gradient-based optimization is performed. The sensitivity derivatives are obtained by means of a discrete adjoint solver. Discrete adjoint method and automatic differentiation (AD) is discussed. Both forward mode and the reverse mode of the adjoint method are explained in detail. The AD tool, *FDOT*, is employed in reverse (adjoint) mode to obtain the adjoint solver. An automatic differentiation example using *FDOT* is presented for a simple objective function. Application of the Class Shape Transformation technique is shown. Subsequently, gradient projection onto the design variables are shown by constructing the shape sensitivity matrix. Finally, the open-source RANS solver used to verify the optimum designs is introduced.

2.1 In-house Panel Code

The existing in-house panel code is based on constant strength source (σ) and vortex (γ) panels distributed on the airfoil surface. The solution is based on the superposition of the potentials induced by the source and the vortex sheets with unknown strengths. The linear system of equations are obtained in order to satisfy the flow tangency over the airfoil surface and the Kutta condition at the airfoil trailing edge for the unknown source and vortex strengths by a Gaussian solver. The velocity field induced by the source and the vortex panels and the surface pressure distribution based on Bernoulli's equation are then evaluated. The panel code written in Fortran 90 is slightly modified for automatic differentiation using *FDOT*. The modifications are further discussed in Chapter 2.2.3.

The panel code solves the Laplacian equation, which governs the steady, incompressible, irrotational and inviscid flows,

$$\nabla^2 \Phi = 0 \tag{2.1}$$

where Φ is the scalar potential function. The potential induced by a source sheet, at a point (x, y) can be obtained by Eq. 2.2,

$$\Phi(x, y) = \int_0^L \frac{\sigma}{2\pi} \ln \sqrt{(x - x_L)^2 + y^2}\, dx_L \tag{2.2}$$

where 0 to L indicate local panel coordinates, and the integration is carried over a panel. The σ distribution over the panels is not known *a priori*.

Additionally, the potential induced at (x, y) by a vortex sheet is given by Eq. 2.3,

$$\Phi(x, y) = \int_0^L \frac{\gamma}{2\pi} \tan^{-1}\left(\frac{y}{x - x_L}\right) dx_L \tag{2.3}$$

where 0 to L indicate local panel coordinates, and the integration is carried over a panel. Similarly, the γ distribution over the panels is not known *a priori*.

The influence of the j^{th} panel on the i^{th} panel can be computed in a discrete manner to obtain the source and vortex influence coefficients,

$$\begin{aligned}\sigma(i, j) = \frac{1}{2\pi}\Bigg(&\sin(\theta_i - \theta_j) \int_0^L \frac{x - x_L}{(x - x_L)^2 + y^2}\, dx_L \\ &+ \cos(\theta_i - \theta_j) \int_0^L \frac{y}{(x - x_L)^2 + y^2}\, x_L \Bigg)\end{aligned} \tag{2.4}$$

$$\begin{aligned}\gamma(i, j) = \frac{1}{2\pi}\Bigg(&\cos(\theta_i - \theta_j) \int_0^L \frac{x - x_L}{(x - x_L)^2 + y^2}\, dx_L \\ &- \sin(\theta_i - \theta_j) \int_0^L \frac{y}{(x - x_L)^2 + y^2}\, dx_L \Bigg)\end{aligned} \tag{2.5}$$

where θ is the angle between the panel and the global x and y axis. After creating the influence coefficients matrix using Eqs. 2.4 and 2.5, local source and vortex panel strengths are obtained with a Gaussian solver. After having obtained these, the local u and v velocities can be computed with,

$$u_L = \frac{\sigma}{2\pi} \int_0^L \frac{x - x_L}{(x - x_L)^2 + y^2} dx_L - \frac{\gamma}{2\pi} \int_0^L \frac{y}{(x - x_L)^2 + y^2} dx_L \tag{2.6}$$

$$v_L = \frac{\sigma}{2\pi}\int_0^L \frac{y}{(x-x_L)^2+y^2}dx_L + \frac{\gamma}{2\pi}\int_0^L \frac{x-x_L}{(x-x_L)^2+y^2}dx_L \tag{2.7}$$

Therefore, using the Eqs. 2.6 and 2.7, it is possible the obtain the velocity distribution over the airfoil and subsequently the pressure distribution with Bernoulli's equation.

2.2 Gradient-Based Optimization

In this study, a gradient based algorithm is employed for the optimization. In general an optimization problem is defined as

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{X}) \\ \text{with respect to} \quad & \mathbf{X} \\ \text{subject to} \quad & g(\mathbf{X}) \leq 0 \\ & h(\mathbf{X}) = 0 \end{aligned} \tag{2.8}$$

In Eq. 2.8, $\mathbf{X}$ is the vector of design variables, f is the objective function, g and h are inequality and equality constraints, respectively [3].

The open source optimization toolbox *DAKOTA* [53] is used to drive the optimization process. The gradient information *DAKOTA* needs is provided by the discrete adjoint solver developed. In this study, *optpp_q_newton* algorithm is chosen, which is the quasi-Newton optimization method suitable for non-linear problems [54]. In this method, a low-rank approximation to the Hessian is computed by means of a BFGS (Broyden-Fletcher-Goldfarb-Shanno) [55, 56, 57, 58] update in each design iteration.

A traditional steepest-descent algorithm requires (which is of first-order):

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \tag{2.9}$$

where α is the stepsize. There is no curvature information is present in Eq. 2.9, which may slow down the convergence. Consider Newton's method given in Eq. 2.10, which also adds curvature information to the problem (second derivatives):

$$x_{k+1} = x_k - \left[{H_k}^{-1}\right] \nabla f(x_k) \tag{2.10}$$

where H_k is the Hessian matrix, which is nothing but the matrix of the second derivatives. Even though the Hessian matrix could be computed using automatic differentiation techniques outlined prior, the computational cost of this procedure is proportional to the number of design variables as shown by Rumpfkeil and Mavriplis [59].

Thus, in this study, the quasi-Newton method is employed where Hessians are approximated. One advantage of quasi-Newton method over the traditional steepest-descent is that convergence is significantly faster in the quasi-Newton method [60]. Jones and Finch [61] suggest that conjugate gradient (CG) and quasi-Newton methods offer clear advantages over the basic steepest-descent algorithm in terms of convergence. In the quasi-Newton optimization algorithm, a new iterate for the design variables is obtained by Eq. 2.11,

$$\mathbf{w}_{k+1} = \mathbf{w}_k - s_k \, B_k^{-1} \, \nabla I(\mathbf{w}_k) \tag{2.11}$$

where B_k is the k^{th} approximation to the Hessian matrix, s_k is the stepsize, and y_k is the yield in the gradients [62] as shown in Eq. 2.12. The optimal stepsize for the new step s_k is obtained via a value based line search, which satisfies the sufficient decrease condition [53].

$$y_k = \nabla_{\mathbf{w}} I(\mathbf{w}_{k+1}) - \nabla_{\mathbf{w}} I(\mathbf{w}_k) \tag{2.12}$$

Hessian matrix is approximated by the BFGS (Broyden-Fletcher-Goldfarb-Shanno) [55, 56, 57, 58] formula for the $(k+1)^{\text{th}}$ iterate using Eq. 2.13

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{2.13}$$

Prior to first update the quasi-Hessian matrix is computed as such,

$$B_0 = \frac{y_k^T y_k}{y_k^T s_k} I \tag{2.14}$$

The convergence is obtained when the relative change in the successive objective function evaluations drop below a certain threshold, i.e. 1×10^{-8}. The optimization process is halted, and the current point is assumed to be optimal. The optimization flowchart is given in Fig. 2.1.

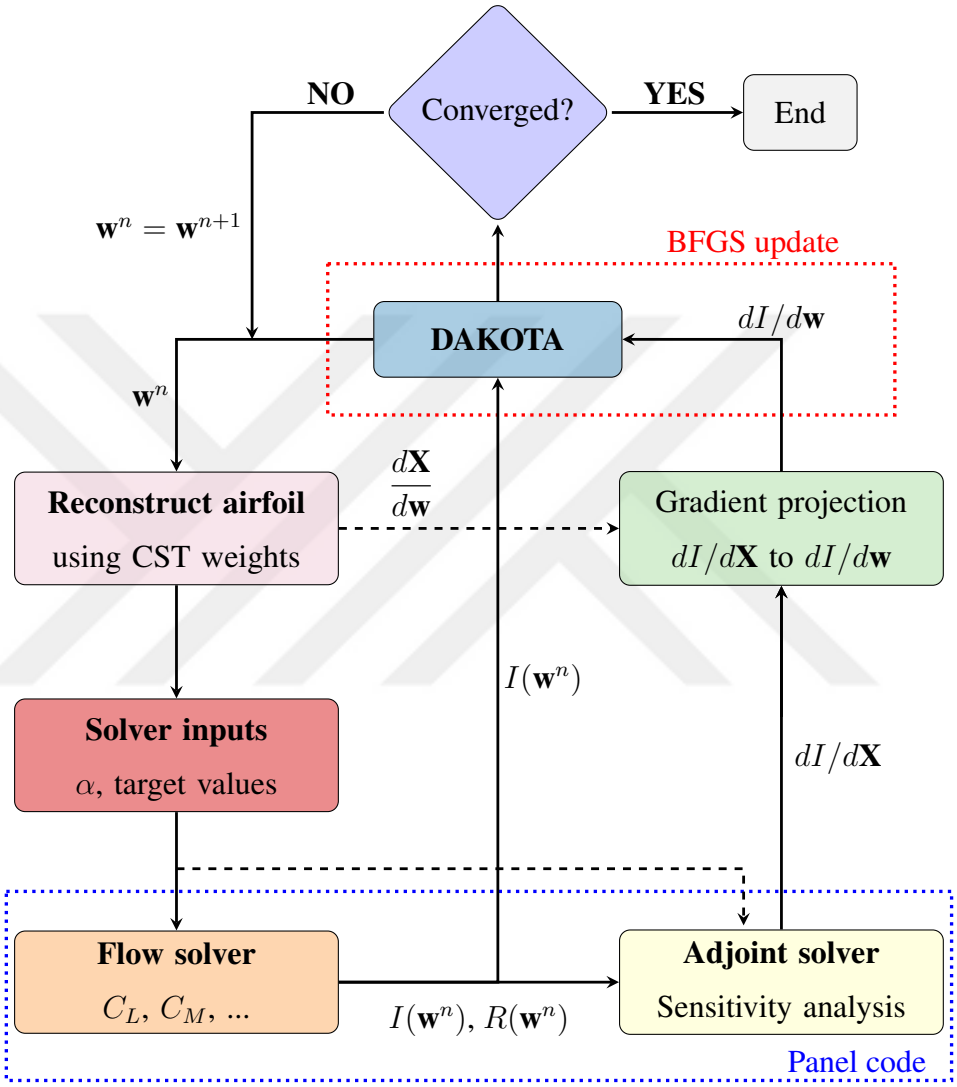


Figure 2.1: Optimization process flowchart

2.2.1 Evaluation of the Sensitivity Derivatives/Gradient Vector

In an aerodynamic shape optimization problem, the objective function is a function of the geometric shape, $\mathbf{X}$, and the flow variables $\mathbf{q}$. The flow variables are a function of $\mathbf{X}$. The objective function generally consists of aerodynamic coefficients such as lift and drag coefficients or a combination of these such as C_L/C_D, which is the aerodynamic efficiency. Without a loss of generality, the choice of objective function, *I*, depends on the optimization problem in hand and is given as

$$I = I\big(\mathbf{X}, \mathbf{q}(\mathbf{X})\big) \tag{2.15}$$

Additionally, since shape parameters, $\boldsymbol{p}$, govern the physical coordinates of the body, $\mathbf{X}$, the physical coordinates are related to the shape parameters such that

$$\mathbf{X} = \mathbf{X}\big(\boldsymbol{p}\big) \tag{2.16}$$

Eq. 2.15 is the objective function definition, which depends only on the shape parameters, $\boldsymbol{p}$. Taking the derivative of the objective function with respect to shape parameters yields the Eq. 2.17.

$$\frac{dI}{d\boldsymbol{p}} = \left(\frac{\partial I}{\partial \mathbf{X}} + \frac{\partial I}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial \mathbf{X}}\right)\frac{\partial \mathbf{X}}{\partial \boldsymbol{p}} = \frac{dI}{d\mathbf{X}}\frac{d\mathbf{X}}{d\boldsymbol{p}} \tag{2.17}$$

Note that $\mathbf{X}$ is the panel coordinates which only depends on the shape parameters as indicated in Eq. 2.16.

In the present study, $dI/d\mathbf{X}$ term given in Eq. 2.17 is evaluated by the algorithmic/automatic differentiation (AD) of the panel code in reverse/adjoint mode. *FDOT* is employed as an AD tool. *FDOT*, which stands for **F**ast Automatic **D**ifferentiation Based on **O**perator-Overloading **T**echnique is a toolbox written by Djeddi and Ekici [63, 51] for Fortran codes. Details are given in Chapter 2.2.3.

Consider the governing equations, which are solved for flowfield evaluation. In Eq. 2.18, the flowfield is denoted by the residual vector $\boldsymbol{R}$. As the governing equations

are always satisfied, the right hand side in the residual equation is set to zero.

$$\mathbf{R} = \mathbf{R}\big(\boldsymbol{p}, \mathbf{q}\left(\boldsymbol{p}\right)\big) = 0 \tag{2.18}$$

In essence, any converged flow solution is of the form,

$$\frac{d\mathbf{R}}{d\boldsymbol{p}} = \frac{\partial \mathbf{R}}{\partial \boldsymbol{p}} + \frac{\partial \mathbf{R}}{\partial \mathbf{q}} \frac{d\mathbf{q}}{d\boldsymbol{p}} = 0 \tag{2.19}$$

where $\mathbf{R}$ is the residual vector. The total derivative of the residual vector with respect to the design variables should vanish as convergence is reached, since the governing equations are satisfied [64].

Therefore, the change in flow variables with respect to design variables term, $d\mathbf{q}/d\boldsymbol{p}$ in equation 2.19, can be solved for

$$\frac{d\mathbf{q}}{d\boldsymbol{p}} = -\left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]^{-1} \frac{\partial \mathbf{R}}{\partial \boldsymbol{p}} \tag{2.20}$$

and Eq. 2.20 may be subsequently inserted into equation 2.17, which yields the basis of sensitivity equation,

$$\frac{dI}{d\boldsymbol{p}} = \frac{\partial I}{\partial \boldsymbol{p}} - \frac{\partial I}{\partial \mathbf{q}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]^{-1} \frac{\partial \mathbf{R}}{\partial \boldsymbol{p}} \tag{2.21}$$

This equation represents the change in the objective function with respect to the design variables, which should be evaluated in the optimization process. In the following subsections, the solution strategies for Eq. 2.21 are discussed. First, the direct method and the finite difference methods are presented, which have cost drawbacks. Then, the adjoint method, which is much more computationally efficient, is introduced.

2.2.1.1 Direct Method

Clearly, one of the challenges is to evaluate the $d\mathbf{q}/d\boldsymbol{p}$ term in Eq. 2.20 due to strong coupling between the design variables and flow variables as mentioned before.

$$-\left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]^{-1} \frac{\partial \mathbf{R}}{\partial \boldsymbol{p}} = \frac{d\mathbf{q}}{d\boldsymbol{p}} \tag{2.22}$$

If the total derivative is to be computed directly, the following linear system [65] must be obtained by manipulating Eq. 2.22

$$-\left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]\frac{d\mathbf{q}}{d\boldsymbol{p}} = \frac{\partial \mathbf{R}}{\partial \boldsymbol{p}} \tag{2.23}$$

Eq. 2.23 may be evaluated via solving the linear system associated. However, the computational cost scales linearly with the number of design variables. In aerodynamic shape optimization problems, the design variables may be on the order of hundreds [66] making the direct solution approach infeasible.

2.2.1.2 Finite Difference Method

In the finite difference method, the gradient vector may be obtained by perturbing each design variable once and obtaining flow solutions by using the finite difference equation:

$$\frac{dI}{d\boldsymbol{p}} = \frac{I(\boldsymbol{p}+h) - I(\boldsymbol{p})}{h} + O(h) \tag{2.24}$$

This method is the easiest one to implement into an optimization framework. However, similar to the direct method, this method also requires repeated flow field evaluation, which is on the order of $(n+1)$ for n design variables for first-order accuracy. The cost scales linearly with the number of design variables, hence this method is also prohibitive in terms of cost. Additionally, this method is prone to truncation errors. Finally, an additional study for the stepsize, h, must be carried out first to determine an optimum perturbation size. This method is very effective for validation purposes as it is straightforward to implement.

2.2.1.3 Adjoint Method

The adjoint method, proposed by Pironneau [25, 26] and extended to compressible flows by Jameson [27] are widely used in aerodynamic optimization problems due to its efficiency. In the adjoint approach, the computational cost scales with the number of objective functions rather than the number of design variables. In general aerodynamic shape optimization problems, the number of design variables are much larger

than the number of objective functions, i.e. $N_{DV} \gg N_{obj}$. For example, the design variables may be spanwise wing stations that control twist, sweep and thickness which may be on the order of hundreds for a large-scale optimization case.

The objective functions are generally integral quantities such as C_D, C_L, C_M which are much less than the number of design variables. Hence, the main advantage of the adjoint method is the dramatic decrease in the computational cost. For example, an adjoint solution has a computational cost that is on the order of 1 flow solution, but finite differences require N_{DV} number of flow solutions, at least.

Consider the following equation which is similar to Eq. 2.21,

$$\frac{dI}{d\boldsymbol{p}} = \frac{\partial I}{\partial \boldsymbol{p}} + \psi^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{p}} \tag{2.25}$$

where ψ^T given in Eq. 2.25 is the *adjoint vector*.

$$\psi^T = -\frac{\partial I}{\partial \mathbf{q}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]^{-1} \tag{2.26}$$

Finally, Eq. 2.26 is solved for the adjoint variable, ψ. After solving for the adjoint vector, the sensitivities with respect to design variables are obtained by carrying out the necessary multiplications shown in Eq. 2.25.

When compared with the direct method or the finite difference methods outlined prior, the adjoint method is much more efficient. However, there may be difficulty in carrying out the necessary derivations such as boundary conditions or new functionals for continuous adjoint formulations. In the continuous adjoint method, by using the newly derived equations, the adjoint variable, ψ, is solved for. In the discrete adjoint approach, the CFD code may be viewed as a series of successive functional operations after initialization [51] such that

$$\begin{aligned} \mathbf{q}_0 &= f_0(\boldsymbol{p}) \\ \mathbf{q}_1 &= f_1(\mathbf{q}_0) \\ &\cdots \\ \mathbf{q}_n &= f_n(\mathbf{q}_{n-1}) \end{aligned} \tag{2.27}$$

where $\mathbf{q}_n$ is the converged solution at n^{th} iteration, which itself is a function of $\mathbf{q}_{n-1}$, and so on. The objective function is evaluated by an operator, $\vec{v}$, acting on the fully-converged solution, yielding

$$I = \vec{v}^T \mathbf{q}_n \tag{2.28}$$

To compute the gradient (sensitivities) of the objective function with respect to the input variables, Eq. 2.28 is differentiated and by the implicit function theorem,

$$\nabla I = \vec{v}^T \left[\frac{\partial f_n}{\partial \mathbf{q}_{n-1}}\right] \left[\frac{\partial f_{n-1}}{\partial \mathbf{q}_{n-2}}\right] ... \left[\frac{\partial f_1}{\partial \mathbf{q}_0}\right] \left[\frac{\partial f_0}{\partial \boldsymbol{p}}\right] \tag{2.29}$$

Eq. 2.29 can be evaluated by n matrix-matrix products and one matrix-vector product at the end. This method is known as the *forward mode* of an adjoint solver. The derivatives are propagated *from the initial solution towards the converged solution*, in the forward direction. The computational effort of matrix-matrix products scale linearly with the number of design variables present in the problem as matrix grows linearly. In aerodynamic shape optimization applications where the number of design variables are quite high, this process leads to significant computational costs. On the contrary, if the Eq. 2.29 is transposed [51],

$$\nabla I^T = \vec{v}^T \left[\frac{\partial f_0}{\partial \boldsymbol{p}}\right]^T \left[\frac{\partial f_1}{\partial \mathbf{q}_0}\right]^T ... \left[\frac{\partial f_{n-1}}{\partial \mathbf{q}_{n-2}}\right]^T \left[\frac{\partial f_n}{\partial \mathbf{q}_{n-1}}\right]^T \tag{2.30}$$

Evaluation of Eq. 2.30 requires n matrix-vector products. This method is known as the *reverse mode* of an adjoint solver. As the name implies, the derivatives are propagated *from the converged solution towards the initial solution*, in the backwards direction. Using the reverse mode, the computational cost is significantly lower but the drawback is that the process requires storing all the intermediate flow solver values in the memory. However, storing all of the intermediate steps require very high memory footprint. More detailed discussions regarding the automatic differentiation procedure, as well as the forward and the reverse modes are given in the Section 2.2.2.

2.2.2 Automatic/Algorithmic Differentiation

Automatic Differentiation -also known as Algorithmic Differentiation- is a method based on the systematic application of the differentiation chain rule to computer programs [50]. In the optimization process, there are need for certain partial derivatives (as shown in the adjoint method) where the specifically developed automatic differentiation (AD) tools are used for evaluation of these terms.

While these derivatives could be computed using finite difference (FD) or complex-step (CS) methods, the computational cost associated with FD and CS are prohibitive. As stated in the preceding parts, the cost of these methods scale linearly with the number of design variables/inputs. However, the AD approach is as accurate as hand-differentiated/analytically differentiated code and requires just the fraction of a cost. The implementation is potentially easier since the process is automated [65], which means that the hassle of lengthy hand derivations or human errors are avoided.

There are two main approaches for AD: *source-code transformation* and *operator-overloading*. In the source-code transformation approach, the code is scanned line by line and new variables are introduced as new source code lines, when necessary. This is done *procedurally* by an AD tool. The AD tool then applies the chain rule in a systematic manner to evaluate the partial derivatives. Finally, the tool outputs an augmented derivative code, which computes the necessary partial derivatives specified by the user.

In the operator-overloading approach (OO/AD), the source-code is not changed significantly like the former approach. In OO/AD, the variable types are of *derived-type*. These variables include both the variable value and also the corresponding derivative [65]. As a consequence, the operations have to be redefined (overloaded) such that they also return the derivative of the operation.

Apart from main approaches to AD, there are also two modes of AD, namely the *forward* and the *reverse* modes. These modes are discussed in-depth in Sections 2.2.2.1 and 2.2.2.2, respectively.

Define the chain rule as follows,

$$\frac{\partial \mathbf{C}}{\partial \boldsymbol{v}} \frac{d\boldsymbol{v}}{d\boldsymbol{c}} = \mathbf{I} = \left[\frac{\partial \mathbf{C}}{d\boldsymbol{v}}\right]^T \left[\frac{\partial \boldsymbol{v}}{d\boldsymbol{c}}\right]^T \tag{2.31}$$

call the left hand side *forward chain rule*, and the right hand side *reverse chain rule* [65].

In an automatically differentiated computer code, the variables $\boldsymbol{v}$ in the Eq. 2.31 are some combinations of assigned variables, which are denoted with t. Line by line, the AD method applies the chain rule for every assignment in the code, which may be considered as functions of their own. These explicit functions are denoted by T, which are functions of the previous assignments and inputs. Hence, T functions depend **only** on previous operations.

