# AERODYNAMIC DESIGN OPTIMIZATION USING 3-DIMENSIONAL EULER EQUATIONS AND ADJOINT METHOD

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

ALİ YILDIRIM

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

JANUARY 2017

Approval of the thesis:

**AERODYNAMIC DESIGN OPTIMIZATION USING 3-DIMENSIONAL EULER EQUATIONS AND ADJOINT METHOD**

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

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Ozan Tekinalp _____
Head of Department, **Aerospace Engineering**

Assoc. Prof. Dr. Sinan Eyi _____
Supervisor, **Aerospace Engineering Dept., METU**

**Examining Committee Members:**

Prof. Dr. İsmail Hakkı Tuncer
Aerospace Engineering Department, METU _____

Assoc. Prof. Dr. Sinan Eyi
Aerospace Engineering Department, METU _____

Prof. Dr. Serkan Özgen
Aerospace Engineering Department, METU _____

Prof. Dr. Nafiz Alemdaroğlu
School of Civil Aviation Department, ATILIM UNI. _____

Assist. Prof. Dr. Durmuş Sinan Körpe
Aeronautical Engineering Department, THK UNI. _____

**Date:** _____

**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, Last Name: ALİ YILDIRIM

Signature :

# ABSTRACT

AERODYNAMIC DESIGN OPTIMIZATION USING 3-DIMENSIONAL EULER EQUATIONS AND ADJOINT METHOD

YILDIRIM, ALİ

M.S., Department of Aerospace Engineering

Supervisor : Assoc. Prof. Dr. Sinan Eyi

January 2017, 127 pages

The tool for the gradient-based optimization of wing-body configuration is developed using three-dimensional Euler equations and Adjoint method. Sensitivities required by design optimization are obtained by three different methods. Finite difference, Direct and Discrete Adjoint methods are used to compute objective sensitivities. A cell-centered, upwind based finite volume method is implemented to discretize the Euler equations. Second-order spatial accuracy is obtained by MUSCL interpolation. The flow solution is obtained by preconditioned, reordered matrix-free Newton-GMRES algorithm. The required derivatives for Adjoint and Direct methods are obtained by analytical derivation of discrete flow equations and finite different approaches. Resulted linear systems are solved by PARDISO solver. Direct and Adjoint sensitivities perfectly match with finite difference gradients. In optimization, design variables are selected as Non-Uniform Rational B-Spline (NURBS) curve and surface control points. Radial Basis Function (RBF) interpolation is used to deform volume mesh. Ad-

joint gradients are used at the optimization procedure due to its superiority on other gradient evaluation techniques. The single point optimization at cruise condition of wing and body are presented according to different design variables selections.

Keywords: optimization, aerodynamics, computational fluid dynamics, Adjoint Method, iterative solvers, Newton-GMRES, preconditioner, reordering, Non-Uniform Rational B-Splines (NURBS), Radial Basis Functions (RBF)

# ÖZ

3 BOYUTLU EULER DENKLEMLERİ VE ADJOİNT YÖNTEMİYLE AERODİNAMİK TASARIM OPTİMİZASYONU

YILDIRIM, ALİ

Yüksek Lisans, Havacılık ve Uzay Mühendisliği Bölümü

Tez Yöneticisi : Sinan Eyi

Ocak 2017 , 127 sayfa

Üç boyutlu Euler çözücü ve Adjoint metodu ile kanat-gövde tasarımının optimizasyonu için gradyan temelli araç geliştirilmiştir. Tasarım optimizasyonu için gerekli olan hassaslık verileri üç farklı metod tarafından elde edilmiştir. Amaç fonksiyonunun hassalık değişimi sonlu farklar metodu, Direct metod ve ayrıklaştırılmış Adjoint metodu tarafından hesaplanmıştır. Hücre merkezli, upwind temelli sonlu hacimler yöntemi Euler denklemlerinin ayrıklaştırılmasında kullanılmıştır. İkinci dereceden uzaysal doğruluk MUSCL interpolasyonu ile elde edilmiştir. Akış denklemlerinin çözülmesinde ön-koşullayıcı ve yeniden düzenleyici, matris gerektirmeyen Newton-GMRES algoritması kullanılmıştır. Adjoint ve Direct metod için gerekli olan türevler analitik türev hesabı ve sonlu farklar yöntemiyle elde edilmiştir. Lineer sistem denklemlerinin çözümünde PARDISO kullanılmıştır. Direct ve Adjoint metodunun hassaslık değerleri sonlu farklar yöntemiyle oldukça uyuşmuştur. Tasarım değişkenleri olarak Non-Uniform Rational

B-Spline (NURBS) eğim ve yüzey kontrol noktaları seçilmiştir. Hacimsel hücreler Radial Basis Function (RBF) kullanılarak deforme edilmiştir. Optimizasyon sırasında diğer metodlara üstünlüğü nedeniyle Adjoint yöntemi izlenmiş olup, kanat-uçak konfigürasyonu belirli bir seyir şartında farklı tasarım değişkenlerinin seçimiyle eniyilenmiştir.

Anahtar Kelimeler: optimizasyon, aerodinamik, hesaplamalı akışkanlar dinamiği, Adjoint metod, tekrarlayıcı çözücüler, Newton-GMRES, ön koşullayıcı, yeniden düzenleyici, Non-Uniform Rational B-Splines (NURBS), Radial Basis Functions (RBF)

Dedicated to my family and

... all aerospace lovers

# ACKNOWLEDGMENTS

I appreciate being part of the METU family as an aerospace engineer throughout my bachelor's and master's degree.

I would like to express my sincere gratitude to my supervisor Assoc. Prof. Sinan Eyi, for giving me the chance of being his student during my Msc study. His invaluable insights on problems, solutions and computational fluid dynamics have always helped me. As a Graduate Research Assistant, I am glad to be a member of this great team of Aerospace Engineering Department. I am thankful for all who always encourage and support me to do my best. A special thanks to Oguz Kaan Onay, Berk Gur, Muharrem Ozgun.

I would also like to thank my office mates Tugba Piskin and Emre Ozmen who have always kept me entertained with their sense of humor, silly questions and rambling conversations not only on CFD but also different aspects of life. Nobody apart from Tugba could have tolerated me and my first year as a MSc student. She has always advised me of being healthy and kept the plants in the room alive by watering them. And Emre has always influenced me with his creative personality and would broaden my horizon on anything.

Last but not the least, I would like to thank my girlfriend Yagmur Yener who has always been by my side supporting me in both personal and professional life with her kindness, caring and loving support.

ALİ YILDIRIM
Middle East Technical University, Aerospace Eng.
January, 25, 2017

# TABLE OF CONTENTS

# LIST OF TABLES

TABLES

# LIST OF FIGURES

FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| FD | Finite difference |
| AD | Automatic differentiation |
| $Q$ | Flow variable vector in cartesian coordinates |
| $F, G, H$ | Flux vectors in cartesian coordinates |
| $\rho$ | Density |
| $p$ | Pressure |
| $u, v, w$ | Velocities in cartesian velocities |
| $e_{in}$,$h$ | Internal energy, enthalpy |
| $\gamma$ | Specific gas constant |
| $J$ | Jacobian of transformation |
| $\hat{Q}$ | Flow variable vector in generalized coordinates |
| $\hat{F}, \hat{G}, \hat{H}$ | Flux vectors in generalized coordinates |
| $U, V, W$ | Contravariant velocity components |
| $i, j, k$ | Node indicies |
| $\xi, \eta, \zeta$ | Coordinates in computational space |
| $\rho_\infty$ | Density in free stream |
| $p_\infty$ | Pressure in free stream |
| $u_\infty, v_\infty, w_\infty$ | Cartesian velocities in free stream |
| $M_\infty$ | Free stream Mach number |
| $T_\infty, P_\infty$ | Free stream static temperature and pressure |
| $\hat{Q}^R, \hat{Q}^L$ | Right-left state variables |
| $\hat{Q}_{i+\frac{1}{2}}$ | Flow variable at cell face |
| $R$ | Residual vector |
| $A$,$x$,$f$ | Sparse Jacobian matrix, solution and right hand side vector |
| $\hat{A}$,$\hat{x}$,$\hat{f}$ | Preconditioned sparse Jacobian matrix, solution and right hand side vector |
| GMRES | Generalized Minimum Residual |
| FOM | Full Orthogonalization Method |

| | |
|---|---|
| $v$ | Krylov subspace vector |
| $r_0$ | Initial residual vector |
| LLS | Linear Least Squares |
| $h_{i,j}$ | Elements of upper Hessenberg matrix |
| $\eta_k$ | Forcing parameter for Newton-GMRES |
| $s_k$ | Initial guess for Newton step |
| $m$ | GMRES iteration index |
| $M$ | Preconditioner matrix |
| $M_L, M_R$ | Left and right preconditioner matrix |
| $\|.\|_2$ | Euicledan norm |
| $\mathbb{K}_k$ | Krylov subspace |
| ILU | Incomplete lower upper factorization |
| ILU$(p, \tau)$ | Incomplete lower upper factorization with level of fill and treshold |
| ND | Nested Dissection ordering |
| MD | Minimum Degree ordering |
| RCM | Reverse Cuthill-Mckee ordering |
| $P_1, P_2$ | Permutation matrices |
| NURBS | Non-Uniform Rational B-Splines |
| $S(\xi, \eta)$ | Coordinates of NURBS points |
| $\xi, \eta$ | Parametric coordinates of NURBS points |
| $n, m$ | Number of control points - 1 in NURBS |
| $p, q$ | Degree of NURBS curves |
| $P_{i,j}$ | Bi-directional control points |
| $N_{i,p}, N_{j,q}$ | Basis functions of p and q degree |
| $u_i$ | NURBS knots |
| $U, V$ | Knot vectors |
| $R_{i,j}$ | Rational NURBS basis function |
| $RBF$ | Radial Basis Functions |
| $S_j$ | Arch-length distance |
| $L_g$ | Grid node lengths |
| $\phi$ | Radial basis function |
| $N_{rbf}$ | Number of RBF points |
| $N_v$ | Number of volume node points |

| | |
|---|---|
| $\alpha_x, \alpha_y, \alpha_z$ | Weigths of RBF points |
| $\Delta x_r, \Delta y_r, \Delta z_r$ | Deflection in RBF points |
| $R_0$ | Support radius |
| $\Delta x_v, \Delta y_v, \Delta z_v$ | Deflection in volume mesh points |
| $I, J, K$ | Computational grid planes |
| $I$ | Objective function |
| $\Omega$ | Objective function gradient |
| $S$ | Design variable vector |
| $N_F$ | Number of flow variables |
| $N_D$ | Number of design variables |
| $\psi$ | Adjoint variables |
| $X$ | Grid points |
| DOT | Design Optimization Tool |
| MMFD | Modified Method of Feasible Directions |

# CHAPTER 1

# INTRODUCTION

## 1.1 Wing-Body Design and Optimization

Aeronautics had started with a simple question "How can a manned flight be possible" in approximately 500 years ago. Now the entire humanity is aware of this fact and can easily answer this question. Today, development in cutting-edge technology and science make people want to fly more efficiently and faster to far away. Even humanity is looking for a way to fly outside of earth ie. imitation of an insect flying to Mars. This is not a fantasy or dream anymore. Now, we can make transatlantic flies with transonic airplanes and the hypersonic missiles are ready to achieve their intercontinental operations in less than one hour.

There are many comprehensive research studies being carried out which aim to make flights more efficient to carry more passengers and freight to further locations with minimum fuel consumption for less carbon footprint and eventually less populated clear skies. Therefore researchers and engineers have been to design more wisely and try to obtain optimum shape that meets requirements. Typical approach that has been found by engineers to design more wisely is to create a design methodology and they have divided the design phases into 3 categories, conceptual, preliminary and detailed design. The developed methodology and design phases are explained in detail in many textbooks [3][4]. A standard approach for typical aircraft is assessing and evaluating each component individually. First, the lifting surfaces should be designed according to standard design parameters such as airfoil thickness, camber, maximum and design lift

coefficients, wing aspect ratio, sweep angle, taper ratio, twist distribution and so on. The second step is to design for minimum parasite drag or induced drag, better fuel efficiency, increased endurance, minimum weight to carry more payload or creating shock-less wing sections. In every phase of design, wing and body both together play a key role in final design since the aerodynamic parameters are strong function of them. In order to show these components' effect on all aircraft, the typical drag and lift breakdown of passenger aircraft at cruise condition are illustrated as below.

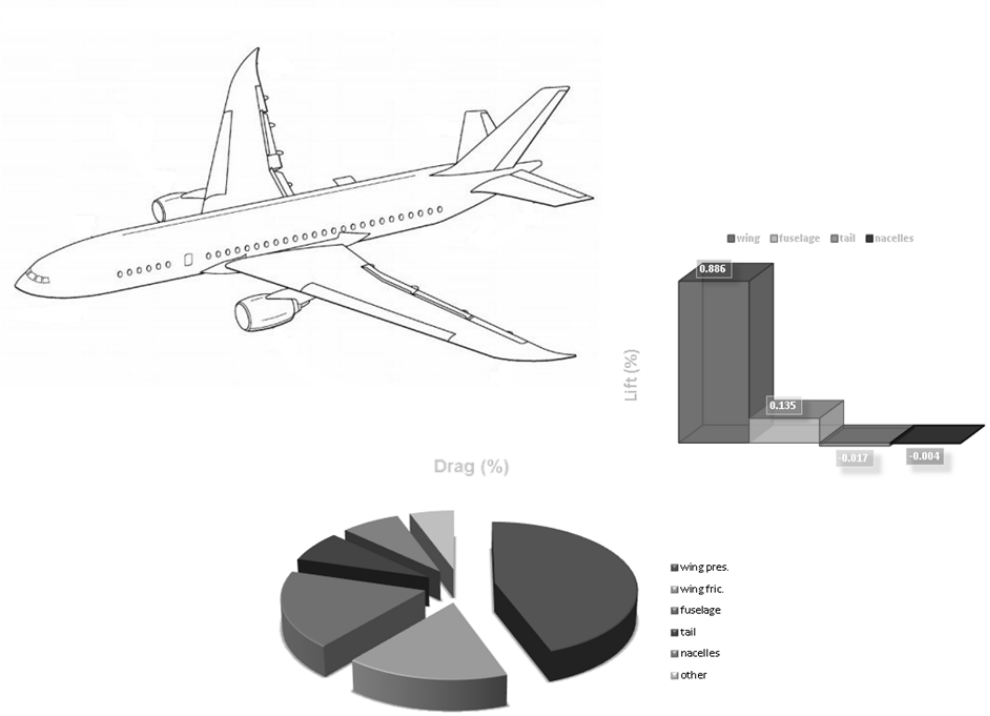



Figure 1.1: Typical lift&drag breakdown of passenger aircraft [1]

As seen in Figure 1.1 most of the lift and drag are provided by wing and body. Therefore their optimal shapes will increase the general performance of the entire aircraft. Thus in this study, we will mainly focus on wing and body to get the optimum shape.

Typical approach is still used widely in conceptual design. However, in later phases of design, more sophisticated tools and approaches are required ie. computational fluid dynamics (CFD). CFD can be used to experiment flow field on designed object. Or even it can be used as specific design tool. Start-

ing with an potential, Euler, Navier-Stokes or even Reynolds-Averaged Navier-Stokes (RANS) solvers today, CFD is known as the best tool for aerodynamic design. The design optimization using CFD requires 4 main elements; optimization technique, gradient calculation (if it is required), design variables and flow solver. In the following sections, 4 main elements of CFD based optimization will be mentioned by focusing on aerodynamics.

## 1.2 Review of Aerodynamic Optimization Using CFD

### 1.2.1 Optimization Techniques

Optimization techniques in aerodynamic design can be considered in two main categories. These are *inverse design* and *numerical optimization*. Figure 1.2 shows the general classification of CFD-based aerodynamic optimization techniques.

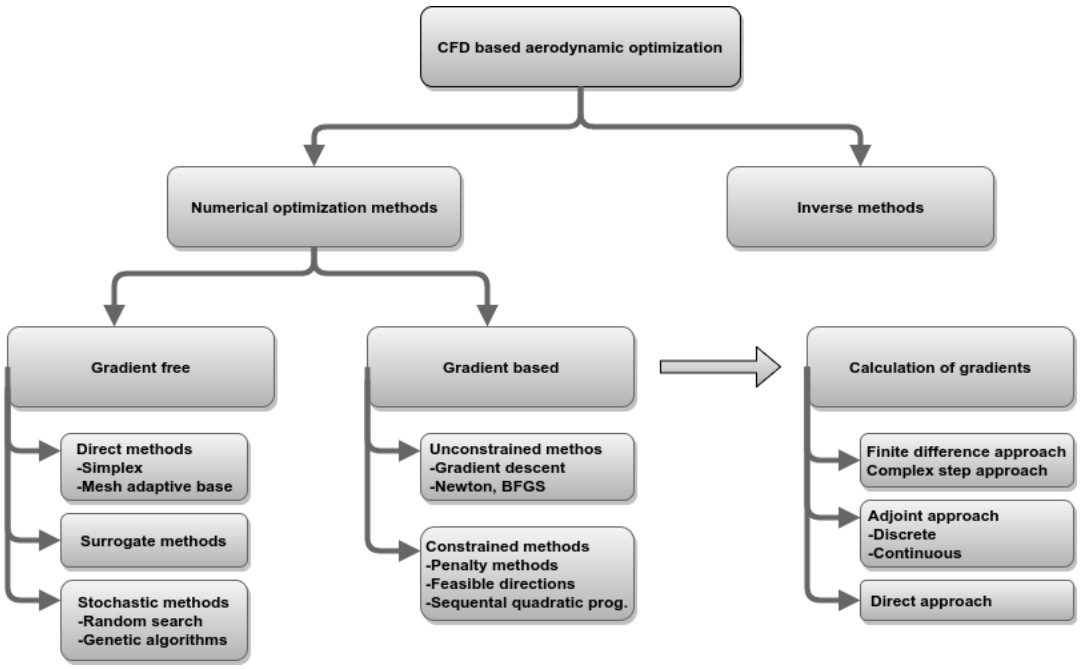



Figure 1.2: Optimization techniques chart in aerodynamics

Inverse design techniques have been commonly used for a long time. It was discovered first by Betz [5] to find an airfoil shape for predetermined pressure distribution. And later, it was developed for the design optimization of various

Jukosky airfoils by Lighthill [6] by using conformal mapping. Further developments to inverse design were succeeded by Volpe & Malnik in 1986 on transonic airfoils using full potential equations and Giles & Drela [7] in 1987 by transonic flow using Euler equations on 2D airfoils. Although the approach is still used and applied to complex flow equations, there is one drawback that is crucial. The designers are not able to always predict the target pressure distribution which satisfies desired performance of optimal shape. Even though, the less computational time is offered by inverse design, the designer selections may fail the optimal shape at the end specifically where separated or turbulent flows take place and the pressure distribution becomes noisy.

On the other hand, second method-numerical optimization-offers a well-posed, robust optimization problem. The required knowledge and experience in inverse design is restricted by rules of numerical optimization technique. Optimization problem is treated a set of linear and nonlinear objective, constraint functions that give the optimal design. The designer is supposed to set up the objective and constraint functions which are specific function of design variables. While objective functions can be lift, drag, pitching moment etc., constraints can be considered as other aerodynamic parameters or geometric variables that have a direct impact on optimal shape. Numerical optimization-in some references mentioned as direct numerical optimization-technique is investigated in two main categories. These are gradient-based methods and gradient-free methods.

The pioneers of numerical optimization technique in aerodynamics are Hicks, Murman and Vanderplaats [8] who applied their methodology to assess the optimal airfoil with various constraints and cost functions. Their approach was the gradient-based but we will mention first the other techniques which are gradient-free. The gradient-free methods can be summarized broadly in three categories; direct methods, surrogate methods and stochastic methods. One of the well known direct methods is *simplex method* which was found by Dantzig [9] for the optimization of linear problems. In later 1965, Nelder and Mead [10] extended the approach for non-linear problems for which the derivatives are not known. The algorithm was promising for notably small number of design variables where objective function was smooth. Probably one of the main advantages was that

there was no need for an expensive gradient calculation. With less effort on aerodynamic side of simplex method, one can find comprehensive research on Duvigneau and Visonneau [11] works.

Stochastic methods are developed mainly on genetic algorithms. The application settles on the biologically inspired fact the operators of mutation, crossover and selection. In algorithm, candidate optimal solutions evolve to better solution by changing their chromosomes. The main advantage of this approach is that it is capable of finding global optimum by comparing to other methods. But it requires tremendous computational cost, function evaluations or in aerodynamics we say flow solutions. First application on aerodynamics which was applied on transonic airfoils was done by Quagliarella and Cioppa [12].

On the other side, gradient-based methods require gradient evaluation or the sensitivity of design variables with respect to objective functions. Naturally they are supposed to converge local minima. They are categorized by many aspects such that constrained, unconstrained, linear and non-linear optimization techniques. Further explanation can be found in the book of Nocedal and Wright [13]. The first application of this kind of approaches was done by Hicks, Murman and Vanderplaats [8] in 1974 on airfoil optimization. Then, the work was extended to three-dimensional wings [14]. The approach was simple, first flow code was developed and with a little modification the gradients were calculated by finite difference approach. They used this approach to solve optimization problem by method of feasible directions which is under the category of constrained optimization. More or less the selection of optimization technique in gradient based methods restricted by the techniques are given in Figure 1.2. However, the calculation of gradients have been investigated for many years. The different approaches have attracted the attention of researchers and they are still a great research subject. In the following section, we will give the history of these approaches, which is also the main focus of this thesis.

### 1.2.2 Gradient Calculation

In the calculation of gradients, fastness and correctness are crucial for gradient-based methods, since the descent or search direction proceeds according to these features. In early times of aerodynamic optimization, the easiest method was considered to be *finite difference* (FD) approach. Its first application is referenced in [8]. Later on, Eyi and Lee [15] applied this approach into high lift optimization of airfoils using Navier-Stokes equations. Finite difference implementation is considered to be fairly straightforward as it only requires the flow solution at perturbed design variables. However, as the number of design variables increases the repeated flow solutions can be inhibitive in terms of computational time. For instance, in central difference scheme, two times of number of design variables flow solutions are required. In addition to this, the calculated gradients accuracy is directly related to step-size choice. While too small steps-sizes increase round-off errors, too large step-sizes increase truncation error. For all these reasons, finite difference approach is not frequently used and only used for comparison reasons with different methods. Nevertheless, the researchers have tried to develop this approach and they have found the *complex step* (CS) methods. Comparing to FD approach, complex step does not need subtraction operation to estimate derivatives, so the cancellation or round-off errors will be diminished. This will allow to use very small perturbation number without considering too much on round-off errors. However, the written flow code must be converted to deal with complex variables in this approach. The detailed study can be found in Marting et al. [16].

In 1983 Pironneau [17] asked the question, "*What is the best shape for physical system?*" and he found the solution by calculus of variation. His book were specialized on optimal shape design using elliptic PDE's. Then in 1988 Jameson [18] published the work on aerodynamic design and he explained its study by the control theory approach. In this method, necessary gradients are obtained by the reformulation of flow equations to solve Adjoint variables that were imposed to Adjoint boundary conditions. The resulting equations can be discretized and solved similar to flow equations. The complete gradients can be extracted from

one flow and adjoint equation solutions which show the superiority of method on FD method. The method is called as *continuous adjoint*. Jameson's research group developed the continuous adjoint during time. In 1994 [19] they used the potential flow equations on airfoils and in 1995 [20] they developed the method for wing and wing-body configurations with Euler equations. At a later time, the planform optimization of Boeing aircraft was succeed by Leoviriyakit [21]. Anderson and Venkatakrishnan [22] applied this method on unstructured grids. Since then there have been many further improvements and different studies done by many researchers on continuous adjoint. Although the method is efficient, the coding time and workforce are great. As the flow equations are getting complex, the continuous adjoint equation cannot be derived or it may even not exist at all. In addition to continuous adjoint, there is one more approach which is called as *discrete adjoint* which provides flexibility in adjoint-based optimization. The method is defined in first-discretize-then-optimize approach. In this approach, flow solver is discretized, then discretized equations are differentiated analytically or by *automatic differentiation* (AD) [23] in line with the required design variables and flow variables. As a result of this, the linear system of adjoint equation can be solved. This method has been studied by many authors in the references, [24][25][22]. The discrete and continuous approaches have been compared to each other by Shubin and Frank [26], and by Nadarajah and Jameson [27]. And they concluded that as the discretization of both approaches are the same the discrete adjoint sensitivities perfectly match the finite difference sensitivities in any circumstances. However, the continuous adjoint can only be the same as the finite difference sensitivities only if the same discretization is applied to continuous problem with the flow equations where is not applicable in most of the time.

On the other hand, *direct method* or occasionally referred as *flow sensitivity* and *direct differentiation* method are another way of dealing with discrete gradients. Hou et. al [28] has described the methods of finite difference, direct and discrete adjoint. In direct sensitivity method, for each design variable linear system of flow sensitivity equation must be solved. This method can be practical when a number of design variables are low or the system is solved by direct methods such

that LU factorization with different right hand sides. Nevertheless, the discrete adjoint has proved itself as being the most robust and accurate, and that no other methods have achieved it yet, thus the thesis will focus on the discrete adjoint approach while the sensitivities are compared to the other methods as well.

### 1.2.3 Design Variables and Grid Motion

One of the crucial elements of optimization is the selection of design variables or namely shape parameterization due to the fact that the optimal shapes are more or less the functions of design variables. While some methods offer less computational time, some methods offer smooth aerodynamic shapes that gives a better general performance even at off-design conditions.

The first approach is applied by choosing grid nodes as design variables. However, this approach is not practical for large grids especially in three-dimensional configurations. For this reason, Hicks and Henne [29] have introduced the bump functions which add the parameterized deflection of baseline geometry. This approach then quite reduced the number of design variables. Then some researchers have worked on the parameterization of known shapes such as wing sections, airfoils. Sobieczky [30] introduced the PARSEC algorithm in 1999. This approach is based on the relation of some parametric features (leading edge radius, maximum camber etc.) of airfoils. Later many other methods have emerged based on this approach. The other method is using the splines ie. Bezier, B-spline, Non-uniform rational b-splines (NURBS) curves and surfaces [31] to create any shape. Spline methods have become very popular and still they are still being used in many aerodynamic optimization studies nowadays [32].

The other concern is that how surface deflections can be applied to control volume mesh in particularly CFD. Compared to unstructured grids, structured grid deformation is relatively easy. Burgreen and Baysal [33] have introduced the algebraic grid deformation in 2D case, then the algorithm has been extended to 3D by them [25] using the arch length parameterization of grid lines. On

the other hand, iterative methods such that linear spring analogy and linear elasticity methods have also been studied and used for long time. However, due to high computational costs they are known as expensive methods. Radial basis functions (RBF) method was introduced by Boer et. al [34] in 2006 to allow large grid deformations in unstructured grids. Fortunately, algorithm was suitable for structured grids as well and then was proposed by Poirier and Nadarajah [35] in detail. In this study, both NURBS surfaces and curves will be used to decide surface point deflections with the conjunction of algebraic grid perturbation and RBF approaches.

### 1.2.4 Flow Solver

In gradient optimization, function evaluations namely flow solutions are used in the line search repeatedly. Therefore a fast and robust solution strategy should be followed. Since the flow Jacobian matrix is created for the linear system of adjoint and direct sensitivity approaches, this matrix can be directly used to solve flow equations with Newton method in implicit manner. The usage of *Newton method* in CFD was initiated by Venkatakrishnan [36] in 1989 on both viscous and inviscid flow solution of airfoils. He linearized the flow equations by Newton method and created the sparse flow Jacobian matrix. Then he solved the linear systems by LU decomposition. The result was remarkable since the machine precision in residual was managed to be obtained in less than ten iterations. The solution had quadratic convergence even after freezing the Jacobian. Nevertheless, the direct solution of the Jacobian matrix was not efficient computationally. In addition, memory requirements were a limiting issue in LU factorization.

At the same time, the researchers were working on robust and fast sparse iterative solvers, Krylov solvers. The first appearance of Krylov methods indicates to Alexei Nikolaevich Krylov in 1930's. In 1952 Hestenes and Stiefel [37] published the first work on solution of symmetric linear system by Krylov methods. They called their methods as *conjutage gradients*. Later, Saad [38] has applied the Krylov methods on unsymmetric matrices and found the Generalized Minimum

Residual (GMRES) method. Anderson and Venkatakrishnan [22] have applied the method on viscous and inviscid flows. Michalak and Gooch [39] have created a guide for higher-order inviscid memory efficient linear system creation and solution by GMRES.

Although, the improvements on iterative solvers make linear system solution fast with less memory requirements, still the creation of Jacobian matrix for every Newton iteration is cumbersome. Then, matrix free Newton-GMRES algorithm has appeared. In this algorithm, Jacobian matrix does not need to be created explicitly. With the developments of different preconditioning and reordering algorithms, preconditioned Newton-GMRES has become very popular in the flow solution of both viscid and inviscid flows. Pueyo et. al [40] has studied the matrix free Newton-GMRES on viscous flow using Baldwin-Lomax turbulence model and Nemec et. al [41] has developed the Spalart Allmaras turbulent model version of previous study. Both studies have been very successful and they have outputted a guideline for generation and solution of system by the help of Newton-GMRES. Although there have been many studies on preconditioning and ordering of Newton-GMRES, the most and current one is by Gatsis [42] and the reader is strongly encouraged to read his doctoral thesis. Because of the powerful background and robustness of the method, in this thesis matrix-free Newton-GMRES algorithm will be investigated with different preconditioner choices and reordering algorithms.

## 1.3 Outline of Thesis

In the introduction part, the key points of CFD based optimization were mentioned. In later chapters these elements will be discussed in more detail. In the second chapter, the governing equations and spatial discretization of these equations will be mentioned including higher order discretization. The application of boundary conditions and the guideline of creation of Jacobian matrix will also be given. In the third chapter, the Newton Method, Newton-GMRES methods with the implementation of preconditioner and reordering algorithms will be discussed. In the fourth chapter, selection of design variables and grid

deformation techniques will be explained. In the fifth chapter, the evaluation of gradient sensitivities, finite difference, adjoint and direct method with optimization technique will be explained in detail. Finally, in the result section, all the results according to the flow solution, sensitivity accuracy and optimization for different design variables will be presented. The results will be concluded and further research areas will be given in the conclusion part. **The scope of the thesis** is the development of adjoint based robust optimization technique on wing-body configuration using Euler equations by focusing on all aspects of optimization ie. flow solution, optimization, design variables and gradient sensitivity calculation.

# CHAPTER 2

# GOVERNING EQUATIONS

## 2.1 3-D Euler Equations in Cartesian Coordinates

Euler Equations in 3-D flow can be written in cartesian coordinates as follows.

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = 0 \tag{2.1}$$

For this flow conservative flow variable vector, $Q$ is given by

$$Q = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{Bmatrix} \tag{2.2}$$

where $\rho$ is density, $u, v$ and $w$ are velocities in cartesian coordinates $(x, y, z)$ and $p$ is the pressure. Then corresponding flux vectors $(F, G, H)$ can be written

$$F = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(\rho e + p) \end{Bmatrix} \qquad G = \begin{Bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ v(\rho e + p) \end{Bmatrix} \qquad H = \begin{Bmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ w(e\rho + p) \end{Bmatrix} \tag{2.3}$$

In Equation (2.3) $e$ is total energy per unit volume. Since pressure, $p$ is not defined in flow variables in Equation (2.2), it is obtained by ideal gas relation such that

$$p = \rho e_{in}(\gamma - 1) \tag{2.4}$$

where $e_{in}$ indicates internal energy and total energy is the sum of internal and kinetic energies

$$e = e_{in} + \frac{1}{2}(u^2 + v^2 + w^2) \tag{2.5}$$

where $\gamma$ is specific heat ratio which is taken as 1.4.

Until now, Euler equations in 3-D cartesian coordinates are formulated. However, this formulation is not practical for the solution of non-regular domains. Therefore, cartesian coordinates must be transformed to generalized curvilinear coordinates.