Therefore, define the variables and constraints,

$$\boldsymbol{v} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{bmatrix} \qquad \mathbf{C}(\boldsymbol{v}) = \begin{bmatrix} t_1 - T_1() \\ t_2 - T_2(t_1) \\ \vdots \\ t_n - T_n(t_1, \ldots, t_{n-1}) \end{bmatrix} \tag{2.32}$$

2.2.2.1 Algorithmic Differentiation in Forward Mode

Applying the chain rule which is given in the left hand side of Eq. 2.31, the *forward mode* of AD is obtained,

$$\begin{bmatrix} 1 & 0 & \ldots & 0 \\ -\dfrac{\partial T_2}{\partial t_1} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ -\dfrac{\partial T_n}{\partial t_1} & \ldots & -\dfrac{\partial T_n}{\partial t_{n-1}} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \ldots & 0 \\ \dfrac{dt_2}{dt_1} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \dfrac{dt_n}{dt_1} & \ldots & \dfrac{dt_n}{dt_{n-1}} & 1 \end{bmatrix} = \mathbf{I} \tag{2.33}$$

Clearly, the forward mode AD shown in Eq. 2.33, yields two lower triangular matrices. The product of these matrices is still a lower triangular matrix. A rather simpli-

fied form of Eq. 2.33 can be obtained by index notation,

$$\frac{dt_i}{dt_j} = \delta_{ij} + \sum_{k=j}^{i-1} \frac{\partial T_i}{\partial t_k} \frac{dt_k}{dt_j} \tag{2.34}$$

By choosing one t_j and keeping j fixed, and marching in index i from 1 to n, a whole column of the lower triangular matrix is obtained. The result is the **derivatives of all the variables with respect to the chosen variable/input.** Moreover, determining the derivatives for another variable require repeating this computation. Thus, forward mode is more efficient when the number of outputs are larger than the number of inputs. This mode is also known as the tangent mode.

2.2.2.2 Algorithmic Differentiation in Reverse Mode

Applying the chain rule which is given in the right hand side of Eq. 2.31, the *reverse mode* of AD is obtained,

$$\mathbf{I} = \begin{bmatrix} 1 & -\frac{\partial T_2}{\partial t_1} & \dots & -\frac{\partial T_n}{\partial t_1} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & -\frac{\partial T_n}{\partial t_{n-1}} \\ 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{dt_2}{dt_1} & \dots & \frac{dt_n}{dt_1} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & \frac{dt_n}{dt_{n-1}} \\ 0 & \dots & 0 & 1 \end{bmatrix} \tag{2.35}$$

The reverse mode AD shown in Eq. 2.35, yields two upper triangular matrices. The product of these matrices is still an upper triangular matrix. A rather simplified form of Eq. 2.35 can be obtained by index notation,

$$\frac{dt_i}{dt_j} = \delta_{ij} + \sum_{k=j+1}^{i} \frac{dt_i}{dt_k} \frac{\partial T_k}{\partial t_j} \tag{2.36}$$

In the reverse mode, a t_i is chosen, which is the quantity to be differentiated. Then, marching *backwards* in the index j from n to 1, a column of the upper triangular matrix is obtained. The result is **the derivatives of the chosen quantity with respect to all other variables/inputs**. Thus, reverse mode is more efficient when the number of inputs are larger than the number of outputs. For example, the sensitivity of lift coefficient (output size 1) with respect to mesh nodes (input size of N_{mesh}).

2.2.2.3 Source Code Transformation

The AD modes are discussed in detail in the previous section. In this section, the source code transformation (SCT) method, which is one of the implementation methods of these modes is discussed. In the source code transformation method, -as the name implies- the source code is differentiated line by line, and the new variables (derivatives and temporary variables) are introduced as necessary by the AD tool. This explicit transformation of the main code (primal) to a separate derivative (adjoint) program requires careful tooling. In source-code transformation approach all the intrinsic functions ($\sin$ $\cos$ etc.) must explicitly be supported [67] otherwise derivatives cannot be computed.

For SCT, there are various tools available for multiple programming languages. For C/C++, AD tools that make use of SCT are ADIC [68], OpenAD [69], TAPENADE [70], to name a few. Similarly, for Fortran, TAF [71] OpenAD/F [72] ADIFOR [73] TAPENADE [70] are some of the toolboxes that make use of SCT. For other programming languages such as MATLAB (ADIMAT [74]), Python (Tangent [67]), Julia, R and so on, the reader is referred to autodiff [1] website for further information about the subject.

2.2.2.4 Operator Overloading

In operator-overloading (OO) approach changes in the source code are negligible. In OO, the operations are "overloaded", i.e. elementary operation semantics are redefined [75]. Newly derived variable types that contain both the value of the variable and a corresponding index are used to record operations in a tape [63]. The recorded tape may be rewound, since each operation is logged to a tape with previously redefined variables at the runtime. With the help of chain rule, the derivatives are then evaluated.

For C/C++, some of the OO-AD tools are FAD [76] and CoDiPack [77], which is used in the discrete-adjoint module of SU^2. For Fortran, ADOL-F [78], ADF95 [79], AUTO_DERIV [80] are some of the AD tools that make use of operator overloading.

[1] https://www.autodiff.org

However, recording each operation in a tape may exceed the available memory resources as the expression tree grows (e.g. more operations, bigger mesh) which is a big problem for operator overloading. According to Djeddi and Ekici, none of the current Fortran OO implementations have adressed the inherently large memory footprint issue [63].

Djeddi [51], Djeddi and Ekici [63] developed a new toolbox (named *FDOT*) based on operator overloading with huge reductions in the memory footprint for Fortran codes. In this study, *FDOT* is used to automatically differentiate an in-house panel code written in Fortran. Details regarding *FDOT* are given in the next section.

2.2.3 Automatic Differentiation with FDOT

In the present study, $dI/d\mathbf{X}$ term, which is given in Eq. 2.17, is evaluated by the automatic differentiation of the panel code in reverse/adjoint mode. *FDOT* is employed as an AD tool. *FDOT*, which stands for **F**ast Automatic **D**ifferentiation Based on **O**perator-Overloading **T**echnique is a toolbox written by Djeddi and Ekici [63, 51] for Fortran codes.

The high-memory requirements of a conventional operator-overloading code is alleviated by the checkpointing feature and the fixed-point iteration approach in *FDOT* [64]. The fixed-point iteration approach for adjoints are proposed in 1994 by Christianson [81] and the idea is implemented in codes like ADOL-C [82] and CoDiPack [77], both of which use the object-oriented features of C/C++ languages. Currently, no other AD tool for Fortran addresses the high memory issues [63].

FDOT can be used as a black-box for automatic differentiation of a solver with minor modifications to the Fortran source code. The differentiated code, known as the adjoint solver, requires the flow solution for initialization. The computational cost of the adjoint solver is proportional to one flow solution [63]. In the end, the adjoint solver returns $dI/d\mathbf{X}$ values, which are the sensitivities of the objective function with respect to the panel coordinates. The size of the computed sensitivity vector is equal to the number of panels, N_{panel}

To automatically differentiate a code, the `REAL` variables are redefined as `AREAL`

which is a derived data type used in the *FDOT* module. The derived data type object holds a *real* component and an *index* component. The derived data type stores 8-byte, double precision float for former and 4-byte double-precision integer as the latter. After that, a tape size is defined and tape allocation subroutine is called in the adjoint code. The expressions in the code are carried out as normal but these expressions are written into the tape simultaneously. Then, the objective function is computed and passed onto the relevant subroutine. Finally, by calling adjoint evaluation subroutine the adjoints are evaluated by employing reverse-mode (i.e. the tape is rewound). The user has control over the tape iteration convergence tolerance, iterative array sizes, maximum iterations for fixed-point iteration procedure and checkpoint locations.

In general, CFD codes consist of 3 parts, pre-iterative, iterative and post-iterative parts. In the pre-iterative section, mesh, solver settings and initialization is carried out. In the iterative part, the solution is obtained iteratively and convergence checks are carried out. Finally, in the post-iterative part, the converged solution, objective function and the necessary files are written into the disk as the outputs.

In the adjoint code, there are minor changes to the code. In the pre-iterative part, the mesh and the solver settings are read as usual. However, the converged flow solution is read instead of initializing. After that, the iterative portion of the solver is marked with checkpoints for better memory efficiency. Then, this portion is run for **exactly 1 iteration** only, since *FDOT* makes use of fixed-point iteration approach. After evaluating the adjoints, the output is written to the disk.

As an example, consider a simple Fortran code given in Fig. 2.2a, which calculates $A = \sin(\mathrm{B}) + \mathrm{C}^2$.

B and C are user inputs taken as 0.0 and 3.0, respectively. the analytic derivatives are,

$$\frac{\partial \mathrm{A}}{\partial \mathrm{B}} = \cos(\mathrm{B}) = 1.0 \tag{2.37}$$

$$\frac{\partial \mathrm{A}}{\partial \mathrm{C}} = 2\mathrm{C} = 6.0 \tag{2.38}$$

Algorithmically differentiated code with *FDOT* provides the adjoint solver, which is

given in Fig. 2.2b. The adjoint solver computes the derivatives of `output`, which is considered as the objective, with respect to the input variables `input1`, `input2` in reverse mode. The print statements at the end outputs the partial derivatives of the objective function with respect to all the variables at once, which shows the efficiency of the adjoint solver in reverse mode. The output is shown in Fig. 2.3, where the derivatives are equal to the analytical values as expected.

The `TAPE_R8(input1%INDEX)%A` term is the derivative of the `output` with respect to `input1`. The `TAPE_R8(input2%INDEX)%A` term is the derivative of the `output` with respect to `input2`.

```
1  program example
2  real :: input1,input2, output
3
4  input1 = 0.0
5  input2 = 3.0
6  output = (sin(input1) + input2**2)
7
8  print*, "input1 = ",input1
9  print*, "input2 = ",input2
10 print*, "output = ",output
11
12 end program
```

(a) Forward code

```
1  program OO_AD_example
2  use FDOT
3  type(AREAL) input1,input2, output
4  TAPE_SIZE = 500
5  call ALLOCATE_TAPES
6
7  input1 = 0.0
8  input2 = 3.0
9  output = (sin(input1) + input2**2)
10
11 call SET_OBJECTIVE(output)
12 call ADJOINT_EVALUATION
13 call PRINT_TAPE
14
15 print*, "input1 = ",input1%v
16 print*, "input2 = ",input2%v
17 print*, "output = ",output%v
18 print*, TAPE_R8(input1%INDEX)%A
19 print*, TAPE_R8(input2%INDEX)%A
20
21 end program
```

(b) Adjoint code

Figure 2.2: Example forward and adjoint source codes for computation of $A = \sin(\mathrm{B}) + \mathrm{C}^2$

```
1  input1 =   0.00000000000000
2  input2 =   3.00000000000000
3  output =   9.00000000000000
4  1.00000000000000
5  6.00000000000000
```

Figure 2.3: Output of the adjoint code

2.2.4 Airfoil Shape Parametrization

In aerodynamic shape optimization problems, there are many ways to tackle the shape parametrization problem. Shape parametrization is one of the crucial steps, since sensitivity of grid coordinates with respect to design variables, namely, $\partial \mathbf{X}/\partial \mathbf{w}$ are obtained in this step. The robustness and design space coverage are important aspects that require close attention. If the parametrization technique is not able to adequately span the design space, then the optimizer may get stuck on a non-optimal point.

In the literature, there are many parametrization methods such as B-Splines, PARSEC and Class Shape Transformations (CST) and Free-Form Deformation (FFD). These parametrizations are useful in a way that they reduce the number of design variables significantly to represent the same shape. Moreover, making use of parametrizations allow obtaining smoother shapes.

2.2.4.1 Class Shape Transformation

In this study, Class Shape Transformation (CST) which is devised by Kulfan [83, 84] is used for the parametrization of airfoil profiles with sharp trailing edges. It is shown by Nadarajah *et al.* [85] that CST performs well in 2-D applications when compared to mesh points, which had the highest accuracy. In the same study it is shown that CST based shape optimization is able to converge to the same minimum drag profile as the ones based on the mesh point and B-spline techniques using only 5 design variables.

CST representation of an airfoil may be obtained by using a *class function* and a *shape function*:

$$\zeta = C_{N2}^{N1}(\psi)\, S(\psi) \, + \psi \, \Delta\tau \tag{2.39}$$

where ζ in Eq. 2.39 is the non-dimensional y-distance, ψ is the non-dimensional x-distance, C_{N2}^{N1} is the class function with coefficients N1 and N2, $S(\psi)$ is the shape function, and $\Delta\tau$ is the trailing edge thickness to chord ratio. Note that due to the sharp TE requirement of the panel code, the trailing edge thickness term is omitted.

The class function is given by

$$C_{N2}^{N1}(\psi) = (\psi)^{N1}\,(1-\psi)^{N2} \tag{2.40}$$

Class functions are used to define "classes" of geometries [86], which are given in Table 2.1. In this study, to create a round-nose, and a sharp trailing edge exponents N1 and N2 are set to 0.5 and 1.0, respectively.

Table 2.1: Classes defined by exponents N1 and N2

N1	N2	Class type
0.5	1.0	Round-nose, pointed aft end airfoil
0.5	0.5	Elliptic airfoil
1.0	1.0	Bi-convex airfoil
0.75	0.25	Low-drag projectile
0.75	0.75	Sears-Haack Body

Using Bernstein polynomials of order n, the shape function is obtained by

$$S(\psi) = \sum_{i=0}^{n} \mathrm{w}_i S_i(\psi) = \sum_{i=0}^{n} \mathrm{w}_i \big(K_i\,\psi^i\,(1-\psi)^i\big) \tag{2.41}$$

in Eq. 2.41, K_i is the binomial coefficient which is equal to $\binom{n}{i}$. Therefore, the class shape transformation is expressed by combining Eqs. 2.39 and 2.41,

$$\zeta = C_{N2}^{N1} \sum_{i=0}^{n} \mathrm{w}_i \left(K_i\,\psi^i\,(1-\psi)^i\right) \tag{2.42}$$

In Eq. 2.42, **w** is a vector of coefficients that is to be determined with a size of $2(n+1)$. This vector of weights determine the unique airfoil shape. Note that Eq. 2.42 uses the class function defined in Eq. 2.40 with coefficients N1 = 0.5 and N2 = 1, which is required for a round LE and sharp TE. Moreover, Eq. 2.42 represents either the lower or the upper surface of the airfoil, which will change depending on the signs of the weights. For upper surface, there is a set of unique w_i values, and similarly there are unique w_i values for the lower surface. Ultimately, these values should be obtained using an optimization or a curve fitting method to represent the baseline airfoil using CST.

Additionally, two of the CST parameters are intuitive. These parameters define the leading edge radius and trailing edge boat-tail angle. Shape function at the ψ = 0 only influences the leading edge radius by,

$$S(0) = \sqrt{\frac{2\, r_{LE}}{c}} \tag{2.43}$$

Shape function at the ψ = 1 influences only the trailing edge boat-tail angle, β,

$$S(1) = \tan\beta + \Delta\tau \tag{2.44}$$

where $\Delta\tau$ is the trailing edge thickness for blunt TE airfoils. It should be noted that due to sharp TE requirement for the panel code, this term is omitted from the formulation. All the other terms affect the thickness distribution of the airfoil. Fig. 2.6 shows the effect of perturbing some CST weights.

By the Weierstrass approximation theorem, there always exists a set of coefficients to bound the approximation error within a small magnitude using finitely many terms [87]. Thus, in the present study, a code is developed which minimizes the difference between the analytical coordinates and the CST reconstruction [46]. The airfoils are approximated with sufficient accuracy using 4 to 10 Bernstein polynomials, depending on the airfoil, with RMS on the order of 10^{-5}. In Figure 2.4, analytic reconstruction of NACA 0012 airfoil is shown with the circles, and 5^{th} order CST reconstruction of the same airfoil is plotted in red. RMS value is on the order of 10^{-5}.

In the Figure 2.5, Bernstein polynomials (shape function components), $w_i\, S_i$, are plotted for NACA0012 using 5^{th} order CST. This is a visual representation of Eq. 2.41, where the Bernstein polynomials of order 5 result in 6 different curves. Additionally, the sum of the Bernstein polynomials are then multiplied with the class function to obtain the CST representation of the NACA0012 airfoil (see Eq. 2.42). Clearly, the CST reconstruction of the airfoil perfectly agrees with the analytical description of NACA0012. In Table 2.2, the CST weights are given for NACA0012 airfoil, which could be used to reconstruct the airfoil with the method outlined above by multiplying the weights with the appropriate Bernstein polynomials. First half corresponds to the lower surface and the other half corresponds to the upper surface.

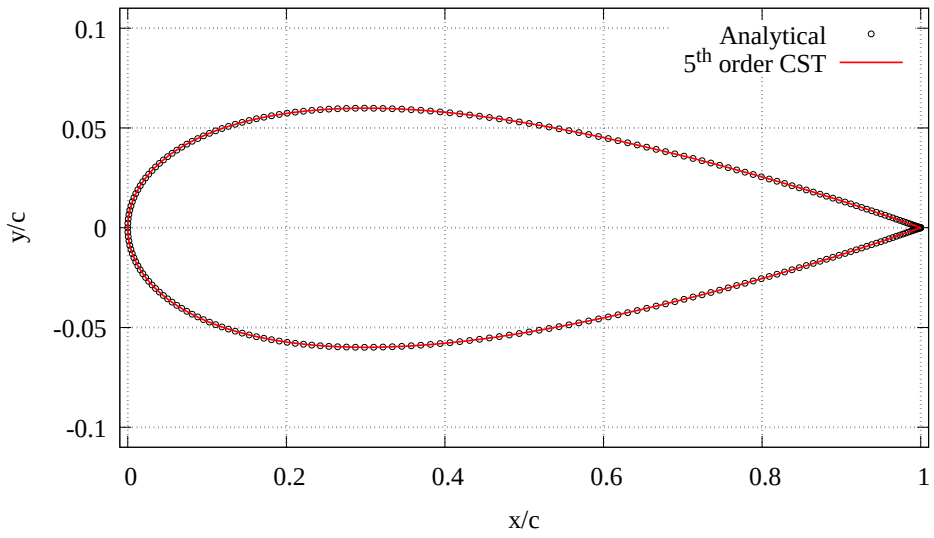


Figure 2.4: 5^{th} order CST representation of NACA0012

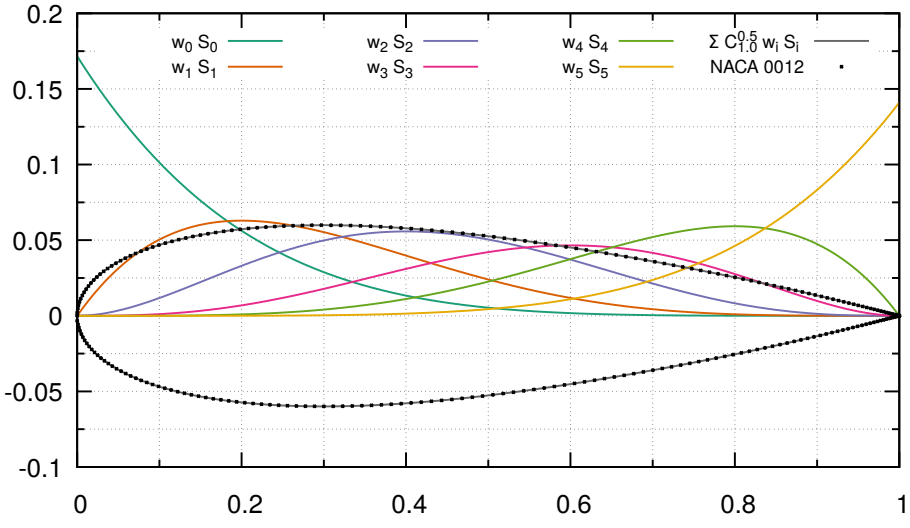


Figure 2.5: 5^{th} order CST components of NACA0012.

Table 2.2: 5^{th} order CST weights for NACA0012

Number (i)	Value
0	-0.171652
1	-0.153572
2	-0.161381
3	-0.134693
4	-0.144539
5	-0.140987
6	0.171652
7	0.153572
8	0.161381
9	0.134693
10	0.144539
11	0.140987

2.3 Gradient Projection

If a shape parametrization technique is used, the computed gradients with respect to panel coordinates need to be projected onto the design variables. The procedure outlined in this section is not necessary if mesh nodes are directly used as design variables since $\boldsymbol{p} = \mathbf{X}$. To accomplish this consider the Eq. 2.17 again, which is the general sensitivity equation,

$$\frac{dI}{d\boldsymbol{p}} = \left(\frac{\partial I}{\partial \mathbf{X}} + \frac{\partial I}{\partial \mathbf{q}} \frac{d\mathbf{q}}{d\mathbf{X}} \right) \frac{\partial \mathbf{X}}{\partial \boldsymbol{p}} = \frac{dI}{d\mathbf{X}} \frac{d\mathbf{X}}{d\boldsymbol{p}} \tag{2.45}$$

In Eq. 2.45, $\mathbf{X}$ denote mesh variables, $\mathbf{q}$ denote flow variables, and $\boldsymbol{p}$ denote design variables. In Section 2.2.1, computation of the gradient with respect to mesh variables are explained. Recall that the computed gradient corresponds to $dI/d\mathbf{X}$ term in the RHS of Eq. 2.45. Afterwards, in Section 2.2.4.1, CST parametrization technique is explained, which represents the airfoil shape. Therefore, only the shape sensitivity matrix is yet to be constructed, which is the $d\mathbf{X}/d\boldsymbol{p}$ term on the RHS of Eq. 2.46. In this section, shape sensitivity matrix and the gradient projection is discussed.

For cases where the volume mesh is of a concern, the volume mesh is deformed after the surface mesh deformation has been applied. However, in a panel code there is only surface mesh and subsequently there is no volume mesh to deform. Hence, $[d\mathbf{X}_V/d\mathbf{X}_S]$ term given in Eq. 2.46 drops out, and in this case $N_V = N_S$. The airfoil x-coordinates are fixed in this procedure. It is done to prevent the airfoil from shrinking or expanding in the chordwise direction.

Generalized gradient projection,

$$\left[\frac{dI}{d\boldsymbol{p}}\right]_{1\times N_{DV}} = \left[\frac{dI}{d\mathbf{X}_V}\right]_{1\times N_V} \left[\frac{d\mathbf{X}_V}{d\mathbf{X}_S}\right]_{N_V\times N_S} \left[\frac{d\mathbf{X}_s}{d\boldsymbol{p}}\right]_{N_S\times N_{DV}} \tag{2.46}$$

For a panel code, the term involving volume mesh drops out and the following is obtained,

$$\left[\frac{dI}{d\boldsymbol{p}}\right]_{1\times N_{DV}} = \left[\frac{dI}{d\mathbf{X}_S}\right]_{1\times N_S} \left[\frac{d\mathbf{X}_s}{d\boldsymbol{p}}\right]_{N_S\times N_{DV}} \tag{2.47}$$

Thus, $d\mathbf{X}/d\boldsymbol{p}$ term is required to project the gradients obtained in the physical space onto the design variables in the design space. Consider the design variable vector

$\boldsymbol{p} = [\alpha_1, \alpha_2, \ldots, \alpha_N]$. If α_i is perturbed by $\Delta\alpha$, a new design (geometry& mesh) is obtained. The $\Delta\alpha$ value should be sufficiently small, but not too small such that numerical errors arise. Example perturbations are shown in Fig. 2.6, which are overlaid over the original airfoil. Every other CST weight is perturbed on the upper surface with $\Delta\alpha = 0.1$. It is evident that the airfoil shapes change drastically. Additionally, $d\mathbf{X}/d\boldsymbol{p}$ magnitudes for CST weights are also shown in the same figure. Every perturbation of a design variable leads to a vector, and perturbing all the design variables form a matrix. Thus, $d\mathbf{X}/d\boldsymbol{p}$ matrix is constructed using finite differences. The shape sensitivity matrix has a size of $N_S \times N_{DV}$. The computational time is insignificant in contrast to a flow/adjoint solution.