## 2.2 3-D Euler Equations in Generalized Coordinates

Governing equations mentioned in previous section can be written in generalized coordinates as

$$\frac{\partial \hat{Q}}{\partial \tau} + \frac{\partial \hat{F}}{\partial \xi} + \frac{\partial \hat{G}}{\partial \eta} + \frac{\partial \hat{H}}{\partial \zeta} = 0 \tag{2.6}$$

For a conservative flow variable vector, $\hat{Q}$

$$\hat{Q} = J^{-1}Q = \frac{1}{J}\begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{Bmatrix} \tag{2.7}$$

Similarly conservative fluxes, $\hat{F}, \hat{G}, \hat{H}$, can be written such that

$$\hat{F} = \tfrac{1}{J}\begin{Bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ (\rho e + p)U \end{Bmatrix} \quad \hat{G} = \tfrac{1}{J}\begin{Bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ (\rho e + p)V \end{Bmatrix} \quad \hat{H} = \tfrac{1}{J}\begin{Bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ (\rho e + p)W \end{Bmatrix} \tag{2.8}$$

In Equation (2.8) formulation small letters for $u, v, w$ correspond to cartesian velocities and capital letters $U, V, W$ are contravariant velocity components, $J$ is the coordinate transformation Jacobian, $\xi$, $\eta$ and $\zeta$ are the curvilinear coordinates, and $\xi_x, \xi_y, \xi_z, \eta_x, \eta_y, \eta_z, \zeta_x, \zeta_y, \zeta_z$ are the transformation metrics. Metric

coefficients are given by

$$\begin{aligned}
\xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta), \quad \xi_y = J(x_\zeta z_\eta - x_\eta z_\zeta), \quad \xi_z = J(x_\eta y_\zeta - x_\zeta y_\eta) \\
\eta_x &= J(y_\zeta z_\xi - y_\xi z_\zeta), \quad \eta_y = J(x_\xi z_\zeta - x_\zeta z_\xi), \quad \eta_z = J(x_\zeta y_\xi - x_\xi y_\zeta) \\
\zeta_x &= J(y_\xi z_\eta - y_\eta z_\xi), \quad \zeta_y = J(x_\eta z_\xi - x_\xi z_\eta), \quad \zeta_z = J(x_\xi y_\eta - x_\eta y_\xi)
\end{aligned} \tag{2.9}$$

Jacobian of transformation is simply,

$$J = \begin{vmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{vmatrix} \tag{2.10}$$

The contravariant velocities are defined as

$$\begin{aligned}
U &= \xi_x u + \xi_y v + \xi_z w \\
V &= \eta_x u + \eta_y v + \eta_z w \\
W &= \zeta_x u + \zeta_y v + \zeta_z w
\end{aligned} \tag{2.11}$$

### 2.3 Non-Dimensionalization

In order to prevent existence of measurement units all governing equations starting from Equation (2.1) to Equation (2.5) are non-dimensionalized. In non-dimensionalization process, following scaling parameters are used [43].

$$x = \frac{\tilde{x}}{\tilde{L}_R}, \quad y = \frac{\tilde{y}}{\tilde{L}_R}, \quad z = \frac{\tilde{z}}{\tilde{L}_R} \tag{2.12}$$

$$\rho = \frac{\tilde{\rho}}{\tilde{\rho}_\infty}, \quad u = \frac{\tilde{u}}{\tilde{a}_\infty}, \quad v = \frac{\tilde{u}}{\tilde{a}_\infty}, \quad w = \frac{\tilde{u}}{\tilde{a}_\infty} \tag{2.13}$$

$$p = \frac{\tilde{p}}{\tilde{\rho}_\infty (\tilde{a}_\infty)^2}, \quad e = \frac{\tilde{e}}{\tilde{\rho}_\infty (\tilde{a}_\infty)^2}, \quad a = \frac{\tilde{a}}{\tilde{a}_\infty}, \quad T = \frac{\tilde{T}}{\tilde{T}_\infty} = \frac{\gamma p}{\rho} = a^2 \tag{2.14}$$

All tilde ($\sim$) symbols indicate the dimensional quantity, the left side of equalities are non-dimensional. The initialization of flow domain is done through non-dimensional free-stream values which are given below.

$$\rho_\infty = 1, \quad a_\infty = 1, \quad p_\infty = \frac{1}{\gamma}, \quad T_\infty = 1, \quad e_\infty = \frac{1}{\gamma(\gamma - 1)} + \frac{(M_\infty)^2}{2} \tag{2.15}$$

$$u_\infty = M_\infty cos(\alpha), \quad v_\infty = M_\infty sin(\alpha), \quad w_\infty = 0 \tag{2.16}$$

## 2.4 Spatial Discretization

The differential form of the steady 3-D Euler equations given in Equation (2.6) can be discretized for an arbitrary hexahedral control volume.

$$\frac{\delta_\xi \hat{F}}{\Delta\xi} + \frac{\delta_\eta \hat{G}}{\Delta\eta} + \frac{\delta_\zeta \hat{H}}{\Delta\zeta} = 0 \tag{2.17}$$

For a cell centered finite volume method Equation (2.17) can be written as

$$(\hat{F}_{i+\frac{1}{2},j,k} - \hat{F}_{i-\frac{1}{2},j,k}) + (\hat{G}_{i,j+\frac{1}{2},k} - \hat{G}_{i,j-\frac{1}{2},k}) + (\hat{H}_{i,j,k+\frac{1}{2}} - \hat{H}_{i,j,k-\frac{1}{2}}) = 0 \tag{2.18}$$

The inviscid fluxes of the Euler equations represent the convective phenomena. The upwind flux splitting schemes are used for the spatial discretization of the flux vectors. In this study the split fluxes, $\hat{F}^+, \hat{F}^-, \hat{G}^+, \hat{G}^-, \hat{H}^+, \hat{H}^-$ are calculated by Steger-Warming [44], Van Leer [45] and AUSM [46] methods.

$$\begin{aligned}
\hat{F}_{i\pm\frac{1}{2},j,k} &= \hat{F}^+(\hat{Q}^L_{i\pm\frac{1}{2},j,k}) + \hat{F}^-(\hat{Q}^R_{i\pm\frac{1}{2},j,k}) \\
\hat{G}_{i,j\pm\frac{1}{2},k} &= \hat{G}^+(\hat{Q}^L_{i,j\pm\frac{1}{2},k}) + \hat{G}^-(\hat{Q}^R_{i,j\pm\frac{1}{2},k}) \\
\hat{H}_{i,j,k\pm\frac{1}{2}} &= \hat{H}^+(\hat{Q}^L_{i,j,k\pm\frac{1}{2}}) + \hat{H}^-(\hat{Q}^R_{i,j,k\pm\frac{1}{2}})
\end{aligned} \tag{2.19}$$

where the $i \pm 1/2, j \pm 1/2$ and $k \pm 1/2$ denote a cell interfaces. The fluxes are calculated at the cell interfaces by using the flow variables interpolated from the cell center. Simple scheme with first-order accuracy in space is obtained by assuming the values at the cell faces are equal to the values at the nearest cell centers such that

$$\begin{aligned}
\hat{Q}^L_{i+\frac{1}{2},j,k} &= \hat{Q}_{i,j,k} \text{ and } \hat{Q}^R_{i+\frac{1}{2},j,k} = \hat{Q}_{i+1,j,k} \\
\hat{Q}^L_{i,j+\frac{1}{2},k} &= \hat{Q}_{i,j,k} \text{ and } \hat{Q}^R_{i,j+\frac{1}{2},k} = \hat{Q}_{i,j+1,k} \\
\hat{Q}^L_{i,j,k+\frac{1}{2}} &= \hat{Q}_{i,j,k} \text{ and } \hat{Q}^R_{i,j,k+\frac{1}{2}} = \hat{Q}_{i,j,k+1}
\end{aligned} \tag{2.20}$$

### 2.4.1 Van Leer Flux Splitting Method

For an example, Van Leer method will be presented here. In Van Leer's method flux are split according to its contravariant mach number for subsonic and supersonic cases. The flux in $\xi$ direction, $\hat{F}$ depends on the sign of $M_\xi$ where

$$M_\xi = \frac{\bar{U}}{a} \tag{2.21}$$

knowing

$$\bar{U} = \frac{U}{|\nabla \xi|} \tag{2.22}$$

$U$ is contravariant velocity in $\xi$ direction as given in Equation (2.11). Then the splitting criteria is given

- For locally supersonic flow, $|M_\xi \geq 1|$;

$$\begin{aligned} \hat{F}^+ &= \hat{F} \quad \hat{F}^- = 0 \quad \text{if } M_\xi \geq +1 \\ \hat{F}^+ &= 0 \quad \hat{F}^- = \hat{F} \quad \text{if } M_\xi \leq -1 \end{aligned} \tag{2.23}$$

- For locally subsonic flow, $|M_\xi < 1|$;

$$\hat{F}^\pm = \frac{|\nabla \xi|}{J} \begin{Bmatrix} f_{mass} \\ f_{mass}\left[\frac{\hat{\xi}_x(-\bar{U}\pm 2a)}{\gamma} + u\right] \\ f_{mass}\left[\frac{\hat{\xi}_y(-\bar{U}\pm 2a)}{\gamma} + v\right] \\ f_{mass}\left[\frac{\hat{\xi}_z(-\bar{U}\pm 2a)}{\gamma} + w\right] \\ f_{mass}\left[\frac{(1-\gamma)\bar{U}^2 \pm 2(\gamma-1)\bar{U}a + 2a^2}{(\gamma^2-1)} + \frac{(u^2+v^2+w^2)}{2}\right] \end{Bmatrix} \tag{2.24}$$

In Equation (2.24) normalized metrics are

$$\hat{\xi}_x = \frac{\xi_x}{|\nabla \xi|} \quad \hat{\xi}_y = \frac{\xi_y}{|\nabla \xi|} \quad \hat{\xi}_z = \frac{\xi_z}{|\nabla \xi|} \tag{2.25}$$

$$\text{where} \quad |\nabla \xi| = \sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2} \tag{2.26}$$

with

$$f_{mass} = \pm \frac{\rho a}{4}(M_\xi \pm 1)^2 \tag{2.27}$$

In previous splitting methods such as Steger-Warming, split flux are not continuously differentiable at the stagnation and sonic points. These discontinuities may cause some oscillations. Van Leer solves this problem by imposing the extra criteria that split fluxes, $\hat{F}^\pm$ must be continuous. By the help of this condition, fluxes are split according to their mach number and shock resolution is improved.

## 2.5 Monotonic Upstream-Centered Scheme Conservation Law

Higher order accuracy in space can be obtained by using Monotonic Upstream-Centered Scheme Conservation Law (MUSCL) interpolation. Its formulation is given below.

$$\hat{Q}^L_{i+\frac{1}{2}} = \hat{Q}_i + \frac{1}{4}\Big\{\phi(r)\big[(1-\kappa)\nabla + (1+\kappa)\Delta\big]\Big\}_i \tag{2.28}$$

$$\hat{Q}^R_{i+\frac{1}{2}} = \hat{Q}_{i+1} - \frac{1}{4}\Big\{\phi(r)\big[(1-\kappa)\Delta + (1+\kappa)\nabla\big]\Big\}_{i+1} \tag{2.29}$$

Upwind-biased schemes are known to under-shoots and over-shoots in the shock region. A limiter can be used to reduce the scheme to a one-sided in the shock regions, then oscillations can be removed. The first variation of this limiter is non-differentiable min-mod limiter. However, its better to use a differentiable limiter such that Van Albada Limiter [47]. It is continuously differentiable and suitable for analytical Jacobian calculation. The limiter is implemented by rewriting Equations (2.28) and (2.29).

$$\hat{Q}^L_{i+\frac{1}{2}} = \hat{Q}_i + \Big\{\frac{s}{4}\big[(1-s\kappa)\nabla + (1+s\kappa)\Delta\big]\Big\}_i \tag{2.30}$$

$$\hat{Q}^R_{i+\frac{1}{2}} = \hat{Q}_{i+1} - \Big\{\frac{s}{4}\big[(1-s\kappa)\Delta + (1+s\kappa)\nabla\big]\Big\}_{i+1} \tag{2.31}$$

where $s, \Delta$ and $\nabla$ are

$$s = \frac{2\Delta\nabla + \epsilon}{\Delta^2 + \nabla^2 + \epsilon} \tag{2.32}$$

$$\Delta_i = \hat{Q}_{i+1} - \hat{Q}_i, \qquad \nabla_i = \hat{Q}_i - \hat{Q}_{i-1} \tag{2.33}$$

The parameter $\epsilon$ is introduced to avoid division by zero. For this study, it is taken [48];

$$\epsilon = 10 * \Omega^{1.25} \tag{2.34}$$

where $\Omega$ is the average volume of neighbor cells. In order to complete the Van Albada Limiter, $\kappa = 0$ can be chosen which is upwind-biased scheme. $\kappa$ does not have to be an integer, however its value must be in region, $[-1, \frac{1}{3}]$. Moreover, accuracy of solutions can be increased by increasing $\kappa$ being aware of divergence pitfall.

## 2.6 Grid Topology and Boundary Conditions

In this study, ARA M100 [2] wing+body configuration is used as test case with single block, C-H type structured grid. In Figure 2.1 boundary conditions are shown.

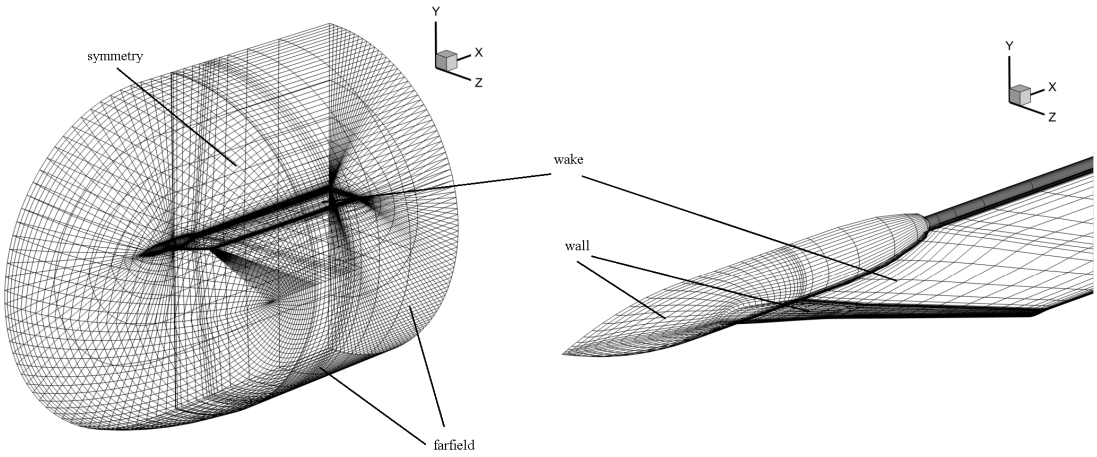



Figure 2.1: Grid topology and boundary conditions

In the application of boundary conditions, ghost cell approach is used due to easiness of implementation in both flow solution and Jacobian matrix generation.

### 2.6.1 Symmetry BCs

On symmetry plane, all conservative scalar variables in inner cells are passed directly to the ghost cells except the velocity normal to symmetry plane. In our case;

$$\begin{aligned} w^1_{ghost} &= w^1_{inner} \\ w^2_{ghost} &= w^2_{inner} \\ w^3_{ghost} &= w^3_{inner} \\ w^4_{ghost} &= -w^4_{inner} \\ w^5_{ghost} &= w^5_{inner} \end{aligned} \tag{2.35}$$

### 2.6.2 Wall (slip) BCs

On wall flow tangency criteria is applied due to inviscid nature of Euler flow. In this case, all normal velocities to the wall are equated to zero. This means contravariant velocity vector at cell face $\bar{V}_{face} = 0$. Then, first and last conservative variables are simply extrapolated. $\bar{V}_{face} = 0$ condition is satisfied as follows.

$$u_{face} = u_{inner} - \eta_x \bar{V}_{inner} \tag{2.36}$$

$$v_{face} = v_{inner} - \eta_y \bar{V}_{inner} \tag{2.37}$$

$$w_{face} = w_{inner} - \eta_z \bar{V}_{inner} \tag{2.38}$$

Multiplying above equations with $\eta_x, \eta_y$ and $\eta_z$, then adding and rearranging;

$$\underbrace{u_{in}\eta_x + v_{in}\eta_y + w_{in}\eta_z}_{\bar{V}_{face}} = \underbrace{u_{in}\eta_x + v_{in}\eta_y + w_{in}\eta_z}_{\bar{V}_{in}} - \underbrace{\eta_x^2\bar{V}_{in} + \eta_y^2\bar{V}_{in} + \eta_z^2\bar{V}_{in}}_{\bar{V}_{in}}$$

By knowing contravariant velocity of inner cells on $\eta$ constant surface, $\bar{V}_{in} = w_{in}^2\eta_x + w_{in}^3\eta_y + w_{in}^4\eta_z$ ghost cell interpolations can be applied by averaging the ghost and inner scalar cell values such that $\bar{w}_{face} = \frac{\bar{w}_{in}+\bar{w}_{ghost}}{2}$, we can write ghost cell values finally

$$\begin{aligned} w_{ghost}^1 &= w_{in}^1 \\ w_{ghost}^2 &= w_{in}^2 - 2\bar{V}_{in}\eta_x \\ w_{ghost}^3 &= w_{in}^3 - 2\bar{V}_{in}\eta_y \\ w_{ghost}^4 &= w_{in}^4 - 2\bar{V}_{in}\eta_z \\ w_{ghost}^5 &= w_{in}^5 \end{aligned} \tag{2.39}$$

### 2.6.3 Far-field BCs

On far-field, Riemann invariants are used [49]. The velocity normal to boundary and speed of sound are obtained from incoming $R^-$ and outgoing $R^+$ characteristic waves. The pressure $p$ and density $\rho$ are calculated from the entropy $s$ and speed of sound $a$.

The Riemann invariants are given by

$$R^+ = U_{in} + \frac{2a_{in}}{\gamma - 1}, \quad R^- = U_0 + \frac{2a_0}{\gamma - 1} \tag{2.40}$$

where

$$U_{in} = \vec{U}_{in} \cdot \hat{n}, \quad \vec{U}_{in} = [u_{in}, v_{in}, w_{in}]^T, \quad a_{in} = \sqrt{\frac{\gamma p_{in}}{\rho_{in}}} \tag{2.41}$$

$$U_0 = \vec{U}_0 \cdot \hat{n}, \quad \vec{U}_0 = [u_0, v_0, w_0]^T, \quad a_0 = \sqrt{\frac{\gamma p_0}{\rho_0}} \tag{2.42}$$

In Equations (2.40) and (2.42) subscript zeros indicate free-stream values. $\hat{n}$ is normalized vector operator ie. on $\xi$ constant surface

$$\hat{n} = \left\langle \frac{\xi_x}{|\nabla\xi|}, \frac{\xi_y}{|\nabla\xi|}, \frac{\xi_z}{|\nabla\xi|} \right\rangle \tag{2.43}$$

The velocity $U_b$ and speed of sound $a_b$ at the boundary are linear combination of invariants such that

$$U_b = \frac{1}{2}(R^+ + R^-) \tag{2.44}$$

$$a_b = \frac{\gamma - 1}{4}(R^+ - R^-) \tag{2.45}$$

The cartesian velocity components and entropy at the ghost cells are calculated according to

$$\vec{U}_b = \begin{cases} \vec{U}_{in} + (U_b - U_{in}) \cdot \hat{n}, & \text{if } U_b > 0 \text{ (outflow)} \\ \vec{U}_0 + (U_b - U_0) \cdot \hat{n}, & \text{if } U_b \leq 0 \text{ (inflow)} \end{cases} \tag{2.46}$$

$$s_b = \begin{cases} \frac{a_{in}^2}{\gamma \rho_{in}^{\gamma-1}} & \text{if } U_b > 0 \text{ (outflow)} \\ \frac{a_0^2}{\gamma \rho_0^{\gamma-1}} & \text{if } U_b \leq 0 \text{ (inflow)} \end{cases} \tag{2.47}$$

The density and pressure at ghost cells can be extrapolated as

$$\rho_b = \left[ \frac{a_b^2}{\gamma s_b} \right]^{\frac{1}{\gamma-1}} \tag{2.48}$$

$$p_b = \frac{\rho_b (a_b)^2}{\gamma} \tag{2.49}$$

Then finally conservative variables can easily be obtained.

## 2.7 Analytical Jacobian Evaluation

In this study analytical Jacobian is used as preconditioner in flow solution. In addition, its transpose is used to create Adjoint equation. Therefore the

accuracy of it evaluation is very important. In this section, analytical derivation of Jacobian will be presented for first-order upwind-biased scheme. The detailed derivation and its comparison with numerical Jacobian evaluation can be found in Eyi's study [50].

If Equation (2.19) is substituted into Equation (2.18), the discretized residual vector at cell $(i, j, k)$ becomes

$$\begin{aligned}\hat{R}(\hat{Q}) =&\left[\hat{F}^+(\hat{Q}^L_{i+\frac{1}{2},j,k}) + \hat{F}^-(\hat{Q}^R_{i+\frac{1}{2},j,k})\right] - \left[\hat{F}^+(\hat{Q}^L_{i-\frac{1}{2},j,k}) + \hat{F}^-(\hat{Q}^R_{i-\frac{1}{2},j,k})\right] \\ &+\left[\hat{G}^+(\hat{Q}^L_{i,j+\frac{1}{2},k}) + \hat{G}^-(\hat{Q}^R_{i,j+\frac{1}{2},k})\right] - \left[\hat{G}^+(\hat{Q}^L_{i,j-\frac{1}{2},k}) + \hat{G}^-(\hat{Q}^R_{i,j-\frac{1}{2},k})\right] \\ &+\left[\hat{H}^+(\hat{Q}^L_{i,j,k+\frac{1}{2}}) + \hat{H}^-(\hat{Q}^R_{i,j,k+\frac{1}{2}})\right] - \left[\hat{H}^+(\hat{Q}^L_{i,j,k-\frac{1}{2}}) + \hat{H}^-(\hat{Q}^R_{i,j,k-\frac{1}{2}})\right]\end{aligned} \tag{2.50}$$

By taking derivatives of $\hat{R}_{i,j,k}$ with respect to $\hat{Q}_{m,l}$

$$\begin{aligned}\frac{\partial \hat{R}_{ir,jr,kr}}{\partial \hat{Q}_{iq,jq,kq}} =&\hat{A}^+_{ir+1/2,jr,kr}\frac{\partial \hat{Q}^L_{ir+1/2,jr,kr}}{\partial \hat{Q}_{iq,jq,kq}} + \hat{A}^-_{ir+1/2,jr,kr}\frac{\partial \hat{Q}^R_{ir+1/2,jr,kr}}{\partial \hat{Q}_{iq,jq,kq}} \\ &-\hat{A}^+_{ir-1/2,jr,kr}\frac{\partial \hat{Q}^L_{ir-1/2,jr,kr}}{\partial \hat{Q}_{iq,jq,kq}} - \hat{A}^-_{ir-1/2,jr,kr}\frac{\partial \hat{Q}^R_{ir-1/2,jr,kr}}{\partial \hat{Q}_{iq,jq,kq}} \\ &+\hat{B}^+_{ir,jr+1/2,kr}\frac{\partial \hat{Q}^L_{ir,jr+1/2,kr}}{\partial \hat{Q}_{iq,jq,kq}} + \hat{B}^-_{ir,jr+1/2,kr}\frac{\partial \hat{Q}^R_{ir,jr+1/2,kr}}{\partial \hat{Q}_{iq,jq,kq}} \\ &-\hat{B}^+_{ir,jr-1/2,kr}\frac{\partial \hat{Q}^L_{ir,jr-1/2,kr}}{\partial \hat{Q}_{iq,jq,kq}} - \hat{B}^-_{ir,jr-1/2,kr}\frac{\partial \hat{Q}^R_{ir,jr-1/2,kr}}{\partial \hat{Q}_{iq,jq,kq}} \\ &+\hat{C}^+_{ir,jr,kr+1/2}\frac{\partial \hat{Q}^L_{ir,jr,kr+1/2}}{\partial \hat{Q}_{iq,jq,kq}} + \hat{C}^-_{ir,jr,kr+1/2}\frac{\partial \hat{Q}^R_{ir,jr,kr+1/2}}{\partial \hat{Q}_{iq,jq,kq}} \\ &-\hat{C}^+_{ir,jr,kr-1/2}\frac{\partial \hat{Q}^L_{ir,jr,kr-1/2}}{\partial \hat{Q}_{iq,jq,kq}} - \hat{C}^-_{ir,jr,kr-1/2}\frac{\partial \hat{Q}^R_{ir,jr,kr-1/2}}{\partial \hat{Q}_{iq,jq,kq}}\end{aligned} \tag{2.51}$$

where

$$\begin{aligned}\hat{A}^+ &= \frac{\partial \hat{F}^+}{\partial \hat{Q}^L}, \quad \hat{A}^- = \frac{\partial \hat{F}^-}{\partial \hat{Q}^R}, \\ \hat{B}^+ &= \frac{\partial \hat{G}^+}{\partial \hat{Q}^L}, \quad \hat{B}^- = \frac{\partial \hat{G}^-}{\partial \hat{Q}^R}, \\ \hat{C}^+ &= \frac{\partial \hat{H}^+}{\partial \hat{Q}^L}, \quad \hat{C}^- = \frac{\partial \hat{H}^-}{\partial \hat{Q}^R}\end{aligned} \tag{2.52}$$

In Equation (2.52) $A, B$ and $C$ corresponds to blocks of $(5 \times 5)$ derivatives of split fluxes with respect to flow variables at cell faces. The derivation of split fluxes are very straight-forward. In this study all three splitting methods are

derived. In addition to these blocks, interpolated residual vectors at cell faces with respect to flow variables have to be derived. This is actually not a derivation but simplification in Equation (2.51).

For the first order discretization scheme, remembering Equation (2.20)

$$\begin{aligned}
\hat{Q}^L_{i+\frac{1}{2},j,k} &= \hat{Q}_{i,j,k} \text{ and } \hat{Q}^R_{i+\frac{1}{2},j,k} = \hat{Q}_{i+1,j,k} \\
\hat{Q}^L_{i,j+\frac{1}{2},k} &= \hat{Q}_{i,j,k} \text{ and } \hat{Q}^R_{i,j+\frac{1}{2},k} = \hat{Q}_{i,j+1,k} \\
\hat{Q}^L_{i,j,k+\frac{1}{2}} &= \hat{Q}_{i,j,k} \text{ and } \hat{Q}^R_{i,j,k+\frac{1}{2}} = \hat{Q}_{i,j,k+1} \\
\hat{Q}^L_{i-\frac{1}{2},j,k} &= \hat{Q}_{i-1,j,k} \text{ and } \hat{Q}^R_{i-\frac{1}{2},j,k} = \hat{Q}_{i,j,k} \\
\hat{Q}^L_{i,j-\frac{1}{2},k} &= \hat{Q}_{i,j-1,k} \text{ and } \hat{Q}^R_{i,j+\frac{1}{2},k} = \hat{Q}_{i,j,k} \\
\hat{Q}^L_{i,j,k-\frac{1}{2}} &= \hat{Q}_{i,j,k-1} \text{ and } \hat{Q}^R_{i,j,k-\frac{1}{2}} = \hat{Q}_{i,j,k}
\end{aligned} \tag{2.53}$$

Residual vector is function of $\hat{R}_{i,j,k}(\hat{Q}_{i-1,j,k}, Q_{i,j,k}, Q_{i+1,j,k})$ in first-order discretization. Substituting Equation (2.53) into Equation (2.51), first-order Jacobian is obtained for $\hat{R}_{i,j,k}$.

$$\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k}} = \hat{A}^+_{i+\frac{1}{2},j,k} - \hat{A}^-_{i-\frac{1}{2},j,k} + \hat{B}^+_{i,j+\frac{1}{2},k} - \hat{B}^-_{i,j-\frac{1}{2}/2,k} + \hat{C}^+_{i,j,k+\frac{1}{2}} - \hat{C}^-_{i,j,k-\frac{1}{2}} \tag{2.54}$$

$$\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i+1,j,k}} = \hat{A}^-_{i+\frac{1}{2},j,k}, \quad \frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j+1,k}} = \hat{B}^-_{i,j+\frac{1}{2},k}, \quad \frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k+1}} = \hat{C}^-_{i,j,k+\frac{1}{2}} \tag{2.55}$$

$$\frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i-1,j,k}} = \hat{-}A^+_{i-\frac{1}{2},j,k}, \quad \frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j-1,k}} = \hat{-}B^+_{i,j-\frac{1}{2},k}, \quad \frac{\partial \hat{R}_{i,j,k}}{\partial \hat{Q}_{i,j,k-1}} = \hat{-}C^+_{i,j,k-\frac{1}{2}} \tag{2.56}$$

When the issue is higher-order discretization, the situation is more complex. MUSCL module must also be differentiated. The use of differentiable limiter such that Van Albada or Venkatakrishnan makes the differentiation less complex and more accurate. In here, higher-order evaluation will not be mentioned anymore and one can refer to [50] for details.

# CHAPTER 3

# SOLUTION STRATEGY

In the solution of Euler equations, the resulting discretized governing equations are composed of nonlinear system of ordinary differential equations. Although the problem is highly time dependent, the great interest in aerodynamics is finding the steady state solution and this can be succeed by neglecting time derivatives and solving the problem. In this study, Newton Method is used to solve steady conservative form of Euler Equations with a Krylov Subspace methods which will be mentioned later.

Newton Method is an implicit, extremely powerful numerical strategy which guarantees quadratic convergence as the solution converges. Moreover, it was being used in CFD from late 80's starting with Venkatakrishnan [36] studies. However this method requires the explicit form of Jacobian Matrix which its entries are the derivatives of residual vector with respect to the flow variables vector and it is called as flow Jacobian. Dimension of Jacobian matrix can be extremely large for even coarse grids in aerodynamics problems and its dimension $n$ depends on grid size and number flow variables. Since it is the matrix, total size is squared $O(n^2)$ if it is dense. Fortunately most of the problems occurs in computational sciences do not end up with dense matrices and they are generally sparse and they have well structured patterns. Even though sparsity has no formal definition it can be designated the matrix which most of its entries are zero. Therefore, storage needed for this matrix is no more $O(n^2)$, but $O(nz)$ where $nz$ is the number of nonzero entries. Sparse flow Jacobian matrix must be created and solved as linear system in each iteration of Newton Method.

Linear system formed in Newton iterations can be solved by direct or iterative methods. Direct solution of these systems is based on some factorizations (LU, SVD, QR, LLT etc.) and its backward solutions. Today many packages and libraries of direct sparse solvers are available for both academic and commercial purposes such as PARDISO [51], UMFPACK [52], SuperLU [53] etc. They generally use similar techniques and show close performances. However direct solvers have crucial bottleneck which is increased memory requirement due to factorization despite its robustness. Therefore, iterative solvers have been gained popularity in that manner.

As an iterative solver Generalized Minimum Residual (GMRES) method is used in this study. GMRES is Krylov subspace technique which is suitable for solution of general non-symmetric sparse matrices and it was first studied by Yousef Saad [38]. However solving the linear system which is arisen from Newton Method is still costly even with the help of robust iterative schemes. Since in every Newton step flow Jacobian must be calculated. This increases both computational time and memory requirements. In order to avoid these difficulties, every Newton step is solved approximately and the new method is called inexact Newton method. A Newton-iterative method is named in this context, in every step approximate solution can obtained by different iterative algorithms such as Newton-SOR, Newton-CG or Newton-GMRES which is agreed name convention by Ortega and Rheinboldt [54]. Beauty of Newton-GMRES is that it requires only matrix-vector products but not Jacobian matrix so it does not need to be created explicitly.

Until now, motivation behind the use of Newton-GMRES method was briefly discussed. In following pages, detailed formulation of Newton-GMRES method will be presented with some improvements on general algorithm such as preconditioning, preconditioner matrix choice and reordering.

## 3.1 Newton Method

The system of non-linear discretized governing equations can be written in the form;

$$\hat{R}(\hat{Q}) = 0 \tag{3.1}$$

where $\hat{Q}$ is the flow variable vector and $\hat{R}$ is the residual vector and defined as

$$\hat{R}(\hat{Q}) = \frac{\partial \hat{F}(\hat{Q})}{\partial \xi} + \frac{\partial \hat{G}(\hat{Q})}{\partial \eta} + \frac{\partial \hat{H}(\hat{Q})}{\partial \zeta} \tag{3.2}$$

Using first-order Taylor series expansion at iteration $n$

$$\hat{R}^{n+1}(\hat{Q}) = \hat{R}^{n}(\hat{Q}) + \left(\frac{\partial \hat{R}}{\partial \hat{Q}}\right)^{n} \hat{\Delta} Q^{n} \tag{3.3}$$

where $\frac{\partial \hat{R}}{\partial \hat{Q}}$ is Jacobian matrix. As solution converges, the equality $\hat{R}^{n+1}(\hat{Q}) = 0$ is satisfied. This makes Newton Method as follows;

$$\frac{\partial \hat{R}}{\partial \hat{Q}}^{n} \Delta \hat{Q}^{n} = -\hat{R}(\hat{Q}^{n}) \tag{3.4}$$

In Equation (3.4) solving for flow variables $\Delta \hat{Q}$ at every iteration and updating, new values are obtained

$$\Delta \hat{Q}^{n+1} = \hat{Q}^{n} + \Delta \hat{Q}^{n} \tag{3.5}$$

## 3.2 Generalized Minimum Residual (GMRES)

GMRES is one of the projection methods in special group namely Krylov subspace methods. Roughly speaking, projection techniques search an approximate solution of the problem from a subspace. Typically solution is searched in $m$-dimensional subspace $K$ with $m$ constraint subspace $L$ which is generally linearly independent to each other, or selected in a different way.

In order to explain Krylov subspace methods, we will follow the Saad's notation [55]. Let us consider a matrix $A \in \mathbb{R}^{nxn}$ and vector $f \in \mathbb{R}^{n}$, the linear system can be written as

$$Ax \;=\; f \tag{3.6}$$

Above equation is simple representation of linear system where $A$ is the coefficient matrix, $f$ is the right hand side vector and $x$ is the solution vector. One can define the residual as

$$r = f - Ax \tag{3.7}$$

Krylov subspace solvers require $m$ dimensional orthogonal basis for this subspace while $m \leq n$. Orthogonalization is done through Arnoldi process for general matrices. Krylov subspace generated by $nxn$ matrix A, $n$-vector $v$ is the subspace spanned by the vectors of Krylov sequence

$$K_m = span\{v, Av, A^2v, ..., A^{m-1}v\} \tag{3.8}$$

The projection method searches an approximate solution $x_m$ from affine subspace $x_0 + K_m$ by imposing constraint condition $f - Ax_m \perp L_m$ where $L_m$ is the constraint subspace of dimension $m$, $x_0$ is initial guess. For the Krylov subspace methods GMRES and Full Orthogonalization Method (FOM) $K_m = K_m(A, r_0)$ and $r_0 = f - Ax_0$. Krylov subspace can be written then

$$K_m = span\{r_0, Ar_0, A^2r_0, ..., A^{m-1}r_0\} \tag{3.9}$$

Krylov subspace methods vary in choice of $L_m$. For two well known method selection of constraint subspace as follows;

$$L_m = K_m \text{ for Full Orthogonalization Method (FOM)} \tag{3.10}$$

$$L_m = AK_m \text{ for GMRES or MINRES} \tag{3.11}$$

In an orthogonal projection, the subspace $L$ is same as $K$ while in oblique projection $L$ is different than $K$. This explains where the FOM name is come from.

To sum up GMRES is a Krylov subspace method in order to solve general non-symmetric linear systems. Algorithm generates $x_k = x_0 + K_m(A, r_0)$ where solution subspace is $K_m = span\{r_0, Ar_0, A^2r_0, ..., A^{m-1}r_0\}$ and $L = AK$, $r_k \perp L$, $r_k = f - Ax_k$ then the solution becomes $x_m = x_0 + V_m y_m$. The algorithm for GMRES is given below as in Saad.

**Algorithm 1** Generalized Minimum Residual (GMRES)

1: $A :=$ system matrix; $f :=$ right hand side
2: $x_0 :=$ initial guess
3: $r_0 = f - Ax_0$
4: $\beta = \|r_0\|_2, v_1 = r_0/\beta$
5: **for** j=1,2,..,m **do**
6: $\quad w_j = Av_j$
7: $\quad$ **for** i=1,2,..,j **do**
8: $\quad\quad h_{ij} = w_j^T v_i$
9: $\quad\quad w_j = w_j - h_{ij}v_i$
10: $\quad$ **end for**
11: $\quad h_{j+1,j} = \|w_j\|_2$
12: $\quad v_{j+1} = w_j/h_{j+1,j}$
13: **end for**
14: Solve minimization problem LLS; $J(y_m) = \|\beta e_1 - \bar{H}_m y_m\|_2$
15: update $x_m = x_0 + V_m y_m$

In Algorithm (1) step 5-13 describes the Arnoldi Orthogonalization by Modified Gram-Schmidt(MGS) method. Arnoldi's method is used to create orthogonal basis of the Krylov subspace $K_m$. At step 9 all created $w_j$ vectors are orthogonal to all previous Arnoldi vector $v_j$'s and $v_j$'s are orthonormal to each other which satisfies equality $\forall i, j :< v_i, v_j >= \delta_{ij}$. In theory, Arnoldi's method creates perfect orthogonal basis to subspace. However in practice due to roundoff errors this is not a case and further improvements can be applied to increase accuracy and efficiency of this procedure such as Householder Arnoldi [56] etc. After the Arnoldi process, overdetermined system can be solved as Linear Least Square(LLS) problem. At step 14, $y_m$ vector which minimizes $J(y_m)$ is found by QR factorization with the help of Givens Rotation. Then last step corresponds to solution update in Krylov subspace.

### 3.3 Newton-GMRES

As mentioned before Newton-GMRES is under the class of Inexact Newton [57] methods. Example of full Newton step was shown in Equation (3.4). However the solution of this full step can be very costly even it is not necessary. Therefore, $\eta_k$ parameter is defined and Newton equation at $k^{th}$ iteration can be solved in reasonable accuracy as shown below.

$$\|\hat{R}'(\hat{Q}_k)\Delta\hat{Q}_k + \hat{R}(\hat{Q}_k)\|_2 \leq \eta_k\|\hat{R}(\hat{Q}_k)\|_2 \tag{3.12}$$

In Equation (3.12) $\eta_k$ is scalar parameter which is $\in [0, 1)$. This means the solution is approximated until residual is dropped its $\eta_k$. We call this parameter *forcing term* as in Eisenstat's & Walker's study [58]. Choice of $\eta_k$ is very important on Newton-GMRES since it is the only parameter that affects performance of algorithm. If forcing term is taken zero then solving Equation (3.12) is simply taking full Newton step as in Equation (3.4);

$$\frac{\partial\hat{R}}{\partial\hat{Q}}^k \Delta\hat{Q}^k = -\hat{R}(\hat{Q}^k) \tag{3.13}$$

Therefore, too small forcing terms must be avoided to get full performance from Inexact methods. There are some well known facts about forcing term selection. While $\eta_k$ increases the risk of divergence will be reduced, however, convergence rate will be very slow due to oversolving the system which is defined in [59]. With a good choice of forcing parameter superlinear or even quadratic convergence can be obtained. It is important to choose right parameter in dynamic fashion if it is possible. There are many studies on this starting with an Eisenstat [58] formula. Moreover, one can find more profound discussion and two more choices of $\eta_k$ in Pernice & Walker [60] studies especially to avoid oversolving. Another way of choosing forcing term is in constant manner as in Pueyo's and Zingg's studies [61]. In our study forcing term is chosen in a constant manner by setting some experiments.

One of the most important feature of Newton-GMRES algorithm is that it is totally matrix free. In other words there is no need to generate Jacobian matrix. By knowing GMRES algorithm requires only matrix-vector products $Av$

in Algorithm (1), Jacobian-Free version appears. The detailed derivation of Jacobian-Free matrix-vector product can be found in Knoll and Keyes [62] work. In this study matrix-vector products are approximated as follows by sticking the notation given in Section 3.1.

$$\hat{R}'(\hat{Q})v = \frac{R(\hat{Q} + \epsilon v) - R(\hat{Q})}{\epsilon} \tag{3.14}$$

where $\hat{R}'(\hat{Q})$ is simply matrix $A$. In Equation (3.14) $\epsilon$ is scalar value which is used to perturb flow variables $\hat{Q}$ for finite difference approximation. Perturbation factor $\epsilon$ can be chosen in different manners. In Equation (3.14) 1st-order forward differencing is used, this means the local truncation error is linearly proportional to the step size $\epsilon$. Therefore, greater accuracy in calculation of $\hat{R}(\hat{Q})$ can be obtained by decreasing the step size. However too small values can cause to round-off errors that can reduce accuracy also. Different choices can be found in reference [62]. In addition, Onur and Eyi [63] have found the optimum perturbation epsilon for Euler Equations as square root of machine precision st. $\epsilon = 4 \times 10^{-8}$. In this study stepsize is simply taken this value for all finite difference cases. Then Newton-GMRES algorithm at $k^{th}$ iteration can be reconstructed as follows.

**Algorithm 2** Newton-GMRES Algorithm

1: $x_k^0$ := initial guess for solution
2: $s_k^0$ := initial guess for Newton update, choose $\eta_k$
3: $r_k^0 = -F'(x_k)s_k^0 - F(x_k)$, $\beta_k = \|r_k^0\|_2$, $v_1 = r_k^0/\beta_k$
4: **While** $\|r_k^0\|_2 > \eta_k \|F(x_k)\|_2$ **do**
5: $m = m + 1$
6: Calculate $Av_m = F'(x_k)v_m$ as in Equation (3.14); $w_m = Av_m$
7: Create Upper-Hessenberg matrix $h_{i,m} = (w_m^T, v_i), \quad \forall i = 1, 2, ..., m$
8: Orthogonalization $\hat{v}_{m+1} = w_m - \sum_{i=1}^{m} h_{i,m} v_i$
9: $h_{m+1,m} = \|\hat{v}_{m+1}\|_2$
10: $v_{m+1} = \hat{v}_{m+1}/h_{m+1,m}$
11: Solve the LLS problem by Givens Rotation $J(y_m) = \min_{y_m} \|\beta_k e_1 - \bar{H}_m y_m\|_2$
12: where $e_1 = [1, ..., 0]_{m+1}$, $y_m = [\ ]_m$ and $\bar{H}_m = [\ ]_{(m+1)\times m}$
13: Update solution $x_k = x_0 + V_m y_m$ where $V_m = [v_1, v_2, ..., v_m]_{m\times m}$

## 3.4 Preconditioning

Since the convergence behavior is strictly related to condition number of the linear system, preconditioning is very important in iterative schemes. Moreover, in aerodynamic problems, the conditioning of system which is arisen from discretization of governing equations are generally poor. Before starting the preconditioning, let us clarify the definition of condition number. Conditioning of system is related to its eigenvalue distribution. If most of the eigenvalues are clustered around 1, it can be called as well conditioned system. The formulation of condition number is given below.

$$\kappa(A) = \frac{|\lambda_{max}(A)|}{|\lambda_{min}(A)|} \tag{3.15}$$

Equation (3.15) shows that condition number, $\kappa(A)$, depends on ratio of absolute values of maximum and minimum eigenvalues of matrix $A$ where the system is defined as $Ax = f$. If $A$ is identity matrix then all eigenvalues of $A$ is one, hence condition number equals to 1. This kind of system is most favorable to solve and all iterative schemes can solve this linear system in one iteration without any effort. Therefore, in preconditioning one seeks the preconditioner matrix that resultant linear system is close identity matrix. This can be handled by choosing preconditioner matrix such that $M \approx A$. Here is the simple representation of left preconditioning.

$$A\,x = f \tag{3.16}$$

$$\underbrace{(M^{-1} * A)}_{\hat{A}}\; x = \underbrace{(M^{-1} * f)}_{\hat{f}} \tag{3.17}$$

$$\hat{A}x = \hat{f} \tag{3.18}$$

In Equation (3.17) $M^{-1}$ is left preconditioner, $\hat{A}$ is preconditioned system matrix, $\hat{f}$ is preconditioned right hand side. The solution of this system is same with unpreconditioned one. However if $M$ is chosen close to $A$ and $M$ is easily invertible, the solution of Equation (3.18) is easier than Equation (3.16). In next section we will explain the right, left and split preconditioners on GMRES method with their use.

### 3.4.1 Left Preconditioned GMRES

Left preconditioned GMRES algorithm can be defined as in Saad [55].

$$\underbrace{M^{-1}A}_{\hat{A}}\,x = \underbrace{M^{-1}f}_{\hat{f}} \tag{3.19}$$

$$\hat{A}x = \hat{f} \tag{3.20}$$

$$\hat{r_0} = \hat{f} - \hat{A}x \tag{3.21}$$

$$\hat{r_0} = M^{-1}(f - Ax) \tag{3.22}$$

$$\hat{r_0} = M^{-1}r_0 \quad \text{and} \quad \hat{x} = x \tag{3.23}$$

where $x \in x_0 + \mathbb{K}_k(M^{-1}A, M^{-1}r_0)$. As one can see the preconditioned residual is different than the original residual. All residual vectors must be preconditioned in order to algorithm works. In addition, some difficulties can be observed if stopping criteria is based on actual residual. Nevertheless, the solution obtained from this approach, $\hat{x}$, gives same solution with unpreconditioned one, $x$, and there is no need for extra calculations. The algorithm for left preconditioned GMRES is given below.

**Algorithm 3** Left Preconditioned GMRES

1: $A :=$ system matrix; $f :=$ right hand side
2: $x_0 :=$ initial guess
3: *Precondition*; $r_0 = M^{-1}(f - Ax_0)$
4: $\beta = \|r_0\|_2, v_1 = r_0/\beta$
5: **for** j=1,2,..,m **do**
6: *Precondition*; $w_j = M^{-1}Av_j$
7: **for** i=1,2,..,j **do**
8: $h_{ij} = w_j^T v_i$
9: $w_j = w_j - h_{ij}v_i$
10: **end for**
11: $h_{j+1,j} = \|w_j\|_2$
12: $v_{j+1} = w_j/h_{j+1,j}$
13: **end for**
14: Solve minimization problem LLS; $J(y_m) = \|\beta e_1 - \bar{H}_m y_m\|_2$
15: update $x_m = x_0 + V_m y_m$

Algorithm 3 is derived for GMRES. Newton-GMRES version is very similar to it. In same way initial residual vector must be preconditioned in Newton-GMRES also. And the Jacobian-vector products can be approximated similar to Equation (3.14) with preconditioner matrix as follows;

$$\text{For the system;} \quad \underbrace{M^{-1}A}_{\hat{A}} x = M^{-1}f \tag{3.24}$$

$$M^{-1}Av = M^{-1}\left[\frac{R(\hat{Q} + \epsilon M^{-1}v) - R(\hat{Q})}{\epsilon}\right] \tag{3.25}$$

### 3.4.2 Right Preconditioned GMRES

Applying very similar procedure in Section 3.4.1 right preconditioned GMRES can be derived as follows.

$$\underbrace{AM^{-1}}_{\hat{A}}\underbrace{Mx}_{\hat{x}} = f \tag{3.26}$$

$$\hat{A}\hat{x} = f \tag{3.27}$$

$$\hat{r_0} = f - \hat{A}\hat{x} \tag{3.28}$$

$$\hat{r_0} = f - Ax \tag{3.29}$$

$$\hat{r_0} = r_0 \quad \text{and} \quad \hat{x} = Mx \tag{3.30}$$

where $x \in x_0 + \mathbb{K}_k(AM^{-1}, r_0)$. Comparing to left preconditioned one, residual is not changed in this algorithm, but solution vector is different now. Since this version GMRES minimizes the original residual, right preconditioning is mostly preferred one. The algorithm is given below.

**Algorithm 4** Right Preconditioned GMRES

1: $A :=$ system matrix; $f :=$ right hand side
2: $x_0 :=$ initial guess
3: $r_0 = f - Ax_0$
4: $\beta = \|r_0\|_2, v_1 = r_0/\beta$
5: **for** j=1,2,..,m **do**
6: *Precondition*; $w_j = AM^{-1}v_j$
7: **for** i=1,2,..,j **do**
8: $h_{ij} = w_j^T v_i$
9: $w_j = w_j - h_{ij}v_i$
10: **end for**
11: $h_{j+1,j} = \|w_j\|_2$
12: $v_{j+1} = w_j/h_{j+1,j}$
13: **end for**
14: Solve minimization problem LLS; $J(y_m) = \|\beta e_1 - \bar{H}_m y_m\|_2$
15: *Find actual solution*; $z_m = M^{-1}y_m$
16: update $x_m = x_0 + V_m z_m$

Algorithm 4 is derived for GMRES. Newton-GMRES version is very similar to it. Jacobian-vector products can be approximated similar to Equation (3.14) with preconditioner matrix as follows;

$$\text{For the system;} \quad \underbrace{AM^{-1}}_{\hat{A}} Mx = f \tag{3.31}$$

$$AM^{-1}v = \left[\frac{R(\hat{Q} + \epsilon M^{-1}v) - R(\hat{Q})}{\epsilon}\right] \tag{3.32}$$

### 3.4.3 Left-Right Preconditioned GMRES

Left-Right precondition derivation is given below.

$$\underbrace{M_L^{-1}AM_R^{-1}}_{\hat{A}}\underbrace{M_R x}_{\hat{x}} = \underbrace{M_L^{-1}f}_{\hat{f}} \tag{3.33}$$

$$\hat{A}\hat{x} = \hat{f} \tag{3.34}$$

$$\hat{r_0} = \hat{f} - \hat{A}\hat{x} \tag{3.35}$$

$$\hat{r_0} = M_L^{-1}(f - Ax) \tag{3.36}$$

$$\hat{r_0} = M_L^{-1}r_0 \quad \text{and} \quad \hat{x} = M_R x \tag{3.37}$$

where $x \in x_0 + \mathbb{K}_k(M_L^{-1}AM_R^{-1}, M_L^{-1}r_0)$. Comparing to previous precondition algorithms both residual and solution is changed now. Then following algorithm appears.

**Algorithm 5** Left-Right Preconditioned GMRES

1: $A :=$ system matrix; $f :=$ right hand side
2: $x_0 :=$ initial guess
3: *Precondition*; $r_0 = M_L^{-1}(f - Ax_0)$
4: $\beta = \|r_0\|_2, v_1 = r_0/\beta$
5: **for** j=1,2,..,m **do**
6: *Precondition*; $w_j = M_L^{-1} A M_R^{-1} v_j$
7: **for** i=1,2,..,j **do**
8: $h_{ij} = w_j^T v_i$
9: $w_j = w_j - h_{ij} v_i$
10: **end for**
11: $h_{j+1,j} = \|w_j\|_2$
12: $v_{j+1} = w_j / h_{j+1,j}$
13: **end for**
14: Solve minimization problem LLS; $J(y_m) = \|\beta e_1 - \bar{H}_m y_m\|_2$
15: *Find actual solution*; $z_m = M_R^{-1} y_m$
16: update $x_m = x_0 + V_m z_m$

Similar to previous algorithms Jacobian-vector product is written such that;

$$\text{For the system;} \quad \underbrace{M_L^{-1} A M_R^{-1}}_{\hat{A}} M_R x = M_L^{-1} f \tag{3.38}$$

$$M_L^{-1} A M_R^{-1} v = M_L^{-1} \left[ \frac{R(\hat{Q} + \epsilon M_R^{-1} v) - R(\hat{Q})}{\epsilon} \right] \tag{3.39}$$

Until now, three different types of preconditioing technique are discussed. The choice depends on the iterative method, characteristics of problem etc. In general, the trend in residual minimizing methods such as GMRES is right preconditioning since the minimized residual is same with actual residual in right preconditioning. Therefore, in this study right preconditioning is chosen. However, the concept of left-right preconditioning will be useful in order to understand reordering algorithms and permutation matrix usage. This will be discussed later.

### 3.4.4 Incomplete Lower Upper (ILU) Preconditioning

Incomplete LU preconditioning consists of a lower-upper factorization of general sparse matrix $A$ with some extra constraints such that the residual matrix $R = LU - A$ satisfies certain criteria. For a typical sparse matrix, full LU factorization causes denser or less sparse matrix comparing to $A$. This is due to fill-in in factorization steps which is a drastic bottleneck of factorization or direct solvers as mentioned at the beginning of this chapter. Fortunately, preconditioning does not require strict factorizations and incomplete variations works great in practice. Before giving too much detail on incomplete factorizations, it will be better to explain generic factorizations. Let us consider a sparse matrix A such that;

$$A = \hat{L}\hat{U} \quad \text{where} \quad \hat{L} + \hat{U} \text{ is generally denser than A} \tag{3.40}$$

The density of matrix increases due to fill-ins resulting from factorization. The general sparse matrix can be shown in Figure 3.1. After factorization process, red circles in Figure 3.2 indicate the fill-ins which is stored in $\hat{L} + \hat{U}$.

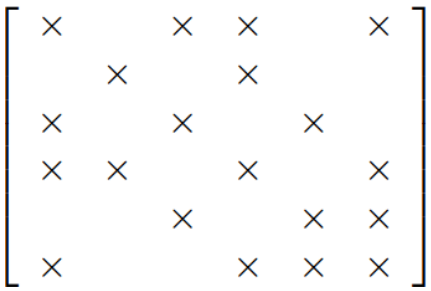


Figure 3.1: Before factorization

$A \Longrightarrow \hat{L}\hat{U}$

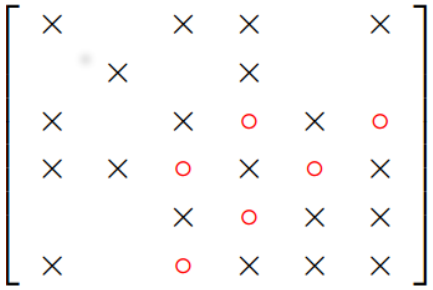


Figure 3.2: After factorization

Classical mathematical representation of LU factorization can be written as;

$$A = \hat{L}\hat{U} \quad \text{or} \quad A = \underbrace{(L + I)}_{\hat{L}}\underbrace{(U + D)}_{\hat{U}} \tag{3.41}$$

In Equation (3.41) $L$ is strictly lower triangular matrix, $U$ is strictly upper triangular matrix and $D$ is diagonal matrix. Some of other variations are;

- Decomposition st. :

$$A = (L + D)D^{-1}(U + D) \tag{3.42}$$

- LDU factorization :

$$A = (L + I)D(U + I) \tag{3.43}$$

- Cholesky factorization :

$$A = (L + D)(L + D)^T \text{valid only for symmetric positive define matrix A} \tag{3.44}$$

We focus on factorizations that correspond to Equation (3.41). Then the algorithm for LU and ILU factorization with Gaussian-Elimination strategy appears [64] as;

**Algorithm 6** LU Factorization-IKJ

1: **for** i=2,...,n **do**
2: **for** k=1,...,i-1 **do**
3: $a_{ik} := a_{ik}/a_{kk}$ while $a_{kk} \neq 0$
4: **for** j=k+1,...,n **do**
5: Compute $a_{ij} := a_{ij} - a_{ik} * a_{kj}$
6: **end for**
7: **end for**
8: **end for**

**Algorithm 7** ILU Factorization-IKJ

1: **for** i=2,...,n **do**
2: **for** k=1,...,i-1 and if $(i,k) \notin S$ **do**
3: $a_{ik} := a_{ik}/a_{kk}$ while $a_{kk} \neq 0$
4: **for** j=k+1,.,n and for $(i,j) \notin S$ **do**
5: Compute $a_{ij} := a_{ij} - a_{ik} * a_{kj}$
6: **end for**
7: **end for**
8: **end for**

In Algorithm 7, $S$ is zero pattern set such that;

$$S \subset \{(i,j) | i \neq j;\ 1 \leq i,j \leq n\} \tag{3.45}$$

General way of building an ILU factorization is derived by Gaussian elimination and dropping(discarding) elements in predetermined off-diagonal positions which is outside of $S$. This method is refered to ILU($p$) where $p$ is the level of fill-in. When $p = 0$ is chosen, ILU(0) refers to simplest factorization that guarantees sparsity pattern of $\hat{L} + \hat{U}$ is same with matrix $A$. In that case $S$ is chosen as $NZ(A)$ in Algorithm 7. One can also obtain more accurate factorizations by increasing level of fill-in such that ILU(1), which has sparsity pattern of $A^2$. This can be generalized as ILU($p$) preconditioner of a matrix $A$ has the sparsity pattern of matrix $A^{p+1}$.

Although the many variations exist for ILU factorization, the well known and used in this study is IKJ variant of SPARSEKIT [64] package of Saad. The algorithm for ILU($p$) is given below, the other variations including treshold and pivoting strategies can be found in [55].

**Algorithm 8** ILU(p) Factorization [64]

1: For all nonzero elements $a_{ij}$ define $lev(a_{ij}) = 0$
2: **for** i=2,...,n **do**
3:  **for** k=1,...,i-1 **do**
4:  **while** $lev(a_{ik}) \leq p$ **do**
5:  Compute $a_{ik} := a_{ik}/a_{kk}$
6:  Compute $a_{i*} := a_{i*} - a_{ik}a_{k*}$
7:  Update the levels of fill of the nonzero $a_{i,j}$'s using;
8:  $lev_{ij} = min\{lev_{ij}, lev_{ik} + lev_{kj} + 1\}$
9:  **end while**
10:  **end for**
11:  Replace any element in row $i$ with $lev(a_{ij}) > p$ by zero
12: **end for**

There is no best way of chosing right precodnitioner for a single problem. Every problem has its own best preconditioner matrix and technique. Therefore, the author of any reference can only guide to choose better preconditioner. Nevertheless, there is guideline to choose preconditioner in incomplete factorizations. Generally, the first strategy is choice of ILU(0) where no fill-in is applied. This preconditioner requires less storage and it is computationally efficient. However because of discarding all fill-ins resulting from matrix $A$ is not much accurate. This accuracy can cause slow convergence or even worse condition number. Hence the second approach is selection of ILU(p) preconditioner which allows some level of fill-in with a limit parameter $p$. However, the dropping strategy of a fill-in entry is independent of its numerical value [65]. Because of this, the large elements discarded in dropping can cause inaccurate factorizations. Therefore, ILU($p, \tau$) is developed by Saad with the following algorithm [55];

**Algorithm 9** ILU($p, \tau$) Factorization

1: **for** i=1,...,n **do**
2: $w := a_{i*}$
3: **for** k=1,...,i-1 and when $w_k \neq 0$ do : **do**
4: $w_k := w_k / a_{kk}$
5: Apply a dropping rule to $w_k$
6: **If** $w_k \neq 0$ **then** $w := w - w_k * u_{k*}$
7: **end for**
8: Apply a dropping rule to row $w$
9: $l_{i,j} := w_j$ for $j = 1, ..., i-1$
10: $u_{i,j} := w_j$ for $j = 1, ..., n$; $w := 0$
11: **end for**

The dropping rule for ILU($p, \tau$) is given by [66];

- The first dropping is applied at line 5 according to value of element $w_k$. In that case $w_k$ is dropped if $w_k < \|a_{i*}\|_2 * \tau$.
- In second dropping at line 10, same dropping rule is applied again. In addition $p$ largest element of L and U matrices are conserved in each row with diagonal entries.

Until now three basic strategy in ILU factorizations were mentioned. They are useful and easily applicable in most of the problems without going too much deep into. However, preconditioning is state of art which has also some drawbacks especially when the problem sets on un-symmetric matrices. The issue is existence and stability of the selected preconditioner [67]. Complete LU factorization of a matrix $A$ exists if and only if all principal sub-matrices of $A$ are nonsingular, and for this non-singular matrix $A$, there must be a permutation $P$ such that $PA$ admits the LU factorization. Furthermore, incomplete factorizations can fail due to occurrence of zero or small pivots regardless of singularity of matrix $A$. This is called as breakdown in literature. Instabilities in preconditioner matrix can be avoided by preprocessing the the coefficient matrix by reordering or

permuting. In next section some basic ordering techniques and their usage will be discussed.

## 3.5 Reordering

Reordering is highly useful technique which can increase the stability and reduce the fill-ins in factorization. Or even it can increase the rate of convergence. Nevertheless, the main concern in the use of reordering is to decrease fill-ins and avoid zero pivots. In our problem, no zero pivots are observed during the factorization due to discretization of governing equations. However, if boundary conditions are added to system matrix, then some zeros can appear at the diagonal, however, this is not a case in our study. Let us consider the system $Ax = b$ and if we apply P and Q permutations to this system we get;

$$Ax = b \tag{3.46}$$

$$PAQy = Pb, \quad x = Qy \tag{3.47}$$

In Equation (3.47) it is desired that $PAQ$ has better stability and less prone to fill-ins. If $A$ is structurally symmetric, it is generally better to preserve symmetry in order to keep stability and diagonal dominance of problem. Hence $Q = P^T$ is chosen. Then $PAP^T$ is called symmetric permutation of $A$ and the symmetry of system is preserved. If $A$ is not structurally symmetric than the structure may be symmetrized by $A + A^T$ or $A * A^T$ to find symmetric permutation $P$ that is applied again to the unsymmetric version of $A$. Typical choice is $A + A^T$. In this study because of the unsymmetry in the system, permutation $P$ is found from the structure of $A + A^T$. Permuting rows and columns of symmetric matrix is simply renumbering the vertices of graph.

### 3.5.1 Graph Theory

All reordering algorithms are related to graph theory. Before going into detail on reordering, graph theory should be mentioned. We will follow the notation given in [68]. We will consider a symmetric-square matrix $A \in \mathbb{R}^{nxn}$. Graph is

simply represented by $V$ and $E$ such that;

$$G = \langle V, E \rangle \tag{3.48}$$

where $V$ consists of a set of $n$ *vertices* or *nodes* such that;

$$V = \{v_1, v_2, ..., v_n\} \tag{3.49}$$

and $E$ specifies the set of unordered pairs of distinct elements namely *edges* or *links* given below

$$E = \{\{v_i, v_j\} \in E,\ \forall\, i \neq j,\ a_{ij} \neq 0\} \tag{3.50}$$

In Equation (3.50), $E$ is formed by adjacent vertices ie. $\{v_1, v_2\} \in E$ then $v_1$ and $v_2$ are said to be adjacent to each other and they are connected by an edge. For this situation, the notation $Adj_G(v_i)$ is used and it refers to set of vertices that adjacent to $v_i$ or simply share edges in given graph $G$. The *degree of vertex* $v_i$ is denoted by $degree_G(v)$ and shows the number of "edge ends" at given vertex [69]. Mathematically it equals to

$$degree_G(v) = |Adj_G(v)| \tag{3.51}$$

Another useful definition is *numbering*, $f$ which is defined as one-to-one map, $f$, from $V(G)$ onto the set $\{1, 2, ..., n\}$. In addition, the bandwidth concept is highly used in many ordering algorithms. The *bandwidth of graph* $G$, is the minimum matrix bandwidth, among all possible adjacency matrices of graph $G$. The *bandwidth of graph* $G$ *relative to numbering* $f$ is denoted $\beta_f(G)$;

$$\beta_f(G) = max\{|f(v_i) - f(v_j)| : \{v_i, v_j\} \in E\} \tag{3.52}$$

In terms of quantity, the minimum of $\beta_f(G)$ in Equation (3.52) among all numberings is called *bandwidth of* $G$, $\beta(G)$.

In graph theory, graphs can be grouped at two distinct categories, directed graphs (digraphs) and undirected graphs. In directed graphs edges are directed. For a finite set of vertices $V$ and set of ordered pairs $(a, b)$ where $(a, b) \in V$ are called edges. For diagraph $a$ is initial node and $b$ is terminal node. In general unsymmetric matrices are represented by directed graphs. In our study the matrix is symmetric, so the corresponding graph is undirected. Figure 3.3 shows the matrix structure at the left and corresponding undirected graph for $5 \times 5$ real example matrix.

$$A = \begin{bmatrix} x & x & x & & x \\ x & x & & x & x \\ x & & x & & \\ & x & & x & \\ x & x & & & x \end{bmatrix}$$

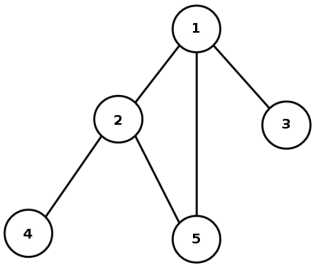



Figure 3.3: Matrix at left, corresponding graph at right

In above example $A \in \mathbb{R}^{5x5}$ and it is structurally symmetric. $x$'s indicates the non-zero entries. At right, the circles with numbers correspond to vertices and the lines between them are edges. Since lines have no directions graph is undirected. In general, for symmetric matrices, adjacency structure of corresponding graph is enough to exploit permutations of reordering. Adjacency structure can be stored in only 2 vectors st. for the system in Figure 3.3; $x_{adj} = [1, 4, 7, 8, 9, 11]$ and $adjncy = [2, 3, 5, 1, 4, 5, 1, 2, 1, 2]$. $adjncy$ stores the adjacency list in consecutive locations for each vertex with dimension "twice number of edges" and $x_{adj}$ is used to specify where it begins and where it ends with a dimension of "number of rows plus one".

In next sections, heuristics in ordering strategy will be discussed by giving the details of Minimum Degree(MD) and Reverse Cuthill-Mckee(RCM) algorithms.

### 3.5.2 Minimum Degree (MD) Ordering

Probably easiest way to explain Minimum Degree ordering is by the help of elimination graphs. At each step of Gaussian Elimination, select a node $v$ of smallest degree then eliminate including its incident edges. After eliminating node $v$, update graph by adding edges to make the nodes adjacent to $v$ into a clique. Process is repeated until the end [70]. Algorithm below is taken from [71] and it can be explained briefly by graph elimination theory.

For a given node $v$, $G^k$ represents the graph $G$ to be factorized after the first $k$ pivots have been chosen. $G^k$ depends on the selection of $k^{th}$ pivot and $G^{k-1}$. Algorithm starts with calculating of each vertex degree on $G^0$ from 4 to 5 as preprocess for remaining of algorithm. The node $v$ is selected from $V^{k-1}$ which

minimizes $t_v$. Throughout the 7-14, pivot node $v$ and its edges are eliminated from graph, $G^k$ and the edges cause to fill-in are added to $E^{k-1}$. Then algorithm appears;

**Algorithm 10** Minimum Degree (MD) Algorithm

1: For a given symmetric graph $G =< V^0, E^0 >$
2: $V^0 = \{1, .., n\}$
3: $E^0 = \{(i, j) : a_{ij} \neq 0 \text{ and } i \neq j\}$
4: **for** i=1,...,n **do**
5: $t_i = |Adj_{G^0}(i)|$
6: **end for**
7: **for** k=1,...,n **do**
8: Select the $k^{th}$ pivot vertex st. $v \in V^{k-1}$ that minimizes $t_v$
9: **for** each $i \in Adj_{G^{k-1}}(v)$ **do**
10: $Adj_{G^k}(i) = \{Adj_{G^{k-1}}(i) \cup Adj_{G^{k-1}}(v)\} - \{i, v\}$
11: $t_i = |Adj_{G^k}(i)|$
12: **end for**
13: $V^k = V^{k-1} - \{v\}$
14: **end for**

In general, MD ordering is used to reduce the number of fill-in which is caused by new edges. Algorithm minimizes the fill-in caused by $k^{th}$ pivot by minimizing the degree. However, there may be several nodes that have minimum degree. So the choice of root node is arbitrary in these situations.

### 3.5.3 Reverse Cuthill-Mckee (RCM) Ordering

The name of Cuthill-Mckee (CM) algorithm comes from the Elizabeth Cuthill and J. McKee that is first found for reduce the bandwidth of symmetric sparse matrix by renumbering the nodes. It is probably the most famous profile reduction algorithm. Then Alan George version [72] CM appears which is called as reverse Cuthill-Mckee (RCM) algorithm. It is simply found by reversing index numbering on original approach. The resultant RCM method is much superior

than its ancestor with less fill-in in factorizations.

Algorithm starts with the root node selection as in MD ordering. Then, it constructs the adjacency structure of graph by finding all un-numbered neighbors and exclude the vertices that is already in set. Adjacency set is sorted in ascending vertex degree. The procedure is very similar to breadth-first search. Finally, order of elements are reversed.

In RCM root node selection and tie-breaking strategies are also important for similar reasons in MD ordering. Starting to an algorithm with different nodes may create totally different graphs with different profiles.

Algorithm 11 is taken from reference [73]. For a given graph $G =< V, E >$, $Q$,$R$ and $S$ are data structures known as queues where $R$ contains the nodes in arbitrary order, $Q$ and $S$ are working queues. Adding element at the end of queue is shown $Q \leftarrow Q, v_i$, removing the first element of queue is denoted by $v_i \leftarrow Top(Q)$.

**Algorithm 11** Reverse Cuthill-Mckee (RCM) Algorithm

1: For a given matrix $A$, define graph $G =< V, E >$
2: Set $i = 1$
3: $Q \leftarrow \emptyset$; $R \leftarrow V(A)$
4: $v_1 \leftarrow Top(R)$
5: $S \leftarrow Adj(v_1) \cap R$
6: $R \leftarrow R - S$
7: **while** $S \neq \emptyset$ **do**
8: $\quad i \leftarrow i + 1$
9: $\quad v_i \leftarrow Top(S)$
10: $\quad Q \leftarrow Q, v_i$
11: **end while**
12: **If** $R = \emptyset$ **then** stop
13: $\{z\} \leftarrow Top(Q)$
14: $S \leftarrow Adj(z) \cap R$
15: $R \leftarrow R - S$; **go to** 7
16: Reverse the ordering of vertices.

### 3.5.4 Implementation of Reordering Algorithms

Application of reordering is very straightforward if the concept of right-left preconditioning is well understood in Section 3.4.3. Since the only application will be held by the permutation matrices obtained from reordering algorithms. In this study, Minimum Degree, Reverse Cuthill-Mckee, One Way Dissection, METIS [74] and Natural reordering strategies are used. For a given symmetric graph, reordering algorithms create a *permutation* matrix, $P$ which is stored in an array. Then simply taking its inverse will crate the *inverse permutation* matrix. One do not have to worry about taking inverse of matrix, since $P$ is *orthonormal* matrix ($P^{-1} = P^T$). The application of $PAP^T$ or $P^TAP$ is known as symmetric permutation of system matrix $A$. For this operator, both rows and columns of $A$ are reordered. For this study $PAP^T$ is applied to system matrix $A$ and $P^TMP$ is applied to preconditioner matrix, $M$. In order to prevent confusion on superscripts, we call $P = P_1$ and $P^T = P_2$. Then system for right preconditioned and reordered system equation can be written as follows.

$$\text{original system} \Rightarrow Ax = f \tag{3.53}$$

$$\text{after right precondition} \Rightarrow AM^{-1}Mx = f \tag{3.54}$$

$$\text{after reordering} \Rightarrow \underbrace{(P_1AP_2)(P_2MP_1)^{-1}}_{\hat{A}}\underbrace{(P_2MP_1)(P_2^{-1} * x)}_{\hat{x}} = \underbrace{P_1 * f}_{\hat{f}} \tag{3.55}$$

Jacobian-vector product is given

$$\hat{A}v = (P_1AP_2)(P_2MP_1)^{-1}v \tag{3.56}$$

$$= P_1\ [A]\ P_2(P_2MP_1)^{-1}\ [v] \tag{3.57}$$

$$\hat{A}v = P_1 * \left[\frac{R(\hat{Q} + \epsilon P_2(P_2MP_1)^{-1}v) - R(\hat{Q})}{\epsilon}\right] \tag{3.58}$$

In addition, residual vector must be reordered

$$r_0 = P_1 * (f - Ax) \tag{3.59}$$

And finally solution vector must be unpermuted as well

$$\hat{x} = (P_2MP_1)\underbrace{(P_2^{-1} * x)}_{y} \ \Rightarrow\ x = P_2 * y \tag{3.60}$$

### 3.6 Efficiency of flow solver

In this section, we will investigate the efficiency of flow solver by using different reordering and preconditioning algorithms. First of all, the results and figures below belong to medium grid $161 \times 29 \times 49$. Throughout the iterations, forcing term $\eta_k$ is kept constant. The selection of forcing parameter is important both for the convergence and robustness. It is decided by trial and error.

For the comparison of preconditioners, $ilu(0)$ and $ilu(k)$, $k = 1, .., 4$ are chosen. Because of the no scaling approach, threshold factorization can drop many elements that are important. In addition, it is hard to decide $\tau$ value in this case. Therefore, ILU$(p, \tau)$ is not used in this study. As reordering algorithms METIS, One Way Dissection (1WD), Reverse Cuthill-Mckee (RCM) and Minimum Degree (MD) are used. Full analytical Jacobian matrix is chosen as preconditioner matrix. Because of the simplicity and better condition, 1st-order Jacobian is used as preconditioner. Figure 3.4 shows non-zero pattern of the first-order Jacobian matrix which is also solved linear system by the help of matrix free approach. The 1st-order Jacobian has number of rows as $n_{row} = 1075200$ and number of non-zero as $nnz = 36956800$.

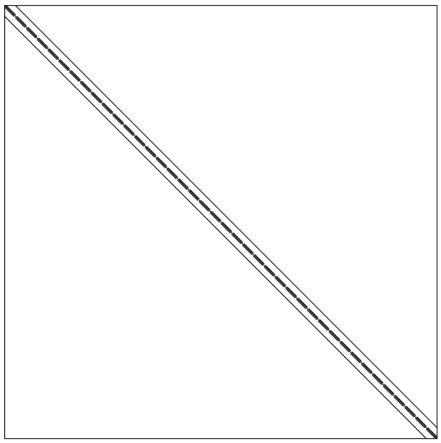


Figure 3.4: Linear system matrix for no reordering case

Second order Jacobian has few sub diagonals according to first-order scheme which increases the condition number of system dramatically from $cond(A) \approx$

$10^7$ to $cond(A) \approx 10^{15}$.

Different orderings for 1st-order Jacobian matrix are given.

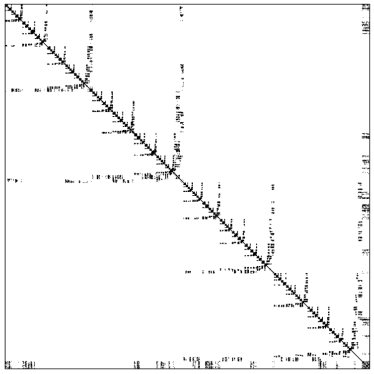


Figure 3.5: METIS reordering case

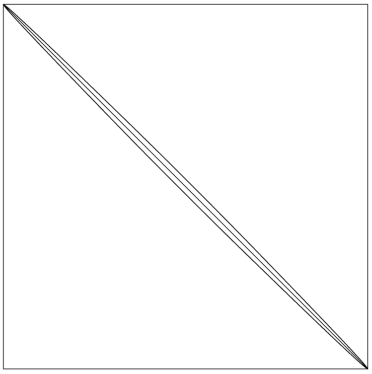


Figure 3.6: RCM reordering case

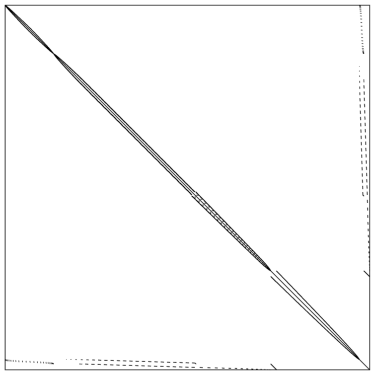


Figure 3.7: 1WD reordering case

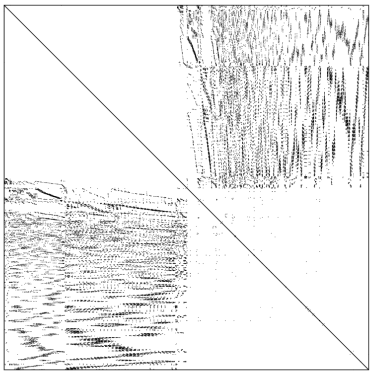


Figure 3.8: MD reordering case

Reordering algorithm choices will not affect the condition number of matrix, but it is expected that overall performance will be different because of the reduced number of nonzero after preconditioning with some level of fill is allowed. We will share the performance table for preconditioners and reorderings for constant $\eta_k = 0.5$ for no ordering, RCM, 1WD and $\eta_k = 0.7$ for METIS, MD. For different forcing parameter choices some of the runs are not converged or weakly converged. Through Table 3.1 to Table 3.5 wall clock times are given in seconds and convergence criteria is chosen as $10^{-6}$ relative residual in continuity equation. Newton and total GMRES iterations are given for full comparison.