Therefore, if the x-components of the nodes are held fixed, the shape sensitivity matrix is given as,

$$\begin{bmatrix} \mathbf{y}(\alpha_1 + \Delta\alpha) & \mathbf{y}(\alpha_2 + \Delta\alpha) & \ldots & \mathbf{y}(\alpha_{N_{DV}} + \Delta\alpha) \end{bmatrix} \tag{2.48}$$

where **y** denotes y-coordinates of the airfoil.

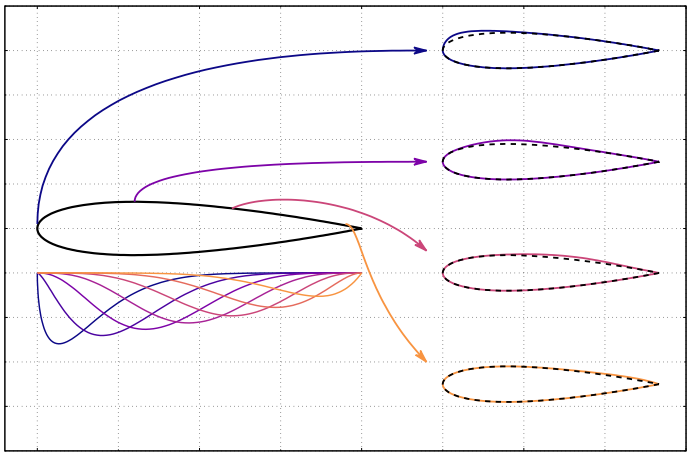


Figure 2.6: Perturbed variants of NACA0012 and shape sensitivity magnitudes

2.4 RANS solver: SU^2

RANS solutions are used to verify attached flow solutions and the aerodynamic loads predicted by the panel code. For this purpose, the well-known, open-source CFD suite, SU^2 is employed [88]. SU^2 stands for *Stanford University Unstructured*. Being an open-source code, there are many groups actively contributing to the code [2].

SU^2 solves the Reynolds-Averaged Navier Stokes equations together with a turbulence model. In the present study SU^2 simulations are carried out on C-grids with $y^+ \approx 1$. The inviscid fluxes are computed using Roe's scheme. The gradients of the flow variables are computed by the Green-Gauss method, which are then used to reconstruct the *second-order* solution. Since Reynolds number is quite high, the flowfield is assumed to be fully turbulent, and Menter's SST model [89, 90] is used for the turbulence closure. An implicit time integration is used to allow high CFL numbers. The resulting linear system of equations is solved with BCGSTAB and ILU preconditioner. Cauchy criteria is used in convergence assessment. The convergence is ensured by keeping the relative changes in C_L, C_D and C_M are less than 1×10^{-5} in the last 250 iterations.

[2] https://github.com/su2code/SU2

CHAPTER 3

RESULTS AND DISCUSSION

In this chapter, validation and verification studies are performed first. Airfoil design optimization studies are then presented. The design optimizations are first performed for target lift coefficient, then for the combination of target lift and moment coefficients. A term to reduce the adverse pressure gradient is added to the objective function in order to improve the stall characteristics of the designed airfoil. Finally, multi-point design optimizations are then carried out in order to further improve the off-design performance of airfoils.

3.1 Validation and Verification Studies

Sensitivity derivatives of the lift and the pitching moment coefficient computed by the adjoint solver are validated first. These sensitivity derivatives are computed over a symmetric and a cambered airfoil, respectively. The verification of RANS solutions with SU^2 and a grid convergence study are performed next.

The sensitivity derivatives computed by the adjoint method are validated with finite difference method. The sensitivity derivatives match well with finite differences, and have an accuracy of at least 5 significant digits. Additionally, the pressure coefficient predictions of the in-house panel code and RANS solver are in very good agreement with the experimental data for attached flows.

3.1.1 Validation of the Sensitivity Derivatives

The computed sensitivity derivatives, i.e. gradient vector, may be validated by using finite differences. The idea is to perturb each design variable one by one by using the following formulae, which yield first-order and second-order accurate derivatives, respectively:

$$\frac{dI}{d\xi} = \frac{I(\xi + h) - I(\xi)}{h} + O(h) \tag{3.1}$$

$$\frac{dI}{d\xi} = \frac{I(\xi + h) - I(\xi - h)}{2h} + O(h^2) \tag{3.2}$$

In essence, the gradients are calculated by repeatedly evaluating the flowfield. The number of required flowfield solutions are double that of a first-order scheme if a second-order accurate scheme is chosen. Moreover, if higher-order accurate gradients are desired, the number of flowfield evaluations required scale linearly. Additionally from Eqs. 3.1 and 3.2, it is clear that finite differences are prone to truncation errors. To summarize, one needs $(n+1)$ flow solutions to obtain first-order accurate gradients and $(2n + 1)$ flow solutions for second-order accurate gradients.

3.1.1.1 Validation of the Sensitivity Derivatives for Target Lift coefficient

The objective function is

$$I = \left| 1 - \frac{C_L}{0.15} \right| \tag{3.3}$$

where C_L is the lift coefficient and 0.15 is the target lift coefficient. Eq. 3.3 represents the optimization case where only the target lift coefficient is set. NACA0012 airfoil is parametrized with 5^{th} order CST, resulting in 12 design variables. Making use of design variable perturbations and Eq. 3.2, the gradients are obtained using finite difference method, which require additional *24 flow solutions*. Also, using only 1 flow solution and 1 adjoint solution the gradients are calculated by the automatically differentiated panel code. The numerical values of the gradient vector are given in Table 3.1. The values obtained by the AD panel code is more accurate than those of FD since in FD there is truncation and round-off errors whereas the AD toolbox, FDOT, is accurate down to the machine precision [51]. From the table, it is clear that at least 5 digits of accuracy is obtained for all the design variables. Additionally, AD

and FD results are plotted in Fig. 3.1. First half of the design variables correspond to the lower surface of the airfoil and the second half of the design variables correspond to the upper surface of the airfoil. The CST design variables start from leading edge (Number 0 & 6) to trailing edge (Number 5 & 11). It is seen that the trailing edge is much more sensitive to the lift coefficient than leading edge. This localized high sensitivity region at the trailing edge may form a cusp in the optimization studies, as shown in Sarıkaya and Tuncer [46]. The behavior of the sensitivity derivatives is also evident from the Fig. 3.2, where dI/dy vs. x/c is plotted over the airfoil surface with vectors. Deforming the airfoil in the direction of the arrows is going to increase the I. Since the aim is to minimize I in the problem formulations, the airfoil is going to be deformed in the opposite sense of the vectors, which is consistent with the sensitivity derivatives obtained in Fig. 3.1 and Table 3.1. For example, if the lower surface of the airfoil is perturbed in the direction of the arrows (-y direction), the dy value is clearly going to be negative. Since the dI/dy value there is also negative, plugging in a negative dy value results in positive dI, which indicates that the perturbed airfoil is going to be located further away from the optimum than the original airfoil.

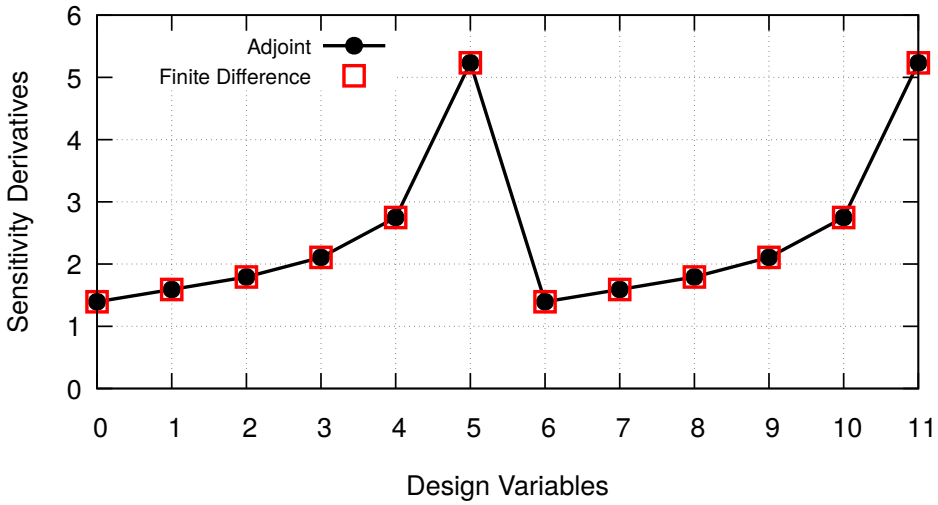


Figure 3.1: AD vs FD for NACA0012, target lift coefficient

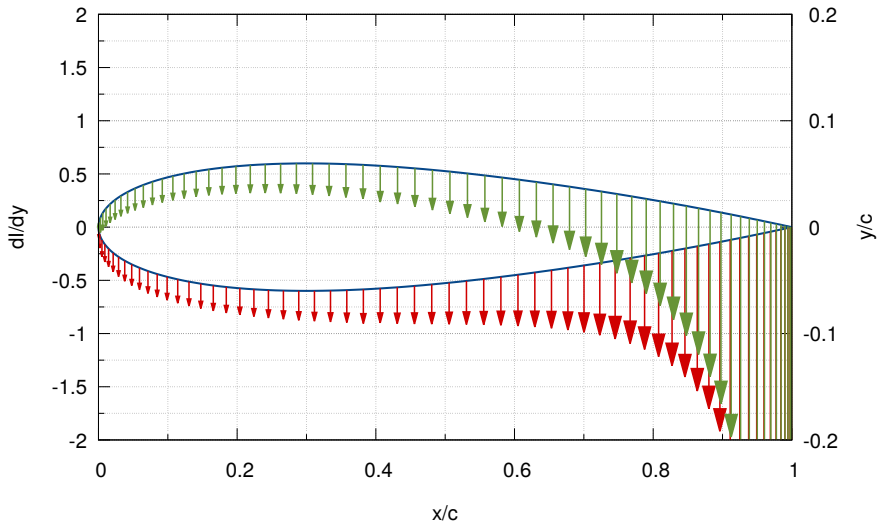


Figure 3.2: dI/dy for NACA2412, target lift coefficient

Table 3.1: Lift Coefficient Sensitivity derivatives, AD and FD, NACA0012

DV	AD	FD
0	**1.3939**7052	**1.3939**6593
1	**1.5914**7946	**1.5914**8526
2	**1.79220**471	**1.79220**757
3	**2.10718**406	**2.10717**891
4	**2.74873**291	**2.74874**206
5	**5.23381**638	**5.23381**985
6	**1.3936135**2	**1.3936130**9
7	**1.5906103**2	**1.5906140**2
8	**1.7918279**9	**1.7918297**1
9	**2.1072**7332	**2.1072**6515
10	**2.74883**751	**2.74883**820
11	**5.2349**1865	**5.2349**0385

3.1.1.2 Validation of the Sensitivity Derivatives for Target Moment Coefficient

For this validation case, the objective function is chosen to be

$$I = \left| 1 - \frac{C_M}{-0.04} \right| \tag{3.4}$$

where C_M is the pitching moment coefficient and -0.04 is the target moment coefficient. Eq. 3.4 represents the case where only the target moment coefficient is set. NACA2412 airfoil is parametrized with 5^{th} order CST, resulting in 12 design variables. Making use of design variable perturbations and Eq. 3.2, the gradients are obtained using finite difference method, which require *24 flow solutions*. Also, using only 1 flow solution and 1 adjoint solution the gradients are calculated by the automatically differentiated panel code. The numerical values of the gradient vector are given in Table 3.2. The values obtained by the AD panel code is more accurate than those of FD since in FD there is truncation and round-off errors whereas the AD toolbox, FDOT, is accurate down to the machine precision. From the table, it is clear that at least 5 digits of accuracy is obtained for all the design variables. Additionally, AD and FD results are plotted in Fig. 3.3. First half of the design variables correspond to the lower surface of the airfoil and the second half of the design variables correspond to the upper surface of the airfoil. The CST design variables start from leading edge (Number 0 & 6) to trailing edge (Number 5 & 11). Again, it is seen that the trailing edge is more sensitive to the moment coefficient than the leading edge, but now it is in the opposite direction. Overall, it is seen that the sensitivities are generally oriented towards decreasing the camber, especially near the trailing edge. Additionally, the sensitivities near leading edge try to push the leading edge region up (increase the camber in leading edge) and try to decrease the camber downstream. This behavior is also evident from the Fig. 3.4, where dI/dy vs. x/c is plotted over the airfoil surface with vectors. Deforming the airfoil in the direction of the arrows is going to increase the I. Since the aim is to minimize I in the problem formulations, the airfoil is going to be deformed in the opposite sense of the vectors, which is consistent with the sensitivity derivatives obtained in Fig. 3.3 and Table 3.2. Additionally, note that changes near the $0.25c$ have little effect, since the pitching moment is computed with respect to this point.

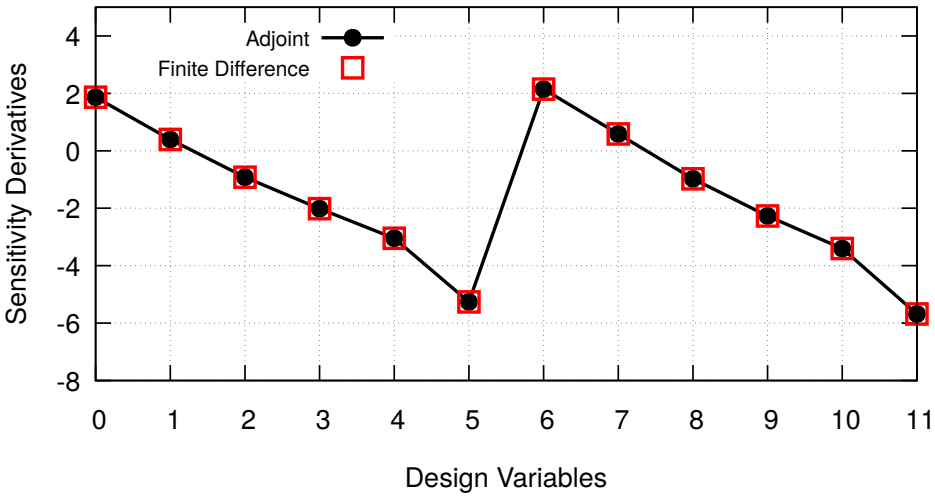


Figure 3.3: AD vs FD for NACA2412, target moment coefficient

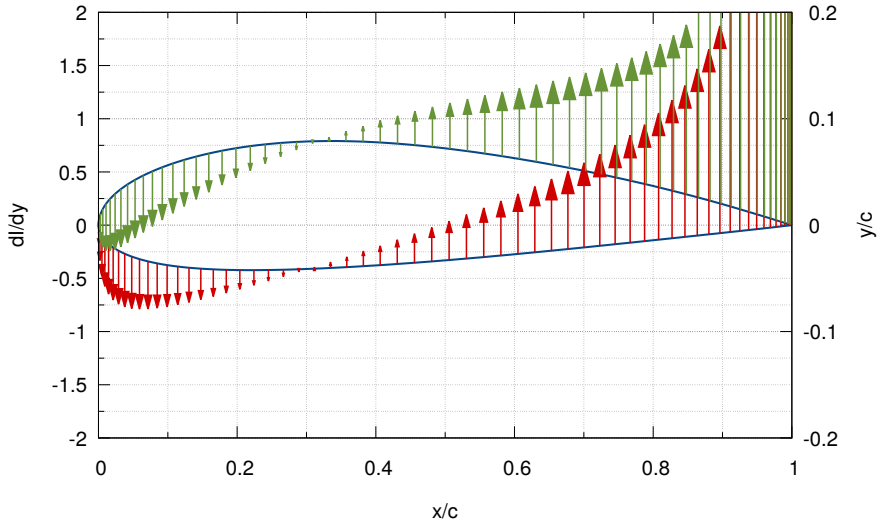


Figure 3.4: dI/dy for NACA2412, target moment coefficient

Table 3.2: Moment Coefficient Sensitivity derivatives, AD and FD, NACA2412

DV	AD	FD
0	**1.86756**408	**1.86756**089
1	**0.3953**6994	**0.3953**7268
2	**-0.9212035**5	**-0.9212036**6
3	**-2.01479**215	**-2.01479**072
4	**-3.0447**7785	**-3.0447**8626
5	**-5.2638**0118	**-5.2638**1702
6	**2.1494695**6	**2.1494691**3
7	**0.58537**172	**0.58537**679
8	**-0.9746**7666	**-0.9746**6980
9	**-2.2706**2750	**-2.2706**1768
10	**-3.4042**7100	**-3.4042**6584
11	**-5.6826**0108	**-5.6826**2290

3.1.2 Verification of Flow Solvers

In this case, the available experimental data from Gregory and O'Reilly [91] are used to validate the pressure coefficient predictions of the flow solvers (i.e. panel and SU^2) over a NACA0012 airfoil. Pressure coefficient distributions for angles of attack of 0, 2, 6 and 10 deg are plotted in Figs. 3.5 to 3.8. Solid lines and dashed lines indicate panel and fully-turbulent RANS predictions, respectively. RANS solutions are carried out using SU^2.

SU^2 simulations are carried out using a proper C-grid with $y^+ \approx 1$ for Re$= 5\times10^6$ and M $= 0.3$. The inviscid fluxes are computed using Roe's scheme, with second-order reconstruction. Since the Reynolds number is quite high, the flowfield is assumed to be fully turbulent. The closure in RANS equations are achieved using Menter's SST model. If the C_L, C_D and C_M change in the last 250 iterations is less than 1×10^{-5} the solution is assumed to be converged. Note that the grids used in this case are generated according to the grid convergence results, which are presented in Part 3.1.2.1.

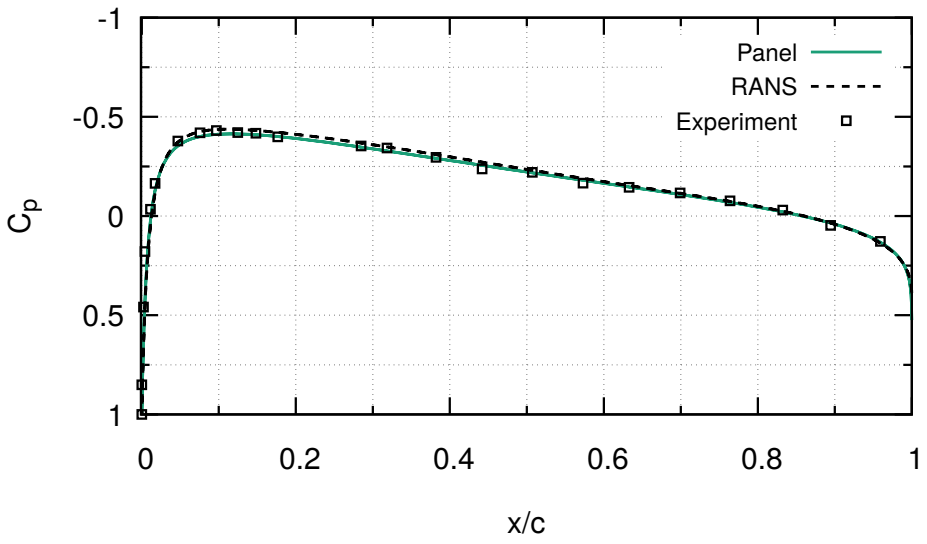


Figure 3.5: Pressure coefficient distributions over NACA0012 airfoil at $0\,\mathrm{deg}$ angle of attack

Fig.3.5 shows the Panel and RANS C_p predictions over NACA0012 airfoil at α = 0 deg. The pressure coefficient distribution is symmetric due to the symmetric nature

of the airfoil, which is expected. The resulting C_p distribution agrees with that of the experiments almost perfectly for both the panel and the RANS solvers.

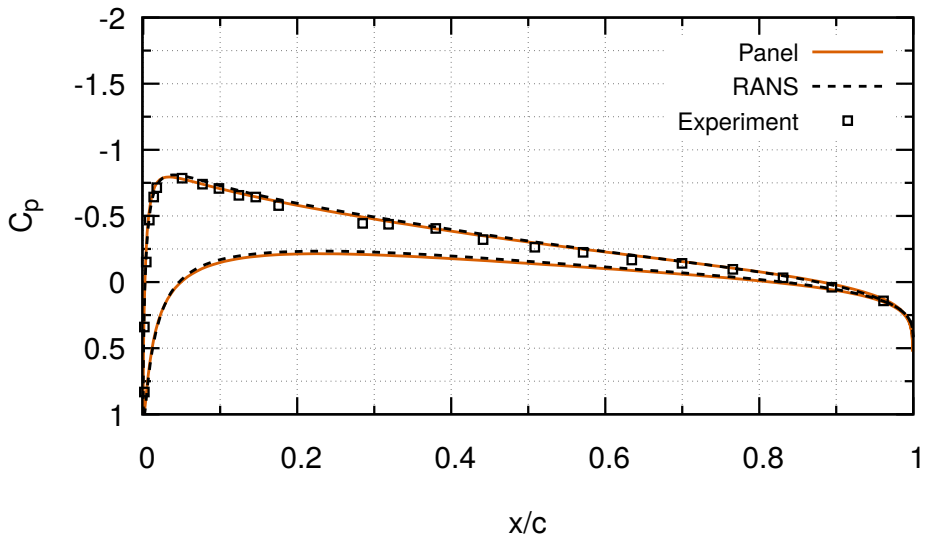


Figure 3.6: Pressure coefficient distributions over NACA0012 airfoil at $2\,\mathrm{deg}$ angle of attack

Fig. 3.6 depicts the Panel code and RANS predictions for NACA0012 airfoil at $2\,\mathrm{deg}$ of angle of attack. Even though the experimental data for lower surface is unavailable for this case, the upper surface predictions are in good agreement for both of the solvers, which predict similar distributions.

For $\alpha = 6\,\mathrm{deg}$, Fig. 3.7 indicates that panel code and RANS predictions are nearly identical with panel code slightly deviating from the minimum C_p at the suction peak. However, this behavior is expected in inviscid flows. Due to the absence of the viscous effects, the suction at the upper surface becomes slightly stronger which result in lower C_p. Both predictions agree with the experiments quite well.

Finally, Fig. 3.8 depicts surface pressure coefficients over NACA0012 airfoil at $10\,\mathrm{deg}$ angle of attack for Panel and RANS solvers. Similar to the previous figure, over-prediction of the suction at the upper surface of the panel code is evident especially between LE and $0.25c$ in the panel code. However, both predictions agree with the experimental data.

To conclude, it is seen that both the panel code and the RANS solver (SU^2) provide

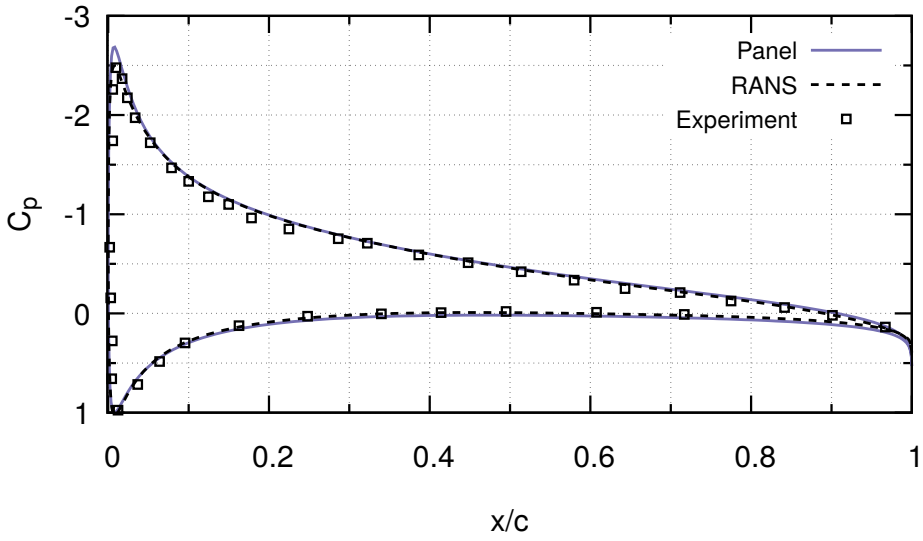


Figure 3.7: Pressure coefficient distributions over NACA0012 airfoil at $6\,\mathrm{deg}$ angle of attack

similar pressure coefficient predictions. Both are able predict the pressure coefficient with sufficient accuracy, and both match the wind tunnel data from low to high angles of attack.

3.1.2.1 Grid Convergence for RANS Solutions

NACA2412 airfoil is used to determine the optimum grid size for RANS solutions. The number of points in both the streamwise points and the normal direction are successively increased until aerodynamic coefficients are determined to be sufficiently converged. $\tanh$-type spacing is employed near leading edge and trailing edge for better resolution of flow features. The change in the aerodynamic coefficients, in this case C_L and C_M, are shown in Fig. 3.9. Note that increasing the normal resolution, J, higher than 150 points does not substantially alter the aerodynamic coefficients. However, as expected, the most dominant effect is the streamwise grid resolution. The aerodynamic coefficients have sufficiently converged at 621 points in the streamwise direction.

The coarsest grid, which has 301 points in streamwise direction and 100 points in the normal direction is shown in Fig. 3.10a. In Fig. 3.10b, an intermediate 361x150 grid

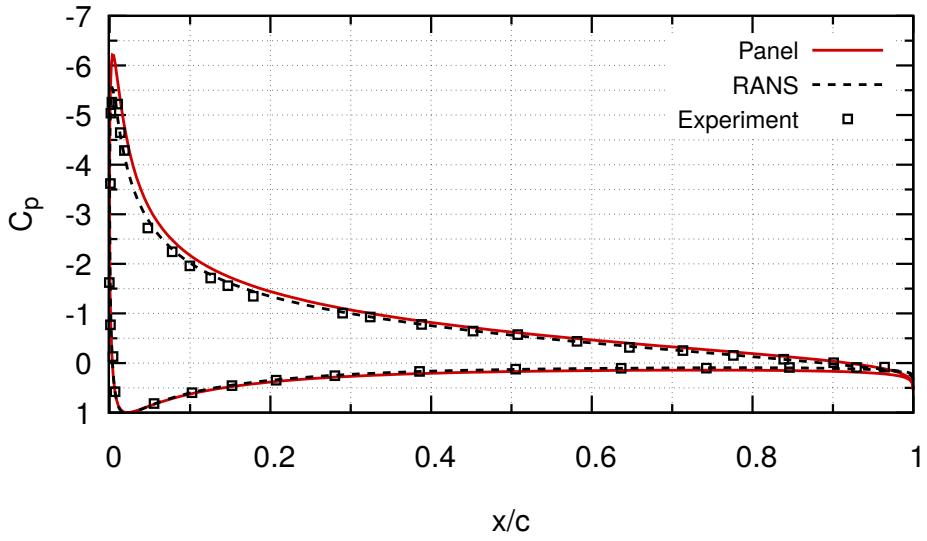


Figure 3.8: Pressure coefficient distributions over NACA0012 airfoil at $10\,\mathrm{deg}$ angle of attack

is shown. Finally, the sufficiently resolved grid, 621x150, is presented in Fig. 3.10c. Finally, a zoomed out view is shown in Fig. 3.10d. It should be noted that the far-field boundary, at its closest, is located at least 30 chord lengths away from the airfoil.

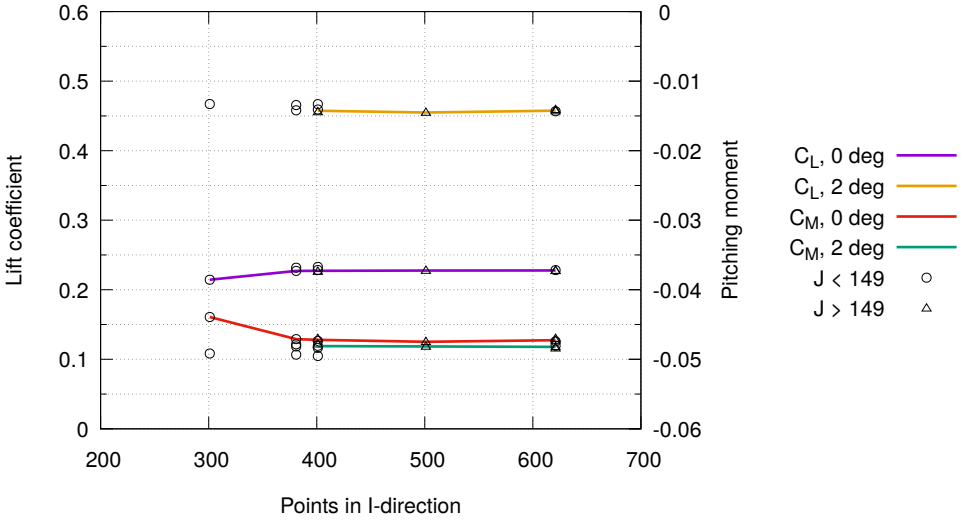
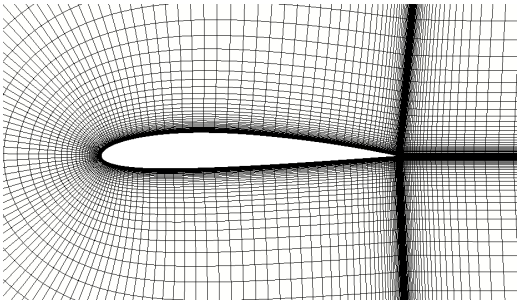
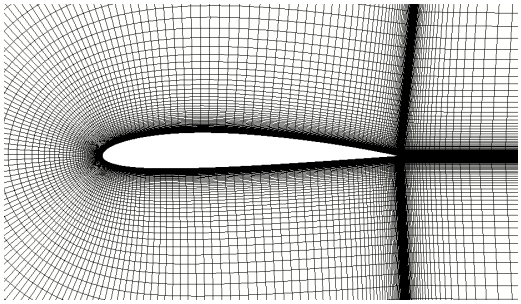


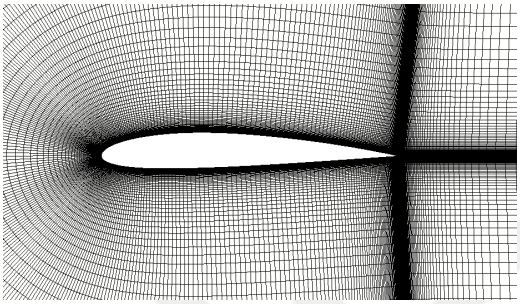
Figure 3.9: C_L and C_M with respect to grid size for NACA2412 airfoil at 0 and $2\,\mathrm{deg}$ of angle of attack



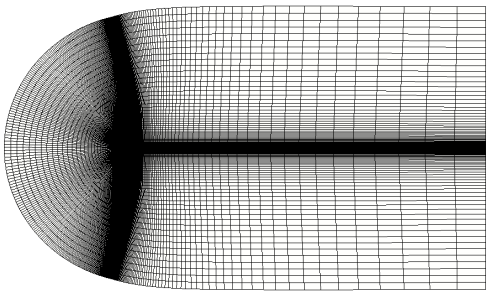
(a) NACA2412 C-Grid with $I = 301$, $J = 100$



(b) NACA2412 C-Grid with $I = 361$, $J = 150$



(c) NACA2412 C-Grid with $I = 621$, $J = 150$



(d) NACA2412 C-Grid, zoomed out

Figure 3.10: C-type grids used in the grid convergence study

3.2 Design Optimizations of Airfoil Profiles

In the optimization studies, the NACA2412 airfoil, which is cambered, is taken as the baseline airfoil. The airfoil upper and lower surfaces are parametrized using 5^{th} order CST resulting in 12 design variables in each case. Four optimization cases are considered with increasing complexity in the objective function. Target lift, pitching moment terms and the adverse pressure gradient terms are successively added case-by-case to the objective function for single-point optimization. The single point optimization studies are carried out at $0\,\mathrm{deg}$ angle of attack. In the fourth and the final case, a multi-point design optimization is presented to extend the optimum characteristics of the designed airfoil to higher angles of attack. In this case, it is aimed to improve the performance of a pre-existing airfoil by increasing the lift and either keeping the pitching moment the same or reducing it over its operational envelope by simultaneously optimizing it at $0\,\mathrm{deg}$ and $4\,\mathrm{deg}$ angle of attack.

3.2.1 Case I: Optimization for Target C_L

For this design optimization case, the lift coefficient of an airfoil is increased at $0\,\mathrm{deg}$ angle of attack, and the objective function is normalized. The objective function to be minimized is given in Eq. 3.5, which is in a normalized form. The target lift coefficient is chosen to be 20% more than that of the baseline airfoil (0.254). The leading edge radius is constrained between 50% and 150% of the baseline airfoil in order to have realistic airfoil profiles. The optimization problem formulation is given as follows in Eq. 3.5,

$$\begin{aligned} \min \quad & I = w_1 \left| 1 - \frac{C_L}{0.305} \right| \\ \text{w.r.t} \quad & \mathbf{w} \\ \text{s.t.} \quad & 1.5 r^*_{LE} \geq r_{LE} \geq 0.5 r^*_{LE} \end{aligned} \tag{3.5}$$

where $\mathbf{w}$ is the vector of CST weights (12 in total) and w_1 is the weight function, which is taken as unity. The constraint imposed on the leading edge radius is computed as given in Eq. 2.43, since the only the first CST weight terms influence the

leading edge radius.

Table 3.3: Aerodynamic coefficients for Case I

Case	C_L		C_M	
	Panel	RANS	Panel	RANS
Baseline	0.254	0.229	-0.0540	-0.0493
Target	0.305	-	-	-
Case I	0.305	0.275	-0.0652	-0.0597

From Fig. 3.11, it is seen that due to having an initially cambered airfoil, not much camber has been added. The airfoil becomes slightly thinner as the lower surface approaches the chord line, with no noticeable change in the camber. The target value for lift is reached in 8 steps (Fig. 3.13) with 28 minor iterations (including trial points), as shown in Fig. 3.12. In the figure, it is clear that while the lift has reached its target value, the value of the pitching moment at the quarter chord has also increased from -0.054 to -0.0652, as shown in Table 3.3. In total, 9 gradient calls have been made with the remainder of the steps being the flowfield solutions. The optimization history is presented in Fig. 3.13. In the final iteration, the objective function value drops below of the threshold value.

Additionally, the aerodynamic coefficients obtained by the SU^2 at 0 degrees angle of attack agree well with that of the potential flow, as shown in Table 3.3. The RANS solutions are obtained using a proper C-grid with $y^+ = 1$ with $\mathrm{M}_\infty = 0.3$ and $\mathrm{Re} = 5 \times 10^6$. Mach number and pressure coefficient contours over the airfoil designed in Case I at $0\,\mathrm{deg}$ angle of attack are given in Figs. 3.15a and 3.15b. No flow separation is observed, as expected. Due to viscous (fully turbulent) solutions, a boundary layer formation over the airfoil is observed. Boundary layer is noted to be relatively thin due to fully attached flow at high Reynolds number.

The optimization outlined in this section is repeated by taking the symmetric NACA0012 profile as the baseline airfoil. The results of the optimization are shown in Fig. 3.14. It is seen that airfoil profiles are not exactly the same yet similar with similar surface pressure distributions. Both airfoils provide the target lift coefficient, but their pitching moments differ slightly ($C_M = -0.0646$ vs $C_M = -0.0652$).

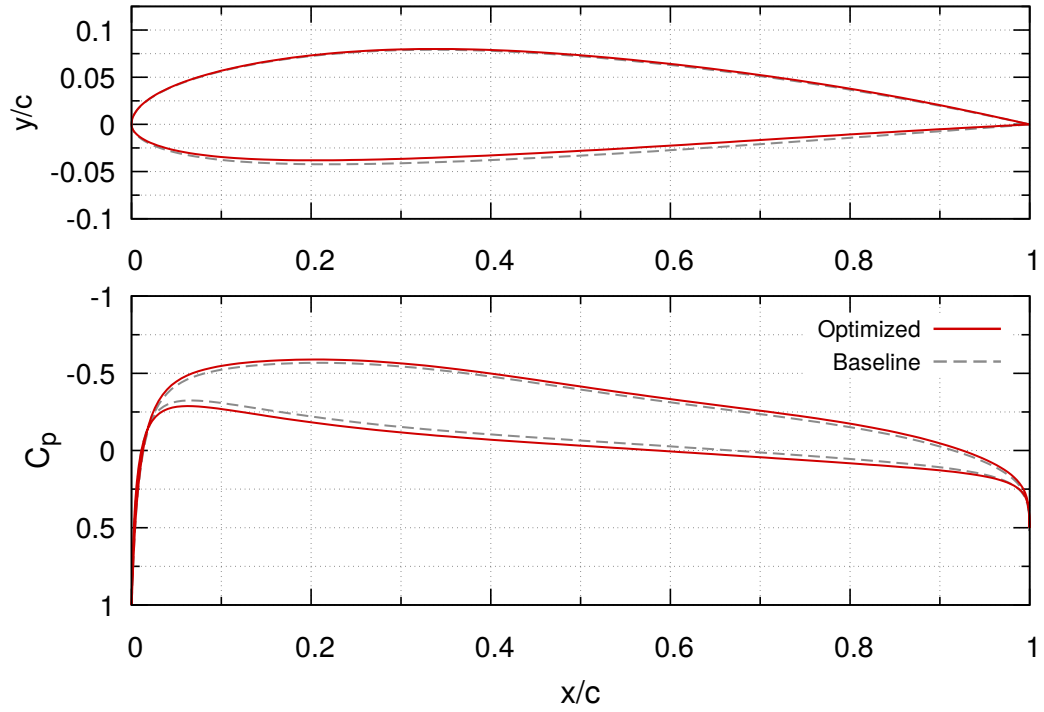


Figure 3.11: Baseline and optimized airfoil profiles for case I

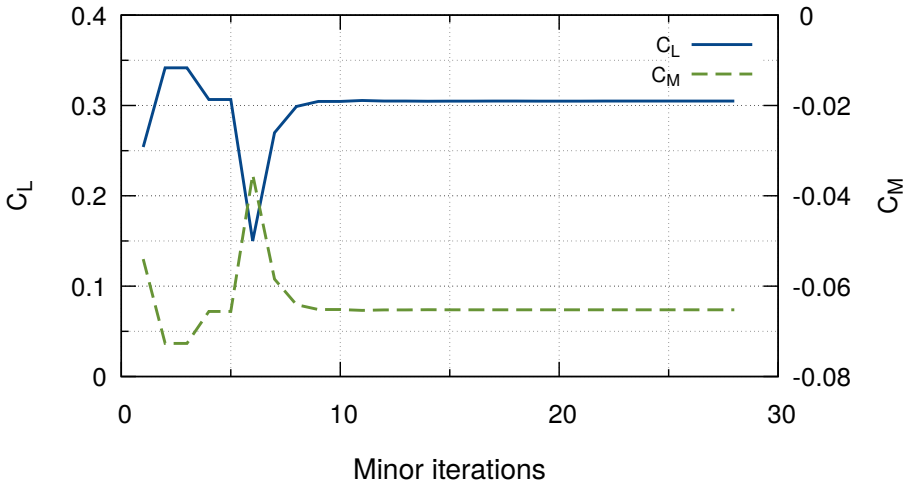


Figure 3.12: Convergence history of the aerodynamic coefficients for case I

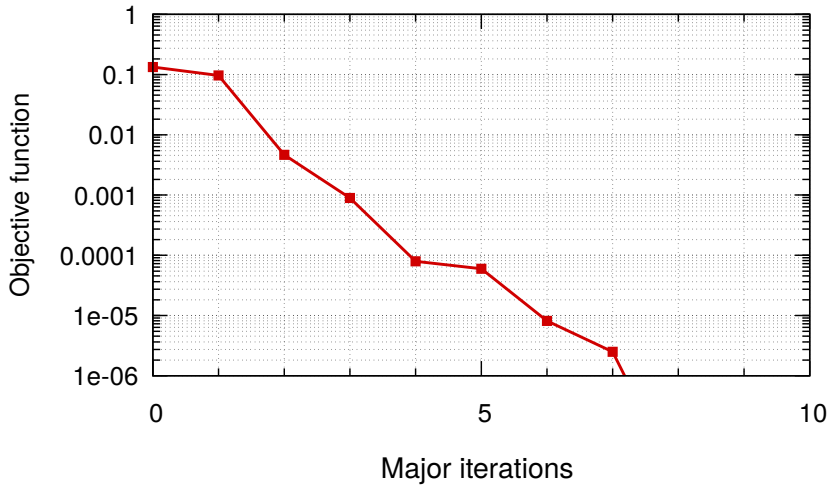


Figure 3.13: The evolution of the objective function with major optimization steps for case I

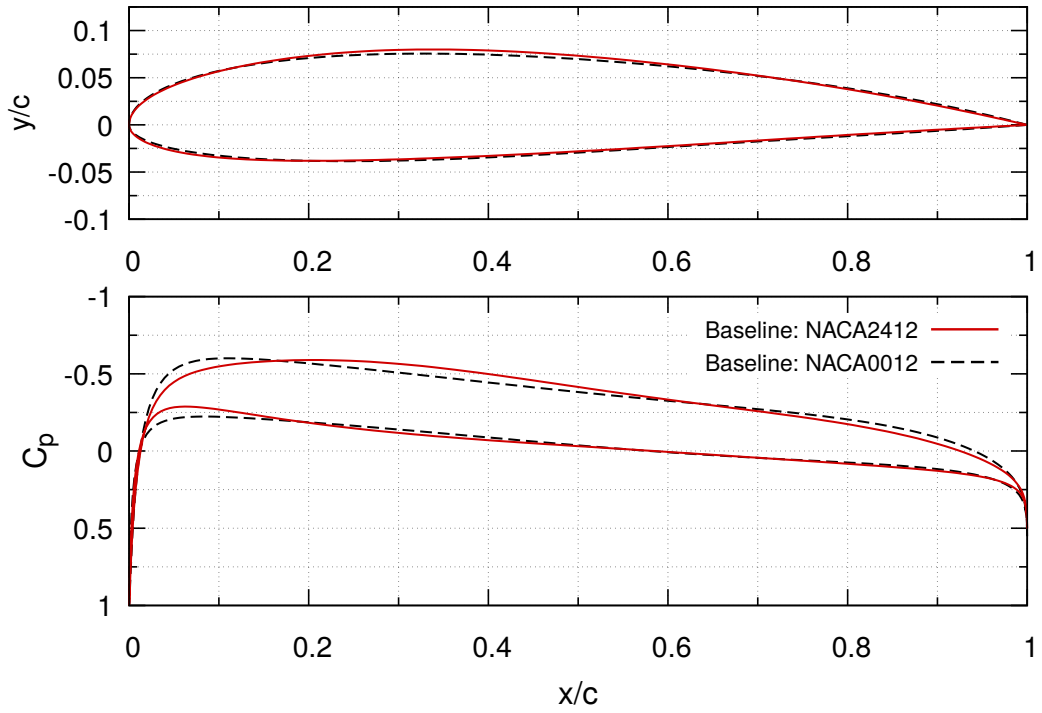
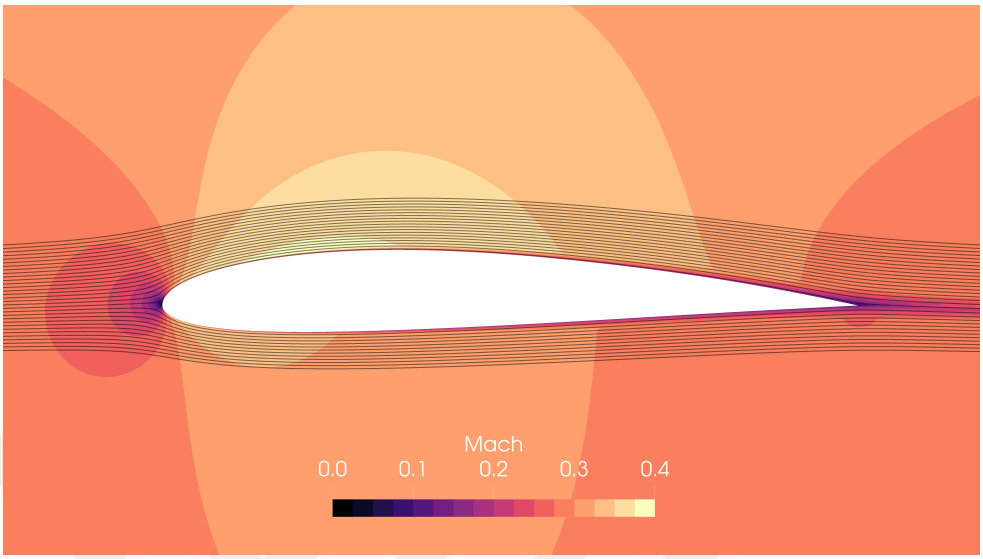
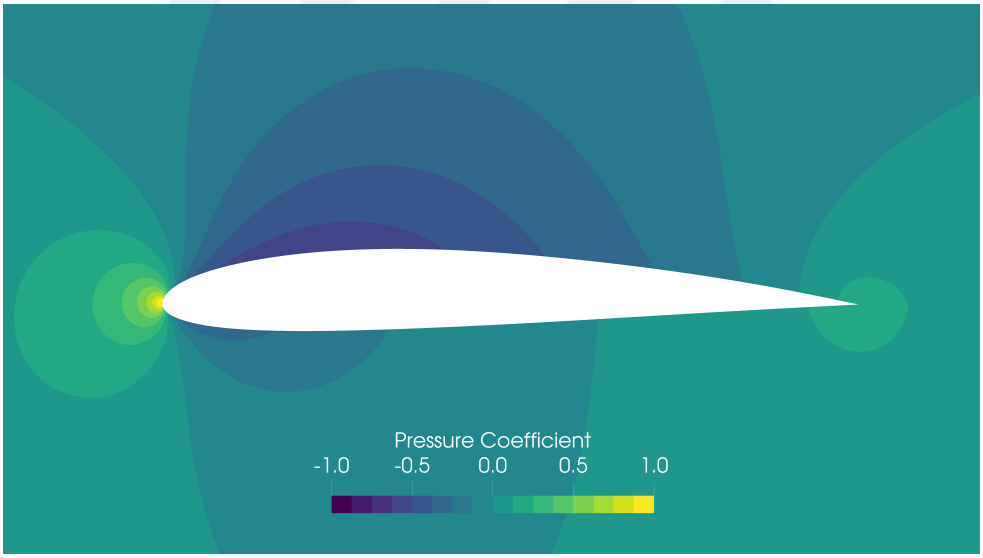


Figure 3.14: Optimized airfoil profiles for case I, different baseline airfoils



(a) Mach number contours for Case I



(b) Pressure coefficient contours for Case I

Figure 3.15: M and C_p contours for airfoil designed in case I, AoA = 0deg

3.2.2 Case II: Optimization for Target C_L and C_M

In this design optimization case, the pitching moment coefficient is added to the normalized objective function definition as given in Eq. 3.6. It should be noted that in the first case, the C_M has increased, which is undesired. Thus, this case aims to increase C_L without changing C_M of the airfoil (-0.054) at $0\,\mathrm{deg}$ angle of attack,. The baseline airfoil is chosen as the optimum airfoil obtained in Case I. The target lift coefficient is unchanged from the previous case, which is equal to 0.305. However, now the pitching moment coefficient target is taken as the initial pitching moment of NACA2412 airfoil, -0.054. The baseline values are also shown in Table 3.4.

$$\begin{aligned} \min \quad & I = w_1 \left| 1 - \frac{C_L}{0.305} \right| + w_2 \left| 1 - \frac{C_M}{-0.054} \right| \\ \text{w.r.t} \quad & \mathbf{w} \\ \text{s.t.} \quad & 1.5 r^*_{LE} \geq r_{LE} \geq 0.5 r^*_{LE} \end{aligned} \tag{3.6}$$

where w_1 and w_2 are the weight terms taken as 0.5 and 0.5 to have an equal bias in the objective function. Similar to the previous case, the constraint imposed on the leading edge radius is computed as given in Eq. 2.43, since the only the first CST weight terms influence the leading edge radius. In the Fig. 3.16, the baseline and the optimized airfoil profiles and their pressure coefficient distributions are shown. Camber addition has been observed, where it is most noticeable between the LE and $0.4c$, but the leading edge radius is kept more or less the same. Compared to the previous case, most of the changes are in the lower surface near leading edge in terms of pressure coefficient. Moreover, the airfoil is now slightly thinner than that of the baseline. In Fig. 3.17, the convergence histories of C_L and C_M are presented. While searching for a feasible direction, the optimizer may sometimes choose the trial point such that the aerodynamic coefficients may seem to be dipping, as similar to the 50^{th} iteration in Fig. 3.17. The optimizer rejects these trial points. The optimization successfully converges to the target values in 16 major optimization steps. In total, 17 gradient calls and 45 flowfield evaluations are required to converge in only 62 minor iterations as depicted in Fig 3.17. The change in the successive objective function evaluations are reduced to below of the threshold in 16 iterations as shown in Fig.