Table 3.1: $ilu(0)$ preconditioning

| | newton itr. | gmres itr. | elapsed time | nnz |
|---|---|---|---|---|
| no reorder | 906 | 1866 | 2855.49 | 36956800 |
| metis | NA | NA | NA | 36956800 |
| rcm | 905 | 1866 | 3575.55 | 36956800 |
| 1wd | 838 | 1879 | 3530.64 | 36956800 |
| md | NA | NA | NA | 36956800 |

Table 3.2: $ilu(1)$ preconditioning

| | newton itr. | gmres itr. | elapsed time | nnz |
|---|---|---|---|---|
| no reorder | 644 | 1476 | 2491.73 | 67874202 |
| metis | NA | NA | NA | 82378052 |
| rcm | 707 | 1561 | 2433.40 | 67874202 |
| 1wd | 688 | 1348 | 2571.34 | 68407402 |
| md | 1617 | 2878 | 8536.21 | 82981202 |

Table 3.3: $ilu(2)$ preconditioning

| | newton itr. | gmres itr. | elapsed time | nnz |
|---|---|---|---|---|
| no reorder | 548 | 1339 | 4076.69 | 118148502 |
| metis | NA | NA | NA | 93855202 |
| rcm | 501 | 1119 | 3200.76 | 118462102 |
| 1wd | 556 | 1359 | 3114.16 | 119046512 |
| md | 1652 | 2924 | 10075.56 | 87051862 |

Table 3.4: $ilu(3)$ preconditioning

| | newton itr. | gmres itr. | elapsed time | nnz |
|---|---|---|---|---|
| no reorder | 428 | 1036 | 3484.1 | 216914952 |
| metis | 575 | 1408 | 3209.83 | 173069802 |
| rcm | NA | NA | NA | 177678402 |
| 1wd | 453 | 1148 | 4037.21 | 179969262 |
| md | 565 | 1297 | 2994.62 | 172425212 |

Table 3.5: $ilu(4)$ preconditioning

| | newton itr. | gmres itr. | elapsed time | nnz |
|---|---|---|---|---|
| no reorder | 411 | 901 | 3566.25 | 351310602 |
| metis | NA | NA | NA | 192610752 |
| rcm | 423 | 961 | 3675.86 | 255229652 |
| 1wd | 430 | 1099 | 4247.54 | 260437752 |
| md | 567 | 1288 | 3064.33 | 179170012 |

In general, best elapsed times are obtained by $ilu(1)$, Table 3.2, while least GMRES iterations are obtained at relatively higher levels of fills such as $ilu(4)$, Table 3.5. After increasing some level of fill, the number of non-zero increases in preconditioner matrix extensively and total time for matrix-vector multiplications becomes more important than the accuracy of incomplete factorization. Therefore, $ilu(1)$ seems the most efficient one in our case. Sample convergence of different preconditioners are given for no reordering case.

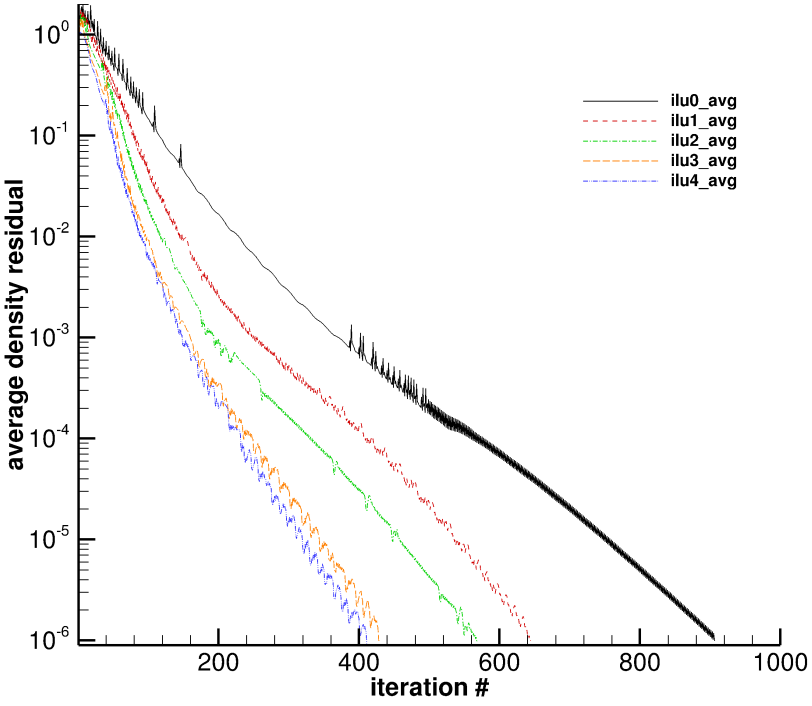



Figure 3.9: Preconditioner effect on convergence using no ordering

As shown in Figure 3.9, early iterations in ilu(0) is extremely zigzaggy. In addition fluctuations which are observed after residual level $10^{-3}$ make scheme slower. Therefore ilu(0) preconditioner is less efficient comparing to ilu(1). Although the less iterations observed in high fill-in preconditioners, total time is greater at them because of the increased matrix-vector multiplications. Therefore the use of ilu(1) preconditioner is found beneficial in terms of stability, less cpu time and less total iterations. Finally we can say that ilu(2) is the most stable one since oscillations are not quite much among the other preconditioners.

Looking the tables given above reordering selection has no effect on elapsed time while its influence can be only understood while level of fill increases. Since in higher accurate factorizations, Table 3.5, some of the algorithms such as METIS and MD decreases the total number of non-zero to half for original matrix. So we can say that reordering algorithms can save too much memory for higher-order factorizations. The comparison on $nnz$ can be seen more clearly in Figure 3.10.

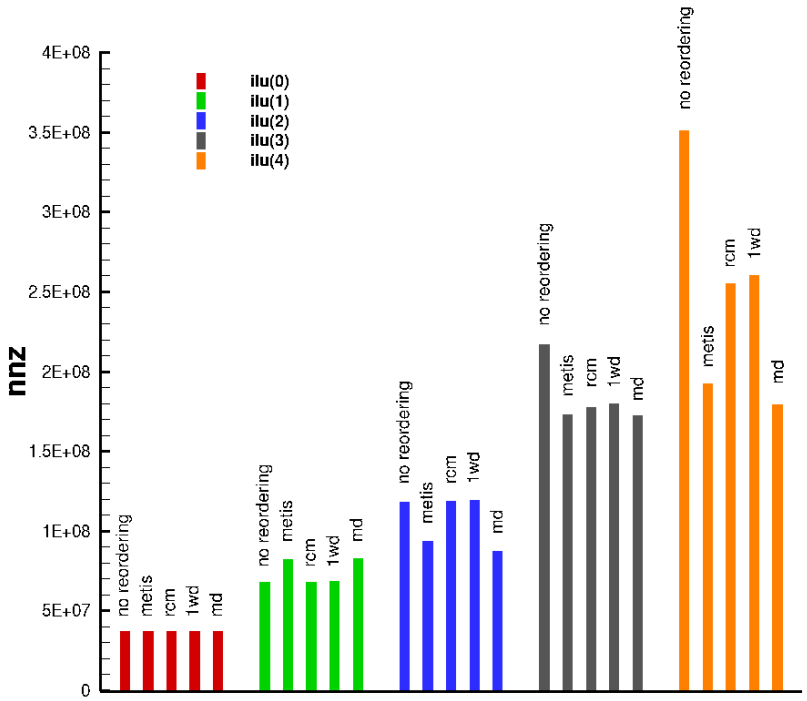



Figure 3.10: Reordering effects on total number of non-zero

Because of the best wall clock time and overall efficiency, no reordering and RCM are used in conjunction with $ilu(1)$ preconditioning throughout the study.

# CHAPTER 4

# DESIGN VARIABLES

## 4.1 Design Variables Selection

The simplest approach for shape parameterization is direct use of grid nodes, but this approach ends up with high number of design variables which is not appropriate for many practical problems. For this reason, parameterization techniques are more appropriate in CFD based optimization.

The most promising shape parameterization approaches are shape functions, polynomials and splines. Shape functions are generally used to create small perturbations on "baseline" 2-D geometries such as airfoils. The well known shape function is Hicks & Henne bump functions [29]. However, bump functions lead to a specific improvement according to design conditions and outside the off-design conditions, optimized shape typically shows poor performance. Nevertheless its formulation is important to show how perturbations are parameterized. Its first application was on airfoil shapes which is only deflection normal to the airfoil is parameterized. This deflection corresponds to right side of positive sign in Equation (4.1).

$$y = y_{base} + \sum_{i=1}^{i=n} \alpha_i f_i(x) \tag{4.1}$$

where $y$ original point of new airfoil surface and $y_{base}$ indicates the base airfoil shape. Name of the shape function comes from the existence of $f_i(x)$. Each function contributes to the final design point according to its participation coefficient

$\alpha_i$. In Equation (4.1) shape function can be defined as

$$f_i(x) = \left[ sin\left( \pi x^{\frac{log\frac{1}{2}}{logt_1}} \right) \right]^{t_2}, \quad 0 \leq x \leq 1 \tag{4.2}$$

where $t_1$ and $t_2$ are the location of maximum thickness and width of the bumps respectively.

The other approach is polynomials. The best example for this method is PARSEC which is abbreviation for *parameterization scheme* developed by Sobieczky [30]. According to method, airfoil shape can be generated by the linear combinations of base functions using 12 different geometric characteristics as control variables. Probably the greatest advantage of this method is that no baseline shape is needed. In addition, geometric constraints can easily be applied since the constraints are used as actual design variables. However, they are generally used for known shapes such as airfoils etc.; therefore, its use on complex geometries are limited.

The effect of shape parameterization techniques on airfoils were investigated by Selvan [75]. He found that increasing the degree of freedom in design variable selection increases the optimization performance. Polynomial approach is found less efficient one comparing to shape functions and splines.

On the other hand, splines are another well known approach for parameterization. General opinion for best splines method is Non-Uniform Rational B-Splines (NURBS) which leads to more realistic optimized shape not only for specific design conditions but also off-design conditions on complex geometries also. In general, NURBS solutions are smooth enough for real world problems with a benefit of high control on shape with less design variables. NURBS surface point representation will be explained in next section.

### 4.1.1 Non-Uniform Rational B-Splines (NURBS) Surfaces

The NURBS [31] surface point is composed of a two parametric coordinates point such as

$$S(\xi,\eta)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(\xi)N_{j,q}(\eta)w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(\xi)N_{j,q}(\eta)w_{i,j}} \tag{4.3}$$

In Equation (4.3) surface points vector $S$ keeps the cartesian coordinates, $x, y$ and $z$ where $\xi$ and $\eta$ corresponds to parametric coordinates, $P_{i,j}$ bi-directional control net, $w_{i,j}$ weight of control net and $N_{i,p}$ and $N_{j,q}$, $p$ and $q$ degree basis functions. The number of control points are $n+1$ and $m+1$ in $i$ and $j$ directions respectively. B-spline basis function on specific knot vector can be derived using Cox-de Boor recurrence relation. The $i$th B-spline basis function of degree $p$ (order $p+1$) is given by

$$N_{i,0}(\xi)=\begin{cases}1 \text{ if } u_i\le\xi<u_{i+1}\\ 0 \text{ otherwise}\end{cases} \tag{4.4}$$

$$N_{i,p}(\xi)=\frac{(\xi-u_i)}{u_{i+p}-u_i}N_{i,p-1}(\xi)+\frac{(u_{i+p+1}-\xi)}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(\xi) \tag{4.5}$$

The other basis function $N_{j,k}(\eta)$ is similar to $N_{i,k}(\xi)$ where $u_i$'s are changed by $v_i$'s and degree $p$ is changed by $q$. In Equation (4.5) $u_i$ and $v_i$ are called knots which are the elements of nondecreasing sequence of real numbers, ie. $u_i \le u_{i+1}$. Knots are stored in knot vectors $U$ and $V$

$$U=\left\{\underbrace{0,...,0}_{p+1},u_{p+1},...,u_i,...,u_n,\underbrace{1,...,1}_{p+1}\right\}_{n+p+2} \tag{4.6}$$

$$V=\left\{\underbrace{0,...,0}_{q+1},v_{q+1},...,v_i,...,v_m,\underbrace{1,...,1}_{q+1}\right\}_{m+q+2} \tag{4.7}$$

The non-uniform knot sequence $u_i$ is given by Nemec [32] due to improved fidelity near leading and trailing edge of wing sections.

$$u_i=\begin{cases}0 & 1\le i\le p+1\\ \frac{n-p+1}{2}\left[1-cos\left(\frac{i-p-1}{n-p+1}\pi\right)\right] & p+2\le i\le n+1\\ n-p+1 & n+2\le i\le n+p+2\end{cases} \tag{4.8}$$

Knot sequence of $v_i$ can be formulated similarly. The basis function is only nonzero inside knot span except everywhere it is equal to zero.

NURBS surface is represented by two parametric coordinates, $\xi$ and $\eta$. For each $\xi$ and $\eta$, there exists a unique NURBS point. The resolution of NURBS surface can be increased by increasing the number of parameters. However, in general it is restricted by the dimension of surface grid points which is to be parameterized. Following strategy is used to create parameter vector in $\xi$ direction.

$$\xi_1 = 0$$
$$\xi_j = \frac{(n-p+1)}{L_T}\sum_{m=1}^{j-1}\sqrt{L_m}, \quad j = 2, ..., N \tag{4.9}$$

where $N$ indicates the number of parameter points. The segment lengths, $L_m$ are chosen as the distance between grid points and $L_T$ is total grid point distance which is given by

$$L_T = \sum_{m=1}^{N-1}\sqrt{L_m} \tag{4.10}$$

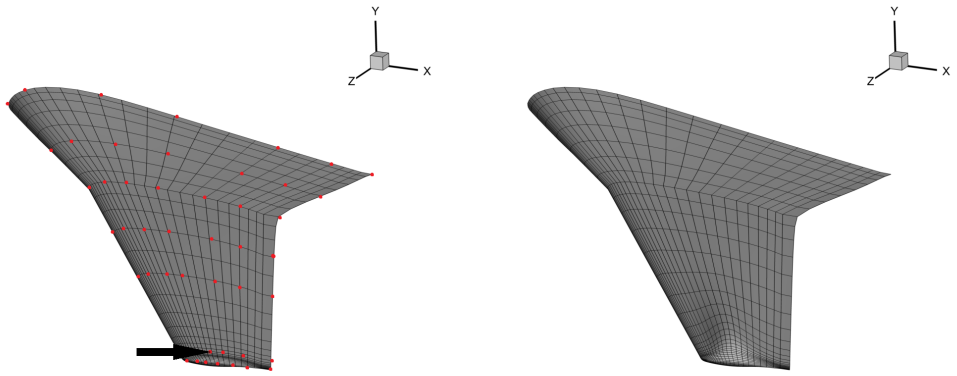
Finally introducing the piecewise rational basis function $R_{i,j}(\xi, \eta)$

$$R_{i,j}(\xi,\eta) = \frac{N_{i,p}(\xi)N_{j,q}(\eta)w_{i,j}}{\sum_{k=0}^{n}\sum_{l=0}^{m}N_{k,p}(\xi)N_{l,q}(\eta)w_{k,l}} \tag{4.11}$$

surface point can be represented as

$$S(\xi,\eta) = \sum_{i=0}^{n}\sum_{j=0}^{m}R_{i,j}(\xi,\eta)P_{i,j} \tag{4.12}$$

In this study only perturbation is parameterized since it is diffucult to create all surface domain by parameterization and so much error is included on baseline geometry [32] even on 2D. NURBS surfaces are used to decide deformation of wing only. Figures 4.1a and 4.1b shows the sample perturbation near wing tip. 3rd degree NURBS surfaces are created with 25x2=50 control points which are chosen on both upper and bottom surface of wing and only y-deflection allowed in Figure 4.1b. On later, x-y deflection will be carried out in optimization to ensure best performance gained from control points. However, if deflection in both x-y is allowed then number of design variables are doubled so 100 control points become a subject. In addition, selection of NURBS degrees will be mentioned in the following sections in detail.

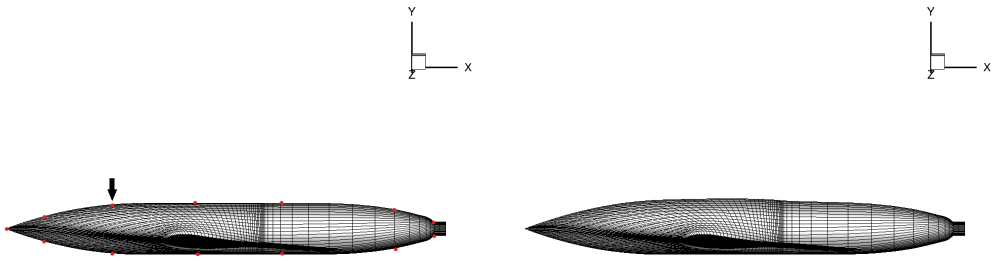



(a) Unperturbed wing surface (b) Perturbed wing surface

Figure 4.1: NURBS surface perturbation on wing

### 4.1.2 Design Variables on Body as NURBS Curves

Body design variables are chosen as 2D NURBS curves on symmetry plane of both upper and lower surface of body. Then the deflection is distributed spanwise direction, through z-plane. Perturbation on x-z is allowed. $5 \times 2 \times 2 = 20$ NURBS curve control points are chosen as in Figure 4.2a.




(a) Unperturbed body surface (b) Perturbed body surface

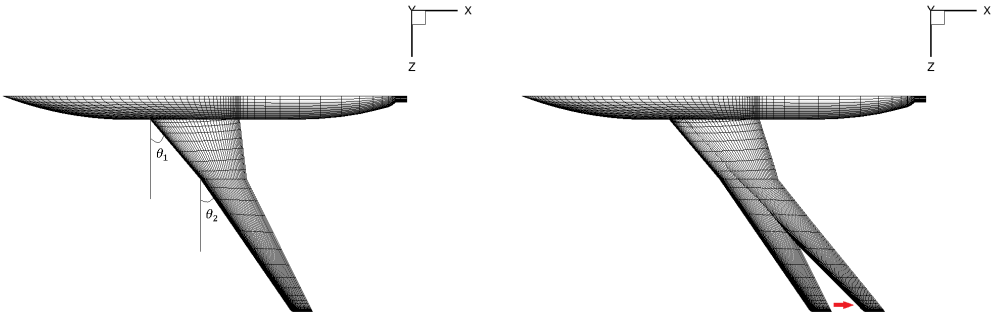
Figure 4.2: NURBS curve perturbation on body

### 4.1.3 Design Variables as Wing Sweep Angles

Wing is composed of two main parts and divided by crank at the middle of span. Two design variables are chosen such that leading edge sweep angles of divided parts as in Figure 4.3a.

$$\Delta\theta_k = tan^{-1}\left[\frac{x_k - x_{k-1}}{y_k - y_{k-1}}\right], \quad k = 1, 2 \tag{4.13}$$

where $k_1$ is root chord station and $k_2$ is crank station. Then, deflections are allowed on the $x$ direction by conserving the chord lengths constant.




(a) Unperturbed wing surface (b) Perturbed wing surface

Figure 4.3: LE sweep angle design variables perturbation on wing

## 4.2 Grid Perturbation Strategy

After shape deformation on surface, volume grid must be deformed to satisfy following criteria

- Grid quality must be conserved.
- The cells with negative volumes must be avoided.
- Grid perturbation strategy can handle with large deformations effectively.

According to criteria given above, many research have been conducted on volume mesh deformations on both structured and unstructured grids. The subject on

volume mesh deformations can be divided into two, the algebraic approaches and iterative approaches. The well known algebraic approach was fist developed by Burgreen and Baysal [33]. This method is easy to apply for structured topologies such that C or C-H type grids on both 2D and 3D. The deflections are propagated from surface to interior nodes according to arc-length distances of grid points. There is no iterative procedure is applied in this method so it is very cheap in terms of computational cost. However grid quality are not satisfied at large deformations so this method is not preferred often.

On the other hand, iterative methods have been gained popularity over the years. Linear spring analogy and linear elasticity methods are well known examples of this kind. Batina [76] found the linear spring analogy and applied on unstructured triangular meshes around an airfoil. Then, he also applied the method on complex aircraft configurations [77] to study static and dynamic deformations of the aircraft grid. In linear spring analogy method, defined linear springs have stiffness which are inversely proportional to the spring length between neighboring nodes. However, edge collapses and skewed cells are frequently encountered for even little deformations and computational cost is found high. The other method treats the mesh as an elastic solid and solves using the equations of linear elasticity. Baker et al. [78] used this method to solve domain and mesh deformation on store separation problem. Afterward Stein et al. [79] investigated method on planar meshed fluid-structure interaction (FSI) problems on translation, rotation and bending modes. The method is also computationally expensive like others.

This brings us to Radial Basis Functions (RBF) method. The method was first developed by Boer et al. [80] for unstructured robust FSI computations. The algorithm is based on interpolation of displacements on surface mesh through the volume mesh with the help of radial basis functions. The required grid connectivity information in linear elasticity and spring methods are no longer needed. Therefore, this method is easy to apply and the computational cost is less since only the boundary or surface nodes involving equations have to be solved. In this study volume mesh is deformed by RBF approach using different basis functions according to parameterized surface. We will present

first algebraic approach and then RBF approach. Afterwards, both approaches will be compared in detail.

### 4.2.1 Algebraic Grid Perturbation Strategy

As mentioned previously, algebraic grid perturbation approach works for C-H type structured grid topology well. Algebraic perturbation definition is given below.

$$y_j^{new} = y_j^{old} + \Delta y(1 - S_j), \qquad j = 1, ..., j_{max} - 1 \tag{4.14}$$

where $\Delta y$ represents deflection in y-direction and $S_j$ is normalized arch-length distance which is calculated by

$$S_1 = 0 \tag{4.15}$$

$$S_j = \frac{1}{L_g} \sum_{i,k=2}^{j} L_{i,k}, \qquad j = 2, ..., j_{max} - 1 \tag{4.16}$$

In Equation (4.16) $L_{i,k}$ corresponds to node line segments according to their indicies and $L_g$ is the total grid line length from surface to outer boundary where

$$L_g = \sum_{i,k=2}^{j_{max}} L_{i,k} \tag{4.17}$$

This version of grid perturbation strategy can be improved in order to obtain higher orthogonal quality grid near the surface as in reference [32] st.

$$y_j^{new} = y_j^{old} + \frac{\Delta y}{2}(1 + cos(\pi S_j)), \qquad j = 1, ..., j_{max} - 1 \tag{4.18}$$

### 4.2.2 Radial Basis Function Strategy

In this method, interpolated deflections on boundary nodes are used to find displacements in internal nodes. The displacement $d$ can be approximated by sum of basis functions where RBF interpolation function appears as

$$d(x) = \sum_{i=1}^{N_{rbf}} \alpha_i \phi(||x - x_i||) \tag{4.19}$$

where $d$ is the displacement at interpolated point $x = x(x, y, z)$, $x_i$ are the known values of RBF points, $\phi$ is radial basis function and $||x - x_i||$ corresponds to euclidean distance between interpolated point and RBF points. The $\alpha_i$'s are the weights and $N_{rbf}$ is the number of RBF points. Equation (4.19) corresponds to a linear system of equation for known displacements on surface nodes. The linear system of dimension $N_{rbf} \times N_{rbf}$ has to be solved to decide weights. Right hand side of linear system is known displacements at RBF points and left hand side matrix contains the information about euclidean distances. In original method, polynomial term $p(x)$ is added to the interpolating function for generally FSI applications to conserve translation motion. However in general optimization based CFD codes, $p(x)$ term is not applied. The corresponding system for Equation (4.19) is in three coordinates

$$\begin{aligned} M\alpha_x &= \Delta x_r \\ M\alpha_y &= \Delta y_r \\ M\alpha_z &= \Delta z_r \end{aligned} \tag{4.20}$$

where $r$ subscript denotes the RBF points. The linear system appears

$$\begin{bmatrix} \phi_{r_1 r_1} & \phi_{r_1 r_2} & \cdots & \phi_{r_1 r_{N_{rbf}}} \\ \phi_{r_2 r_1} & \ddots & & \\ & & & \\ \vdots & & & \vdots \\ \phi_{r_{N_{rbf}} r_1} & & \cdots & \phi_{r_{N_{rbf}} r_{N_{rbf}}} \end{bmatrix} \times \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \\ \vdots \\ \alpha_{N_{rbf}} \end{bmatrix}_{x,y,z} = \begin{bmatrix} \Delta_1 \\ \Delta_2 \\ \\ \vdots \\ \Delta_{N_{rbf}} \end{bmatrix}_{x_r,y_r,z_r}$$

Since $M$ matrix on the left includes only euclidean distances, it is independent of coordinates. The deflection on different spatial coordinates appears at the right hand side. Above system can be solved by first direct factorization of coefficient matrix $M$ by LU or SVD then simple forward-backward substitution. Once the factorization is applied, deflection for RBF points in three spatial direction can be easily obtained by simple arithmetic operations. Therefore, direct approach is followed in this study. The elements of radial basis functions can be scaled by support radius $R_0$ when compact basis functions are used especially.

$$\phi_{r_1,r_2} = \phi(\xi_{r_1,r_2}) = \phi\left(\frac{||x - x_i||}{R_0}\right) \tag{4.21}$$

The evaluation of basis functions is relatively straight-forward. Three types of basis functions exist, global, local and compact. Global functions are always non-zero and growing through outer boundary. Similar to previous one, local functions are always non-zero, but its trend is decaying through the outer boundary. Compact basis functions conserve the decaying property of local functions, but at some distance its effect vanishes. The distance is called as support radius which is given in Equation (4.21). Rendall and Allen [81] have defined the basis functions nicely. The details of global and local basis functions can be found in [80]. Some of them are tabulated below.

Table 4.1: Some global & local basis functions

| Name | Definition |
|---|---|
| Gaussian (G) | $\phi(\xi) = e^{-\alpha\xi}$ typically $10^{-5} \leq a \leq 10^{-3}$ |
| Thin plate spline (TPS) | $\phi(\xi) = \xi^2 ln(\xi)$ |
| Volume spline (V) | $\phi(\xi) = \xi$ |

Thin plate spline (TPS) and volume splines (V) are well known examples of global functions while Gaussian spline (G) is local function. Global support basis functions can lead to unwanted deformation at outer boundaries since the support radius are defined for these kind of functions. They are generally used in neural networks, FSI problems and computer graphics applications.

The Wendland's functions [82] are known as compact functions. In Table 4.2, they are presented according to their degrees.

Table 4.2: Compact basis functions

| Degree | Name | Definition that satisfies $\phi(\xi) = 0 \ \forall \ \xi > 1$ |
|---|---|---|
| deg = 1 | Wendland's C0 | $\phi(\xi) = (1-\xi)$ |
| | Wendland's C2 | $\phi(\xi) = (1-\xi)^3(3\xi+1)$ |
| | Wendland's C4 | $\phi(\xi) = (1-\xi)^5(8\xi^2+5\xi+1)$ |
| deg = 3 | Wendland's C0 | $\phi(\xi) = (1-\xi)^2$ |
| | Wendland's C2 | $\phi(\xi) = (1-\xi)^4(4\xi+1)$ |
| | Wendland's C4 | $\phi(\xi) = (1-\xi)^6(35\xi^2+18\xi+3)/3$ |
| | Wendland's C6 | $\phi(\xi) = (1-\xi)^8(32\xi^3+25\xi^2+8\xi+1)$ |
| deg = 5 | Wendland's C0 | $\phi(\xi) = (1-\xi)^3$ |
| | Wendland's C2 | $\phi(\xi) = (1-\xi)^5(5\xi+1)$ |
| | Wendland's C4 | $\phi(\xi) = (1-\xi)^7(16\xi^2+7\xi+1)$ |

In this work, compact basis functions are used because of the smooth mesh deformation and support radius availability in order to prevent far-field perturbations. In addition, decaying property of compact functions allows to higher deformations rate at near the surface while lower deformations at near of outer boundary. This is also good for the sake of accuracy in flow solution. According to trial and errors degree 3 Wendland's C0 function is chosen to use in this study.

The compact basis functions are drawn below

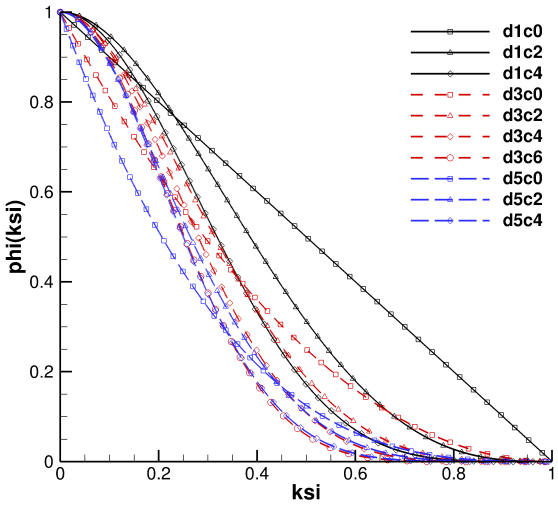



Figure 4.4: Compact basis functions

And finally displacements on the volume mesh can be found

$$\begin{aligned} \Delta x_v &= A\alpha_x = AM^{-1}\Delta x_r \\ \Delta y_v &= A\alpha_y = AM^{-1}\Delta y_r \\ \Delta z_v &= A\alpha_z = AM^{-1}\Delta z_r \end{aligned} \tag{4.22}$$

where A matrix

$$A = \begin{bmatrix} \phi_{v_1 r_1} & \phi_{v_1 r_2} & \cdots & \phi_{v_1 r_{N_{rbf}}} \\ \phi_{v_2 r_1} & \ddots & & \\ & & & \\ \vdots & & & \vdots \\ \phi_{v_{N_v} r_1} & & \cdots & \phi_{v_{N_v} r_{N_{rbf}}} \end{bmatrix}_{N_v \times N_{rbf}}$$

where $N_v$ is number of volume mesh points. Equalities at the right hand side are more useful since $M^{-1}$ is computed at the beginning of interpolation of RBF points. If the parameterization is applied from the deformation of original grid, $M^{-1}$ is computed just once and stored in the memory as in matrix $A$ also. Only changed component is deflection in RBF points at repeated design cycles. Rest of the computation is then simple matrix-vector multiplications. This approach is followed in the thesis. However, if the number of surface grid points increase, memory problems can occur easily.

Some RBF points reduction strategies are studied by various researchers such as Jakobsson and Amoignon [83], Rendall and Allen [81] later. In this method, all surface points are not selected as RBF points. Non-RBF points are accumulated by secondary mesh movement ie. algebraic manner [35]. However, this approach can create error at the surface point deflection which is decided by design variables as stated [81]. Although the usage of full set is costly, in this study it is preferred to use all surface points as RBF points to adopt desired accuracy in gradients.

### 4.2.3 Comparison of grid perturbation strategies

In this section, both algebraic and RBF grid perturbation strategies will be compared on $50 \times 50 \times 50$ structured cube domain. Before showing the volumes mesh deformation, it is better to demonstrate that how NURBS surface degrees affect the surface grid deformation. In surface parameterization $5 \times 5$ control points are used with degree $1, 2, 3, 4$ and $5$. The location of control points are not kept constant and all weights are set to 1.0. The largest deformation on volume

grid is observed at half cut plane which is $I = 25$. Hence the deformations are presented at that plane for volume deformations.