3.18. The target values for lift and moment are comfortably reached, as also shown in Table 3.4.

Additionally, a fully-turbulent RANS solution at 0 deg angle of attack is used to validate the aerodynamic coefficients. Indeed, it is seen that the lift coefficient has increased around 20% and the pitching moment is the equal to that of the NACA2412 (Table 3.4) which verifies that the optimization targets have been reached. The surface pressure coefficient distributions are plotted in Fig. 3.20. In general, the panel code compares favorably with the SU^2 predictions. Overall, a very good agreement is noted.

The optimization study is similarly repeated by taking the symmetric NACA0012 profile as the baseline airfoil. The results of the optimization are shown in Fig. 3.19. The resulting surface pressure distributions are observed to be similar, with comparable camber and thickness distributions that are very much alike. Both airfoil profiles provide the target aerodynamic coefficients.

Table 3.4: Aerodynamic coefficients for Case II

Case	C_L		C_M	
	Panel	RANS	Panel	RANS
Baseline	0.305	0.275	-0.0652	-0.0597
Target	0.305	-	-0.0540	-
Case II	0.305	0.282	-0.0540	-0.0491

Additionally, off-design performance of airfoils designed in Case I and II are assessed with angle of attack sweeps using fully-turbulent solutions obtained via SU^2. The change in the aerodynamic coefficients are given in Fig. 3.21. From the figure, it is obvious that the designed airfoil in Case I and II have more or less the same lift curve slope, which is expected since in both cases the target lift coefficient was 20% increase over the baseline (NACA2412). Same delta in the lift curve has been maintained throughout the operational envelope for both of the airfoils. In terms of pitching moment, it is clear that due to the unconstrained pitching moment in case I, the value at $0\,\mathrm{deg}$ angle of attack is around -0.06 whereas in the constrained case, Case II, it is the same as the baseline airfoil (NACA2412). However, as the angle of

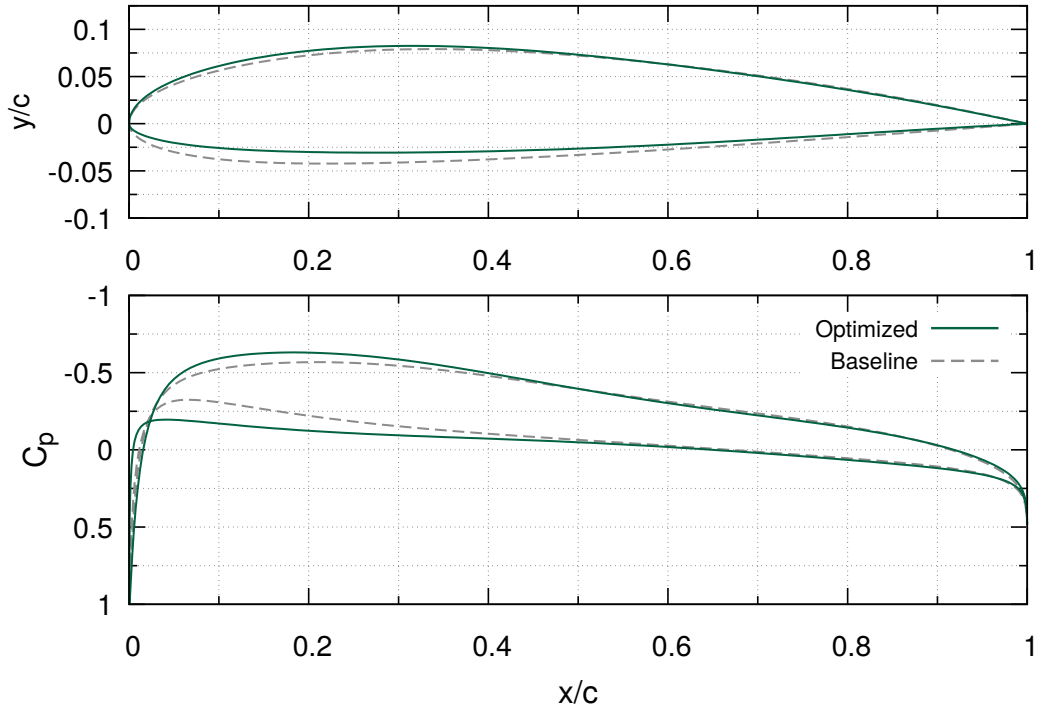


Figure 3.16: Baseline and optimized airfoil profiles for case II

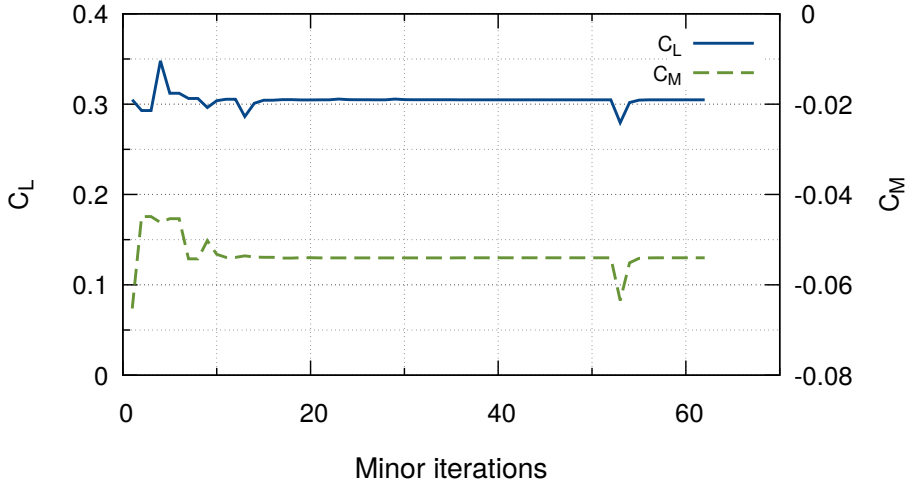


Figure 3.17: Convergence history of the aerodynamic coefficients for case II

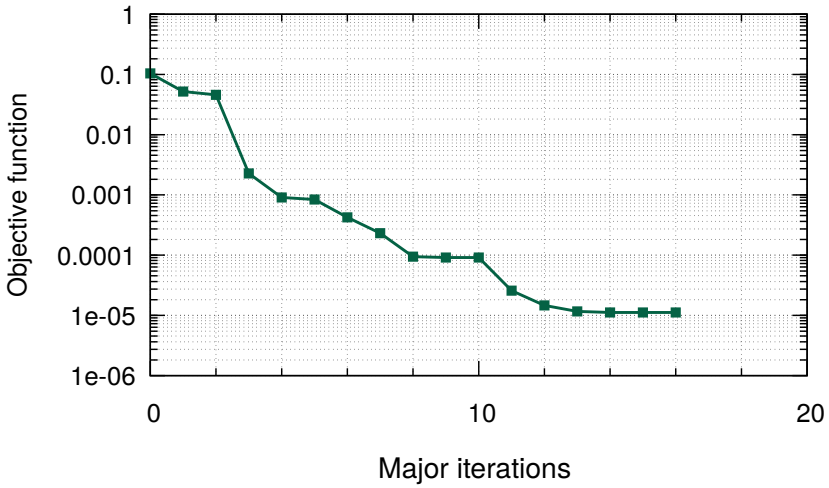


Figure 3.18: The evolution of the objective function with major optimization steps for case II

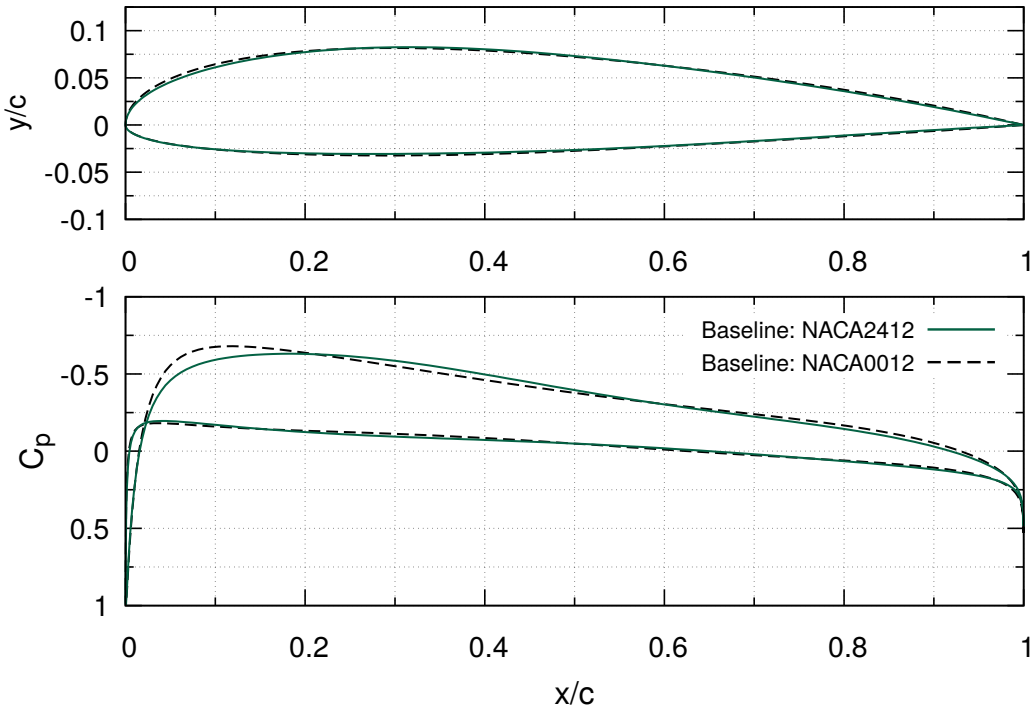


Figure 3.19: Optimized airfoil profiles for case II, different baseline airfoils

attack increases the pitching moment deviates from the baseline airfoil. For drag, the optimized airfoils do not incur massive penalties, with changes being in the order of 5 drag counts.

Finally, M and C_p contours over the airfoil designed in Case II at $0\,\mathrm{deg}$ angle of attack are given in Figs. 3.22a and 3.22b. Clearly the flow does not separate, which is expected. Due to viscous (fully turbulent) solutions, there is a boundary layer formation over the airfoil, which is relatively thin due to fully attached flow at high Reynolds number. The boundary layer thickness is comparable to that of Case I.

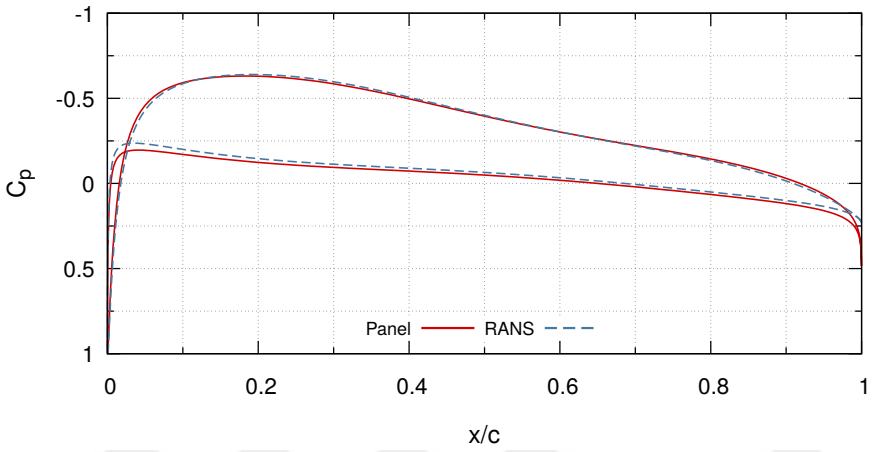


Figure 3.20: Surface pressure comparisons for panel vs. RANS solvers, case II

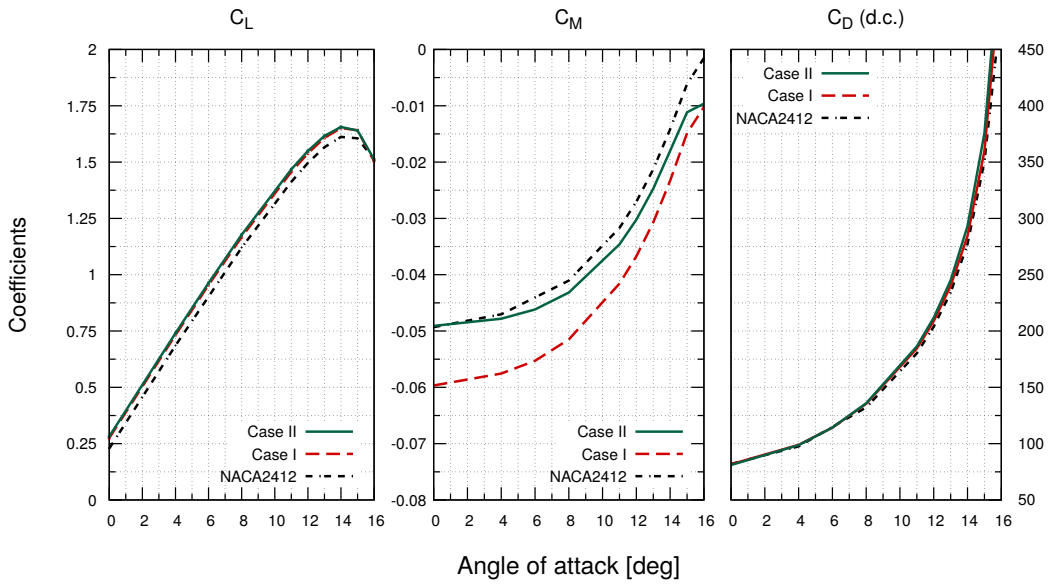
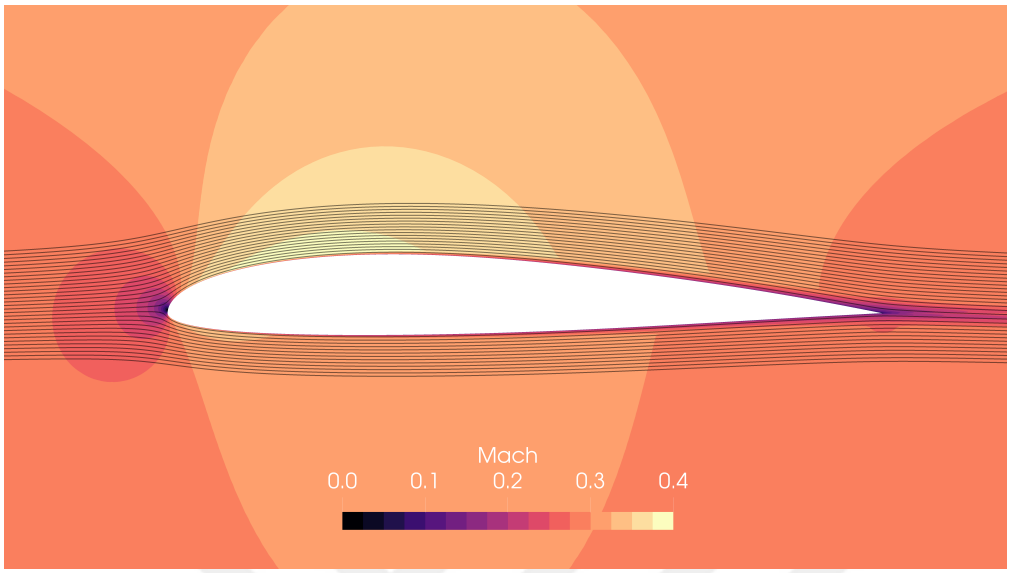
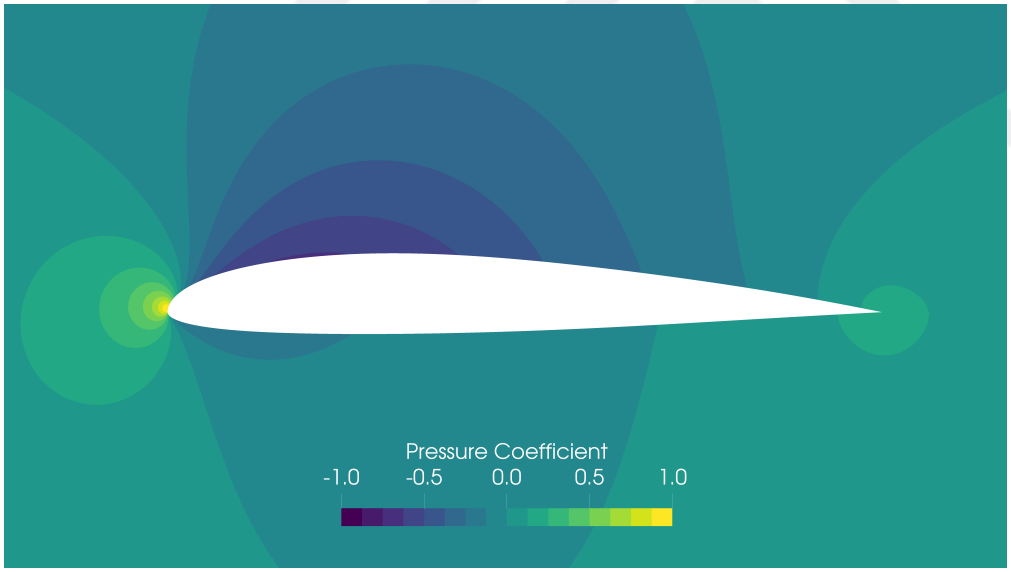


Figure 3.21: Aerodynamic coefficients based on SU^2 solutions over the designed airfoils of cases I and II



(a) Mach number contours for Case II



(b) Pressure coefficient contours for Case II

Figure 3.22: M and C_p contours for airfoil designed in case II, AoA = 0deg

3.2.2.1 Target Pitching Moment Reduced

In this design optimization subcase, the lift coefficient and pitching moment coefficient are used in the normalized objective function definition as shown in Eq. 3.7. The baseline airfoil is chosen to be NACA0012 to demonstrate the robustness of the framework. This case aims to increase lift with nearly zero C_M (-0.005). The target lift coefficient is lower than those of previous cases, which is equal to 0.15. However, now the pitching moment coefficient target is taken as -0.005 which is nearly one tenth of the pitching moment of the NACA2412 airfoil (-0.054). The baseline values are also shown in Table 3.5.

$$\begin{aligned} \min \quad & I = \left|1 - \frac{C_L}{0.15}\right| + \left|1 - \frac{C_M}{-0.005}\right| \\ \text{w.r.t} \quad & \mathbf{w} \\ \text{s.t.} \quad & 1.5r^*_{LE} \geq r_{LE} \geq 0.5r^*_{LE} \end{aligned} \tag{3.7}$$

Similar to the previous cases, the constraint imposed on the leading edge radius is computed as given in Eq. 2.43, since the only the first CST weight terms influence the leading edge radius.

In the Fig. 3.23, baseline and the optimized airfoil profiles are plotted, as well as their pressure coefficient distributions. From the figure, it is observed that camber is added starting from the leading edge throughout the whole chord. Most of the pressure changes are observed in the upstream of the $0.6c$ region. However, due to nearly zero pitching moment target, the airfoil has a slightly reflexed profile. The convergence of aerodynamic coefficients are plotted in Fig. 3.24. The aerodynamic coefficients are converged in 35 minor steps, as shown in the figure. In total, 8 gradient calls have been made and the remainder are flow solutions, which shows the efficiency of the optimization algorithm and the method. Additionally, the relative change in the successive objective function evaluations drops below of the threshold as seen in Fig. 3.25. The optimization converges in 7 major iterations. The resulting aerodynamic coefficients are shown in Table 3.5. The targets are matched comfortably for the airfoil designed using the panel solver. Additionally, fully-turbulent RANS solution over the airfoil using the method outlined in the previous chapters is obtained for

$0\,\mathrm{deg}$ angle of attack. In the end, it is observed that the aerodynamic loads are in very good agreement (Table 3.5). Also, the surface pressure coefficient predictions of potential flow and RANS solvers are plotted in Fig. 3.26. It is seen that the potential flow compares favorably with the RANS solutions, with very good agreement overall.

Finally, the Mach number and pressure coefficient contours are presented in Figs. 3.27a and 3.27b for $0\,\mathrm{deg}$ angle of attack. Clearly the flow does not separate over the designed airfoil, which is expected. Due to fully turbulent solutions, a boundary layer forms over the airfoil, which is relatively thin due to fully attached flow at high Reynolds number.

Thus, a lifting, low pitching moment is designed by using a symmetrical NACA0012 airfoil as the baseline airfoil, and the optimum airfoil is verified with a RANS solution.

Table 3.5: Aerodynamic coefficients for very low pitching moment case

Case	C_L		C_M	
	Panel	RANS	Panel	RANS
Baseline	0.0	0.0	0.0	0.0
Target	0.150	-	-0.005	-
Optimized	0.150	0.149	-0.005	-0.003

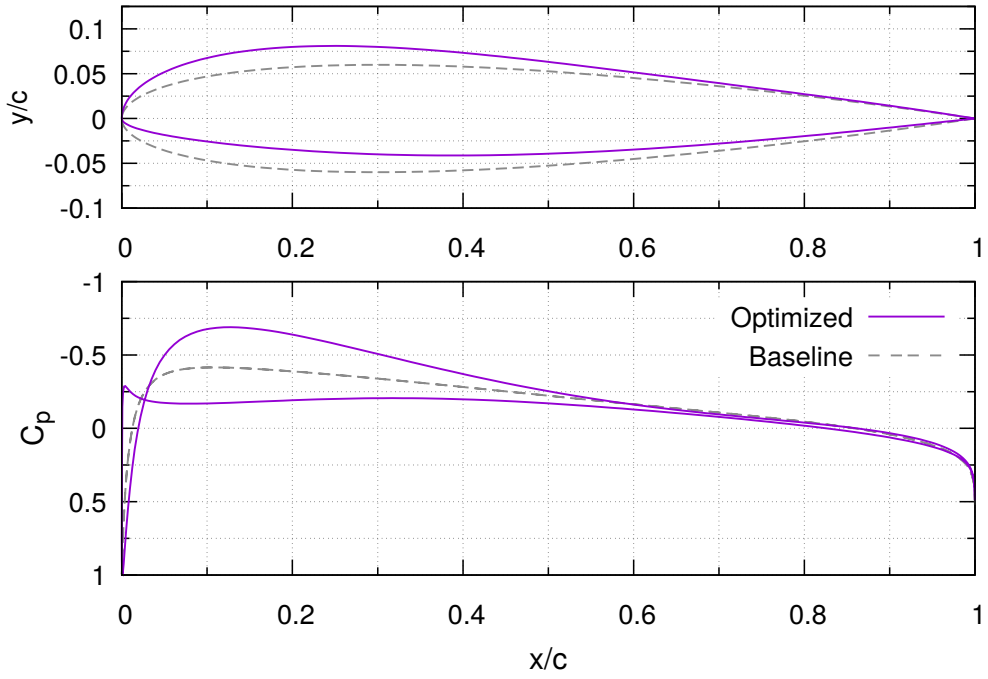


Figure 3.23: Baseline and optimized airfoil profiles for very low pitching moment case

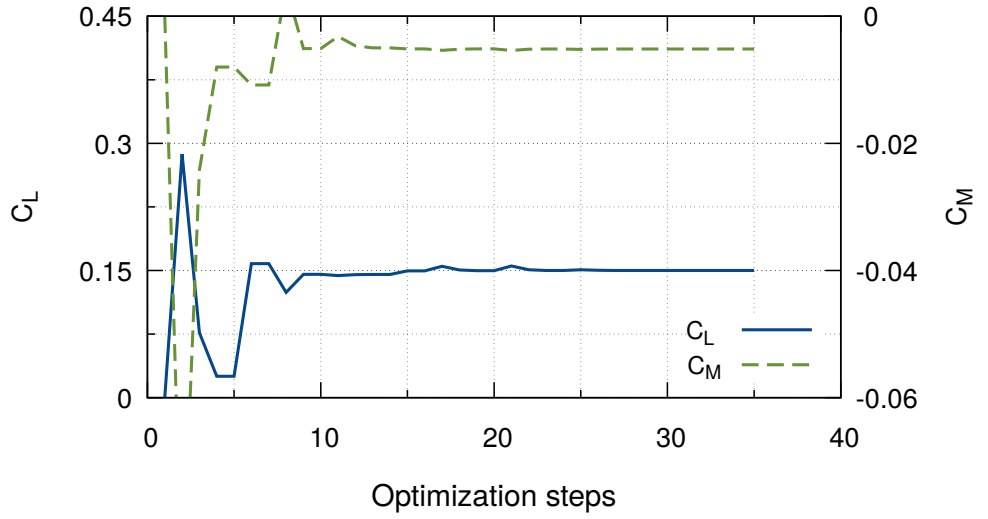


Figure 3.24: Convergence history of the aerodynamic coefficients for very low pitching moment case

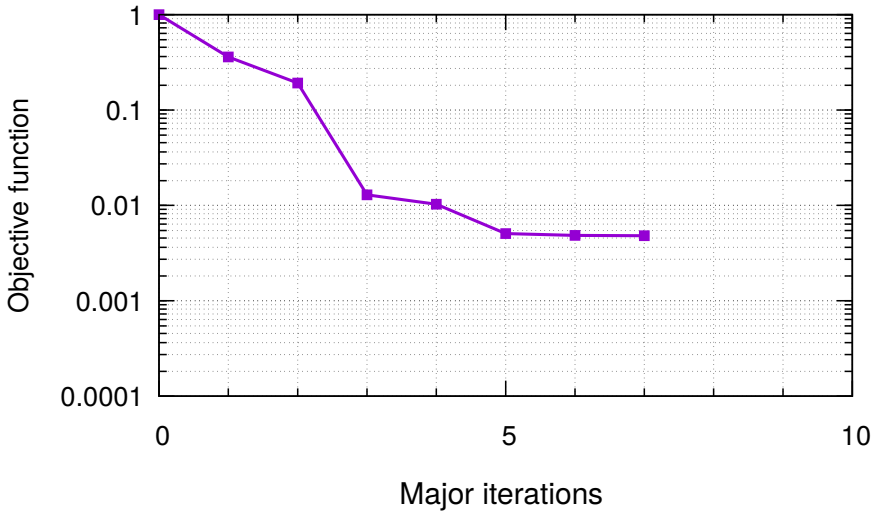


Figure 3.25: The evolution of the objective function with major optimization steps for very low pitching moment case

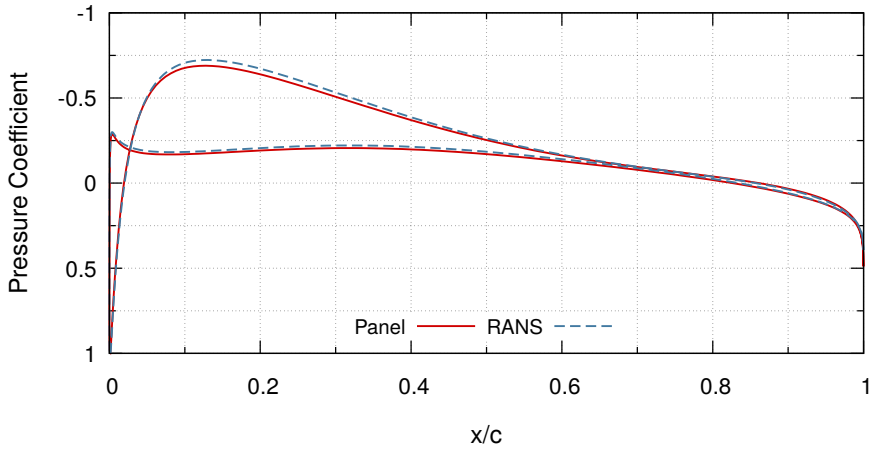
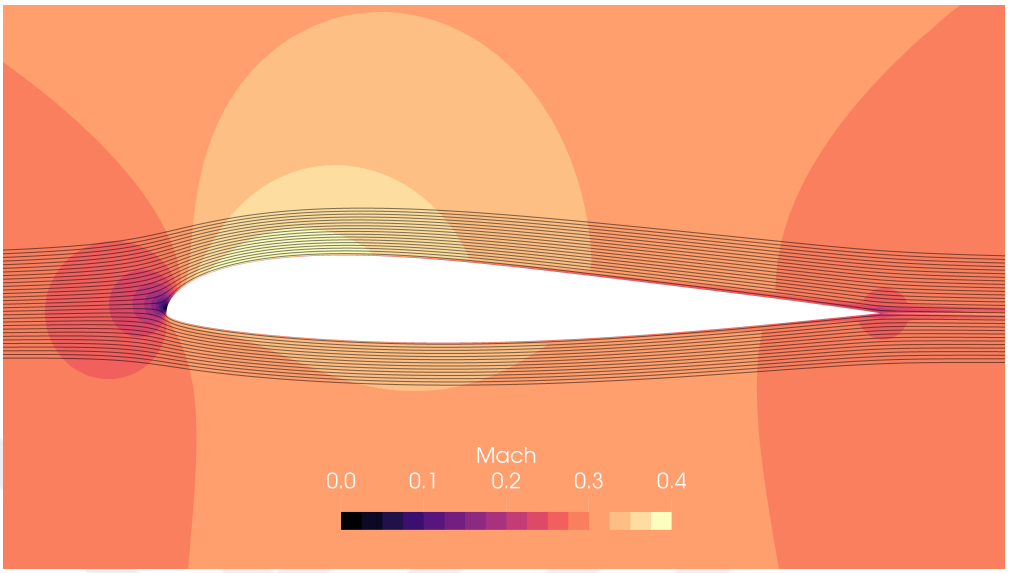
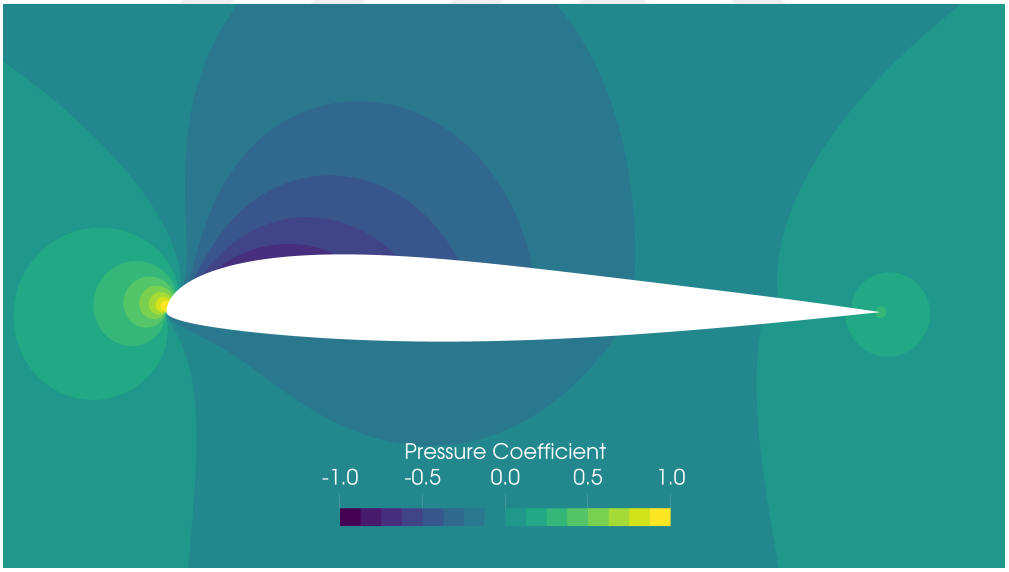


Figure 3.26: Surface pressure comparisons for panel vs. RANS solvers, very low pitching moment case



(a) Mach number contours for very low pitching moment case



(b) Pressure coefficient contours for very low pitching moment case

Figure 3.27: M and C_p contours for airfoil designed in case II, AoA = 0deg

3.2.3 Case III : Optimization for Target C_L and C_M with Reduced dP/dx over the Upper Airfoil Surface

Strong adverse pressure gradients located in the vicinity of the leading edge may lead to massive flow separation at higher angles of attack, especially at the suction side. Thus, reducing the pressure gradient on the upper surface leads to milder stall characteristics. Therefore, in this design optimization case, the adverse pressure gradient term [23, 46] is added to the normalized objective function definition as shown in the Eq. 3.8. The target lift coefficient is unchanged from the previous case, which is equal to 0.305, as well as the target pitching moment coefficient, which is equal to -0.054. Finally, the targeted adverse pressure gradient value is -0.005, which still provides a small suction on the upper surface.

$$\begin{aligned} \min \quad & I = w_1 \left| 1 - \frac{C_L}{0.305} \right| + w_2 \left| 1 - \frac{C_M}{-0.054} \right| + w_3 \left| 1 - \frac{\Delta C_p}{-0.005} \right| \\ \text{w.r.t} \quad & \mathbf{w} \\ \text{s.t.} \quad & 1.5 r^*_{LE} \geq r_{LE} \geq 0.5 r^*_{LE} \end{aligned} \tag{3.8}$$

where ΔC_p is approximated by $C_p(x/c = 0.5) - C_p(x/c = 0.1)$. Weights w_1 to w_3 are chosen such that $w_1 = w_2 = 0.495$ and $w_3 = 0.01$ (sum of the weights are equal to unity). w_3 is chosen as 0.01 to have similar order of magnitude in the objective and the gradient when compared to C_L and C_M. The constraint imposed on the leading edge radius is computed as given in Eq. 2.43, since the first CST weight terms influence the leading edge radius, only.

In the Fig. 3.28, the optimized airfoil and the baseline airfoil are depicted. Much of the changes are indeed focused near leading edge, which is expected in an adverse pressure gradient minimization case. Thinner LE radius is required to maintain the suction for a longer percent of the chord on the upper surface. Even though a modified airfoil profile with significantly thinner LE is obtained, the pressure coefficient distribution indicates a successful optimization, where the point of minimum pressure moves nearly to $0.35c$ (from $\approx 0.15c$ previously). The convergence history of the aerodynamic coefficients are given in Fig. 3.29. The aerodynamic coefficients converge without issues in 75 minor steps where 16 of them are gradient calls and the

remainder are flowfield evaluations. While searching for a feasible direction, the optimizer may sometimes choose the trial point such that the aerodynamic coefficients may seem to be dipping, similar to dips and peaks in Fig. 3.29. The optimizer rejects these trial points. Additionally, the change in ΔC_p with respect to minor iterations are shown in Fig. 3.30. Clearly, ΔC_p is successfully minimized from its initial value to its target value, which is -0.005, when compared to its baseline value. Table 3.6 indicates that both aerodynamic coefficients have been reached, and the ΔC_p term is successfully driven into its target value. Finally, as depicted in Fig. 3.31, the optimization converges successfully in 15 major iterations and the relative change in successive objective function evaluations is lower than the threshold.

Table 3.6: Aerodynamic coefficients for Case III

Case	C_L		C_M		ΔC_p	
	Panel	RANS	Panel	RANS	Panel	RANS
Baseline	0.305	0.282	-0.0540	-0.0491	0.157	0.140
Target	0.305	-	-0.0540	-	-0.005	-
Case III	0.305	0.290	-0.0540	-0.0496	-0.005	-0.028

With SU^2, the high angle of attack behavior is assessed with fully-turbulent RANS solutions, with the method outlined in the preceding sections. The change in the aerodynamic coefficients with respect to the angle of attack are depicted in Fig. 3.32. As expected, much more gradual lift loss is observed with a slight change in stall angle. Unlike the Case II or the NACA2412 airfoil, the pitching moment does not exceed -0.015. Since gradual lift and moment loss are desired stall characteristics (milder), and the optimization performs as expected. For the $0\,\mathrm{deg}$ angle of attack, the RANS solutions are visualized in Figs. 3.34a and 3.34b. Since the point of minimum pressure moves downstream, separation risk is reduced compared to the previous cases. The C_p distributions obtained by the potential flow and the RANS solver are compared in Fig. 3.33 for zero degrees angle of attack. Potential flow C_p predictions compare favorably with the fully-turbulent RANS solutions with similar minimum pressure predictions. As expected, the point of minimum pressure is obtained at roughly the same location for both solvers. The aerodynamic loads (Table 3.6) are also in good agreement. Therefore, the optimum airfoil is verified by the SU^2.

However, since the pitching moment values are higher than that of baseline in the angle of attack range of 0 to 11 degrees (Fig. 3.32), a multi-point design optimization is performed next for better off-design performance.

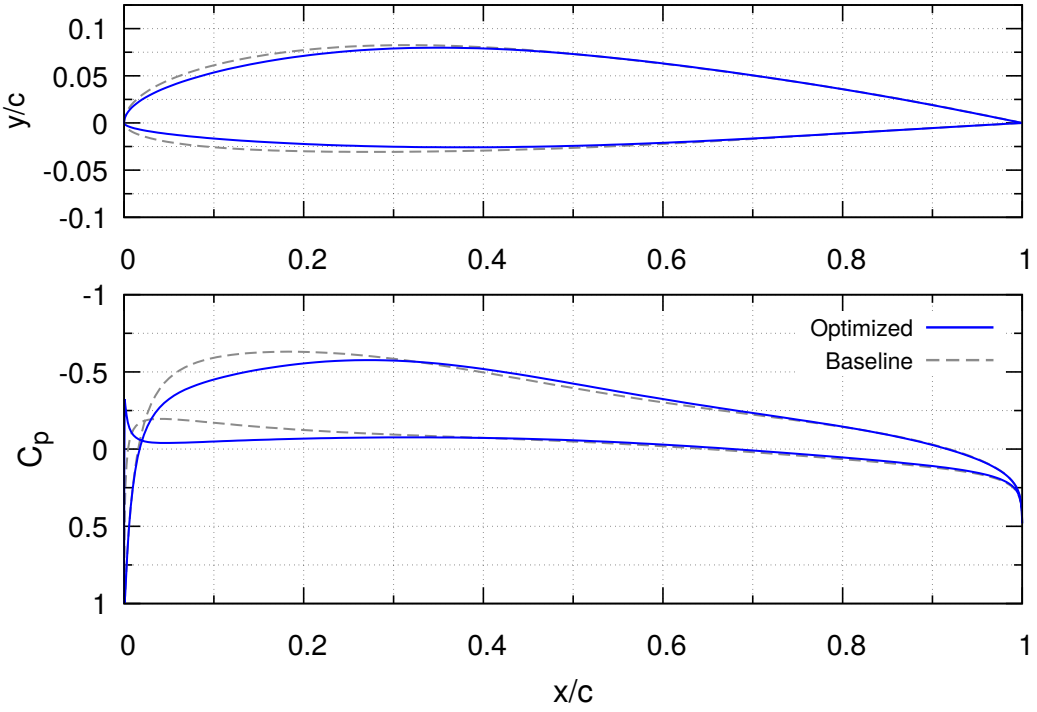


Figure 3.28: Airfoil profiles and pressure distributions for case III

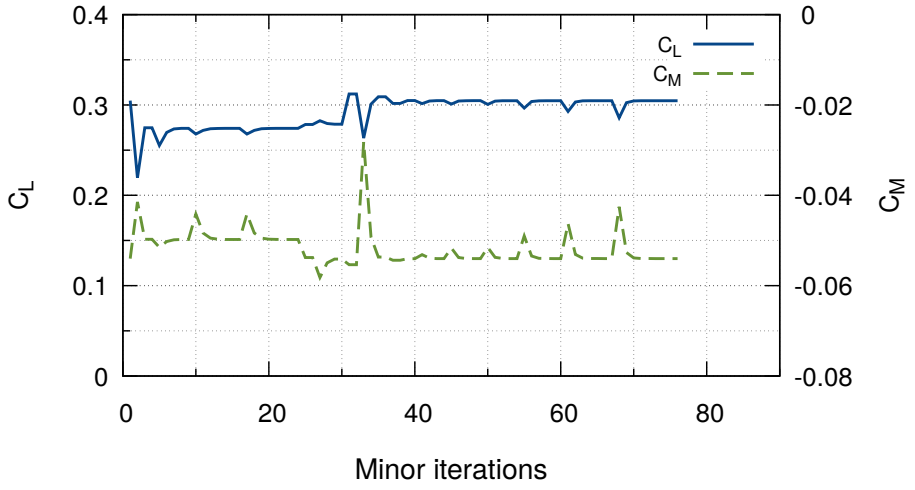


Figure 3.29: Convergence history of case III

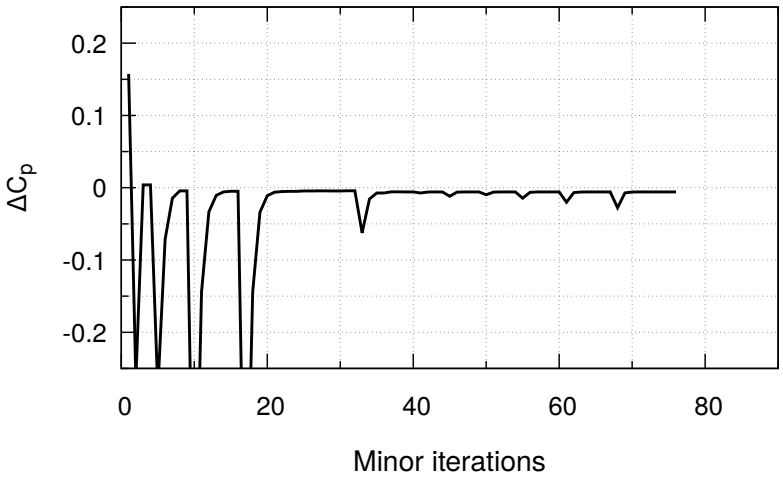


Figure 3.30: Convergence history of ΔC_p in case III

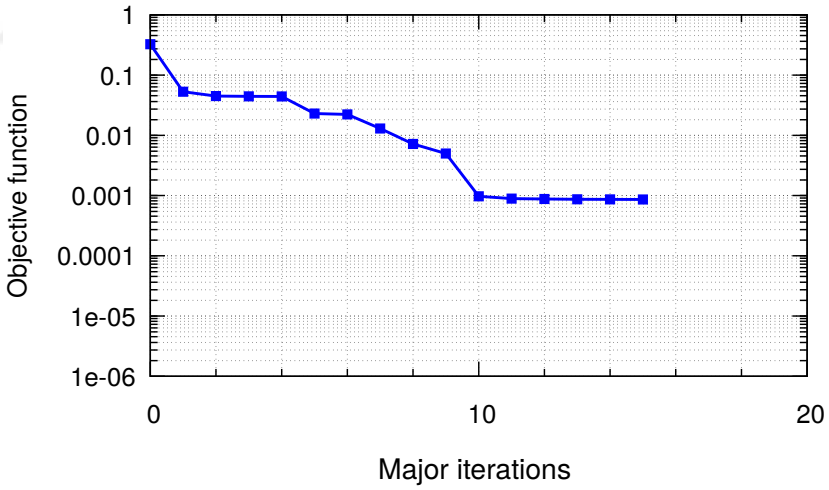


Figure 3.31: The evolution of the objective function with major optimization steps for case III

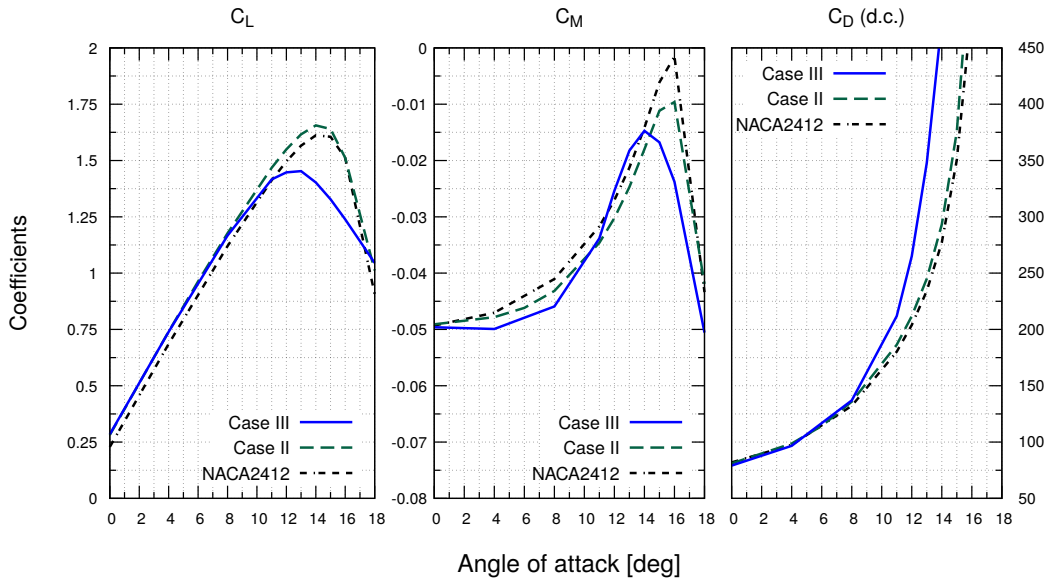


Figure 3.32: Aerodynamic coefficients based on SU^2 solutions over the designed airfoil of case III