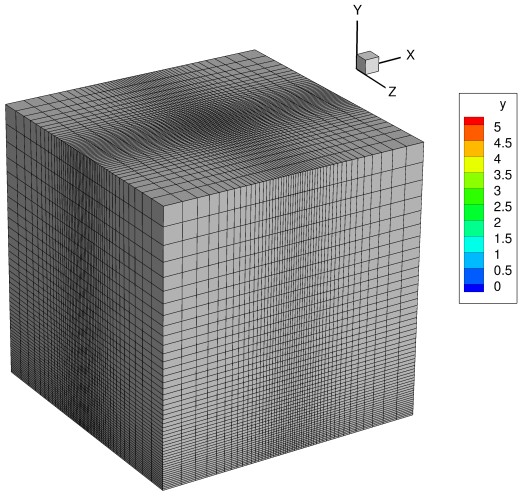



Figure 4.5: Cube grid

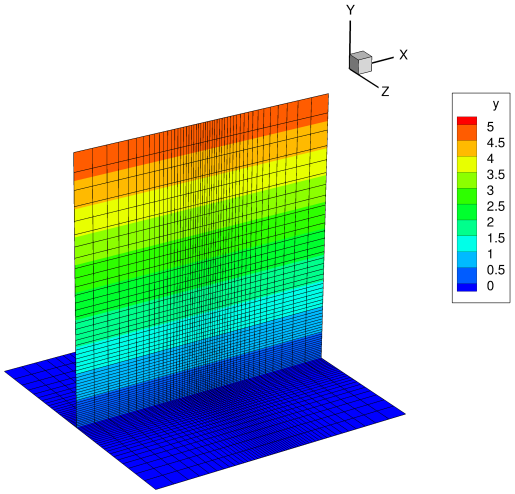
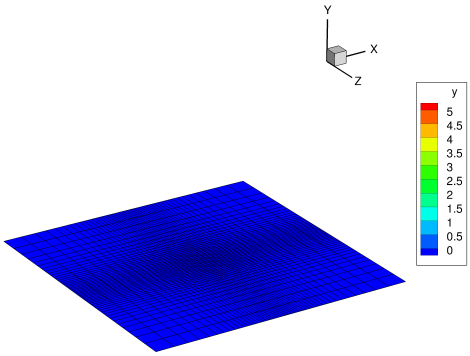
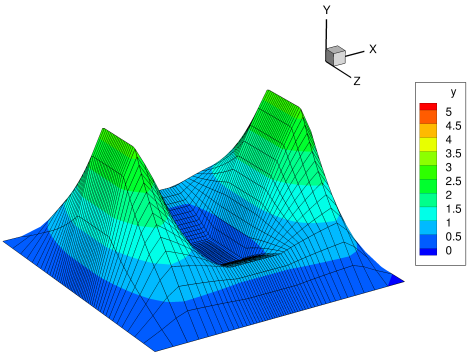


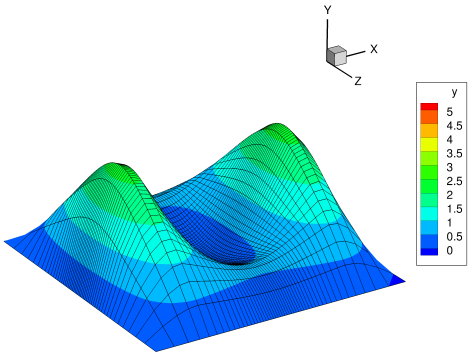

Figure 4.6: Bottom surface plane (k=1) and half cut plane (i=25)



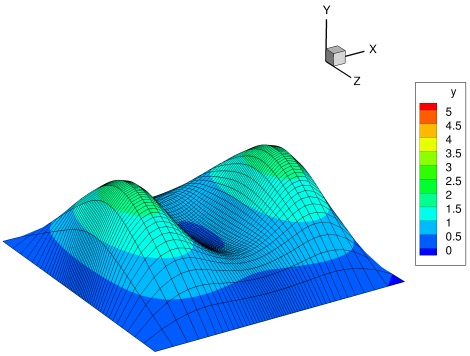

(a) Original surface



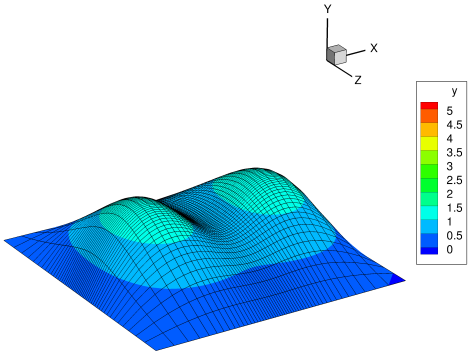

(b) NURBS d=1 deformed surface



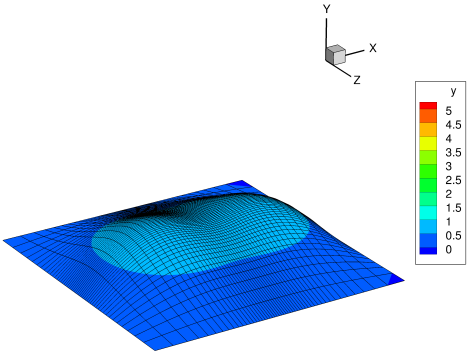

(c) NURBS d=2 deformed surface




(d) NURBS d=3 deformed surface

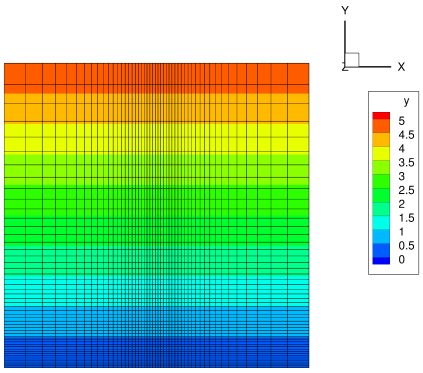



(e) NURBS d=4 deformed surface

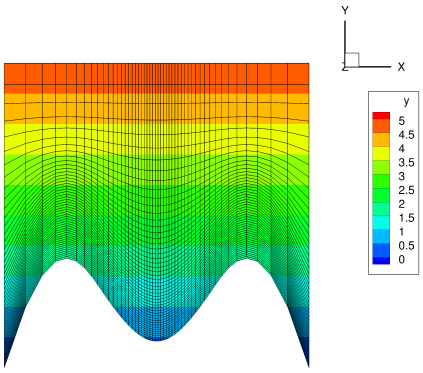



(f) NURBS d=5 deformed surface

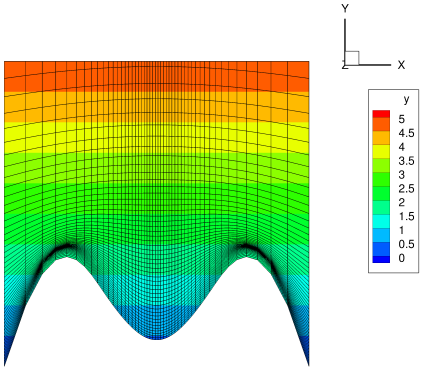
Figure 4.7: Comparison of NURBS control point degrees on surface



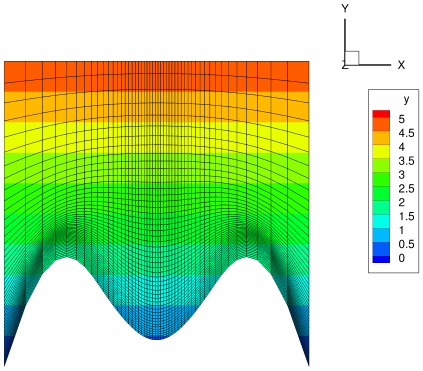

(a) Original



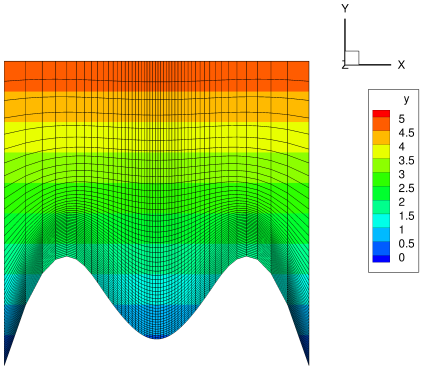

(b) Algebraic perturbation



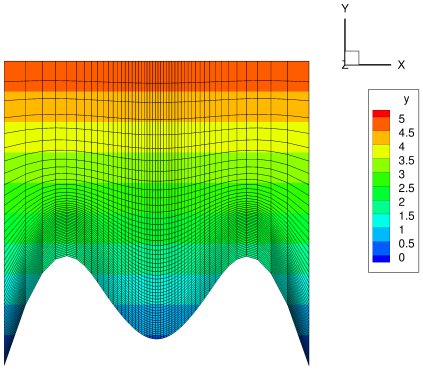

(c) RBF d=3 Wendland's C0 function




(d) RBF d=3 Wendland's C2 function




(e) RBF d=3 Wendland's C4 function




(f) RBF d=3 Wendland's C6 function

Figure 4.8: Comparison of algebraic and RBF perturbation

In Figure 4.7, effect of degree choice on surface deflections are presented. As shown, increasing control point degrees generates smoother deflections especially on curvature regions. However, higher degrees also decreases the effect of control points. In Figure 4.7d, hollow at the middle can be easily distinguished. However, in Figure 4.7f the hollow region can be barely seen. Higher degrees can smooth the deflections unnecessarily which is not desired for the optimization procedure. Therefore, the optimum degree must be chosen for the sake of optimization. In general, degree 3 is used as rule of thumb which is also chosen in this study on both NURBS curve and surface parameterization.

On the other hand, sample volume deformations at half cut plane is shown in Figure 4.8. RBF approach creates more dense cells on curvature areas which is critical for the flow resolution. Density of the cells in RBF on curvatures are decreasing as function polynomial degree increases through Figure 4.8c to Figure 4.8f. Also algebraic approach gives same density to cells independently from the deformation. This situation is expected since in algebraic approach the distribution of step-sizes in cells depends only the undeformed nodes arc-length distances. Moreover, RBF approach helps to reduce average skewness on the cells, Table 4.3. All grid information is imported to mesh generation software Pointwise [84] and the average cell quality metrics are taken.

Table 4.3: Volume deformation strategies on grid quality metrics

| | *Centroid skewness* | *Equiangle skewness* | *Aspect ratio* |
|---|---|---|---|
| Original grid | 0 | 0 | 3.429 |
| Algebraic perturbation | 0.155 | 0.304 | 4.246 |
| RBF Wendland's C0 | 0.097 | 0.216 | 4.465 |
| RBF Wendland's C2 | 0.120 | 0.257 | 4.334 |
| RBF Wendland's C4 | 0.128 | 0.264 | 4.335 |
| RBF Wendland's C6 | 0.135 | 0.274 | 4.382 |

RBF approach both reduces the centroid and equiangular skewness and slightly decreases the aspect ratio also. The benefits of RBF may seen little comparing code development or CPU time, however the actual use on real life problems can be life saver because real life problems are more complex than the given cube example Figure 4.5. The negative cells are often encountered situations. Algebraic approach is barely useful to prevent negative cells. To sum up, RBF

seems useful for large deformations and it is chosen as volume mesh deformation strategy. Because of the good resultant grid quality and safety for negative cell volumes, degree 3 Wendland's C0 basis function is selected in this study for different shape parameterizations with a constant support radius 100 as in [80].

# CHAPTER 5

# ADJOINT METHOD

## 5.1 Evaluation of Discrete Gradients

Discrete gradients can be evaluated in three different way, direct, adjoint and finite difference approaches. Although the scope of thesis is implementation of Adjoint approach, finite difference and direct gradients are also given for comparison purposes. Therefore all three methods will be mentioned below.

### 5.1.1 Direct Method Sensitivities

The definition of direct discrete gradient, $\Omega$ of the objective function $I\big[S, Q(S)\big]$ is given by

$$\Omega = \frac{dI}{dS} = \frac{\partial I}{\partial S} + \frac{\partial I}{\partial Q}\,\frac{dQ}{dS} \tag{5.1}$$

where vector of design variables, $S$, is dimension of $N_D$ and vector of flow variables, $Q$, is dimension of $N_F$. Then $\Omega$ and $\frac{\partial I}{\partial S}$ are $[1 \times N_D]$ row vectors. $\frac{\partial I}{\partial Q}$ is $[1 \times N_F]$ row vector. $\frac{\partial Q}{\partial S}$ is $[N_F \times N_D]$ matrix. However, this equation requires the assumption of $Q(S)$ is sufficiently smooth or its derivative exists for specified design variable. This condition is satisfied by the spatial discretization of flow equations where the splitting and MUSCL schemes are used.

Let us first differentiate the flow equations in $R(Q, S) = 0$ with respect to design variables as in Equation (5.1)

$$\frac{dR}{dS} = \frac{\partial R}{\partial S} + \frac{\partial R}{\partial Q}\,\frac{dQ}{dS} \tag{5.2}$$

For any design variable $R(Q) = 0$ must be satisfied hence $\frac{dR}{dS} = 0$ is also satisfied. We can write linear system of equations for the flow sensitivities, $\frac{dQ}{dS}$, as follows

$$\left[\frac{\partial R}{\partial Q}\right] \frac{dQ}{dS} = -\left[\frac{\partial R}{\partial S}\right] \tag{5.3}$$

where $\frac{\partial R}{\partial Q}$ is $[N_F \times N_F]$ flow Jacobian matrix, $\frac{dQ}{dS}$ and $\frac{\partial R}{\partial S}$ is $[N_F \times 1]$ column vectors. However, Equation (5.3) must be solved for each design variable. The cost of solving this equation for each design variable can be tremendous according to dimension of optimization problem. At the end obtained flow sensitivities can be used in direct sensitives given in Equation (5.1). It is important to note that residual vector, $R$, has contribution from all grid points in flow domain including boundary conditions also. The other derivatives which are required in direct sensitivities, $\partial I/\partial S$ and $\partial I/\partial Q$ are relatively straight-forward and will be mentioned in following sections.

### 5.1.2 Adjoint Method Sensitivities

The Adjoint approach is developed to avoid repeated solution of flow sensitivity equations. Hence Equation (5.3) is inserted into objective sensitivity function in Equation (5.1), and discrete adjoint gradient can be written.

$$\frac{dI}{dS} = \frac{\partial I}{\partial S} - \underbrace{\frac{\partial I}{\partial Q}\left(\frac{\partial R}{\partial Q}\right)^{-1}}_{\psi^T} \frac{\partial R}{\partial S} \tag{5.4}$$

The Adjoint equation is derived from the right side of Equation (5.4) by introducing the Lagrange multiplier vector $\psi$. The linear system of equation can be written for Lagrange multipliers as follows

$$\left[\frac{\partial R}{\partial Q}\right]^T \psi = \left[\frac{\partial I}{\partial Q}\right]^T \tag{5.5}$$

where $\psi$ and $\frac{\partial I}{\partial Q}$ are $[N_F \times 1]$ column and row vectors. The solution of linear system for Lagrange multipliers naturally satisfies the flow equations. At one solution of Adjoint equation all design variable sensitivities can be obtained.

Finally gradient of objective function becomes

$$\frac{dI}{dS} = \frac{\partial I}{\partial S} - \psi^T \frac{\partial R}{\partial S} \tag{5.6}$$

In Adjoint equation, $\frac{\partial I}{\partial Q}$ is needed. In this study objective functions are simply chosen as lift coefficient ($C_L$), drag coefficient ($C_D$) and pitching moment coefficient ($C_M$). For the Euler equations, the aerodynamic coefficients are direct function of surface pressure $p_{surf}$. Therefore, its derivative is relatively straight-forward and only depends on the ghost cells and first inner cells on target surface.

In gradient computation, the residual sensitivities, $\frac{\partial R}{\partial S}$, is another required derivative which is calculated in hybrid manner with the combination of analytical and finite difference methods. Since at the ghost cells which is required in $\frac{\partial R}{\partial S}$, no residual is computed, so it must be computed in analytical way. But for inner cells, finite difference approach gives satisfactory results. The central divided-difference method is given as

$$\frac{\partial R}{\partial S_n} = \frac{R(S + he_n, Q) - R(S - he_n, Q)}{2h} \quad n = 1, ..., N_D \tag{5.7}$$

where $h$ is step-size. The calculation of residual $R(S, Q)$ is not computationally intense. For the boundary cells, for algebraic derivation, $R(S, Q)$ depends on the grid points, $X$ such that

$$\frac{\partial R}{\partial S} = \frac{\partial R}{\partial X} \frac{\partial X}{\partial S} \tag{5.8}$$

In Equation (5.8), $\frac{\partial X}{\partial S}$ represents the grid node sensitivities which can be approximated easily by finite differences.

After creating all necessary derivatives, adjoint equation appears as sparse linear system of equations which can be solved iteratively or directly. In this study, solution is obtained by PARDISO [51] solver in direct manner. One can obtain different sensitivities in that equation by changing right hand sides only. Therefore multiple right hand side solution is required. In direct approach, factorized $A$ matrix can be used in solution of different right hand sides without additional memory requirements. After solving Adjoint equation, implementation of Equation (5.6) is straight-forward which is the combination of sparse matrix-vector products etc.

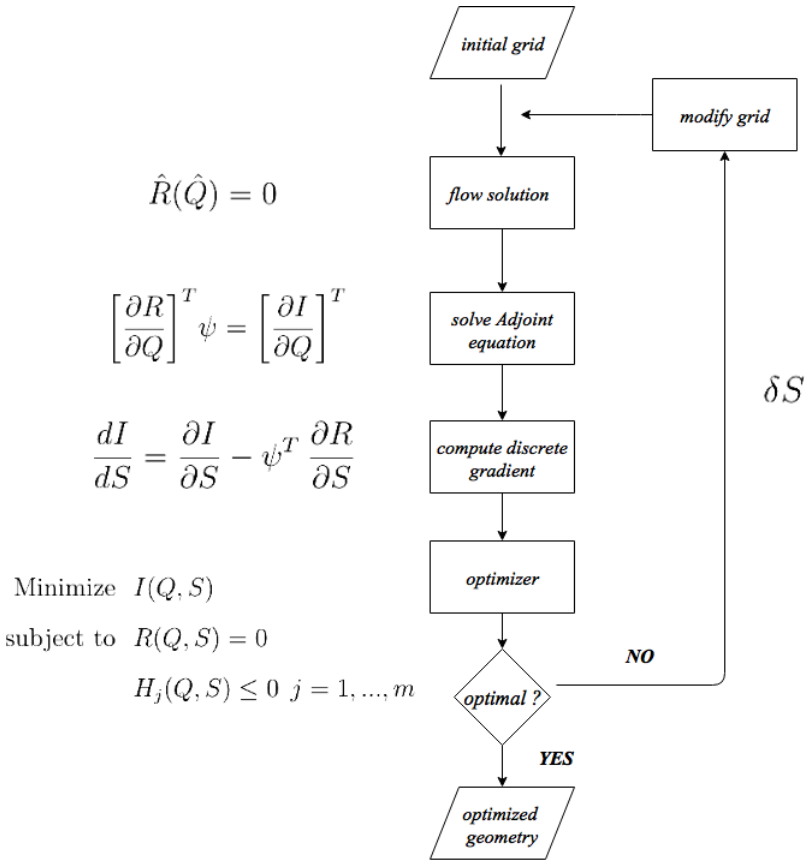



Figure 5.1: Flow-chart of optimization with Adjoint method

### 5.1.3 Finite Difference Method Sensitivities

Forward difference formula is used in this study to calculate finite difference gradients.

$$\frac{\partial I}{\partial S_n} = \frac{I(S + he_n, Q(S + he_n)) - I(S, Q(S))}{h} \quad n = 1, ..., N_D \tag{5.9}$$

where $h$ is step-size and $e_n$ is unit vector where $n$th element is 1. However, this method requires $N_D$ flow solution similar to direct method. Therefore its cost is too much and it is not applicable for large problems. Moreover, the accuracy of finite difference gradients strictly depends on step-size, $h$, selection. Since method is $O(h)$ accurate, larger the step-size increases truncation error, smaller stepsize increases round-off errors. Finite difference sensitivities are used to validate direct and Adjoint sensitivities in results section.

## 5.2 Optimization Tool

The optimizer used in this study is Design Optimization Tool (DOT) [85] which is developed by Vanderplaats at late 80's. DOT is general purpose gradient-based numerical optimization library that consists of Fortran codes and is used to solve generic non-linear optimization problems. The algorithm or package used in optimization is very critical that affects the final performance of design. In this study, DOT with Modified Method of Feasible Directions (MMFD) Method is chosen as optimization algorithm.

For general non-linear, constrained optimization problem, one seeks to find set of design variables, $X_i,\ i = 1, ..., N$ such that

$$\begin{aligned} &\text{Minimize} \quad F(X) \\ &\text{Subject to;} \\ &\qquad g_j(X) \leq 0 \quad j = 1, M \\ &\qquad X_i^L \leq X_i \leq X_i^U \quad i = 1, N \end{aligned} \tag{5.10}$$

where $F(X)$ is objective function, $g(X)$ are inequality constraints and $X^L$ and $X^U$ are lower-upper bounds of side constraints. Then MMFD can be formulated

$$X^q = X^q + \alpha S \tag{5.11}$$

where $X^q$ and $X^{q-1}$ are $q$-th and $(q-1)$-th design variable vectors. $S$ is search direction vector that minimizes the objective function and $\alpha$ is suitable step length that comes from line search. The algorithm composed of two main parts.

1. Finding the search direction, $S$. In this step, algorithm seeks the search direction to find out which constraints active or violated. DOT requires the constraint values have to be negative for feasible design. Then, immediately it can be concluded that constraints are active when its value equals to zero. However, due to machine precision it is not quite applicable in real life; therefore two parameters are defined, $CTMIN$ and $CT$, which

are small positive number and small negative number respectively.

$$\begin{aligned} g_j(X) &\leq CT && \text{Inactive} \\ CT \leq g_j(X) &\leq CTMIN && \text{Active} \\ g_j(X) &> CTMIN && \text{Violated} \end{aligned} \tag{5.12}$$

(5.13)

In this study, these parameters are used to relax the constraints. Defaults are $-0.03$ and $0.005$ for $CT$ and $CTMIN$ respectively. For further details in search algorithm one can look at [85].

2. Finding the scalar step-length $\alpha$. One dimensional search algorithm is used in DOT to find optimum scalar step size. During the process objective and constraint functions are evaluated repeatedly.

MMFD approach of DOT can be summarized in an algorithm Algorithm 12.

**Algorithm 12** Modified Method of Feasible Directions in DOT

1: Start, $q = 0$, $X = X^0$
2: $q = q + 1$
3: Evaluate $F(X)$ and $g_j(X)$ $j = 1, .., M$
4: Identify the set of critical or near critical constraints $J$.
5: Calculate $\nabla F(X)$ and $\nabla g_j(X)$
6: Determine a usable-feasible search direction, $S^q$
7: Perform a one-dimensional search to find $\alpha^*$
8: Set $X^q = X^{q-1} + \alpha^* S^q$
9: Check convergence to the optimum. If satisfied, exit otherwise go to step 2.

# CHAPTER 6

# RESULTS

## 6.1 Test Case

In this thesis, transonic test case ARA-M100 is used in order to investigate the efficiency of different sensitivity approaches and solution strategies. ARA-M100 model is one of the test/validation cases of NASA CFL3D solver. It consists of wing and body on three-dimensional structured C-H grid topology on wind tunnel with sting. The original grid has dimensions of $321 \times 57 \times 49$ in computational coordinates $\xi$, $\eta$ and $\zeta$. Test case solution is result of the Navier-Stokes equations with Spalart-Allmaras turbulence model. Therefore, quite fine mesh is used in RANS solution to meet requirements of Euler grid.

The wing has a strong normal shock on upper surface and shock-free lower surface at free stream conditions where $M_\infty = 0.8027$ and $\alpha = 2.873$ degree. Therefore, it is difficult to solve the problem accurately. Since there is no Euler solution of this test case, obtained results will be compared to the RANS solutions and the experiment results. As a result of this, some differences especially on upper surface can be observed due to shock wave.

The grid independence study is applied for three different meshes; coarse grid $81 \times 21 \times 49$; medium grid $161 \times 29 \times 49$ and fine grid $321 \times 57 \times 49$ which are obtained from the CFL3D archive [2].

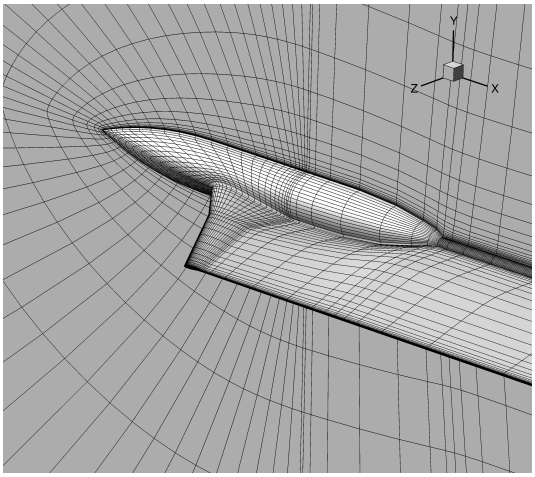



Figure 6.1: Coarse grid

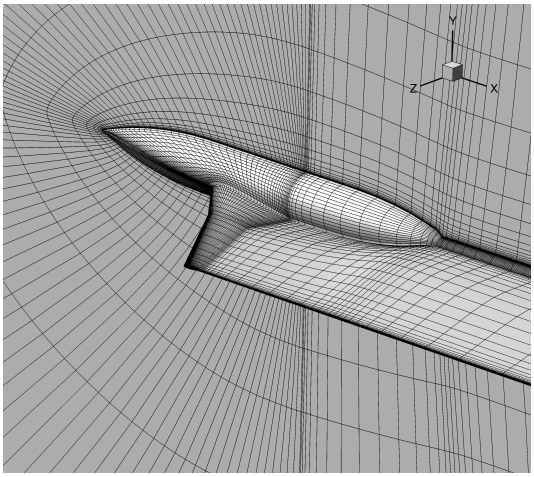



Figure 6.2: Medium grid

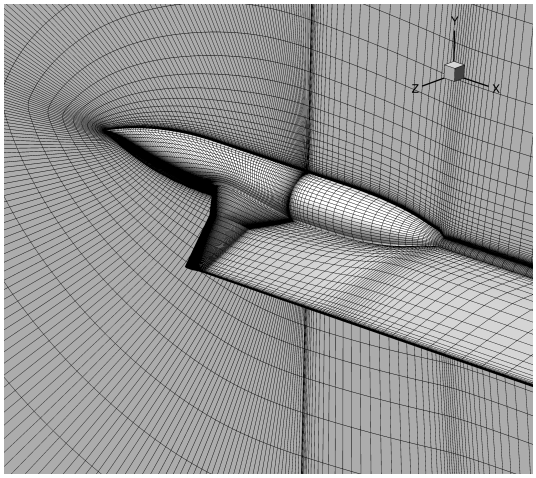



Figure 6.3: Fine grid

The grid consists of the boundary conditions of symmetry, wake, far-field and wall. The fine grid is absolutely same as CFL3D grid, the medium one is coarsened in $\xi$ direction and surface normal direction, $\eta$. For the Euler solution medium grid is sufficient especially when optimization is the main concern. Repetitive solutions can be cumbersome in terms of CPU time. Moreover, memory consumption might be a concerning problem when grid size is increased.

In addition to NASA, Epstein et. al [86] have investigated the ARA-M100 configuration at different angle of attacks (aoa) using higher-order essentially non-oscillatory scheme with the help of algebraic turbulence model Baldwin-Lomax and they have compared their results with TLNS3D code [87]. They have both used similar grid to our fine grid.

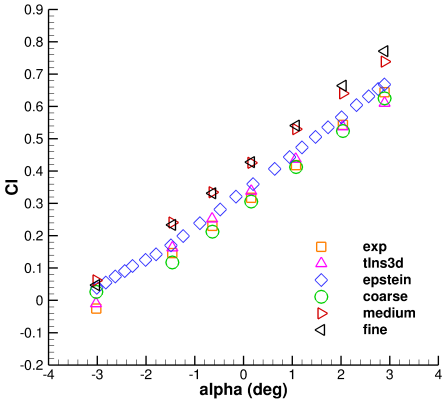



Figure 6.4: $C_L$ vs. $\alpha$ graph

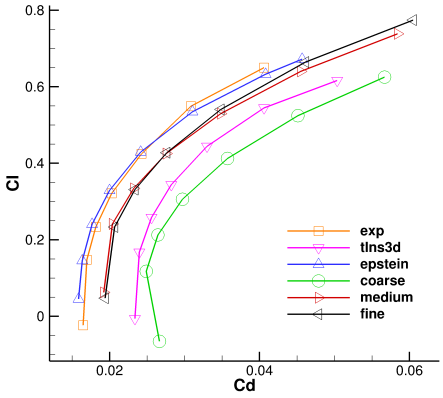



Figure 6.5: Drag polar

As seen in Figure 6.4 beside the lowest aoa, our Euler solution always overshoots the lift coefficient. Experimental and both Marconi's and Epstein's studies have lower $C_l$ values. This result is expected as in Euler equations generally shock is more stronger and it occurs at more aft position in chord-wise direction. This will cause higher lift, drag and pitching moment. In addition, fine and medium grids show good agreement for all angles of attack. They agree well for relatively small angles of attack. At higher angles of attack medium grid results slightly differ from fine grid. However, because of the computational cost and the potential memory issues on fine grid, the medium grid is used throughout the study. $C_p$ plots on different spanwise locations for medium grid are illustrated below.

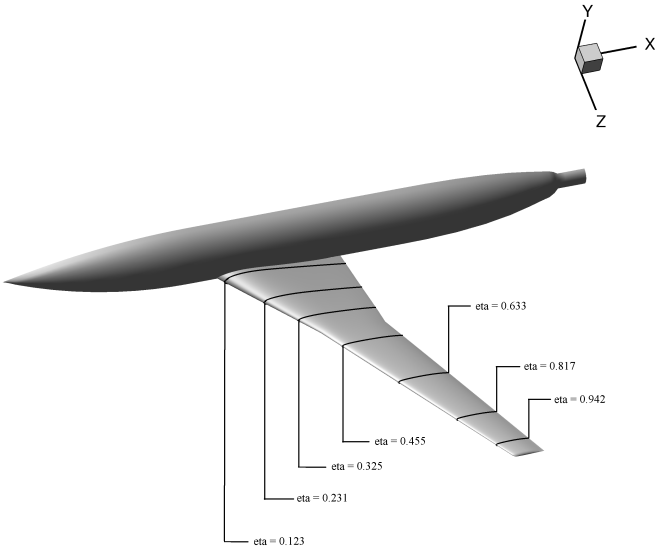
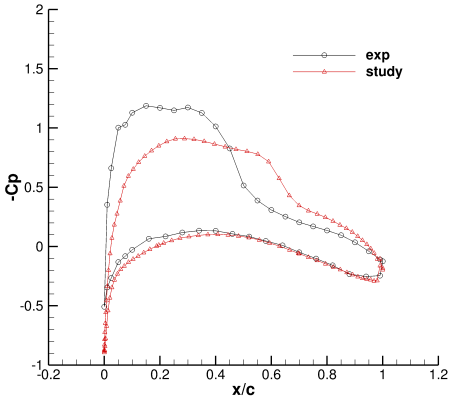
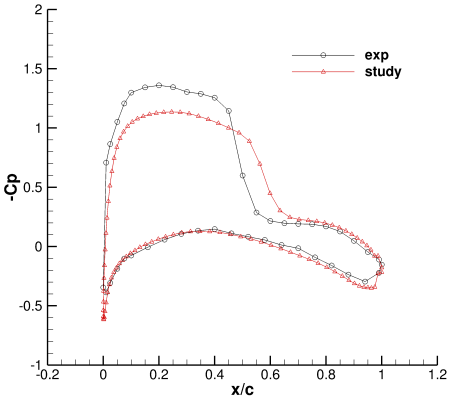


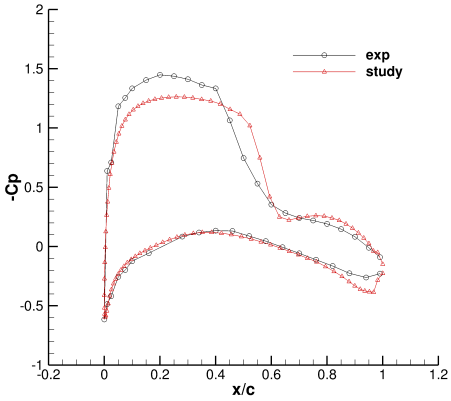

Figure 6.6: Section cuts on wing



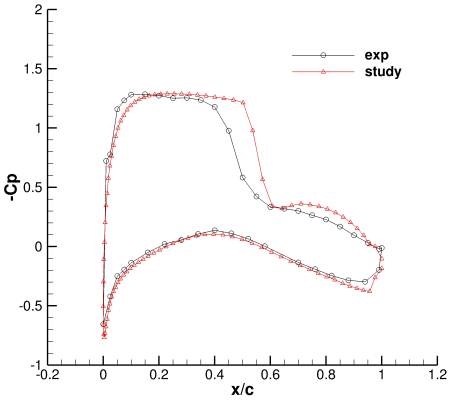

(a) $\eta = 0.123$ plane



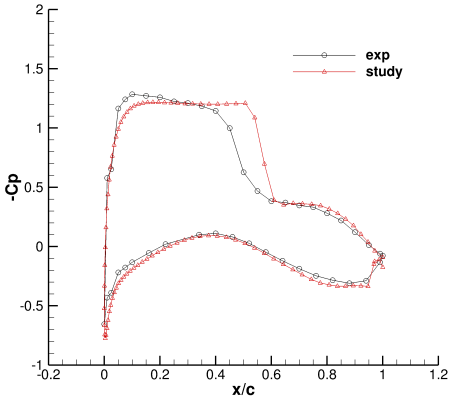

(b) $\eta = 0.231$ plane



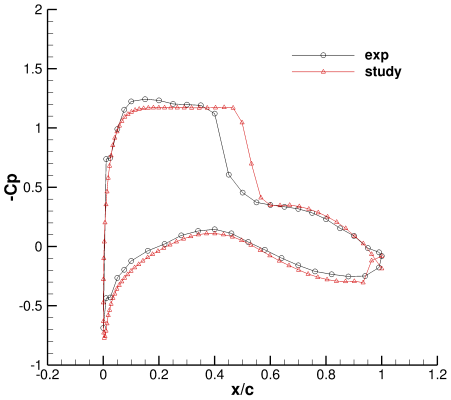

(c) $\eta = 0.325$ plane




(d) $\eta = 0.455$ plane




(e) $\eta = 0.633$ plane




(f) $\eta = 0.817$ plane

Figure 6.7: $C_p$ graphs for spanwise locations

Upper and lower surface pressure coefficients distribution are given above for

different chordwise locations from root to tip. For all sections, lower surface $C_p$ values match up with the experimental data. However, upper surface pressures are different than the pressures that are found in this study due to the presence of shock wave. In Euler solution, shock is located at more aft location of chord compared to Navier-Stokes solution. Moreover, strength of shock wave is stronger and steeper in this case. Before starting the gradient accuracy section, it is recommended to check the convergence and $C_p$ contours on this study solution according to CFL3D results. Relative convergence criteria is chosen as $10^{-9}$ for cell average continuity residual.

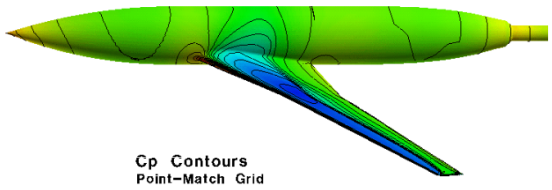



Figure 6.8: CFL3D $C_p$ contours [2]

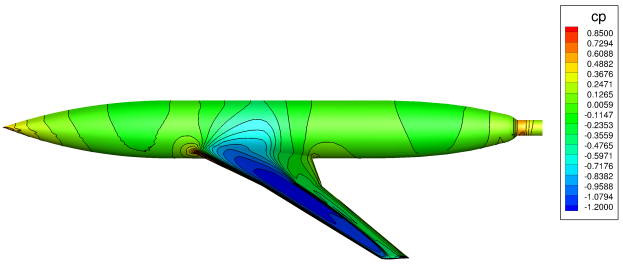



Figure 6.9: Van-Leer $C_p$ contours

Figure 6.10 shows the convergence history of this study where *rsdavg*, average continuity residual, *rsdmax* is maximum continuity residual and $C_l, C_d$ are aerodynamic coefficients. Table 6.1 shows the CFL3D and our results on force coefficients where $s_{ref} = 0.1835\ m^2$ and $c_{ref} = 0.245\ m$ for $\alpha = 2.873^0$ and $M_\infty = 0.8027$. It should be noted that body-axis coordinate system is used for pitching moment value rather than z-axis. Moment point is selected arbitrarily as no data is available.

Table 6.1: Results for CFD solution

| | CFL3D | Study |
|---|---|---|
| $C_l$ | 0.6820 | 0.7361 |
| $C_d$ | 0.0512 | 0.0580 |
| $C_m$ | 0.1437 | 0.1765 |
| $log(res)$ | -0.9e+01 | -0.9e+01 |

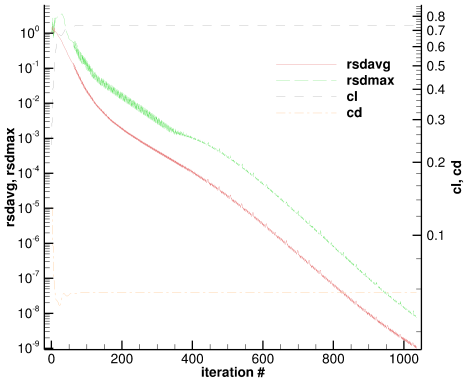



Figure 6.10: Convergence history

$C_p$ contours are very close to each other in Figure 6.8 and Figure 6.9. Both RANS and Euler solutions are good to predict shock wave on upper surface of wing. Euler solution overpredicts the aerodynamic coefficients as expected. On the other hand, the convergence of solution is steep enough even with some oscillations due to common nature of inner GMRES iterations. In Figure 6.10, green line shows maximum residual bounces at early iterations as no globalization or initial startup algorithm are applied. This phenomenon is expected and there is no direct impact on general convergence behavior as long as forcing term is chosen wisely.

## 6.2 Gradient Accuracy

In this section, accuracy of gradients will be investigated by three different approaches. It is important to show that objective and constraint functions gradi-

ents are accurate before starting the optimization procedure. Therefore, finite difference, direct and adjoint sensitivity methods will be used and compared to each other.

In aerodynamic optimization, generally objective functions are chosen as force coefficients such as lift coefficient, $C_l$, drag coefficient, $C_d$, or pitching moment coefficient, $C_m$, to increase performance of aircraft for different flight conditions and operations. Occasionally, geometric objective functions such as fuel tank volume or spar dimensions are used as objective or constraints but they are generally related to subject of structural optimization or multi-objective optimization problems. In this study, aerodynamic optimization will be focused on wing-body, so it should be meaningful to use the aerodynamic force coefficients as objective and constraint functions. Sensitivities given in Figure 6.11 correspond to surface design variables which are defined in design variable section previously. $25 \times 2 = 50$ design variables are chosen on both upper and lower surface of wing with degree 3 NURBS surfaces given in Section 4.1.1. Deflections are only allowed in y-direction to keep number of design variables small.

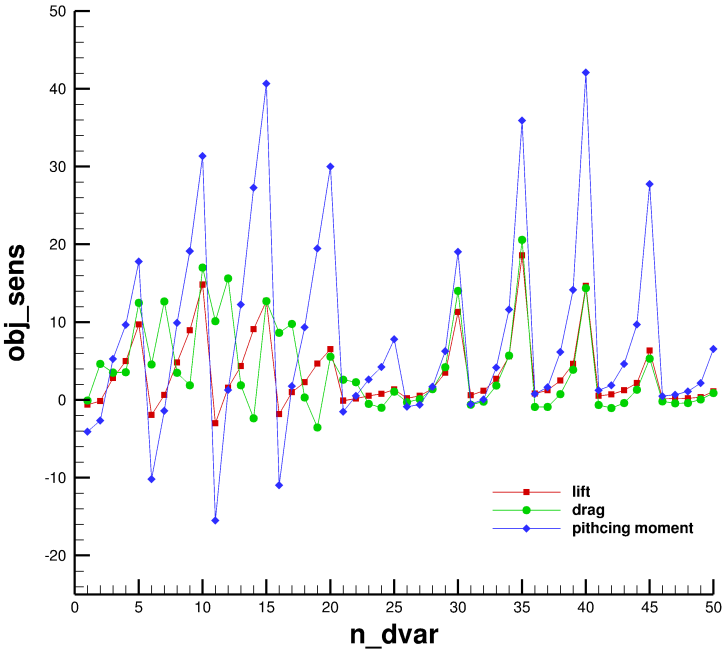



Figure 6.11: Sensitivities on surface design variables

In order to compare sensitivities normalized values are used such that objective functions are chosen $obj = \frac{Cl}{C_{l_0}}$, $obj = \frac{Cd}{C_{d_0}}$ and $obj = \frac{Cm}{C_{m_0}}$ where subscript 0 denotes aerodynamic coefficients at reference geometry. They are all normalized to 1 in order to compare difference in cost functions easily. In addition, constraint functions are normalized to zero ie. if drag is chosen as constraint then the following is used $cons = \frac{Cd}{C_{d_0}} - 1 \leq 0$. This type of choices make the comparison of aerodynamic coefficients on optimized and reference geometry easier. Moreover, it is a must for DOT optimizer due to subjects which were mentioned in Section 5.2, Equation (5.10).