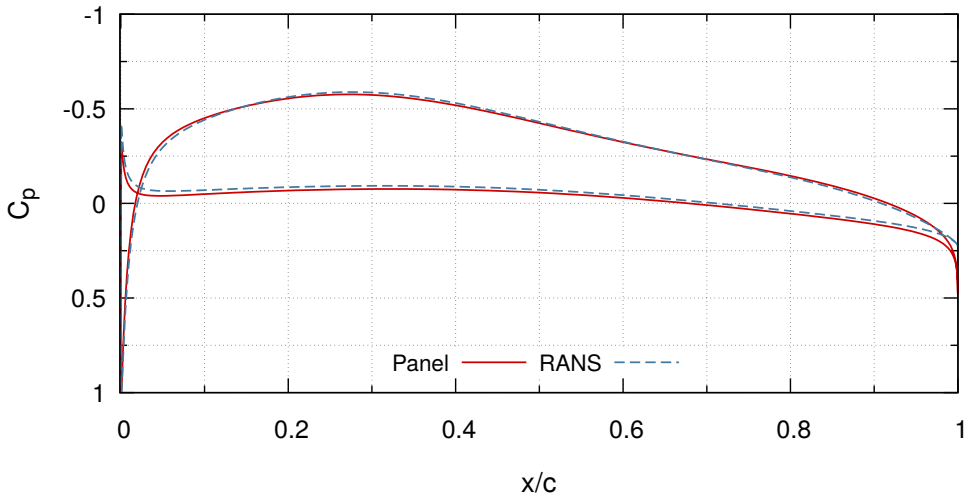
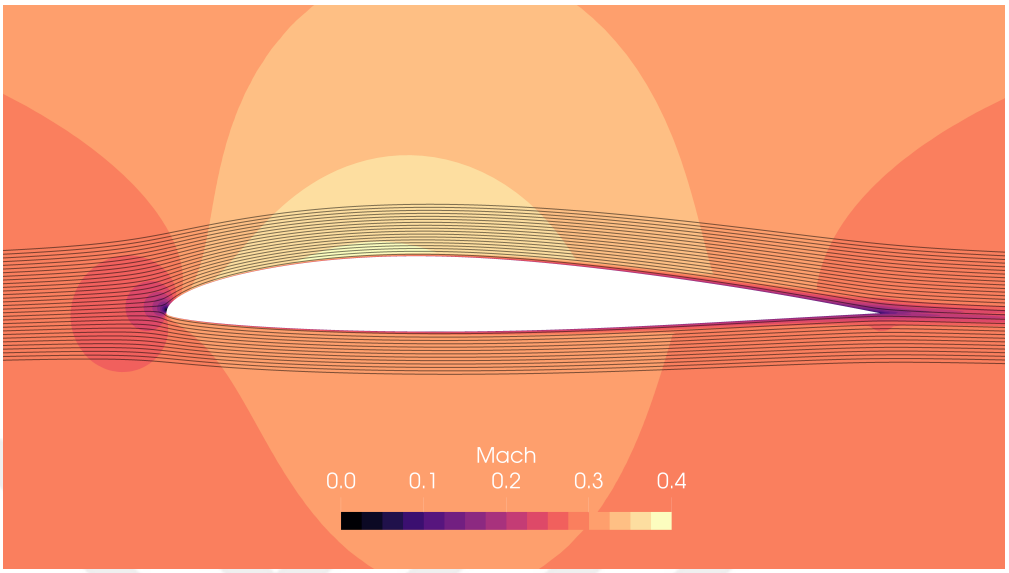
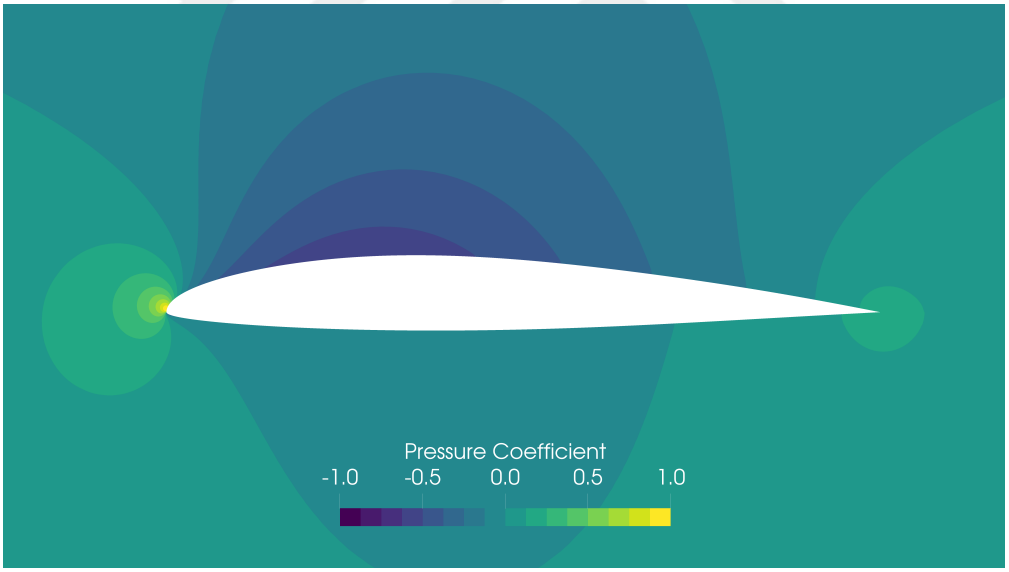


Figure 3.33: Potential flow and RANS surface pressure coefficient distributions, case III



(a) Mach number contours for Case III



(b) Pressure coefficient contours for Case III

Figure 3.34: M and C_p contours for airfoil designed in case III, AoA = 0deg

3.2.4 Case IV: Multi-point Design Optimization

In the previous cases, the optimization studies are done at a single design point, at $0\,\mathrm{deg}$ angle of attack. In this case, the optimization is performed for multiple design points, i.e. multiple angles of attack for better performance throughout the operational envelope.

Thus, in this case, the baseline airfoil is chosen to be the optimized airfoil obtained in Case II. The goal is to increase the performance of this airfoil by simultaneously optimizing at 0 and $4\,\mathrm{deg}$ for target C_L and C_M values. The target is to keep the pitching moment lower than that of NACA2412 over the full range angle of attack. Multi-point optimization flowchart is depicted in Fig. 3.35.

The multi-point optimization problem is formulated in Eq. 3.9.

$$\begin{aligned} \min \quad & I = \sum_{i=1}^{2} \beta_i \left(w_{1,i} \left| 1 - \frac{C_L}{C_{L,\text{target}}} \right|_i + w_{2,i} \left| 1 - \frac{C_M}{C_{M,\text{target}}} \right|_i \right) \\ \text{w.r.t} \quad & \mathbf{w} \\ \text{s.t.} \quad & 1.5 r^*_{LE} \geq r_{LE} \geq 0.5 r^*_{LE} \end{aligned} \tag{3.9}$$

where β_i are the weights for each optimization point and are taken as 0.5, as both optimization points are weighted equally. Similarly, $w_{j,i}$ are the relative weights of C_L and C_M, which are taken as 0.5, to have equal bias in the objective function. Sum of the weights at each design point should add up to one. The target values for the aerodynamic coefficients are given in Table 3.7, where the same increment in lift at $\alpha = 0\,\mathrm{deg}$ is imposed at $\alpha = 4\,\mathrm{deg}$ and the pitching moment is reduced by 5% at $4\,\mathrm{deg}$ while the baseline value at $0\,\mathrm{deg}$ is kept the same since it is equal to that of NACA2412. Finally, the leading edge radius constraint is imposed with the first CST weights for both surfaces of the airfoil as per Eq. 2.43, similar to the previous cases.

To summarize, the lift of the the optimum airfoil obtained in Case II is kept constant at 0 and $4\,\mathrm{deg}$. Additionally, the pitching moment at $0\,\mathrm{deg}$ is unchanged (which is already the same as the NACA2412 airfoil) and a 5% decrease in the pitching moment is targeted at $4\,\mathrm{deg}$ to keep the pitching moment below the NACA2412 airfoil throughout the operational envelope. The target aerodynamic coefficients are given

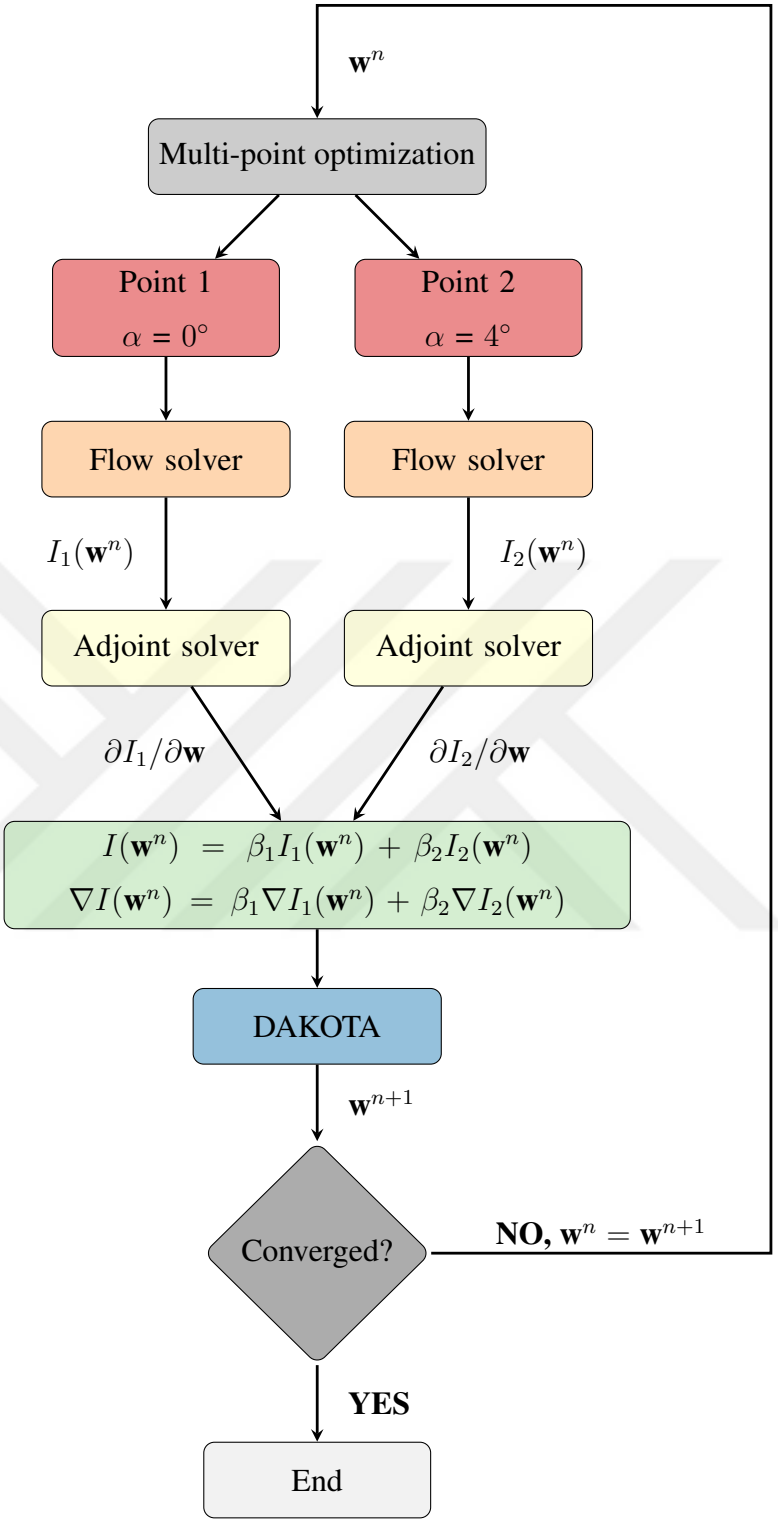


Figure 3.35: Multi-point optimization flowchart

in Table 3.7 for 0 and $4\,\mathrm{deg}$ angle of attack.

The multi-point optimized and the case II (i.e. baseline) airfoils and their pressure coefficient distributions are shown with solid and dashed lines in Fig. 3.36. From the figure, it is noted that the leading edge radius has increased compared to the baseline airfoil. Airfoil thickness increases until $0.25c$ and subsequently decreases until $0.9c$. Additionally, on the upper and the lower surfaces, stronger leading edge suction is obtained to reach the target aerodynamic coefficients where the majority of the lift is generated. Due to reduced thickness, the suction between $0.3c$ and $0.8c$ is slightly reduced.

Convergence history for the multi-point optimization case is presented in Fig. 3.37. From the figure, it is seen that the aerodynamic coefficients converge in 30 optimization steps with 84 minor iterations where 31 of them are gradient calls. The aerodynamic coefficients are also given in Table 3.7, where the pitching moment reduction is evident under constant lift for the multi-point optimized airfoil. The targets are reached comfortably using the potential flow solver. Additionally, the evolution of the objective function with respect to the major optimization steps are depicted in Fig. 3.38. The objective function converges in 30 steps without issues and the relative change is lower than the threshold value in the final step.

Table 3.7: Aerodynamic coefficients in the multi-point design optimization

Case	C_L		C_M	
	$0\,\mathrm{deg}$	$4\,\mathrm{deg}$	$0\,\mathrm{deg}$	$4\,\mathrm{deg}$
Baseline, Panel	0.305	0.783	-0.0540	-0.0588
Baseline, RANS	0.282	0.741	-0.0491	-0.0478
Target	0.305	0.783	-0.0540	-0.0558
Optimized, Panel	0.305	0.783	-0.0540	-0.0558
Optimized, RANS	0.282	0.744	-0.0489	-0.0445

In Fig. 3.39, the aerodynamic characteristics of the multi-point optimized airfoil are assessed with RANS solutions obtained by using SU^2. Aerodynamic load predictions of RANS also agree with that of potential flow as shown in Table 3.7. From the figure, the C_L curves match for the single and multi-point optimized airfoils. Slightly higher

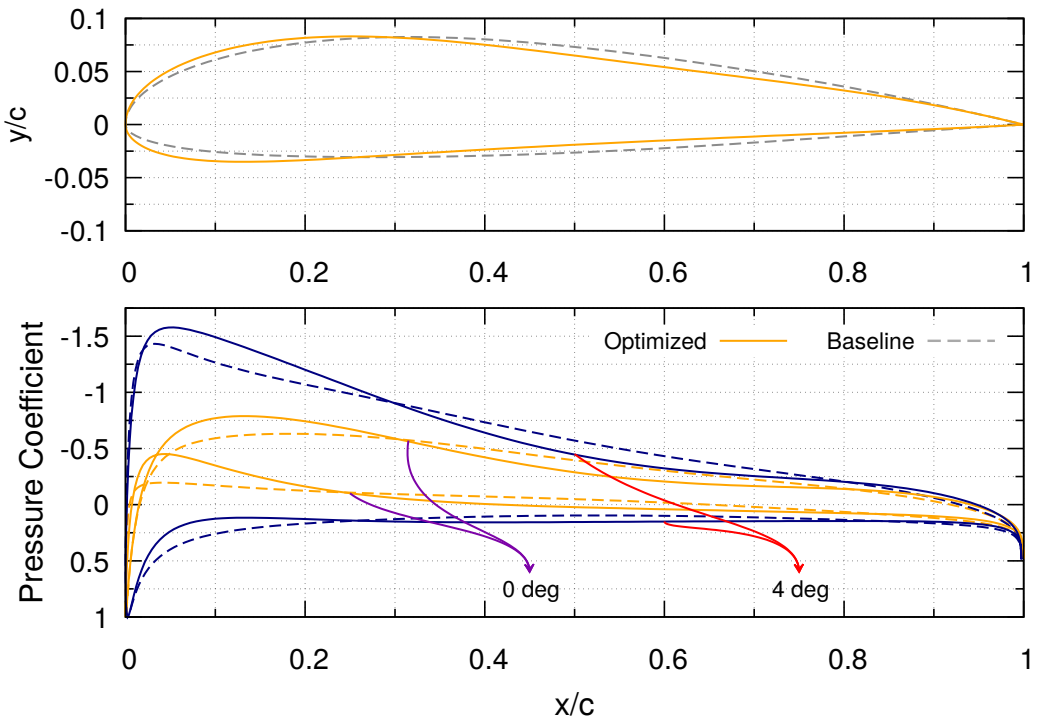


Figure 3.36: Baseline and optimized airfoil profiles for case IV

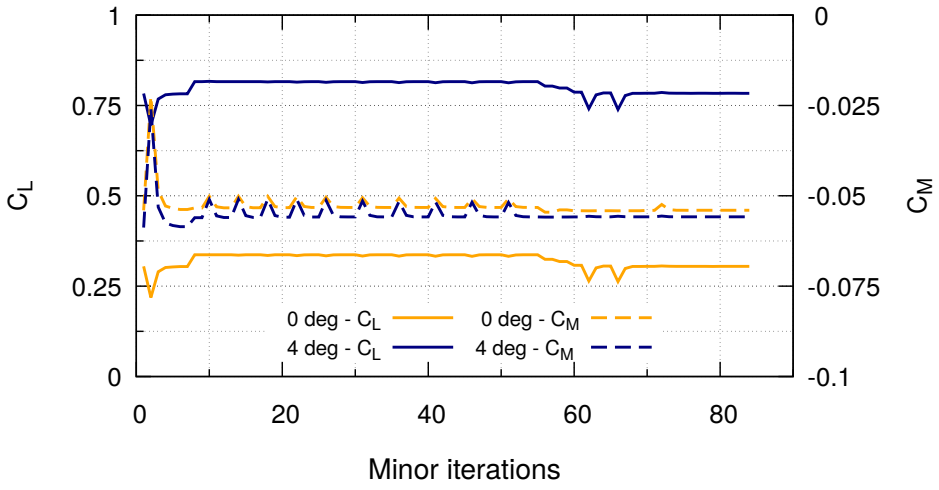


Figure 3.37: Convergence history of the aerodynamic coefficients for case IV

C_{Lmax} is obtained in the multi-point optimized case but no change is observed in the stall angle. The pitching moment behaves as expected, and the pitching moment obtained in case II is successfully reduced in multi-point optimization throughout the operational envelope of the airfoil. Also, the drag change of the multi-point optimization is negligible as the differences are on the order of 1 drag count. Finally, the M and C_p contours obtained with SU^2, are given for the multi-point optimized airfoil at $0\,\mathrm{deg}$ angle of attack in Figs. 3.40a and 3.40b. As expected, no flow separation or abrupt changes are observed over the airfoil.

To conclude, the performance of NACA2412 airfoil is successfully enhanced by increasing the lift coefficient at two design points as well as reducing the pitching moment coefficient to below of NACA2412.

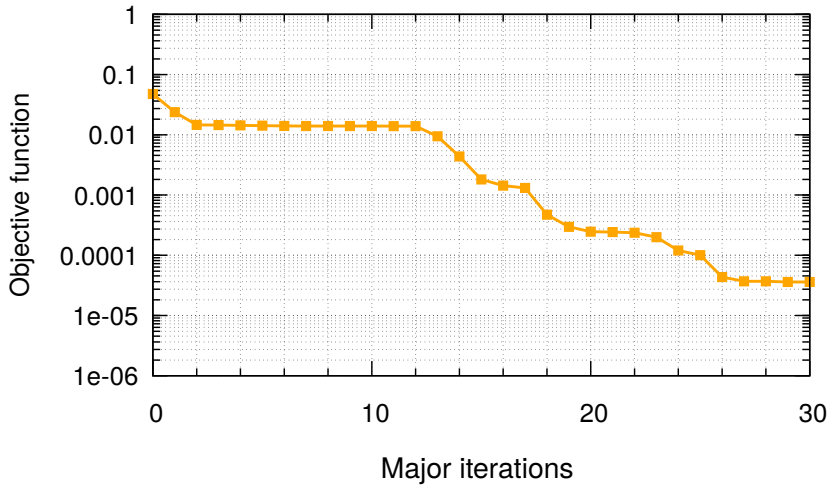


Figure 3.38: The evolution of the objective function with major optimization steps for multi-point design optimization case

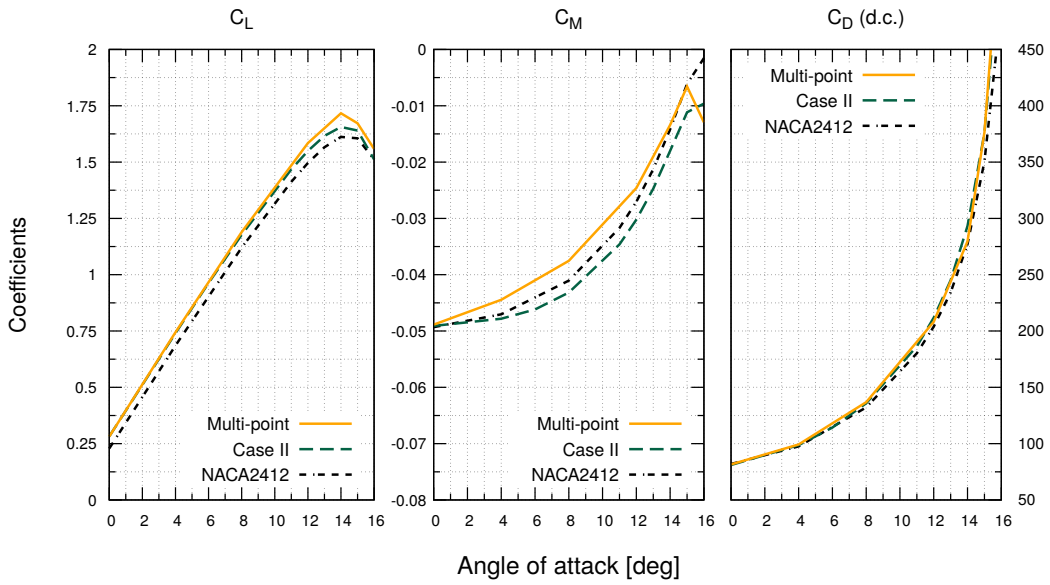
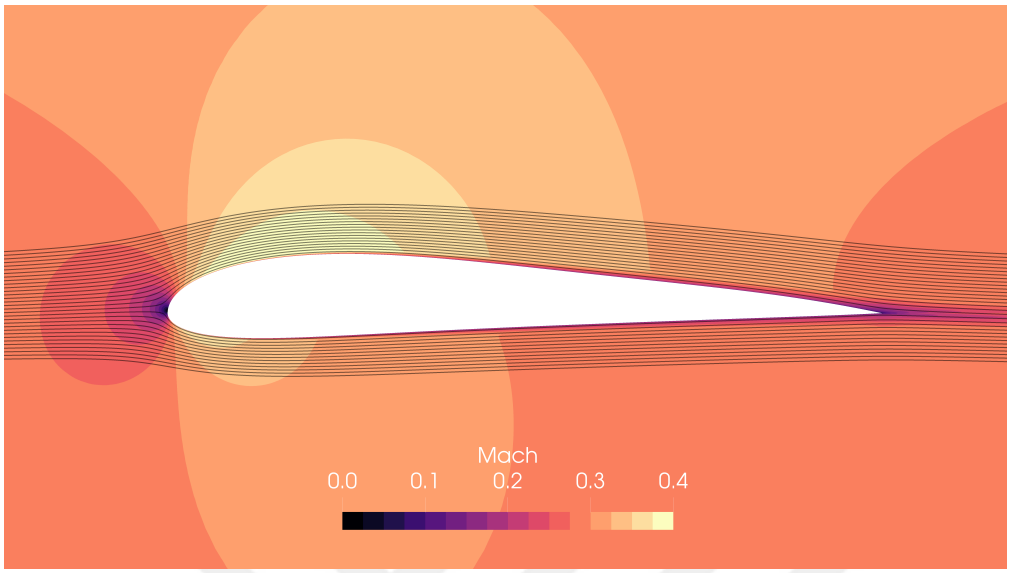
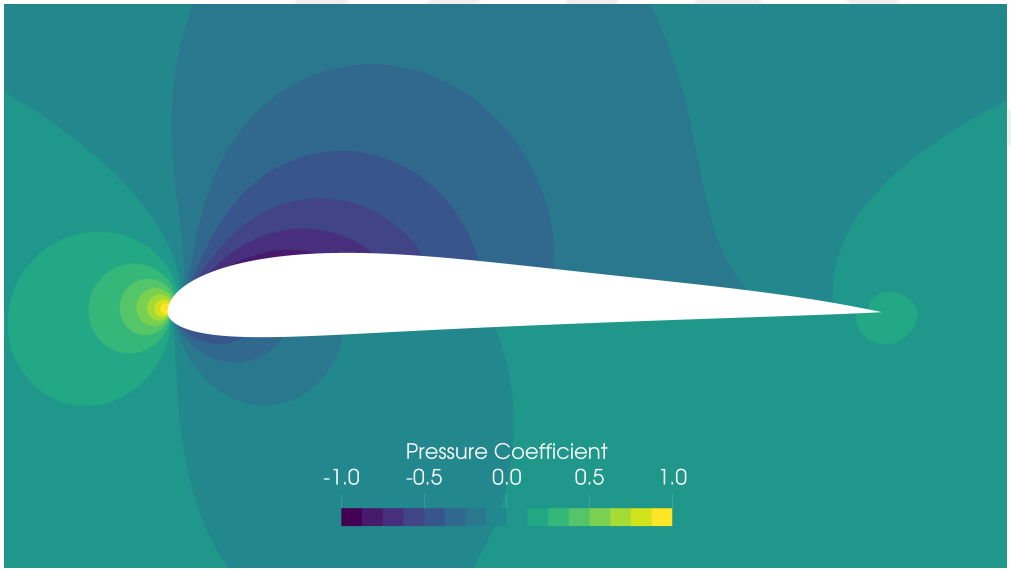


Figure 3.39: Aerodynamic coefficients based on SU^2 solutions over the designed airfoil of case IV



(a) Mach number contours for Case IV



(b) Pressure coefficient contours for Case IV

Figure 3.40: M and C_p contours for multi-point designed airfoil, AoA = 0deg

CHAPTER 4

CONCLUSIONS

A discrete-adjoint based aerodynamic design optimization framework for subsonic airfoils is successfully developed. First, the in-house panel code is discussed. After that, the adjoint equations and automatic differentiation techniques are discussed. Examples are given on forward and reverse mode AD, also some forward and adjoint source codes are also shown. The Class Shape Transformation parametrization technique is discussed, which forms the vector of design variables. Additionally, an open-source optimizer, *DAKOTA*, that makes use of the gradient computed by the adjoint solver is utilized and the quasi-Newton method with BFGS update is introduced.