In Figure 6.11 adjoint, direct and finite difference gradients are plot at the same time. However, due to accuracy in gradients the difference cannot be easily noticed. Therefore, ten different design variables are selected randomly for each objective function and compared in tables. Also % differences correspond to finite difference gradients are given to make comparison complete.

Table 6.2: $C_l$ sensitivity comparison

| | adjoint | direct | finite difference | $\Delta(AD)$ % | $\Delta(DIR)$ % |
|---|---|---|---|---|---|
| 1 | -0.6098769168 | -0.6098769168 | -0.6097683787 | 0.01780 | 0.01780 |
| 6 | -1.9184382805 | -1.9184382805 | -1.9184226576 | 0.00081 | 0.00081 |
| 11 | -2.9827437481 | -2.9827437481 | -2.9826753318 | 0.00229 | 0.00229 |
| 18 | 2.2851706407 | 2.2851706407 | 2.285644991 | -0.02075 | -0.02075 |
| 23 | 0.5322094914 | 0.5322094914 | 0.5323821999 | -0.03244 | -0.03244 |
| 26 | 0.2025730911 | 0.2025730911 | 0.2028785659 | -0.15057 | -0.15057 |
| 32 | 1.1660273409 | 1.1660273409 | 1.1662627845 | -0.02019 | -0.02019 |
| 37 | 1.2628239603 | 1.2628239603 | 1.2628195788 | 0.00035 | 0.00035 |
| 41 | 0.5180330468 | 0.5180330468 | 0.518275164 | -0.04672 | -0.04672 |
| 45 | 6.3665620305 | 6.3665620305 | 6.366991368 | -0.00674 | -0.00674 |

As seen in Table 6.2 direct and Adjoint sensitivities are completely matched until ten digit decimal. Finite difference gradients change after 3 digit decimal which correspond to ($< 0.1\%$) difference for most of the design variables.

Drag and pitching moment gradients are also given in tables below.

Table 6.3: $C_d$ sensitivity comparison

| | adjoint | direct | finite difference | $\Delta(AD)$ % | $\Delta(DIR)$ % |
|---|---|---|---|---|---|
| 4 | 3.5495734455 | 3.5495734954 | 3.5501009532 | -0.01486 | -0.01486 |
| 10 | 17.02600704 | 17.026007076 | 17.0263162955 | -0.00182 | -0.00182 |
| 11 | 10.1301277182 | 10.1301276829 | 10.1301925765 | -0.00064 | -0.00064 |
| 19 | -3.5415569679 | -3.541556952 | -3.5415182021 | 0.00109 | 0.00109 |
| 25 | 1.0716031572 | 1.0716031951 | 1.0721046575 | -0.04678 | -0.04677 |
| 27 | 0.1053596677 | 0.1053596998 | 0.1053627955 | -0.00297 | -0.00294 |
| 35 | 20.5822687783 | 20.5822687376 | 20.5834487971 | -0.00573 | -0.00573 |
| 42 | -1.0321008493 | -1.0321008218 | -1.0324572554 | -0.03452 | -0.03452 |
| 44 | 1.3048270343 | 1.3048270306 | 1.3042490362 | 0.04432 | 0.04432 |
| 50 | 0.8884147809 | 0.8884147963 | 0.8880559293 | 0.04041 | 0.04041 |

Table 6.4: $C_m$ sensitivity comparison

| | adjoint | direct | finite difference | $\Delta(AD)$ % | $\Delta(DIR)$ % |
|---|---|---|---|---|---|
| 2 | -2.6337136711 | -2.6337136501 | -2.6335640503 | 0.00568 | 0.00568 |
| 6 | -10.1760565762 | -10.176056493 | -10.1752690058 | 0.00774 | 0.00774 |
| 12 | 1.2741748617 | 1.2741746861 | 1.2762033343 | -0.15895 | -0.15896 |
| 14 | 27.2962541299 | 27.2962540726 | 27.2964544234 | -0.00073 | -0.00073 |
| 17 | 1.783636509 | 1.7836364141 | 1.7877238868 | -0.22864 | -0.22864 |
| 27 | -0.5932178663 | -0.5932178437 | -0.5912149267 | 0.33878 | 0.33878 |
| 38 | 6.1696333293 | 6.1696333693 | 6.1716662841 | -0.03294 | -0.03294 |
| 43 | 4.6464533788 | 4.6464534075 | 4.6505283 | -0.08762 | -0.08762 |
| 44 | 9.7083098747 | 9.7083098254 | 9.7081320139 | 0.00183 | 0.00183 |
| 49 | 2.1882579693 | 2.1882579159 | 2.1916902398 | -0.15660 | -0.15661 |

It can be deduced that, adjoint and direct gradients perfectly match for three cases. Moreover, their difference with finite difference gradients are very small which are not greater than 0.33% and for most of the design variables they are less than 0.01%. It is also important that finite difference sensitivities can not be completely accurate. Their accuracy can change according to finite difference order and step size used in perturbation. For example in this study, first-order difference with $\epsilon = 4 \times 10^{-8}$ is used. The use of central difference scheme or smaller perturbation value may decrease the difference of FD gradients with others. However, even for 50 design variables, $50 + 1$ flow solutions are needed to obtain FD sensitivities. This requires large computational time, and it is a big burden for repeated sensitivity calculations in optimization procedure.

In Table 6.5 wall clock times are given in sensitivity calculation procedures. Table indicates the required time in seconds. RBF interpolation time is the time that is passed on singular value decomposition of RBF weight matrix. Since direct solver is used to solve linear system, reordering, factorization and back substitution times are also given in PARDISO part. Finite difference solution does not require linear system solution; therefore, it is not taken into calculation. However, repetitive solution times are reflected at wall clock time. Miscellaneous time is indicates the initialization, post process, allocation of arrays time which are small comparing to other contributions. For comparison purposes, pie charts of time passed in both adjoint and direct methods are also plot.

Table 6.5: Wall clock time for sensitivity calculations

| | **Adjoint** | **Direct** | **Finite Difference** |
|---|---|---|---|
| RBF interpolation | 74.08 | 70.41 | 85.29 |
| Flow solution | 2999.1 | 2543.6 | 151435 |
| Pardiso (reorder) | 51.43 | 61.04 | - |
| Pardiso (factorization) | 7073.95 | 9593.07 | - |
| Pardiso (back subs.) | 177.57 | 2205.96 | - |
| Miscellaneous | 133.87 | 111.92 | 5913.71 |
| **Total** | 10510 | 14586 | 157434 |

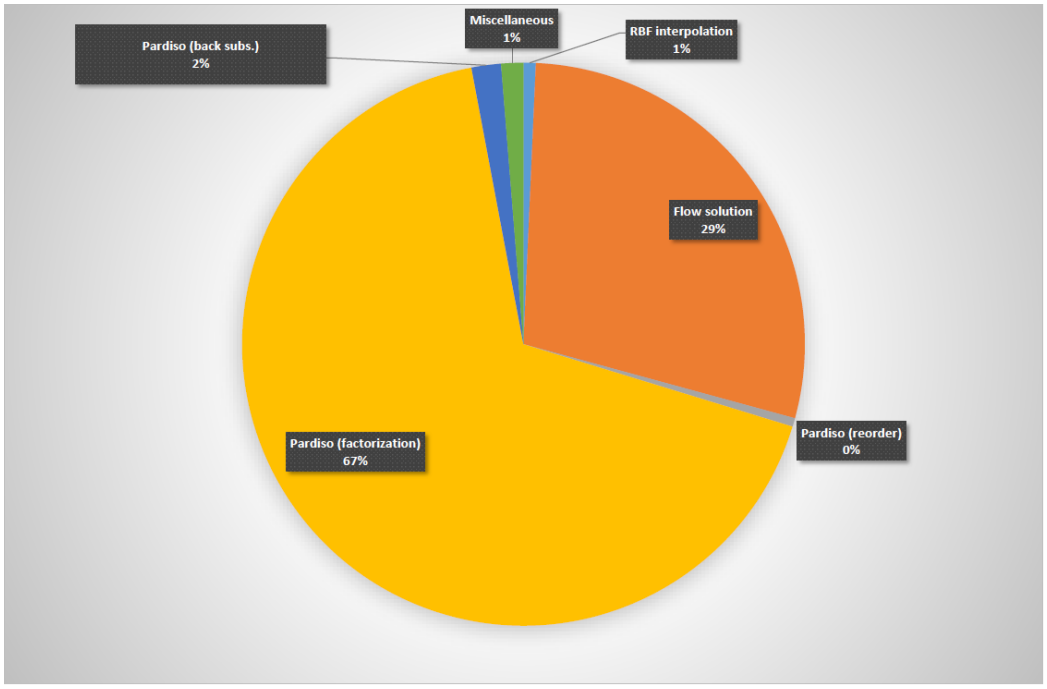



Figure 6.12: Adjoint sensitivity time table

As seen in Figure 6.12 the time dominated in sensitivity calculations is due to PARDISO solver factorization (2/3 of total time). The reordering time is very low and generally back substitution does not require time that is greater than 120 seconds. Another important time piece is due to flow solver as expected. However, because of the direct solver use, factorization is critical point in terms of CPU and memory.

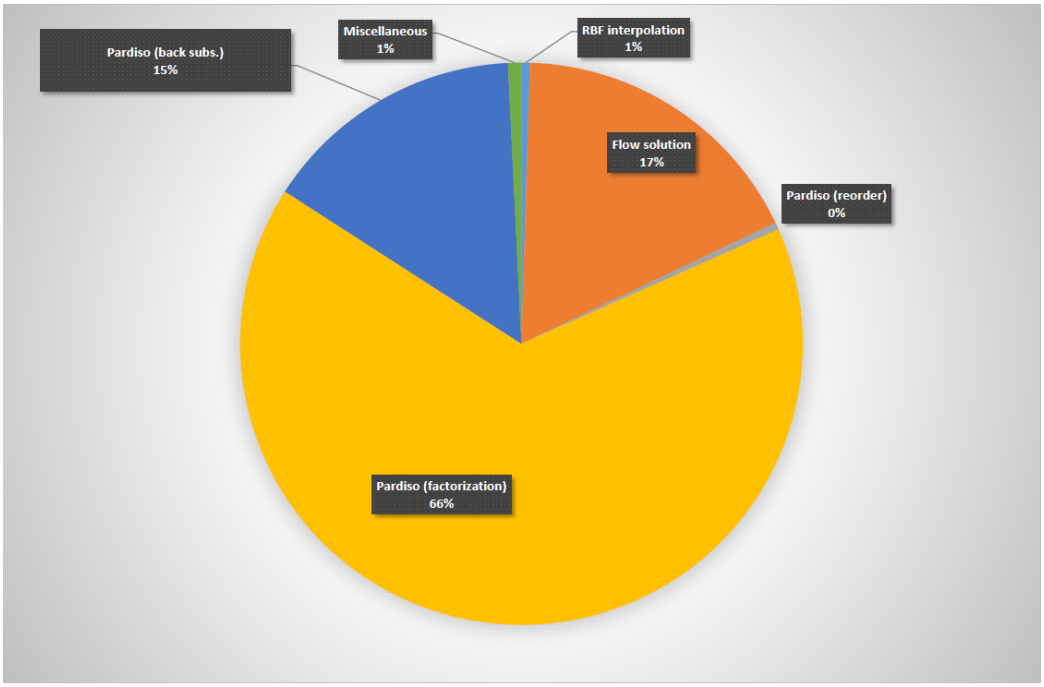



Figure 6.13: Direct sensitivity time table

The crucial point in Figure 6.13 is the region dominated by back substitution which is nearly equal to flow solution time. As it was explained in Equation (5.3), direct sensitivity computation requires the solution of number of design variable (ndv) right hand side vector. This means ndv times back substitution are required. For even larger design variables, this time may dominate the factorization time. Hence, adjoint method is the most preferred and efficient way of calculation of gradients in terms of both wall clock time and accuracy.

## 6.3 Optimization Results

For optimization 3 different design variables type are chosen. The NURBS control points of wing surface, 2D body NURBS control points and wing leading edge sweep angle points are the design variable types that are used in this scope. The details of design variables selection can be found in Chapter 4. Design variable choices and their performances on different objective functions and constraints will be focused. The detail of optimization cases will be investigated in next sections centering upon optimized shape, $C_p$ and Mach contours of reference and optimized geometry and the performance on off-design conditions. Single point optimization is carried out only for $\alpha = 2.873^0$ and $M_\infty = 0.8027$.

### 6.3.1 NURBS surface on wing

In this section, lift will be maximized by minimizing drag and pitching moment coefficients separately on wing-body configuration using NURBS control points of wing surface. 100 design variables are chosen as stated previously for this case. Control net is composed of $5 \times 5$ control points on upper and lower surface of wing with all weights are initialized with 1.0 and weights are kept constant throughout the optimization iterations.

#### 6.3.1.1 Lift maximization case with drag and pitching moment constraints

Problem statement can be described with subscript 0 denoting the value at first design cycle.

$$\begin{aligned} \text{Maximize} \quad & \frac{C_l}{C_{l_0}} \\ \text{Subject to;} \quad & \frac{C_d}{C_{d_0}} - 1 \leq 0.001 \\ & \frac{C_m}{C_{m_0}} - 1 \leq 0.001 \\ & -1.5 \times 10^{-3} \leq X_i \leq +1.5 \times 10^{-3} \quad i = 1, ..., 100 \end{aligned}$$

In this case, lift is tried to be increased with decreased drag and pitching moment by the help of inequality constraints. $CT = 0.001$ is defined for numerical precision issue in DOT. The side constraints are added to the problem in order to avoid unnecessary iterations. At the end, the optimized design variables are far from the side constraints so side constraints have no effect on optimized shape. Convergence history for optimization iterations is given in Figure 6.14 while *ndcyc* corresponds to number of design cycles. The change in objective

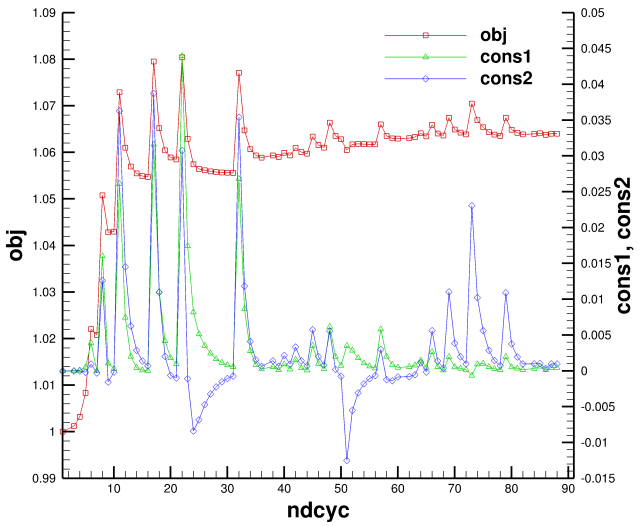



Figure 6.14: Convergence history of lift maximization case

function and constraints can be seen in figure. The objective function is initially equal to 1 since normalized values are used. Also, constraints are starting from 0. At the end of iterations, lift coefficient is increased beyond 6% and constraints are not violated. Table for aerodynamic coefficients is given also.

Table 6.6: Results for lift maximization case

| | Initial | Optimized |
|---|---|---|
| Obj | 1.0000000 | 1.0640230 |
| Cons1 | 0.0000000 | 0.0004465 |
| Cons2 | 0.0000000 | 0.0009884 |
| Cl | 0.7361341 | 0.7832637 |
| Cd | 0.0580534 | 0.0580793 |
| Cm | 0.1765676 | 0.1767421 |

Pressure coefficients and Mach contours are drawn on upper surface of wing-body where the normal shock is present.

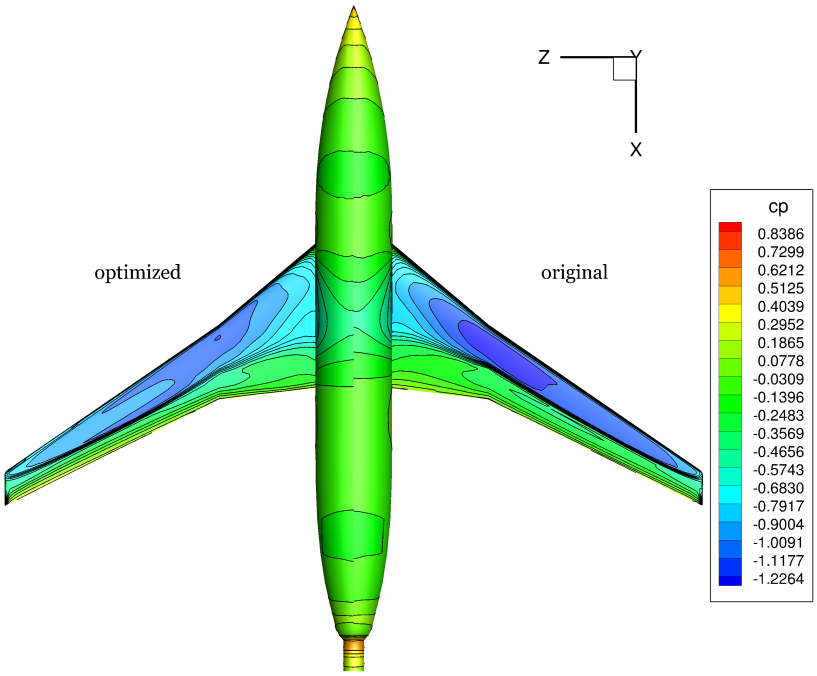



Figure 6.15: Lift maximization $C_p$ contours on upper surface

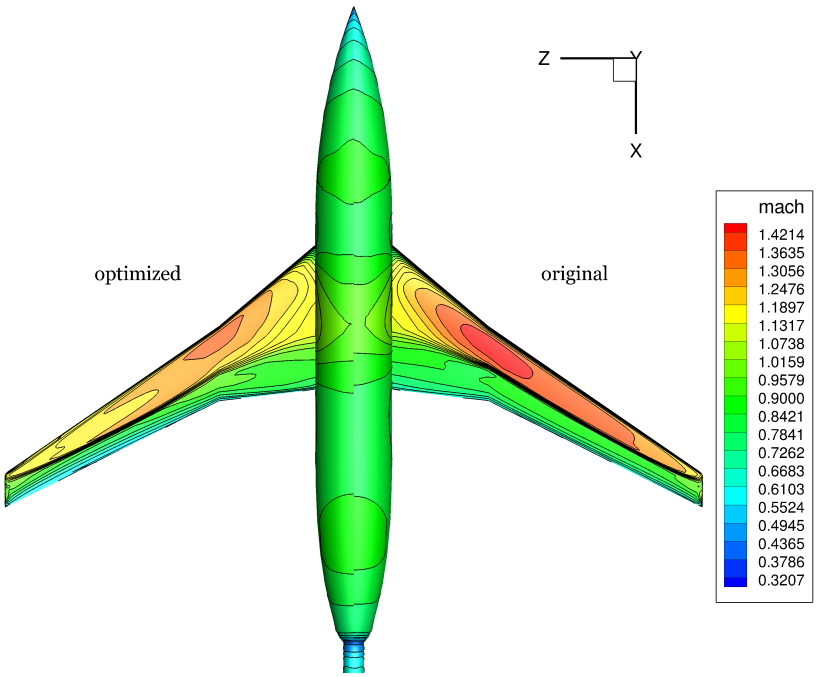



Figure 6.16: Lift maximization Mach contours on upper surface

As seen in Figures 6.15 and 6.16 density of contour lines are decreased around mid-chord and especially wing tip. Although shock still exists on upper surface, its strength is decreased to increase lift coefficient and to keep drag in allowable level. Maximum local Mach number on upper surface in optimized geometry is decreased and further increase in Mach number is prohibited. In addition to previous benefits of optimized geometry, wing tip vortices are reduced to increase lift coefficient by decreasing the effect of induced angle of attack. Spanwise section cuts are taken at different spanwise locations and given below.

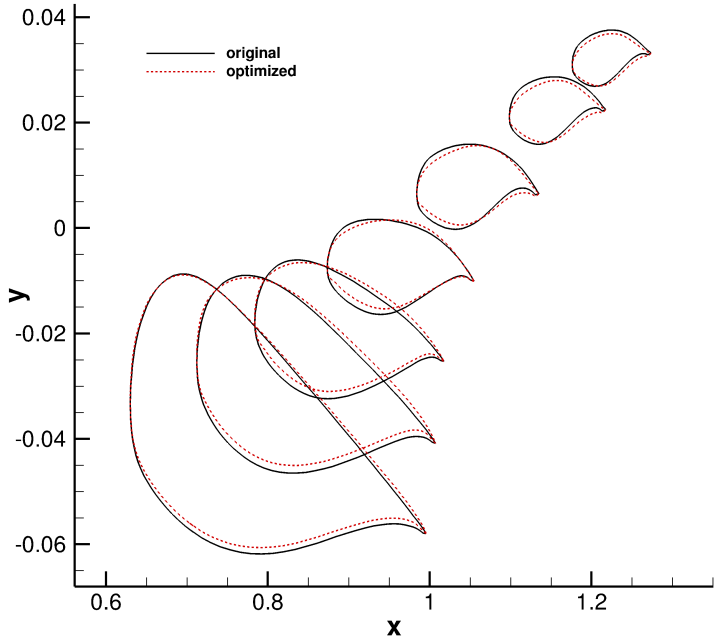
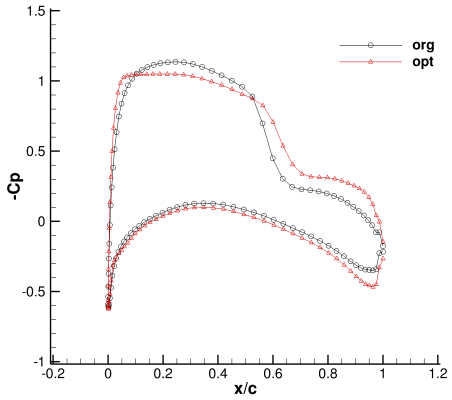
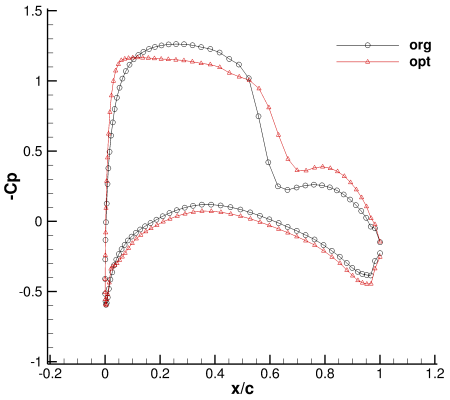



Figure 6.17: Airfoil sections on spanwise locations

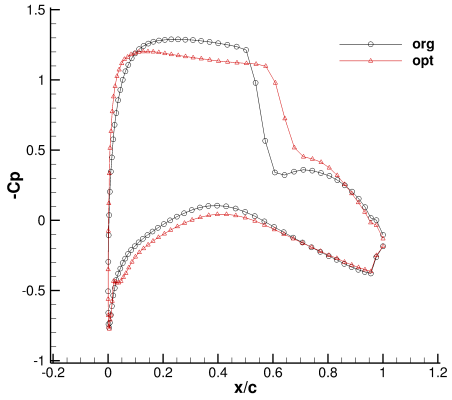
In Figure 6.17 airfoil sections are extracted from root to tip. The outermost airfoil section corresponds to tip airfoil while innermost indicates the root airfoil. Red sections show the optimized geometry and black sections belong to reference geometry. While in root airfoils cambers are located at more front, at the tip sections they are located at back. Moreover, upper surface becomes more flat and trailing edges become thicker compared to reference geometry. Reduced curvature of the mid-chord provides reduction of the Mach number before shock, Figure 6.16. The distribution of pressure coefficients will be given in next figures.



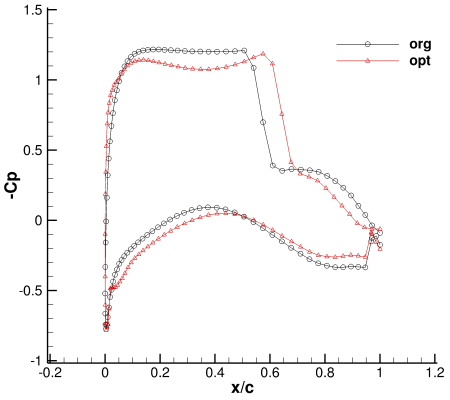

(a) $\eta = 0.231$ plane



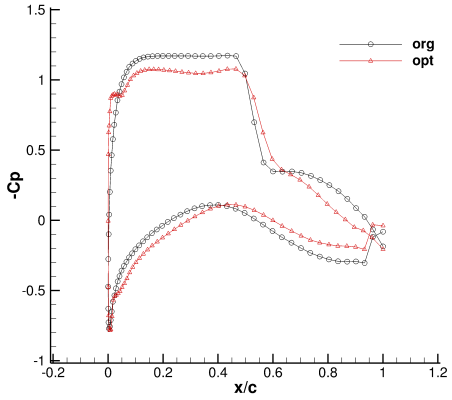

(b) $\eta = 0.325$ plane



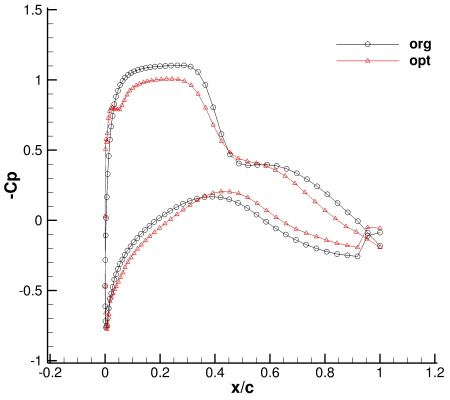

(c) $\eta = 0.455$ plane




(d) $\eta = 0.633$ plane




(e) $\eta = 0.817$ plane




(f) $\eta = 0.942$ plane

Figure 6.18: $C_p$ graphs for spanwise locations

From Figure 6.18a to Figure 6.18d shock moves towards the trailing edge and its strength is reduced near root and tip sections. Flat surface of new wing tip decreases the effect of wave drag by preventing acceleration of local flow. In addition, strength of shock is highly reduced. Leading edges of lower surface are another critical point that most of the pressure coefficient change is observed at wing tip. The difference between $C_p$ values are decreased through the tip. This means wing tip vortices are also reduced. Consequently, we can generalize the trend in lift maximization case as shock dislocation and lower surface pressure coefficients minimization with shock damping at root and tip. The performance at off-design conditions is also important issue. Since optimization is held for only Mach number 0.8027 and $\alpha = 2.873^0$ and by knowing aircraft has different operation conditions, it is better to look at drag polar and lift curve at off-design conditions also.

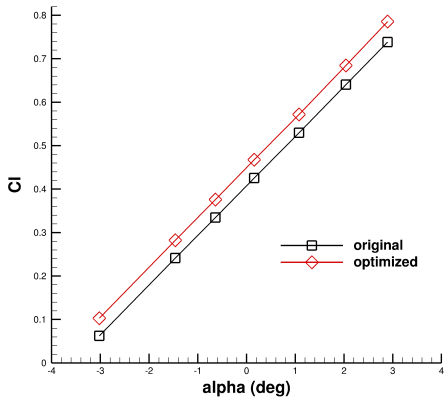



Figure 6.19: $C_l - \alpha$ off design conditions

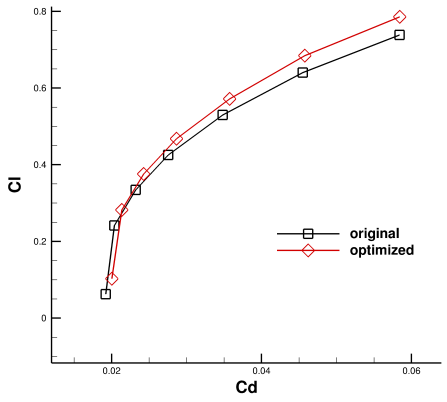



Figure 6.20: $C_l - C_d$ off design conditions

As it can be seen in Figure 6.19 lift curve slope is maintained and lift curve is translated upward for all angles of attack, so that the lift increment is satisfied at off-design conditions. In Figure 6.20 we expect the shift in drag polar in upward direction. For relatively large angles of attack drag is kept constant and lift is increased. However, the little drag increment is also observed at lower angles of attack. Unfortunately, for negative aoa optimized geometry shows poor performance.

#### 6.3.1.2 Drag minimization case with lift and pitching moment constraints

Similar to previous optimization case, drag minimization problem is defined as follows.

$$\begin{aligned} \text{Minimize} \quad & \frac{C_d}{C_{d_0}} \\ \text{Subject to;} \quad & 1 - \frac{C_l}{C_{l_0}} \leq 0.001 \\ & \frac{C_m}{C_{m_0}} - 1 \leq 0.001 \\ & -1.5 \times 10^{-3} \leq X_i \leq +1.5 \times 10^{-3} \quad i = 1, ..., 100 \end{aligned}$$

Convergence history through optimization and final results are given in figures.

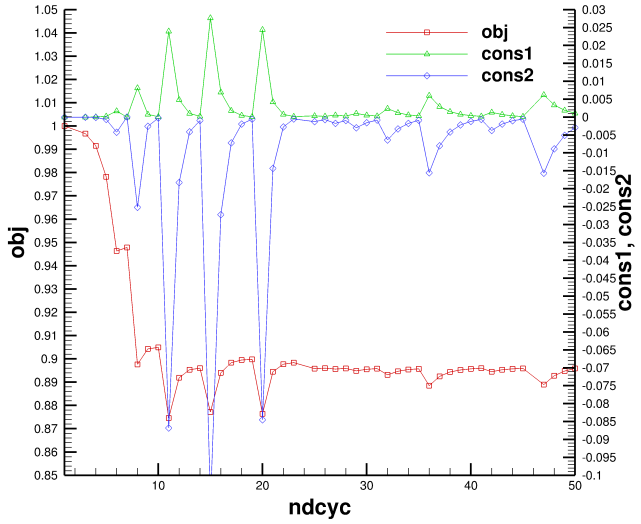



Figure 6.21: Convergence history of drag minimization case

Results show that drag is reduced by 10.5% with 0.03% lift reduction and 0.09% pitching moment reduction.

Table 6.7: Results for drag minimization case

| | Initial | Optimized |
|---|---|---|
| Obj | 1.0000000 | 0.8957720 |
| Cons1 | 0.0000000 | 0.0003247 |
| Cons2 | 0.0000000 | -0.0009067 |
| Cl | 0.7361341 | 0.7358951 |
| Cd | 0.0580534 | 0.0520026 |
| Cm | 0.1765676 | 0.1764075 |

Mach and $C_p$ contours are drawn for better comparison on optimized and reference geometry. From the figures given below, one can see that shock is delayed

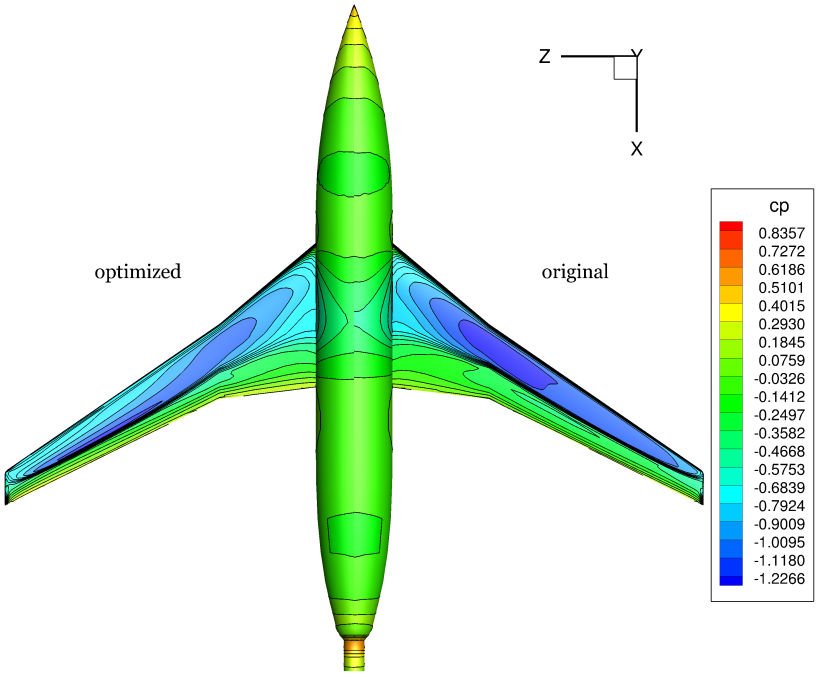



Figure 6.22: Drag minimization $C_p$ contours on upper surface

to more aft location and its strength is reduced. At the leading edge acceleration of local flow is prevented. In addition $C_p$ values at the tip of wing is reduced to suppress wing tip vortices. Mach contours density minimization at crank location is visible. This will also help to decrease in shock strength at that region.

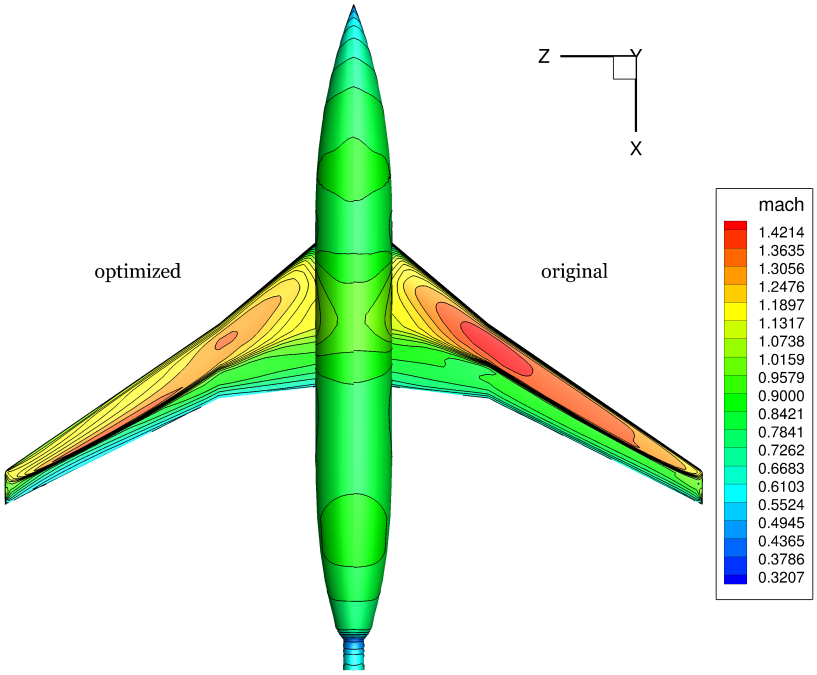
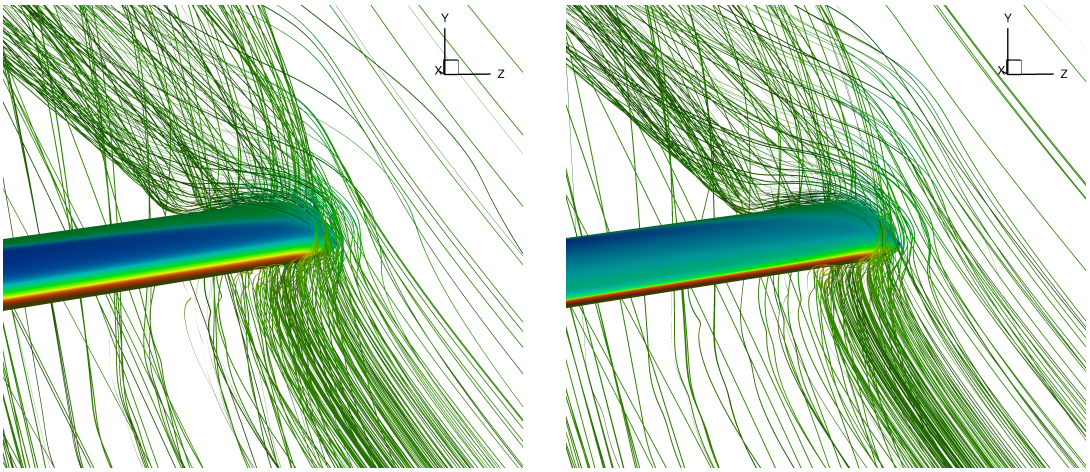



Figure 6.23: Drag minimization Mach contours on upper surface




(a) Baseline geometry (b) Optimized geometry

Figure 6.24: Wing tip vortices and $C_p$ distribution

As it is seen in Figure 6.24b, shock is removed and density of wing tip vortices are reduced for lower induced drag.

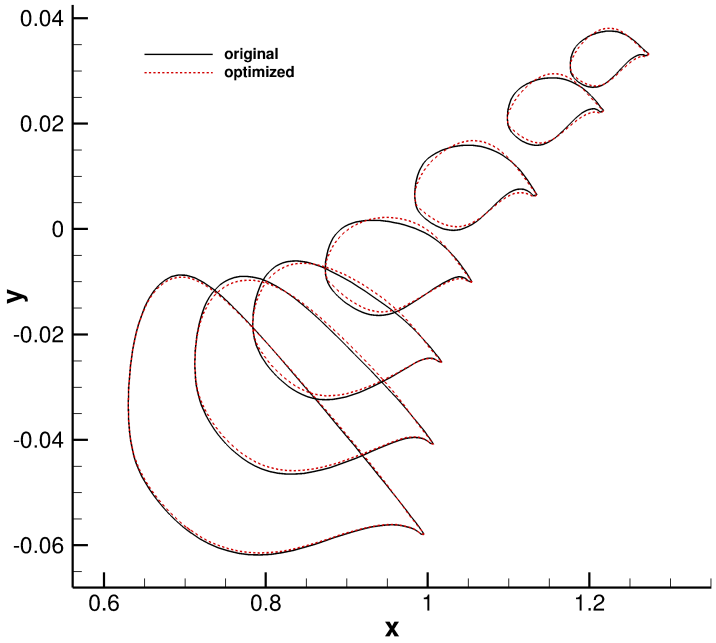
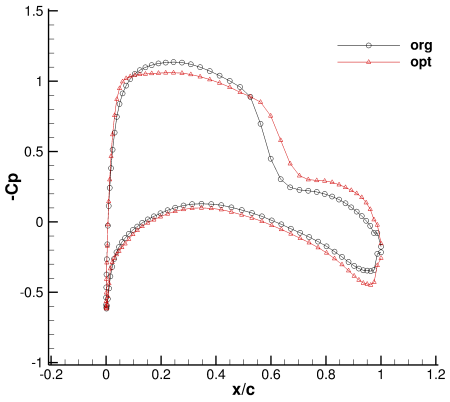


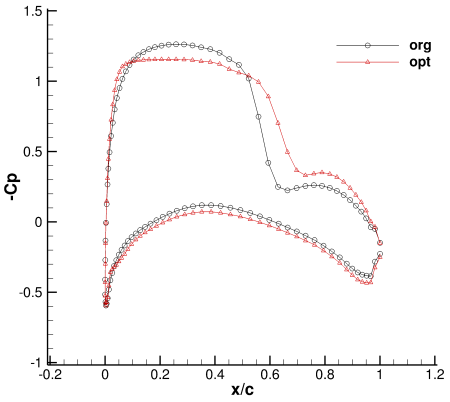

Figure 6.25: Airfoil sections on spanwise locations

As seen in Figure 6.25 curvature of leading upper sections are reduced for first 4 spanwise locations. The upper surface is now more flat compared to baseline shape and camber is shifted to trailing edge at bottom surface. This configuration is similar to supercritical airfoil design concept which provides higher drag rise Mach number since flat upper surface prevents the further acceleration of flow before the shock wave as seen in Figure 6.23. Then the strength of shock is reduced and shock is observed at more aft location so the total drag easily becomes less. From root to tip, flat upper surface transforms into cambered airfoil sections. This is well-known phenomenon which shows the importance of crest location on airfoil when flow is nearly transonic. The crest is defined as the location where incoming flow becomes tangent. In our design point, free-stream flow has 2.8027 angle of attack. In optimized sections, crest location moves backward by creating the camber at further aft location. Thus, supersonic local velocity which occurs ahead of crest does not lead to extreme drag increment because the local flow velocity decreases after the crest which can be seen quite easily after midspan of Figure 6.23 of optimized shape. If we look at the all sections from root to tip, we can see the pattern of airfoil section changes. Not

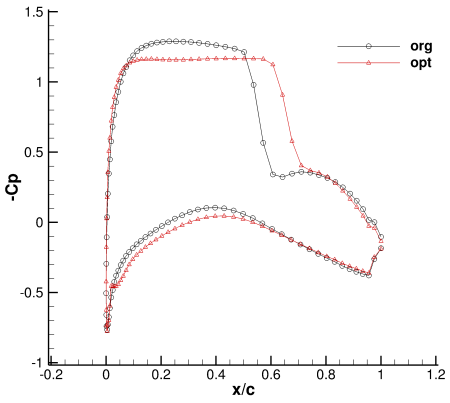
a real twist but artificial twist can be easily distinguished as the tip sections are twisted more downward. Hence both reduction of wing tip vortices and increment in span efficiency are succeeded. In following figures we will clearly see the effect of this artificial twist on pressure differences of upper and lower surfaces of wing tip. $C_p$ distribution will be given in order to investigate shock pattern.



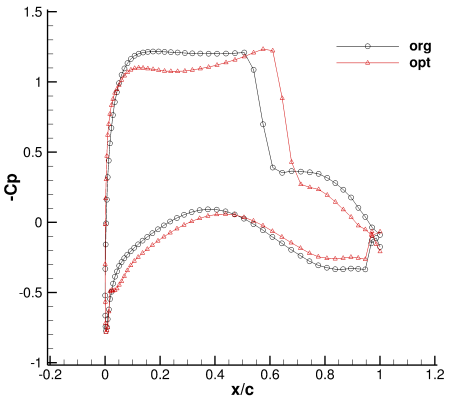

(a) $\eta = 0.231$ plane $C_p$ graph



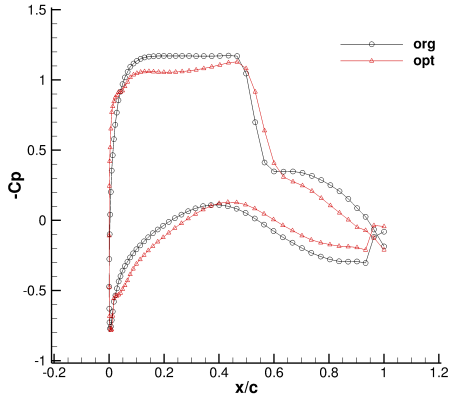

(b) $\eta = 0.325$ plane $C_p$ graph



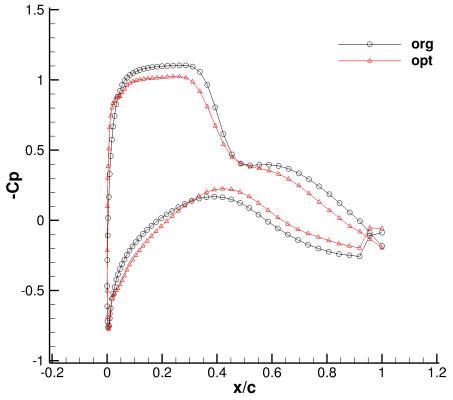

(c) $\eta = 0.455$ plane $C_p$ graph




(d) $\eta = 0.633$ plane $C_p$ graph




(e) $\eta = 0.817$ plane $C_p$ graph




(f) $\eta = 0.942$ plane $C_p$ graph

Figure 6.26: $C_p$ graphs for spanwise locations

Beside the Figure 6.26d, shock strength is reduced for all angles of attack by decreasing the maximum $C_p$ and sharpness of shock wave line, Figures 6.26a and 6.26b. As seen in Figures 6.26e and 6.26f, induced drag which is caused by pressure difference near tip is also reduced.

For off-design conditions, both drag polar and drag divergence graphs are drawn for different Mach numbers.

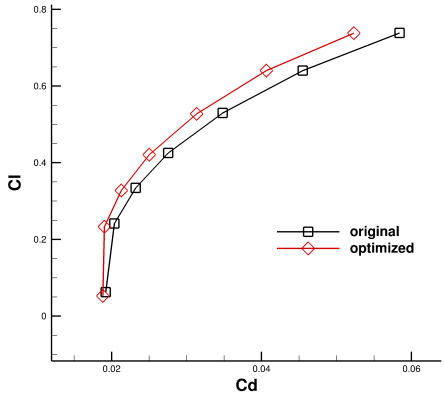



Figure 6.27: $C_l - C_d$ off-design

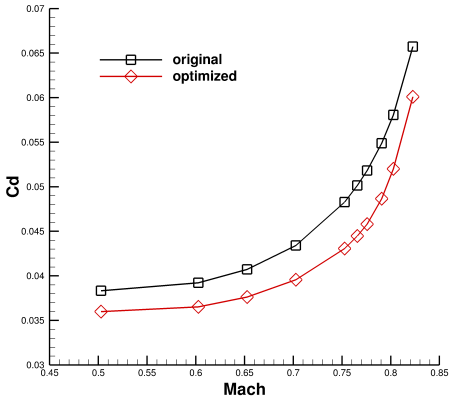



Figure 6.28: $C_d - M_\infty$ off-design

Drag polar is shifted to left as expected in Figure 6.27. Lift is only decreased at very low angles of attack. Moreover, $C_d - M_\infty$ curve is shifted downward so drag values are reduced for operating range. The highest difference in drag

values are observed beyond 0.8327 Mach which shock is stronger than the Mach 0.8027. This shows also shock strength is reduced for even higher Mach numbers. Because of the reduction in wave drag, drag divergence Mach number is also increased. As a consequence, the higher cruise speed is now possible with optimized wing. Lift distribution is also given in Figure 6.29. New spanwise distribution is more elliptic compared to baseline geometry.

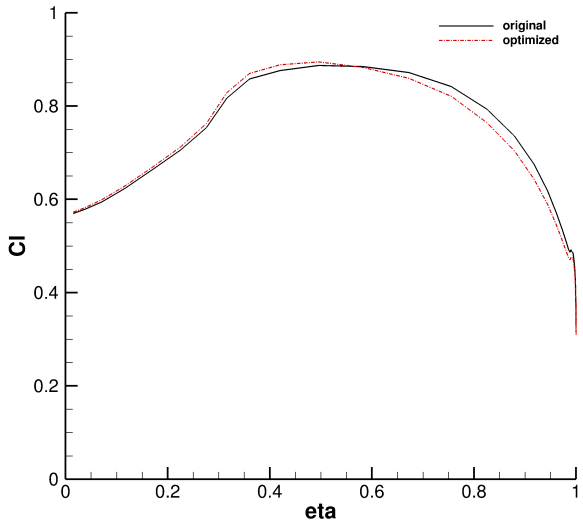



Figure 6.29: Spanwise lift distribution

As lift is decreased at lower angles of attack, it is decided to use higher degree NURBS control points to see its efficiency on off-design conditions. Then at the off design conditions some improvements on drag and lift are observed.

Table 6.8: Results for lift case

| | Initial | Optimized(d3) | Optimized(d4) |
|---|---|---|---|
| Obj | 1.0000000 | 0.8957720 | 0.8758510 |
| Cons1 | 0.0000000 | 0.0003247 | 0.0001856 |
| Cons2 | 0.0000000 | -0.0009067 | -0.0006381 |
| Cl | 0.7361341 | 0.7358951 | 0.7359974 |
| Cd | 0.0580534 | 0.0520026 | 0.0508461 |
| Cm | 0.1765676 | 0.1764075 | 0.1764549 |

In the higher degree case, optimization requires more time compared to lower degree control points because of the smoothness of new control points.

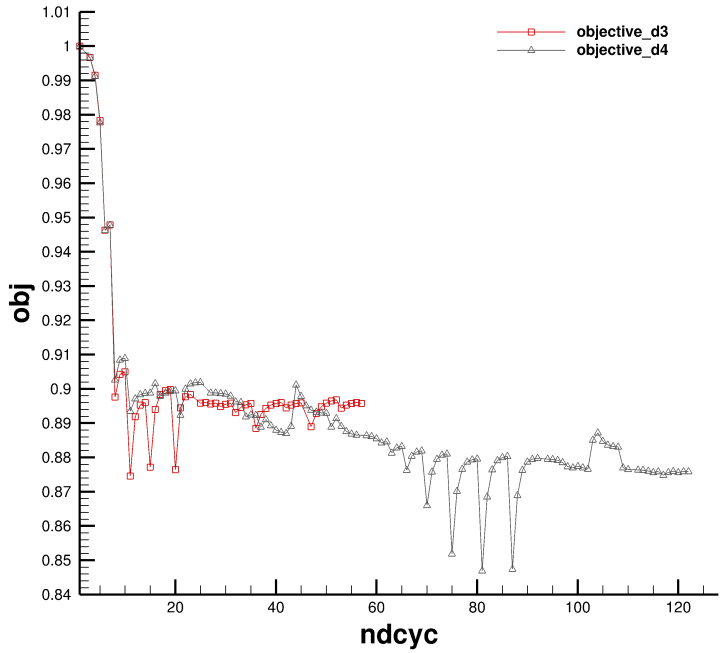
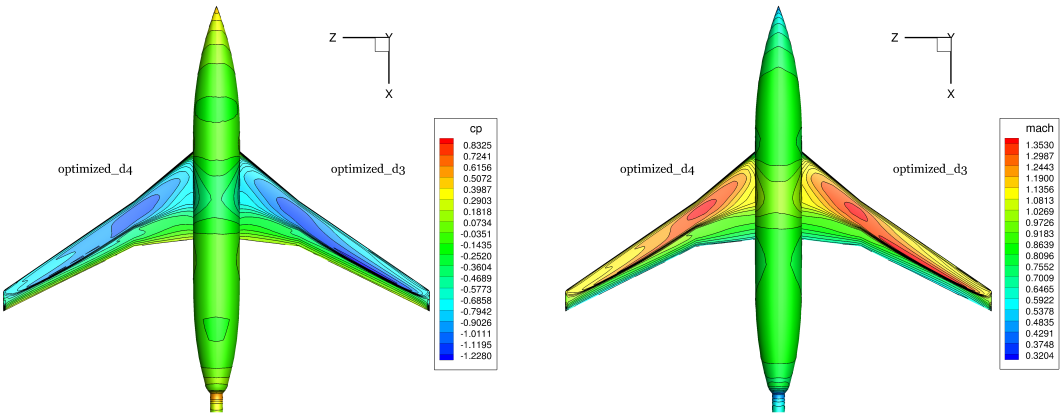



Figure 6.30: Convergence history of drag minimization case

$C_p$ and Mach contours show that the flow is accelerated near wing tip, also shock is weaken due to decreased pressure coefficients. Midspan is also effected by degree selection.




(a) $C_p$ contours on upper surface (b) Mach contours on upper surface

Figure 6.31: $C_p$ and Mach distribution on degree 3 and 4 optimization

Different choice of degree selection on wing sections can be found in Figure 6.32.

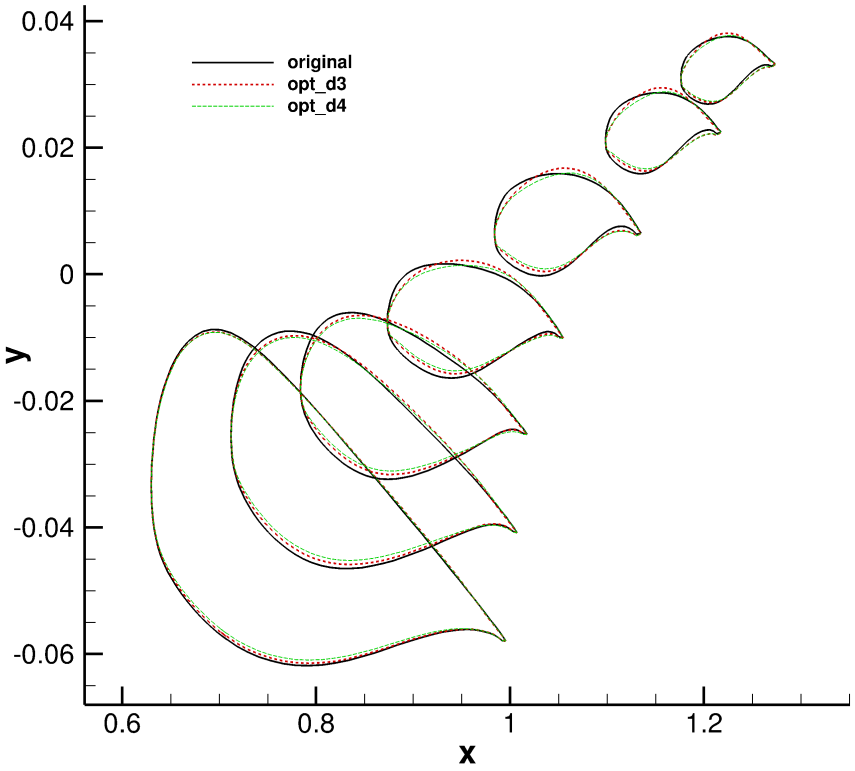



Figure 6.32: Airfoil sections on spanwise locations

Degree 4 selection results the smaller thickness and camber comparing to degree 3 choice. More smooth and aft cambered sections can be seen in after midspan locations. This will reduce the flow separation near wing tip due to high camber. The parameter selection is very important in NURBS for final geometry.

At the off design conditions better performance of 4th degree NURBS can be seen.

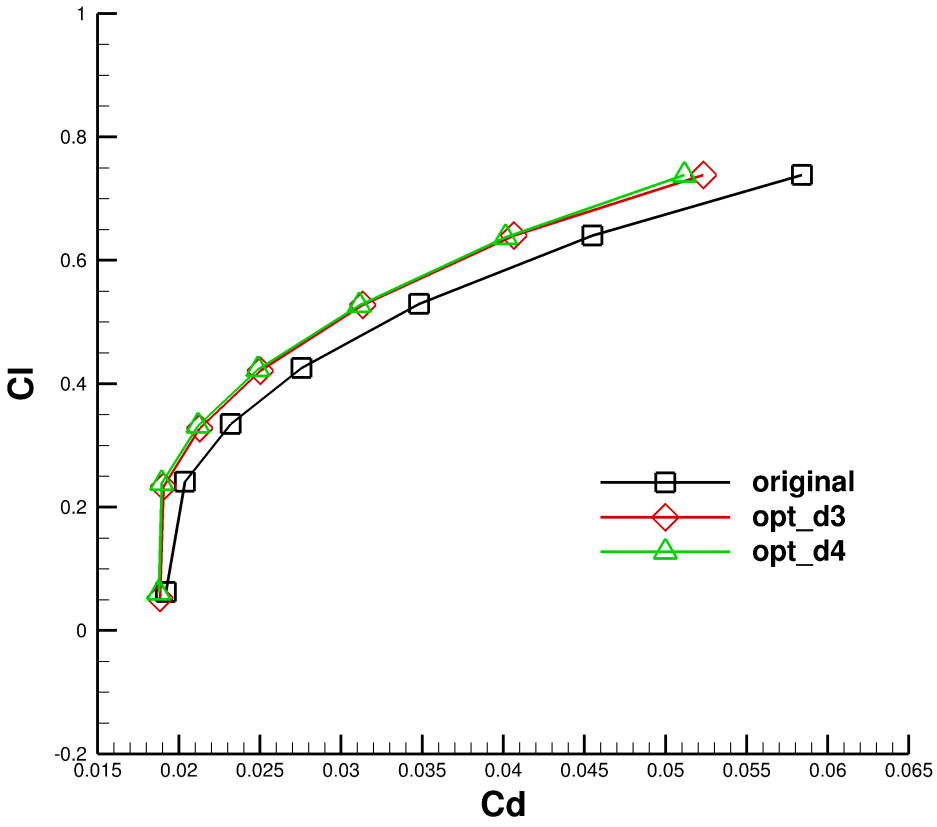



Figure 6.33: $C_l - C_d$ off-design

As illustrated in Figure 6.33 at lower angles of attack lift is conserved and drag polar is shifted for all angle of attack range. However, difference between optimized drag polar and baseline drag polar get smaller through the negative angles of attacks. This shows that when shock is not present the efficiency of optimization vanishes.

#### 6.3.1.3 Pitching moment minimization case with lift and drag constraints

Problem statement is written in this case as follows;

$$\begin{aligned} \text{Minimize} \quad & \frac{C_m}{C_{m_0}} \\ \text{Subject to;} \quad & \frac{C_d}{C_{d_0}} - 1 \leq 0.001 \\ & 1 - \frac{C_l}{C_{l_0}} \leq 0.001 \\ & -1.5 \times 10^{-3} \leq X_i \leq +1.5 \times 10^{-3} \quad i = 1, ..., 100 \end{aligned}$$

In this case pitching moment coefficient will be tried to minimized while $C_l$ and $C_d$ will be kept in tolerable levels.

Convergence history is given in Figure 6.34.

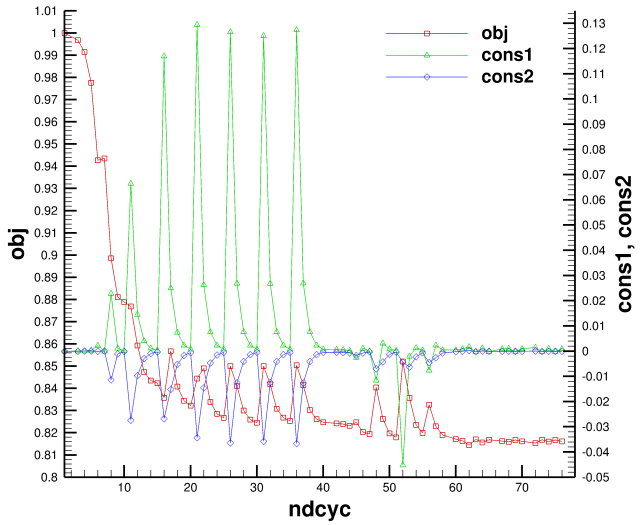



Figure 6.34: Convergence history of pitching moment minimization case

After fast decrease in 10 iterations at pitching moment, the constraints are pushed up to 0.1 and -0.03 level. Releasing constraints may result in probably less pitching moment coefficient.

Table 6.9: Results for pitching moment minimization case

| | Initial | Optimized |
|---|---|---|
| Obj | 1.0000000 | 0.8161653 |
| Cons1 | 0.0000000 | 0.0009307 |
| Cons2 | 0.0000000 | -0.0000608 |
| Cl | 0.7361341 | 0.7361790 |
| Cd | 0.0580534 | 0.0581075 |
| Cm | 0.1765676 | 0.1441084 |

The pressure coefficient and Mach contours are drawn on wing body upper surface where the shock region is visible.

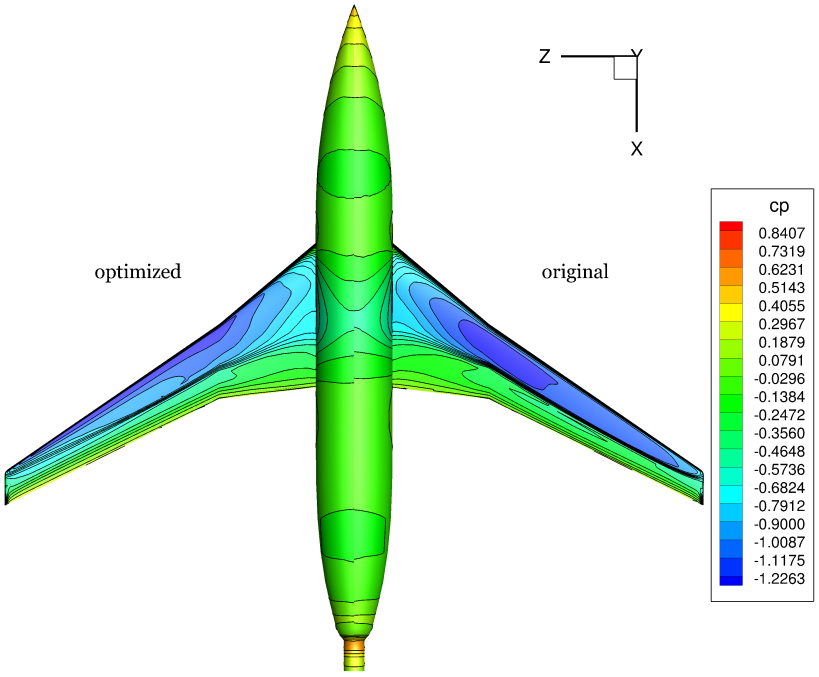



Figure 6.35: Pitching moment minimization $C_p$ contours on upper surface

Figure 6.35 shows the pressure coefficient reduction near the wing tip. Shock is completely removed after midspan. Moreover, maximum Mach number is quite reduced as seen in Figure 6.36. Section span-wise distribution is given below.

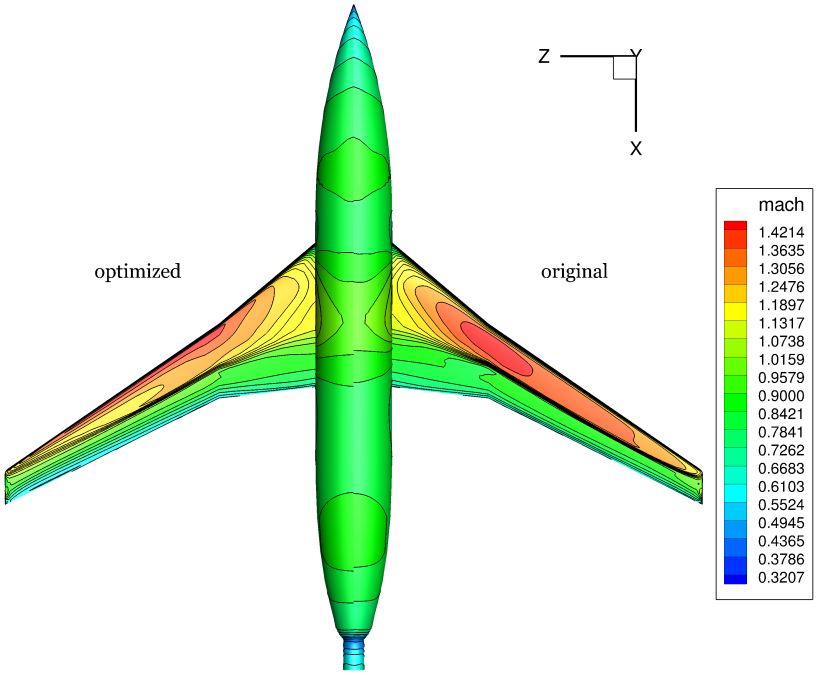



Figure 6.36: Pitching moment minimization Mach contours on upper surface

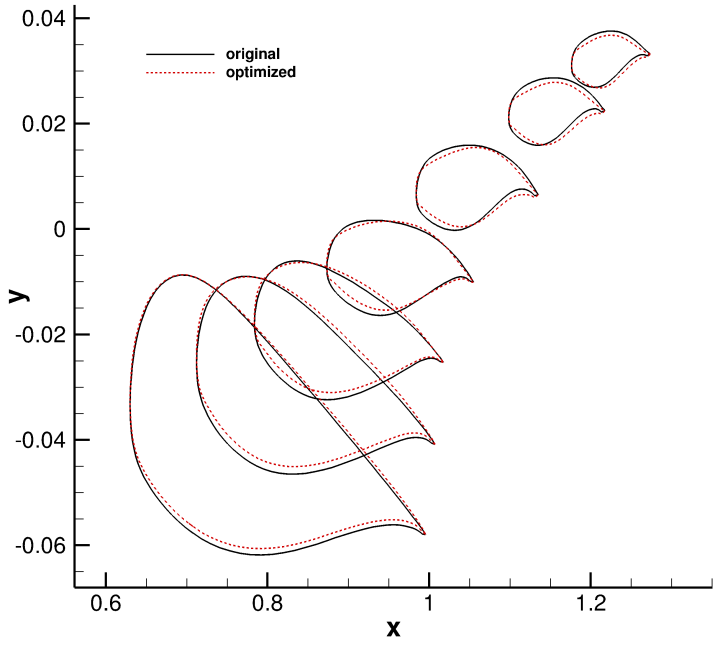
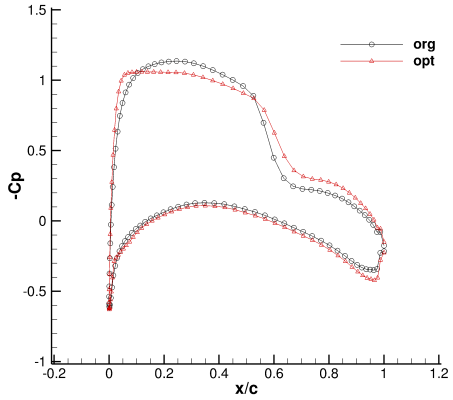



Figure 6.37: Airfoil sections on spanwise locations

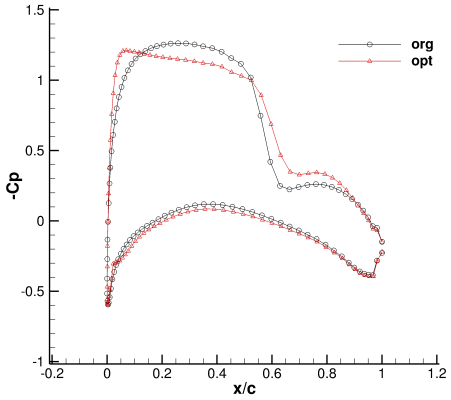
As shown in Figure 6.37 new leading edge curvature is obtained which is also

found at optimized shape of reference [32]. Unlike the supercritical airfoils, which have larger leading edge radius, pitching moment optimized geometry has lower nose radius. Also flat upper surface and aft cambered lower surface can be easily distinguished. Comparing to lift and drag optimized wing sections, pitching moment optimized airfoils are more similar to supercritical airfoils. In addition to new deformed leading edges and flat upper surface, upward bended trailing edges which are also known as reflex airfoils are present now. Reflexed airfoil shapes are very common in low Mach number flights which tailless aircrafts use this concept due to their innate positive pitching moment. The deformed leading edge is also commonly known pattern in transonic aircrafts and its curvature is usually related to optimization point angle of attack.

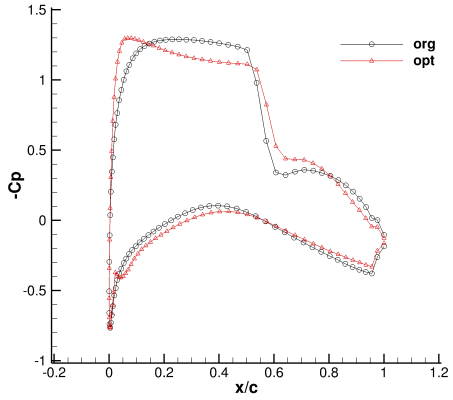
Pressure coefficient distribution is given below.



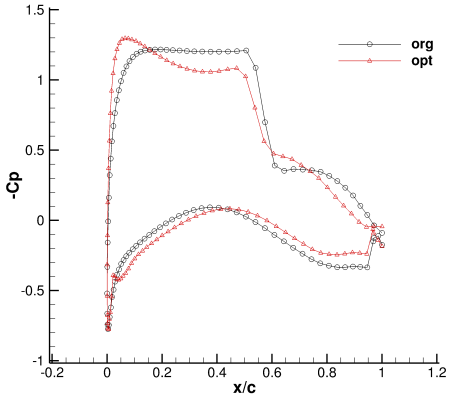

(a) $\eta = 0.231$ plane $C_p$ graph



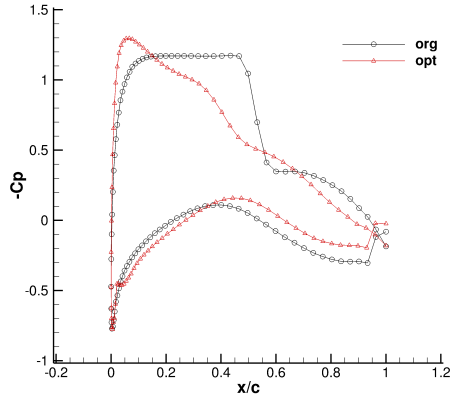

(b) $\eta = 0.325$ plane $C_p$ graph



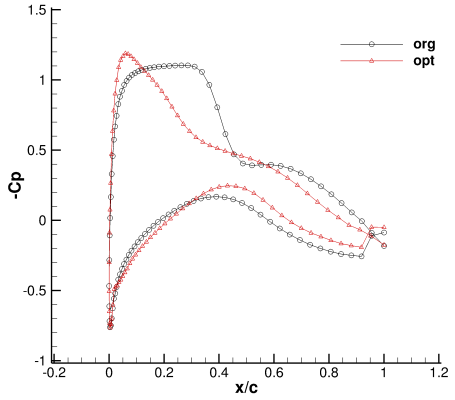

(c) $\eta = 0.455$ plane $C_p$ graph




(d) $\eta = 0.633$ plane $C_p$ graph




(e) $\eta = 0.817$ plane $C_p$ graph




(f) $\eta = 0.942$ plane $C_p$ graph

Figure 6.38: $C_p$ graphs for spanwise locations

Optimized geometry has no shock wave near wing tip. $C_p$ values make their peak at near leading edge of suction side due to new leading edge shape. Strong suction at front locations prevents the acceleration of flow. So the increment in Mach number is prohibited. Like previous cases, pressure differences between upper and lower surfaces are decreased.

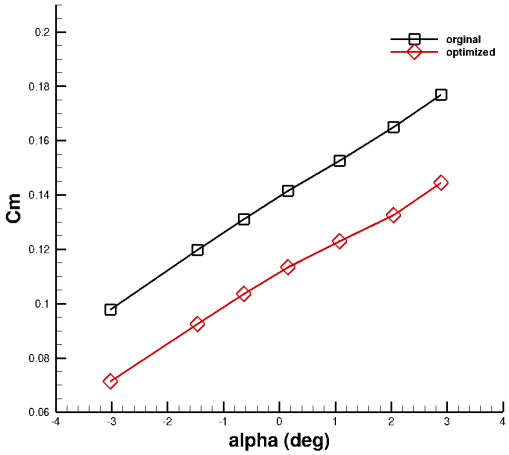



Figure 6.39: $C_m - \alpha$ off design conditions

At the off design conditions pitching moment curve is shifted downward for all angles of attack. Less positive pitching moment is more preferable since it causes less nose-up moment about center of gravity. Then, elevator is trimmed at lower degrees. Thus, the trim drag is reduced extremely. However, wing+body is

still statically unstable as expected. Shock waves have also strong influence on pitching moment. At different Mach numbers pitching moment can increase dramatically as seen in drag case. Therefore, pitching moment-Mach curve is drawn.

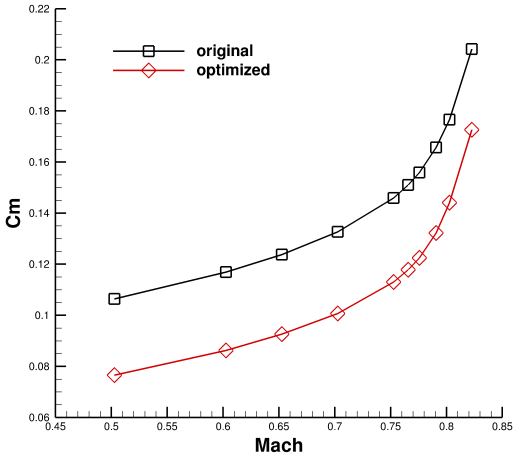



Figure 6.40: $C_m - Mach$ off design conditions

Now all $C_l, C_d$ and $C_m$ optimized wing sections will be given in single figure.