The automatic differentiation tool, *FDOT*, is successfully used to develop an adjoint solver using an in-house panel code written in Fortran. FDOT toolbox is based on operator-overloading approach, which requires minor changes to the Fortran source code. The differentiated code runs on the reverse mode and the sensitivities of an objective function are efficiently obtained with the computational cost equivalent of one flow solution. Once the sensitivities of the objective function are evaluated with respect to panel nodes, the sensitivities are projected onto the design variables (CST weights).

The validation studies are presented for flow and adjoint solvers. The panel code and an open-source RANS solver, SU^2, are used as the flow solvers. The panel method agrees well with the fully-turbulent solutions for attached flows, and both solvers are able to match the experimental data. The good agreement between the panel code and RANS solver also holds for high angles of attack, where inviscid C_p does not massively differ from the viscous C_p.

The adjoint solver is validated by means of finite differences. It is observed that to obtain the gradient of any objective function, 1 adjoint solution is required which has the equivalent cost of 1 flow solution. If finite differences are to be used 24 flow solutions are required for 2^{nd} order accuracy. The AD based sensitivity derivatives agree very well with FD for lift and pitching moment coefficients over both symmetric and cambered airfoils, respectively. The AD based sensitivity derivatives are accurate down to the machine precision whereas FD based derivatives suffer from numerical truncation error.

A build-up approach in the objective function is used to create airfoils that satisfy the prescribed target lift and moment coefficients. In the first case, a target lift coefficient is prescribed and in the following case its pitching moment is successfully reduced to that of the baseline. Additionally, an adverse pressure gradient term is employed to obtain milder stall characteristics. The angle of attack sweeps carried out using SU^2 indicate that this is indeed the case and gradual loss of lift and pitching moment are observed. The off-design performance is improved by keeping the pitching moment below that of the baseline for all angles of attack by using a multi-point optimization approach, without actually changing the lift obtained in the optimum airfoils.

The developed framework is used to improve performance of NACA2412 using a build-up approach in the objective function definition as mentioned above. Additionally, the method is also used to design a lifting, very low pitching moment airfoil. The robustness of the optimization process is shown by starting from a symmetrical airfoil and converging to very similar designs for the first two cases. Overall, the optimum airfoils are further verified with fully-turbulent RANS solutions as well as the aerodynamic loads. It is seen that the in-house panel code and SU^2 solutions are in very good agreement. The developed aerodynamic shape optimization framework based on a panel code using the discrete-adjoint method is shown to be quite efficient, robust and accurate for subsonic airfoil profiles.

REFERENCES

[1] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther, "High-fidelity aerostructural design optimization of a supersonic business jet," *Journal of Aircraft*, vol. 41, pp. 523–530, May 2004.

[2] W. Bacha and W. Ghaly, "Drag prediction in transitional flow over two-dimensional airfoils," in *44th AIAA Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Jan. 2006.

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

[4] J. E. Green, "Civil Aviation and the Environment – The Next Frontier for the Aerodynamicist," *The Aeronautical Journal (1968)*, vol. 110, no. 1110, p. 469–486, 2006.

[5] O. Chernukhin and D. W. Zingg, "Multimodality and global optimization in aerodynamic design," *AIAA Journal*, vol. 51, pp. 1342–1354, June 2013.

[6] I. Newton, *Philosophiæ Naturalis Principia Mathematica*. Jussu Societatis Regiae ac Typis Josephi Streater; prostat apud plures Bibliopolas Londini, 1687.

[7] A. Betz, "Modification of Wing-Section Shape to Assure a Predetermined Change in Pressure Distribution," Tech. Rep. NACA-TM-767, National Advisory Committee for Aeronautics, March 1935. ID: 19930094650.

[8] M. J. Lighthill, "A new method of two dimensional aerodynamic design," *Rep. Memor. 1111, Aeronautical Research Council*, vol. 2112, 1945.

[9] J. D. Hawk and D. R. Bristow, "Development of MCAERO Wing Design Panel Method with Interactive Graphics Module," Contractor Report 3775, NASA, April 1984.

[10] G. Volpe and R. E. Melnik, "The design of transonic aerofoils by a well-posed inverse method," *International Journal for Numerical Methods in Engineering*, vol. 22, no. 2, pp. 341–361, 1986.

[11] A. Hassan, H. Sobieczky, and A. R. Seebass, "Subsonic airfoils with a given pressure distribution," *AIAA Journal*, vol. 22, pp. 1185–1191, Sept. 1984.

[12] A. HASSAN and H. SOBIECZKY, "Transonic airfoils with a given pressure distribution," in *14th Fluid and Plasma Dynamics Conference*, American Institute of Aeronautics and Astronautics, June 1981.

[13] M. S. Selig and M. D. Maughmer, "Multipoint inverse airfoil design method based on conformal mapping," *AIAA Journal*, vol. 30, pp. 1162–1170, May 1992.

[14] M. S. Selig, "Multipoint inverse design of an infinite cascade of airfoils," *AIAA Journal*, vol. 32, pp. 774–782, Apr. 1994.

[15] J. B. Malone, J. Vadyak, and L. N. Sankar, "Inverse aerodynamic design method for aircraft components," *Journal of Aircraft*, vol. 24, pp. 8–9, Jan. 1987.

[16] A. Jameson and J. Reuther, "Control theory based airfoil design using the Euler equations," in *5th Symposium on Multidisciplinary Analysis and Optimization*, American Institute of Aeronautics and Astronautics, Aug. 1994.

[17] J. Reuther and A. Jameson, "Aerodynamic shape optimization of wing and wing-body configurations using control theory," in *33rd Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Jan. 1995.

[18] A. Jameson, L. Martinelli, and N. Pierce, "Optimum Aerodynamic Design Using the Navier-Stokes Equations," *Theoretical and Computational Fluid Dynamics*, vol. 10, pp. 213–237, Jan. 1998.

[19] F. Morlando, "Adjoint-based sensitivity analysis by panel methods and CAS," *Optimization Letters*, vol. 11, pp. 739–752, May 2016.

[20] Y. Zhang, C. Yan, and H. Chen, "An inverse design method for airfoils based on pressure gradient distribution," *Energies*, vol. 13, p. 3400, July 2020.

[21] Z. Lyu, G. K. W. Kenway, and J. R. R. A. Martins, "Aerodynamic shape optimization investigations of the common research model wing benchmark," *AIAA Journal*, vol. 53, pp. 968–985, Apr. 2015.

[22] R. M. Hicks, E. M. Murman, and G. N. Vanderplaats, "An assessment of airfoil design by numerical optimization," Technical Memorandum X-3092, NASA, July 1974.

[23] R. M. Hicks and G. N. Vanderplaats, "Design of low-speed airfoils by numerical optimization," in *SAE Technical Paper Series*, SAE International, Feb. 1975.

[24] R. M. Hicks and P. A. Henne, "Wing design by numerical optimization," *Journal of Aircraft*, vol. 15, no. 7, pp. 407–412, 1978.

[25] O. Pironneau, "On Optimum Profiles in Stokes flow," *Journal of Fluid Mechanics*, vol. 59, no. 1, p. 117–128, 1973.

[26] O. Pironneau, "On Optimum Design in Fluid Mechanics," *Journal of Fluid Mechanics*, vol. 64, no. 1, p. 97–110, 1974.

[27] A. Jameson, "Aerodynamic design via control theory," *Journal of Scientific Computing*, vol. 3, pp. 233–260, Sept. 1988.

[28] O. Baysal and M. E. Eleshaky, "Aerodynamic design optimization using sensitivity analysis and computational fluid dynamics," *AIAA Journal*, vol. 30, pp. 718–725, Mar. 1992.

[29] S. Kim, J. J. Alonso, and A. Jameson, "Multi-element high-lift configuration design optimization using viscous continuous adjoint method," *Journal of Aircraft*, vol. 41, pp. 1082–1097, Sept. 2004.

[30] W. Anderson and V. Venkatakrishnan, "Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation," *Computers & Fluids*, vol. 28, no. 4, pp. 443–480, 1999.

[31] W. K. Anderson and D. L. Bonhaus, "Airfoil design on unstructured grids for turbulent flows," *AIAA Journal*, vol. 37, pp. 185–191, Feb. 1999.

[32] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, pp. 287–404, May 2019.

[33] N. Marco, S. Lanteri, J.-A. Désidéri, B. Mantel, and J. Périaux, "Parallel genetic algorithms applied to optimum shape design in aeronautics," in *Euro-Par'97 Parallel Processing*, pp. 856–863, Springer Berlin Heidelberg, 1997.

[34] Y. He and R. K. Agarwal, "Shape Optimization of NREL S809 Airfoil for Wind Turbine Blades Using a Multi-Objective Genetic Algorithm," in *32nd AIAA Applied Aerodynamics Conference*, American Institute of Aeronautics and Astronautics, June 2014.

[35] Y. Yu, Z. Lyu, Z. Xu, and J. R. Martins, "On the Influence of Optimization Algorithm and Initial Design on Wing Aerodynamic Shape Optimization," *Aerospace Science and Technology*, vol. 75, pp. 183–199, Apr. 2018.

[36] D. W. Zingg, M. Nemec, and T. H. Pulliam, "A Comparative Evaluation of Genetic and Gradient-Based Algorithms Applied to Aerodynamic Optimization," *European Journal of Computational Mechanics*, vol. 17, pp. 103–126, Jan. 2008.

[37] T. Pulliam, M. Nemec, T. Holst, and D. Zingg, "Comparison of evolutionary (genetic) algorithm and adjoint methods for multi-objective viscous airfoil optimizations," in *41st Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Jan. 2003.

[38] T. Holst and T. Pulliam, "Aerodynamic shape optimization using a real-number-encoded genetic algorithm," in *19th AIAA Applied Aerodynamics Conference*, American Institute of Aeronautics and Astronautics, June 2001.

[39] J. R. R. A. Martins, "Perspectives on Aerodynamic Design Optimization," in *AIAA Scitech 2020 Forum*, American Institute of Aeronautics and Astronautics, Jan. 2020.

[40] E. N. Tinocco, *CFD Applications to Complex Configurations: A Survey*, ch. Chapter 15, pp. 559–615.

[41] S. A. Ragab, "Shape Optimization of Surface Ships in Potential Flow Using an Adjoint Formulation," *AIAA Journal*, vol. 42, pp. 296–304, Feb. 2004.

[42] G. Kennedy and J. R. R. A. Martins, "A comparison of metallic and composite aircraft wings using aerostructural design optimization," in *12th AIAA*

Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, American Institute of Aeronautics and Astronautics, Sept. 2012.

[43] G. Kennedy and J. R. R. A. Martins, "An adjoint-based derivative evaluation method for time-dependent aeroelastic optimization of flexible aircraft," in *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, American Institute of Aeronautics and Astronautics, Apr. 2013.

[44] G. Kennedy, G. Kenway, and J. R. R. A. Martins, "Towards gradient-based design optimization of flexible transport aircraft with flutter constraints," in *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, June 2014.

[45] C. Conlan-Smith, N. Ramos-García, O. Sigmund, and C. S. Andreasen, "Aerodynamic shape optimization of aircraft wings using panel methods," *AIAA Journal*, vol. 58, pp. 3765–3776, Sept. 2020.

[46] B. Sarikaya and I. H. Tuncer, "Adjoint Based Design Optimization of Subsonic Airfoils using a Panel Code Together with a RANS Solver," in *AIAA SciTech 2022 Forum, San Diego, CA*, American Institute of Aeronautics and Astronautics, Jan. 2022.

[47] N. Gauger and J. Brezillon, "The continuous adjoint approach in aerodynamic shape optimization," in *MEGAFLOW - Numerical Flow Simulation for Aircraft Design*, pp. 181–193, Springer Berlin Heidelberg, 2005.

[48] A. Jameson, "Efficient Aerodynamic Shape Optimization," in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, Aug. 2004.

[49] H. Kaya, *Development of a discrete adjoint-based aerodynamic shape optimization tool for natural laminar flows*. PhD thesis, Middle East Technical University, 2020.

[50] U. Naumann, *The Art of Differentiating Computer Programs*. Society for Industrial and Applied Mathematics, Jan. 2011.

[51] S. Djeddi, *Towards Adaptive and Grid-Transparent Adjoint-Based Design Optimization Frameworks*. PhD thesis, University of Tennessee, 2018.

[52] R. Djeddi, C. D. Floyd, J. G. Coder, and K. Ekici, "Adjoint-based uncertainty quantification and calibration of RANS-based transition modeling," in *AIAA AVIATION 2021 FORUM*, American Institute of Aeronautics and Astronautics, July 2021.

[53] B. M. Adams, W. J. Bohnhoff, K. R. Dalbey, M. S. Ebeida, J. P. Eddy, M. S. Eldred, G. Geraci, R. W. Hooper, P. D. Hough, K. T. Hu, J. D. Jakeman, M. Khalil, K. A. Maupin, J. A. Monschke, E. M. Ridgway, A. A. Rushdi, J. A. Stephens, L. P. Swiler, D. M. Vigil, T. M. Wildey, and J. G. Winokur, "Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.11 User's Manual," Tech. Rep. SAND2014-4633, Sandia National Laboratories, July 2014. updated November 2019.

[54] J. C. Meza, R. A. Oliva, P. D. Hough, and P. J. Williams, "Opt++: An object-oriented toolkit for nonlinear optimization," *ACM Trans. Math. Softw.*, vol. 33, p. 12–es, jun 2007.

[55] C. G. BROYDEN, "The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations," *IMA Journal of Applied Mathematics*, vol. 6, pp. 76–90, 03 1970.

[56] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Mathematics of Computation*, vol. 24, no. 109, pp. 23–26, 1970.

[57] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, pp. 317–322, 01 1970.

[58] D. F. Shanno, "Conditioning of quasi-newton methods for function minimization," *Mathematics of Computation*, vol. 24, no. 111, pp. 647–656, 1970.

[59] M. P. Rumpfkeil and D. J. Mavriplis, "Efficient hessian calculations using automatic differentiation and the adjoint method with applications," *AIAA Journal*, vol. 48, pp. 2406–2417, Oct. 2010.

[60] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 2e ed., 2006.

[61] D. I. Jones and J. W. Finch, "Comparison of optimization algorithms," *International Journal of Control*, vol. 40, pp. 747–761, Oct. 1984.

[62] F. Gallard, M. Meaux, M. Montagnac, and B. Mohammadi, *Aerodynamic aircraft design for mission performance by multipoint optimization*.

[63] R. Djeddi and K. Ekici, "Fdot: A fast, memory-efficient and automated approach for discrete adjoint sensitivity analysis using the operator overloading technique," *Aerospace Science and Technology*, vol. 91, p. 159–174, 2019.

[64] R. Djeddi and K. Ekici, "Helicopter rotor optimization via operator overloading-based discrete adjoint approach," in *AIAA Scitech 2021 Forum*, American Institute of Aeronautics and Astronautics, Jan. 2021.

[65] J. R. R. A. Martins and J. T. Hwang, "Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models," *AIAA Journal*, vol. 51, pp. 2582–2599, Nov. 2013.

[66] Z. Lyu and J. R. R. A. Martins, "Aerodynamic design optimization studies of a blended-wing-body aircraft," *Journal of Aircraft*, vol. 51, pp. 1604–1617, Sept. 2014.

[67] B. van Merrienboer, D. Moldovan, and A. Wiltschko, "Tangent: Automatic differentiation using source-code transformation for dynamically typed array programming," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[68] C. H. Bischof, L. Roh, and A. Mauer, "ADIC — An extensible automatic differentiation tool for ANSI-C," *Software–Practice and Experience*, vol. 27, no. 12, pp. 1427–1456, 1997.

[69] J. Utke, "OpenAD: Algorithm implementation user guide," Technical Memorandum ANL/MCS–TM–274, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 2004. available online at `ftp://info.mcs.anl.gov/pub/tech_reports/reports/TM-274.pdf`.

[70] L. Hascoet and V. Pascual, "The Tapenade Automatic Differentiation Tool," *ACM Transactions on Mathematical Software*, vol. 39, pp. 1–43, Apr. 2013.

[71] R. Giering, T. Kaminski, and T. Slawig, "Generating efficient derivative code with TAF: Adjoint and tangent linear Euler flow around an airfoil," *Future Generation Computer Systems*, vol. 21, no. 8, pp. 1345–1355, 2005.

[72] J. Utke, U. Naumann, M. Fagan, N. Tallent, M. Strout, P. Heimbach, C. Hill, and C. Wunsch, "OpenAD/F: A modular, open-source tool for automatic differentiation of Fortran codes," *ACM Transactions on Mathematical Software*, vol. 34, no. 4, pp. 18:1–18:36, 2008.

[73] C. H. Bischof, A. Carle, G. F. Corliss, A. Griewank, and P. D. Hovland, "ADIFOR: Generating derivative codes from Fortran programs," *Scientific Programming*, vol. 1, no. 1, pp. 11–29, 1992.

[74] C. H. Bischof, H. M. Bücker, B. Lang, A. Rasch, and A. Vehreschild, "Combining source transformation and operator overloading techniques to compute derivatives for MATLAB programs," in *Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2002)*, (Los Alamitos, CA, USA), pp. 65–72, IEEE Computer Society, 2002.

[75] A. Baydin, B. Pearlmutter, A. Radul, and J. Siskind, "Automatic differentiation in machine learning: A survey," *Journal of Machine Learning Research*, vol. 18, pp. 1–43, 04 2018.

[76] P. Aubert and N. Di Césaré, "Expression templates and forward mode automatic differentiation," in *Automatic Differentiation of Algorithms: From Simulation to Optimization*, Computer and Information Science, ch. 37, pp. 311–315, New York, NY: Springer, 2002.

[77] M. Sagebaum, T. Albring, and N. Gauger, "High-performance derivative computations using CoDiPack," *ACM Transactions on Mathematical Software*, vol. 45, no. 4, 2019.

[78] D. Shiriaev, A. Griewank, and J. Utke, "A user guide to ADOL–F: Automatic differentiation of Fortran codes," Tech. Report IOKOMO–04–1995, TU Dresden, Dept. of Mathematics, 1996.

[79] C. W. Straka, "Adf95: Tool for automatic differentiation of a fortran code designed for large numbers of independent variables," *Computer Physics Communications*, vol. 168, no. 2, pp. 123–139, 2005.

[80] S. Stamatiadis, R. Prosmiti, and S. C. Farantos, "AUTO_DERIV: Tool for automatic differentiation of a FORTRAN code," *Comput. Phys. Commun.*, vol. 127, pp. 343–355, may 2000. Catalog number: ADLS.

[81] B. Christianson, "Reverse accumulation and attractive fixed points," *Optimization Methods and Software*, vol. 3, no. 4, pp. 311–326, 1994.

[82] A. Walther and A. Griewank, "Getting started with adol-c," in *Combinatorial Scientific Computing* (U. Naumann and O. Schenk, eds.), ch. 7, pp. 181–202, Chapman-Hall CRC Computational Science, 2012.

[83] B. Kulfan, "A Universal Parametric Geometry Representation Method - "CST"," in *45th AIAA Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Jan. 2007.

[84] B. M. Kulfan, "Universal Parametric Geometry Representation Method," *Journal of Aircraft*, vol. 45, pp. 142–158, Jan. 2008.

[85] S. Nadarajah, P. Castonguay, and A. Mousavi, "Survey of Shape Parameterization Techniques and its Effect on Three-Dimensional Aerodynamic Shape Optimization," in *18th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, June 2007.

[86] B. Kulfan, ""CST" Universal Parametric Geometry Representation Method With Applications to Supersonic Aircraft," in *Fourth International Conference on Flow Dynamics*, Sendai International Center, Japan, Sept. 2007.

[87] E. D. Olson, "Three-Dimensional Piecewise-Continuous Class-Shape Transformation of Wings," in *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, June 2015.

[88] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso, "SU2: An Open-Source Suite for Multiphysics Simulation and Design," *AIAA Journal*, vol. 54, pp. 828–846, Mar. 2016.

[89] F. Menter, “Zonal two equation k-w turbulence models for aerodynamic flows,” in *23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference*, American Institute of Aeronautics and Astronautics, July 1993.

[90] F. R. Menter, “Two-equation eddy-viscosity turbulence models for engineering applications,” *AIAA Journal*, vol. 32, pp. 1598–1605, Aug. 1994.

[91] N. Gregory and C. O’Reilly, *Low-speed Aerodynamic Characteristics of NACA 0012 Aerofoil Section, Including the Effects of Upper-surface Roughness Simulating Hoar Frost*. Aeronautical Research Council Reports and Memoranda, R. & M. No. 3726, H.M. Stationery Office, 1973.