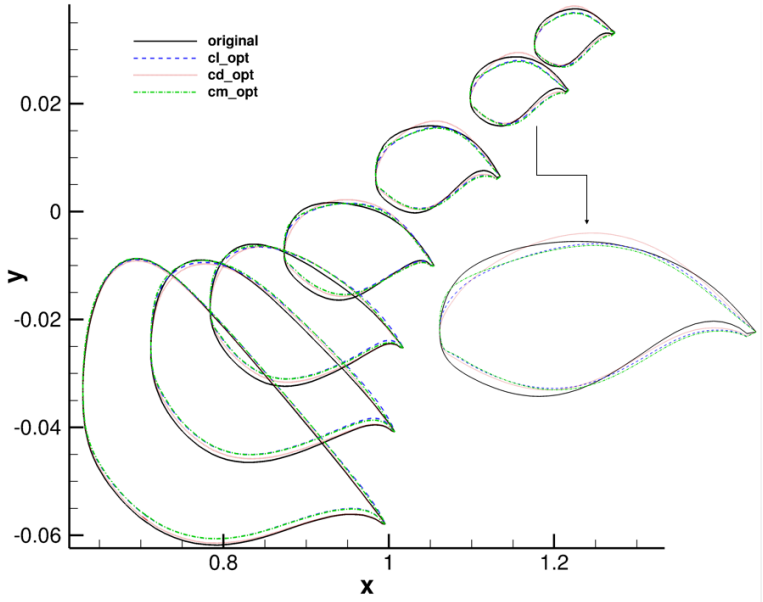



Figure 6.41: Sections for all cases

As it is seen in Figure 6.41 for all cases lower surfaces are very similar to each other. While thickness reduces at leading edge, it increases at trailing edge. Lift and pitching moment cases show similar trend for all the sections. $C_m$ optimized sections only differ from lift optimized ones at leading edge as pitching moment optimized airfoils end up with the nose shape similar to whale nose then the flow is rapidly sucked at upper surface and it decelerates extremely through the trailing edge. Comparing to previous cases the higher and the more aft camber are observed in drag minimized case. The optimization results for wing tip show that the airfoils have trend to transform into NACA-6 digit arfoils which have cusped and thin trailing edges. At the middle sections, supercritical airfoils are obtained. Although the 6-digit airfoils are famous with low drag for range of small design conditions, results show that drag is minimized in our study for the wide range of design conditions.

### 6.3.2 Body design variables

In this part, focus will be on body by keeping wing is untouched. The effect of design variables on body will be investigated. $5 \times 2$ design variables on upper and lower parts of body with degree 3 control points will be used. Drag will be tried to be minimized by applying volume and lift constraints. The problem statement is given below.

$$\begin{aligned} \text{Minimize} \quad & \frac{C_d}{C_{d_0}} \\ \text{Subject to;} \quad & 1 - \frac{C_l}{C_{l_0}} \leq 0.001 \\ & 1 - \frac{V}{V_0} \leq 0.001 \end{aligned}$$

Results for drag minimization on body and optimization convergence history are given in Table 6.10 and Figure 6.42 Body change is less effective comparing to shape optimization of wing to reduce drag since most of the drag is composed of wave and induced drag on wing surface. As a result, approximately 6.5% drag reduction can be achieved. It is important to put a volume constraint to restrict body to going too thinner and skinny as volume reduction is the easiest

Table 6.10: Results for body case

| | Initial | Optimized |
|---|---|---|
| Obj | 1.0000000 | 0.9350334 |
| Cons1 | 0.0000000 | 0.0008851 |
| Cons2 | 0.0000000 | -0.0002000 |
| Cl | 0.7361341 | 0.7354826 |
| Cd | 0.0580534 | 0.0542819 |
| Cm | 0.1765676 | 0.1982510 |

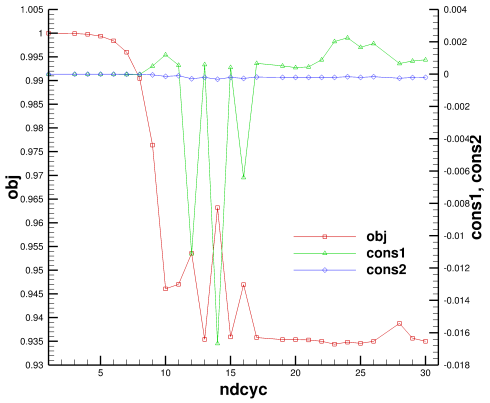



Figure 6.42: Convergence history

way to obtain less drag which is composed of body form drag. However, this configuration may result in less payload or passenger which have to be carried.

Optimized and reference bodies are given below.

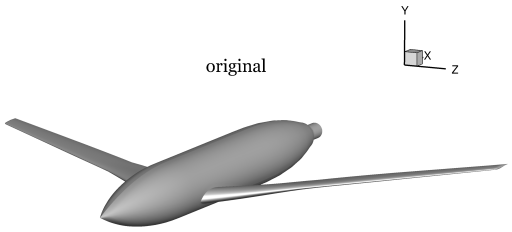



Figure 6.43: Baseline wing body

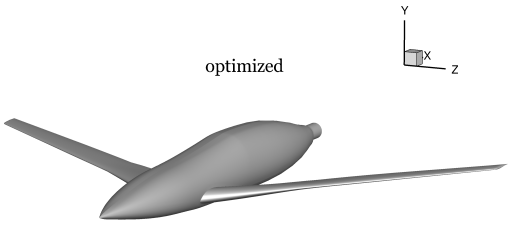



Figure 6.44: Optimized wing body

As seen from Figure 6.44, body nose becomes thinner and aft part becomes more bulky like a bullet. This configuration is probably result of optimization using Euler equations where no viscous effects and separation take place. In RANS solution fatter backside can create undesirable turbulence and flow separation. $C_p$ contours on symmetry plane is given below.

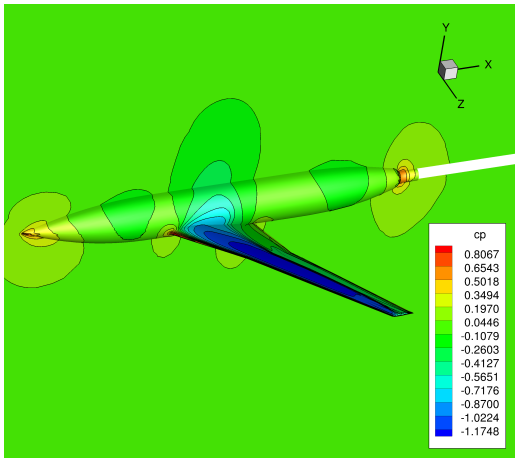



Figure 6.45: Symmetry plane $C_p$ contours base

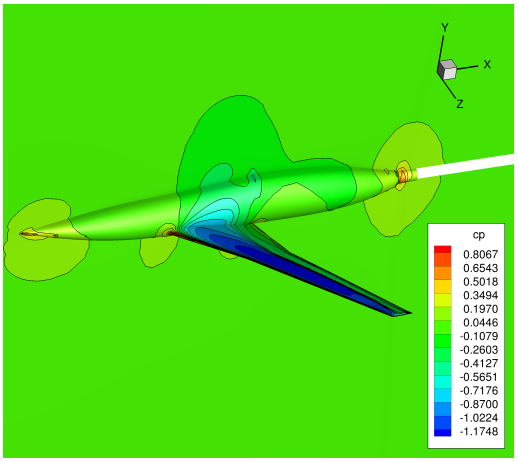



Figure 6.46: Symmetry plane $C_p$ contours optimized

Stagnation point on reference geometry is removed in Figure 6.46 by making nose thinner. Then the $C_p$ contours density is reduced at nose. Optimized body has little effect on wing root section, and other than this no change is observed.

Drag polar for different angles of attack is also plot to show performance on off-design conditions.

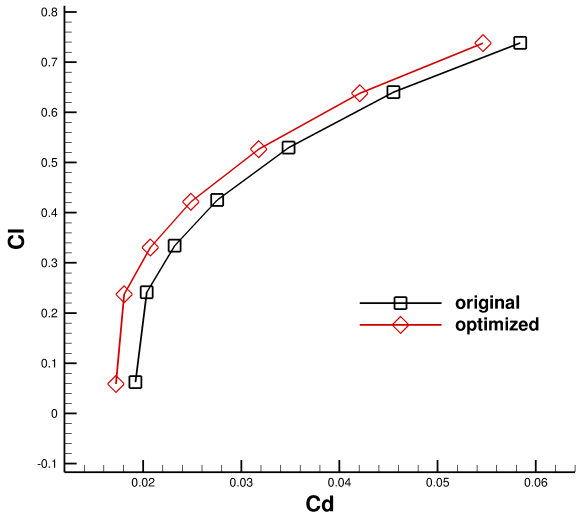



Figure 6.47: Drag polar for body optimized case

### 6.3.3 Sweep angle design variables

In this part wing planform will be optimized by changing leading edge sweep angles. Two design variables in terms of degrees are designated for this purpose. The reference area and chord dimensions are kept constant during the iterations. Optimization problem is now defined as follows.

$$\begin{aligned} \text{Maximize} \quad & \frac{C_d}{C_{d\,0}} \\ & \frac{C_l}{C_{l_0}} - 1 \leq 0.01 \end{aligned}$$

Drag will be tried to be decreased by keeping the lift unchanged too much. Pitching moment constraint is not defined, as larger the sweep angle causes destabilizing effect on entire aircraft. This result can be seen in Table 6.11 on $Cm$ value.

Table 6.11: Results for sweep case

| | Initial | Optimized |
|---|---|---|
| Obj | 1.0000000 | 0.9728462 |
| Cons1 | 0.0000000 | 0.0079058 |
| Cl | 0.7361341 | 0.7303142 |
| Cd | 0.0580534 | 0.0564769 |
| Cm | 0.1765676 | 0.2688364 |
| Cl/Cd | 12.680289 | 12.931194 |

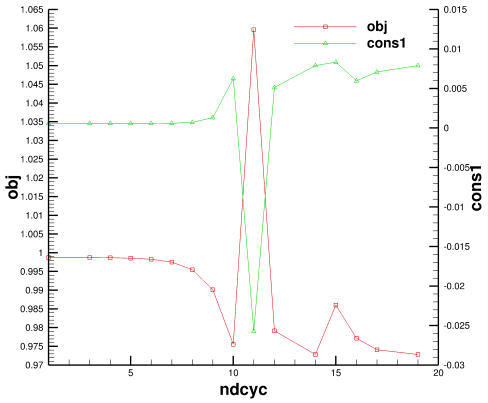



Figure 6.48: Convergence history

Drag coefficient is reduced by approximately 3% at the end of design iterations. Pitching moment is increased nearly 52%. Excessive increment in pitching moment is due to the change in center of pressure location on entire aircraft. In addition, it can be deduced that more backward sweep causes destabilizing effect on aircraft. Sweep angle change is demonstrated on following figure.

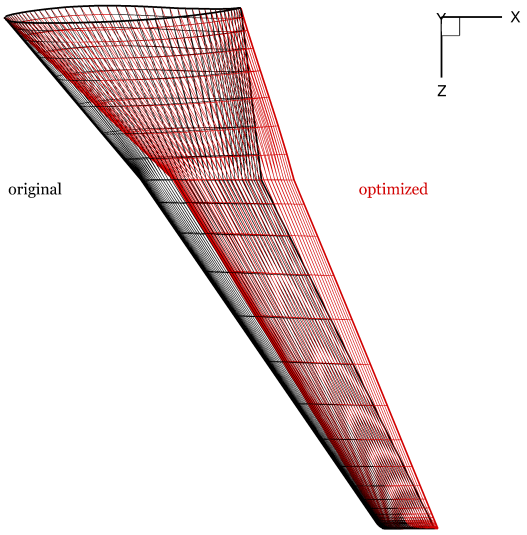
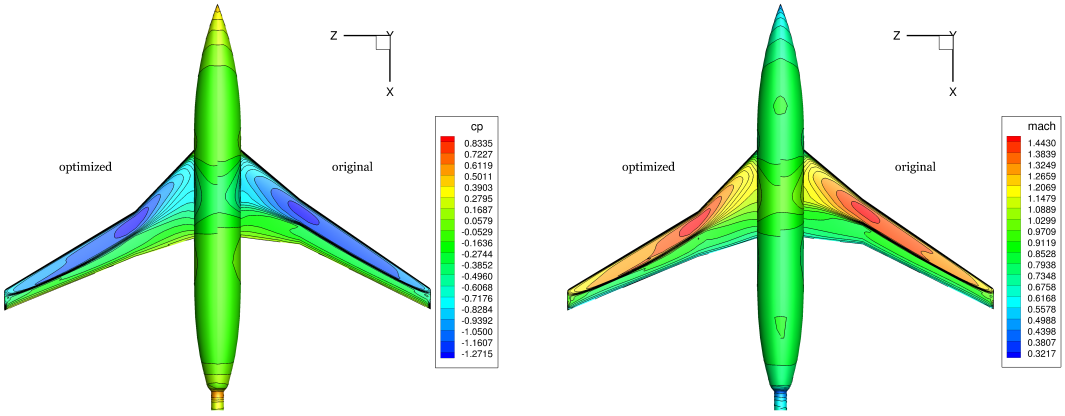



Figure 6.49: Sweep angles comparison on wing

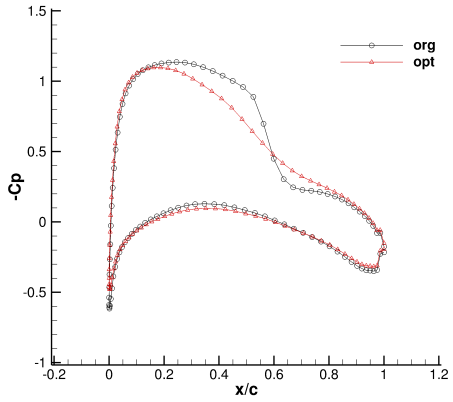
Spanwise sections have more backward sweep at the end of optimization. The difference is visible especially around crank location. Through the wing tip, effect of leading edge sweep vanishes.



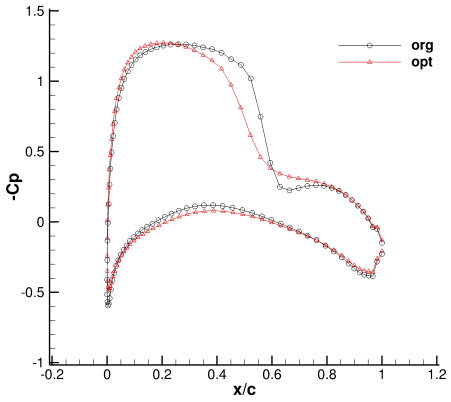

(a) $C_p$ contours (b) Mach contours

Figure 6.50: $C_p$ and mach contours

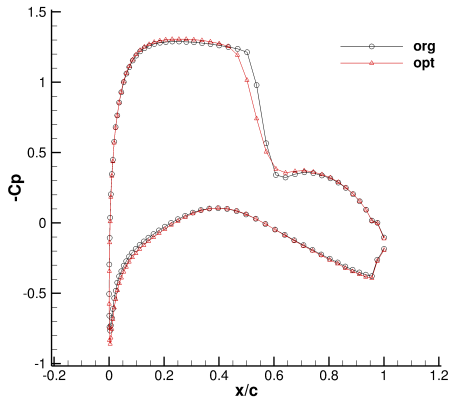
It is hard to conclude from Figures 6.50a and 6.50b. $C_p$ and Mach contours are slightly smooth around crank. Maximum Mach number which is achieved on crank is reduced. Also shock strength is decreased a little around the midspan as shown in Figure 6.51.



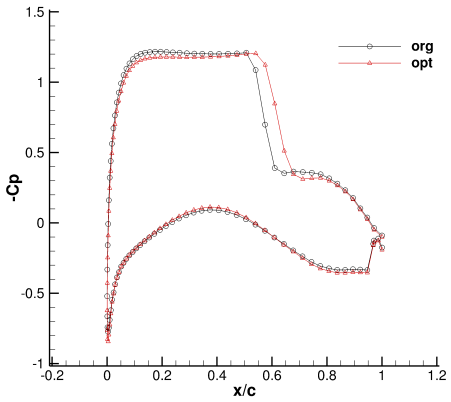

(a) $\eta = 0.231$ plane $C_p$ graph



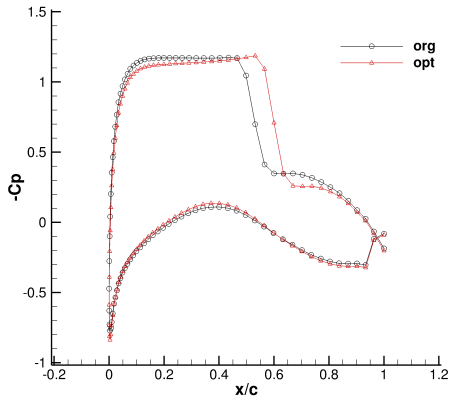

(b) $\eta = 0.325$ plane $C_p$ graph



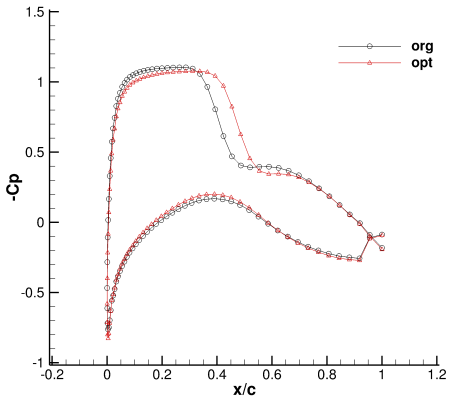

(c) $\eta = 0.455$ plane $C_p$ graph




(d) $\eta = 0.633$ plane $C_p$ graph




(e) $\eta = 0.817$ plane $C_p$ graph




(f) $\eta = 0.942$ plane $C_p$ graph

Figure 6.51: $C_p$ graphs for spanwise locations

As it can be seen, effect of shock is diminished at root sections. In addition to this fact shock dislocation is observed in wing tip airfoils. Results given in this section shows that reference geometry is already optimized in terms of leading edge sweep. However, quarter-chord sweep angle or taper ratio may have larger influence on drag as design variables but this kind of planform changes are beyond the scope of this thesis. Finally, span efficiencies of swept wing will be given in terms of sectional lift coefficients.

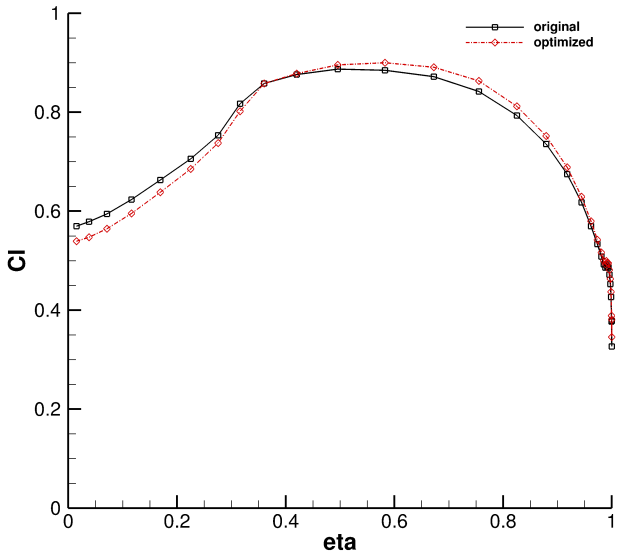



Figure 6.52: Sweep angles comparison on wing

As seen, span efficiency is reduced at near root because of the slipstream downwash effect of body. For optimized case sectional lifts are increased through the tip where shock moves backward. Also higher backward sweep in root sections causes less sectional lift due to reduction in local flow velocity.

# CHAPTER 7

# CONCLUSIONS

## 7.1 Conclusion

This thesis has been written to produce a guideline for CFD-based optimization of wing-body configuration in line with all aspects of optimization explained in detail. Starting from the discretization of Euler equations, flow solution, design variables, volumes mesh deformation, iterative solver, and gradient computation methods have been discussed. For design variables, volume mesh deformation and flow solution strategies different approaches have also been demonstrated. At the end of these chapters, the methods and approaches have been compared with examples and the parametric studies. In consideration of these results, different optimization procedures and optimized shapes with their performances have been given in order to make this thesis a guideline for Aerodynamic Designers also.

Euler solution has proved a subtle consistency with RANS solution especially where no shock wave exists such that the lower surface of wing. However, when shock has been present, there have been much more difference between two results as expected. In Euler solution normal shock has been more powerful and located at more aft position comparing to RANS solution. Consequently, both lift and drag have been found greater. $C_p$ contours have matched well on wing and body also. Nevertheless, in the early phase of design Euler solution can be used. Due to the less computational time and sparse Jacobian matrix, it is highly preferable for the optimization also.

In the third chapter, we have discussed the flow solution strategy and we have found the Newton-GMRES method is more useful than the Newton method because Newton-GMRES requires no Jacobian matrix generation. Solving Newton iterations inexactly may increase the efficiency of flow solution. Among the different preconditioners, ILU (1) has given the best efficiency in terms of computational time. As the level of fill increases, the number of non-zero elements increases tremendously in preconditioner matrix. Therefore, for high level of fills reordering strategies have been applied to decrease number of non-zeros in sparse preconditioner matrix. After fill-in level 2nd, all of the reordering strategies have proved good performance and made preconditioner matrix sparser comparing to natural ordering. Minimum Degree and METIS have been found most efficient reordering strategies for high fill-ins. However, out of RCM all reordering have given poor performance with ILU (1) preconditioner. Therefore, the RCM and natural ordering have been used throughout the study.

After a literature survey on design variables selection, NURBS has been found efficient by many researchers as it gives a reliable result on off-design conditions also. In addition to this, it creates a logical, real life shapes comparing to other parameterization schemes. In our results we have demonstrated these advantages of NURBS by investigating the optimized shape performance on off-design conditions where the angle of attack and mach numbers have been different. While lower degree NURBS creates sharper representation of geometry, higher degrees can create smoother shapes. Therefore, degree selection should be made very carefully.

In the design variable chapter, we have also discussed the volume mesh deformation strategies. We have created a simple test problem on a structured cube mesh. We have looked at the performance of RBF and algebraic perturbation strategies by applying very large deformations. As illustrated in the given example case, the RBF method has created a detailed representation of flow field. Algebraic approach has not given enough attention to curvature areas and become tenuous. RBF has also helped to decrease skewness on cells comparing to algebraic approach. Moreover, the negative cells have not been observed in RBF approach.

At the results section, three gradient computation approaches have been compared to each other. All of the sensitivities have proved a good consistency in gradients accuracy. Most of the time, difference between gradients have been less than 0.2%, which is a good accuracy level for optimization. Among the computation approaches for gradients, finite difference was found least efficient one. Since it requires at least number of design variable flow solution which is not practical for larger problems. Adjoint and direct methods require only one flow and linear system solution. Since, direct approach requires number of design variables right hand side solution, the time passed for back-substitution after factorization may be large. In Adjoint method, linear system with only one right hand side is solved. Wall clock time comparisons in this chapter also highlighted the efficiency of Adjoint method. Therefore, Adjoint method were used in optimization to calculate sensitivities.

In optimization, firstly wing surface control points have been used for wing optimization. Lift maximization, drag and pitching moment minimization have succeeded with different constraints. Shock has been removed and its strength has been reduced at the root and tip of wing for all cases. In drag minimization case, airfoil cambers have been transferred to aft in order to decrease strength of normal shock. For all cases, new leading edge curvature has been obtained. For lift maximization, tip airfoil sections have been flattened to decrease effect of tip vortices. As a result, it has been deduced that without changing planform total shock removal is not possible. Finally, although the single point optimization has been applied, the resultant shape has produced a good performance on off-design conditions. Changing angle of attack and mach number have not reduced the performance of optimized shape.

As a future plan, we are planning to work on iterative, robust solution of linear systems of Adjoint method as after some increment in grid dimension, direct solution of Adjoint equation requires too much memory, which is not available in our workstations. Also the CPU time requirement goes over 10 hours. Viscous terms of Navier-Stokes equations will be also added to sensitivities and discretization. Then, more compact optimization will be held. As a doctoral study unsteady Adjoint method will be investigated with aeroelastic effects.

# REFERENCES


[1] M. Drela. Development of the d8 transport configuration. 29th AIAA Applied Aerodynamics Conference, 2011.

[2] Cfl3d test cases. `http://cfl3d.larc.nasa.gov/Cfl3dv6/cfl3dv6_testcases.html`. Accessed: 2016-03-25.

[3] D. P. Raymer. *Aircraft design: A conceptual approach*. American Institute of Aeronautics and Astronautics, Reston, VA, 1999.

[4] L. M. Nicolai, G. Carichner, and L. M. Nicolai. *Fundamentals of aircraft and airship design*. American Institute of Aeronautics and Astronautics, Reston, VA, 2010.

[5] A. Betz. Modification of wing-section shape to assure a predetermined change in pressure distribution. *NASA Technical Report Server*, 11, 1935.

[6] M. J. Lighthill. A new method of two-dimensional aerodynamic design (rep. *No*, 2112, 1945.

[7] M. B. Giles and M. Drela. Two-dimensional transonic aerodynamic design method. *AIAA Journal*, 25(9):1199–1206, 1987.

[8] R. M. Hicks, E. M. Murman, and G. N. Vanderplaats. An assessment of airfoil design by numerical optimization. Technical report, NASA TM X, 1974.

[9] G. B. Dantzig. *Origins of the simplex method*. Stanford University, Dept. of Operations Research, Systems Optimization Laboratory, Stanford, CA, 1987.

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

[11] R. Duvigneau and M. Visonneau. Shape optimization of incompressible and turbulent flows using the simplex method. 15th AIAA Computational Fluid Dynamics Conference, 2001.

[12] D. Quagliarella and A. D. Cioppa. Genetic algorithms applied to the aerodynamic design of transonic airfoils. *Journal of Aircraft*, 32(4):889–891, 1995.

[13] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, New York, 1999.

[14] R. M. Hicks and P. A. Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412, 1978.

[15] S. Eyi, K. D. Lee, S. E. Rogers, and D. Kwak. High-lift design optimization using navier-stokes equations. *Journal of Aircraft*, 33(3):499–504, 1996.

[16] J. Martins, I. Kroo, and J. Alonso. An automated method for sensitivity analysis using complex variables. 38th Aerospace Sciences Meeting and Exhibit, 2000.

[17] O. Pironneau. *Optimal Shape Design for Elliptic Systems*. Springer Series in Computational Physics, New York, 1983.

[18] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 1988.

[19] J. Reuter and A. Jameson. Control theory based airfoil design for potential flow and a finite volume discretization. 32nd Aerospace Sciences Meeting and Exhibit, 1994.

[20] J. Reuther and A. Jameson. Aerodynamic shape optimization of wing and wing-body configurations using control theory. 33rd Aerospace Sciences Meeting and Exhibit, 1995.

[21] K. Leoviriyakit. *Wing Planform Optimization via An Adjoint Method*. PhD thesis, 2005.

[22] W. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers and Fluids*, 28(4-5):443–480, 1999.

[23] R. E. Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464, 1964.

[24] I. A. Taylor, G. W. Hou, and V. M. Korivi. Methodology for calculating aerodynamic sensitivity derivatives. *AIAA Journal*, 30(10):2411–2419, 1992.

[25] G. W. Burgreen and O. Baysal. Three-dimensional aerodynamic shape optimization using discrete sensitivity analysis. *AIAA Journal*, 34(9):1761–1770, 1996.

[26] G. R. Shubin and P. D. Frank. A comparison of two closely-related approaches to aerodynamic design optimization. pages 67–78. Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-3); p, 1991.

[27] S. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. 38th Aerospace Sciences Meeting and Exhibit, 2000.

[28] G. J. Hou, A. C. Taylor, and V. M. Korivi. Discrete shape sensitivity equations for aerodynamic problems. *International Journal for Numerical Methods in Engineering*, 37(13):2251–2266, 1994.

[29] R. M. Hicks and P. A. Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412, 1978.

[30] H. Sobieczky. Parametric airfoils and wings. *Notes on Numerical Fluid Mechanics (NNFM) Recent Development of Aerodynamic Design Methodologies*, pages 71–87, 1999.

[31] L. Piegl and W. Tiller. *The NURBS book (2nd ed.)*. Springer, Berlin, 1997.

[32] M. Nemec. *Optimal Shape Design of Aerodynamic Configurations: A Newton-Krylov Approach*. PhD thesis, 2003.

[33] G. W. Burgreen, O. Baysal, and M. E. Eleshaky. Improving the efficiency of aerodynamic shape optimization. *AIAA Journal*, 32(1):69–76, 1994.

[34] A. D. Boer, M. S. Schoot, and H. Bijl. Moving mesh algorithm for unstructured grids based on interpolation with radial basis functions. In *III European Conference on Computational Mechanics*, pages 418–418, 2006.

[35] V. Poirier and S. Nadarajah. Efficient reduced-radial basis function-based mesh deformation within an adjoint-based aerodynamic optimization framework. *Journal of Aircraft*, pages 1–17, 2016.

[36] V. Venkatakrishnan. Newton solution of inviscid and viscous problems. *AIAA Journal*, 27(7):885–891, 1989.

[37] M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409, 1952.

[38] Y. Saad and M. H. Schultz. Gmres: a generalized minimal residual method for solving nonsymmetric linear systems. *SIAM Journal*, 7:856–869, 1986.

[39] K. Michalak and C. Ollivier-Gooch. Matrix-explicit gmres for a higher-order accurate inviscid compressible flow solver. 18th AIAA Computational Fluid Dynamics Conference, 2007.

[40] A. Puyero, D. Zingg, A. Puyero, and D. Zingg. An efficient newton-gmres solver for aerodynamic computations. 13th Computational Fluid Dynamics Conference, 1997.

[41] M. Nemec and D. W. Zingg. Newton-krylov algorithm for aerodynamic design using the navier-stokes equations. *AIAA Journal*, 40:1146–1154, 2002.

[42] J. Gatsis. *Preconditioning Techniques for a Newton-Krylov Algorithm for the Compressible Navier-Stokes Equations*. PhD thesis, 2013.

[43] Cfl3d version 5.0 manual. `https://cfl3d.larc.nasa.gov/Cfl3dv6/cfl3dv6_v5manual.html`. Accessed: 2016-09-22.

[44] J. L. Steger and R. F. Warming. Flux vector splitting of the inviscid gas-dynamic equations with application to finite-difference methods. *Journal of Computational Physics*, 40:263–293, 1981.

[45] B. Van Leer. Flux vector splitting for the euler equations. *ICASE Report*, pages 82–30, September 1982.

[46] M.-s. Liou. A sequel to ausm: Ausm+. *Journal of Computational Physics*, 129:364–382, 1996.

[47] W. Anderson, J. Thomas, and B. V. Leer. A comparison of finite volume flux vector splittings for the euler equations. 23rd Aerospace Sciences Meeting, 1985.

[48] P. Kalita, A. K. Dass, and A. Sarma. Effects of numerical diffusion on the computation of viscous supersonic flow over a flat plate. *International Journal of Applied and Computational Mathematics*, 2015.

[49] J. R. (n.d.) Carlson. Inflow-outflow boundary conditions with application to fun3d (pdf). Technical report, Hampton, Virginia.

[50] S. Eyi, M. Camci, M. Yumusak, and A. Ezertas. Three dimensional design optimization using analytical and numerical jacobians. 20th AIAA Computational Fluid Dynamics Conference, 2011.

[51] Pardiso version 5.0.0. `http://www.pardiso-project.org/`. Accessed: 2016-11-15.

[52] Umfpack solver. `http://faculty.cse.tamu.edu/davis/suitesparse.html`. Accessed: 2016-12-01.

[53] Superlu is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. `http://crd-legacy.lbl.gov/`. Accessed: 2016-11-07.

[54] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Academic Press, New York, 1970.

[55] Y. Saad. *Iterative methods for sparse linear systems (2nd ed.)*. SIAM, Philadelphia, 2003.

[56] H. F. Walker. Implementation of the gmres method using householder transformations. *SIAM Journal on Scientific and Statistical Computing*, 9(1):152–163, 1988.

[57] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[58] S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact newton method. *SIAM Journal on Scientific Computing*, 17(1):16–32, 1996.

[59] R. Idema, D. Lahaye, and C. Vuik. *On the convergence of inexact Newton methods*. Delft University of Technology, Delft, 2011.

[60] M. Pernice and H. F. Walker. Nitsol: A newton iterative solver for nonlinear systems. *SIAM Journal on Scientific Computing*, 19(1):302–318, 1998.

[61] A. Pueyo and D. Zingg. Efficient newton-krylov solver for aerodynamic computations. *AIAA Journal*, 36:1991–1997, 1998.

[62] D. Knoll and D. Keyes. *Jacobian-free Newton-Krylov methods: A survey of approaches and applications*. Journal of Computational Physics, 193(2), 357-397, 2004.

[63] O. Onur and S. Eyi. Effects of the jacobian evaluation on newton's solution of the euler equations. *International Journal for Numerical Methods in Fluids*, 49(2):211–231, 2005.

[64] A basic tool-kit for sparse matrix computations (version 2). `saad/software/SPARSKIT/index.html`. Accessed: 2016-11-03.

[65] I. Arany. The preconditioned conjugate gradient method with incomplete factorization preconditioners. *Computers & Mathematics with Applications*, 31(4-5):1–5, 1996.

[66] Y. Saad. Ilut: A dual threshold incomplete lu factorization. *Numerical Linear Algebra with Applications*, 1(4):387–402, 1994.

[67] M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[68] L. C. Dutto. The effect of ordering on preconditioned gmres algorithm, for solving the compressible navier-stokes equations. *International Journal for Numerical Methods in Engineering*, 36(3):457–497, 1993.

[69] Graph theory tutorials. `http://primes.utm.edu/graph/index.html`. Accessed: 2016-11-05.

[70] A. George and J. W. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31(1):1–19, 1989.

[71] P. R. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[72] A. George. *Computer Implementation of the Finite Element Method*. PhD thesis, 1971.

[73] W. Liu and A. H. Sherman. Comparative analysis of the cuthill-mckee and the reverse cuthill-mckee ordering algorithms for sparse matrices. *SIAM Journal on Numerical Analysis*, 13(2):198–213, 1976.

[74] Serial graph partitioning and fill-reducing matrix ordering. `http://glaros.dtc.umn.edu/gkhome/views/metis`. Accessed: 2016-12-10.

[75] K. M. Selvan. On the effect of shape parameterization on aerofoil shape optimization. *International Journal of Research in Engineering and Technology*, 4:02, 2015.

[76] J. T. Batina. Unsteady euler airfoil solutions using unstructured dynamic meshes. *AIAA Journal*, 28(8):1381–1388, 1990.

[77] J. T. Batina. Unsteady euler algorithm with unstructured dynamic mesh for complex-aircraft aerodynamic analysis. *AIAA Journal*, 29(3):327–333, 1991.

[78] T. Baker and P. Cavallo. Dynamic adaptation for deforming tetrahedral meshes. 14th Computational Fluid Dynamics Conference, 1999.

[79] K. Stein, T. Tezduyar, and R. Benney. Mesh moving techniques for fluid-structure interactions with large displacements. *Journal of Applied Mechanics*, 70(1):58, 2003.

[80] A. D. Boer, M. V. Schoot, and H. Bijl. Mesh deformation based on radial basis function interpolation. *Computers & Structures*, 85(11-14):784–795, 2007.

[81] T. Rendall and C. Allen. Efficient mesh motion using radial basis functions with data reduction algorithms. *Journal of Computational Physics*, 228(17):6231–6249, 2009.

[82] Holger Wendland. Error estimates for interpolation by compactly supported radial basis functions of minimal degree. *Journal of Approximation Theory*, 93(2):258–72, 1998.

[83] S. Jakobsson and O. Amoignon. Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. *Computers and Fluids*, 36(6):1119–136, 2007.

[84] Software for cfd - pointwise. `http://www.pointwise.com/index.shtml`. Accessed: 2016-12-10.

[85] G. N. Vanderplaats and S. R. Hansen. *DOT Users Manual*. Vanderplaats, Miura & Associates, Inc, Goleta, California.

[86] B. Epstein, T. Rubin, and S. and Sé. Accurate multiblock navier-stokes solver for complex aerodynamic configurations. *AIAA Journal*, 41(4):582–594.

[87] F. Marconi, N. Siclary, G. Carpenter, and R. Chow. Comparison of tlns3d computations with test data for a transport wing/simple body configuration. *AIAA paper*, pages 94–2237.