Adjoint based compressible Euler shape optimization code for propeller driven aircraft

A Thesis

Submitted for the Degree of

MASTER OF SCIENCE (ENGINEERING)

by

MILIND PRAKASH DHAKE

ENGINEERING MECHANICS UNIT
JAWAHARLAL NEHRU CENTRE FOR ADVANCED SCIENTIFIC RESEARCH
(A Deemed University)
Bangalore – 560 064

MAY 2014

To

my parents

DECLARATION

I hereby declare that the matter embodied in the thesis entitled "**Adjoint based compressible Euler shape optimization code for propeller driven aircraft**" is the result of investigations carried out by me at the Engineering Mechanics Unit, Jawaharlal Nehru Centre for Advanced Scientific Research, Bangalore, India under the supervision of Prof. Suresh Madusudhan Deshpande and Prof. Roddam Narasimha and that it has not been submitted elsewhere for the award of any degree or diploma.

In keeping with the general practice in reporting scientific observations, due acknowledgment has been made whenever the work described is based on the findings of other investigators.

Milind Prakash Dhake

CERTIFICATE

We hereby certify that the matter embodied in this thesis entitled "**Adjoint based compressible Euler shape optimization code for propeller driven aircraft**" has been carried out by Mr. Milind Prakash Dhake at the Engineering Mechanics Unit, Jawaharlal Nehru Centre for Advanced Scientific Research, Bangalore, India under our supervision and that it has not been submitted elsewhere for the award of any degree or diploma.

Prof. Suresh Madusudhan Deshpande	Prof. Roddam Narasimha
(Research Supervisor)	(Research Supervisor)

List of Figures

Acknowledgements

It give me immense pleasure to express my profound gratitudes to my research supervisors Prof. S. M. Deshpande and Prof. Roddam Narasimha for their invaluable guidance, continuous support and kindness. I am immensely grateful to Prof. S. M. Deshpande for his insightful discussions on topics on almost everything. I am immensely grateful to Prof. Roddam Narasimha for all the discussions we have had. His insights and unending passion inspires me a lot.

I would like to thank Intel Laboratories, Bangalore for the collaborations we have had for making the Euler code optimized. I would like to thank Dr. Anand Deshpande for his discussions on optimization. Stay in JNC was indeed memorable. I would like to thank my friends who always encouraged me and made my life in JNC memorable. I would like to thank the dandapinda galu group Sidharth, Sunil, Shashank, Mamta, Rohan Subbaiah, Sharmila, nisha and vaibhav for all those happy moments. A special thanks to Kanwar, Prasanth, Priyanka, Depthi, Punnu, Meha, Chakri, Rajesh, Navneeth, Dheeraj, Vicky, Sankalp, Deepak , Manjusha. Finally I thank my parents for their unconditional support.

Abstract

High fidelity aerodynamic shape optimization using the adjoint method for gradient calculation is presented for an aircraft wing with particular emphasis on designing wings for propeller-driven aircraft with lower induced drag with lower aspect ratio.

Rakshith *et al.* (2011) exploited the aerodynamic interaction of a wing with the slipstream of a propeller mounted in front of the wing, in order to design wings of lower drag. They got novel wing designs with lower induced drag for a propeller driven aircraft in tractor configuration using their own specially developed software PROWING, which includes the coupling of an optimizer with lifting line theory. The latter was modified for including the effect of the propeller slipstream. Inspired by this work the present work addresses a higher fidelity optimization problem by coupling the Euler equation with optimization techniques for obtaining wings with lower induced drag. Rakshith *et al.* (2011) optimized wings with aspect ratio of 12 which was suitable for the lifting line theory. Present work verifies the optimized shape got by Rakshith *et al.* (2011) and also includes optimization with small aspect ratio wing. A software package was developed for this purpose and its implementation and validation have been performed.

A general Aerodynamic Shape Optimization procedure includes nonlinear constraints such as a PDE (partial differential equation, in this case Euler equation). The present work solves an optimization problem to minimize the induced drag calculated by Euler equations with other nonlinear constraints which include aerodynamic and geometric properties like lift coefficient, aspect ratio etc. Hence there was a need to have a constrained nonlinear programming algorithm for minimization of a specified cost function. A C++ software package PROP-OPT was developed for this purpose. This has been coupled to a flow solver, gradients solver, shape parametrization and domain mesh deformation, in order to automate the optimization cycle. PROP-OPT uses the open source C++ library NLOPT, which gives a choice of using various optimization techniques available on the Internet such as Sequential Quadratic Programming (SQP) which approximates the Hessian with Broyden Fletcher Goldfarb Shanno (BFGS) algorithms, to reach the optimum faster. With NLOPT library the PROP-OPT can solve the optimization problem with nonlinear constraints of PDE with aerodynamic and geometric properties of the wing-propeller system.

A gradient-based optimization procedure requires the converged flow solution for each optimization cycle. Flow solutions by Euler Equations are considered and an in-house

Euler solver PROP-EULER (Rakshith 2013) was optimized in time in the present to enable quick solution. With the implementation of open-MP, SIMD and MPI with domain decomposition, 3x per processor speedup was achieved in collaboration with Intel Technologies, Bangalore. Optimized PROP-EULER was validated with the standard test case of the ONERA M6 wing. The Euler equations are solved by finite volume methods with a second order accurate Kinetic Flux Vector Scheme (KFVS). For faster convergence options such as LU-SGS (Lower Upper Symmetric Gauss Seidel) or point Jacobi solver have been used. PROP-EULER has a propeller module based on blade element theory, which is used to determine appropriate sources distribution on an actuator disc in the Euler field.

Gradient calculations are known to be computationally costly in general. A simple way to calculate a gradient is by the finite difference method, but the computational cost increases with increase in the number of design variables. One way to reduce this cost is by the adjoint method. The Discrete Adjoint method is explained, implemented and validated in the thesis. A C++ code PROP-ADJ was developed to solve the adjoint equations, which are formulated by augmenting the cost function with the flow residual flux in order to desensitize the cost function to the number of design variables used in the problem. Such benefits in the computational cost are greater as the number of design variables increases, as is the case in the present work. The Adjoint equation was coupled with the blade element theory to include the effects of the propeller slipstream over the wing. The derivatives of the cost function with respect to the design/control variables were obtained by using Automatic Differentiation techniques by the open source software tool TAPENADE, which includes an output subroutine that calculates the derivative for another input subroutine which calculates the function. The PROP-ADJ code solves for a steady state solution of an adjoint equation which includes the pseudo-time derivative of the adjoint variables. PROP-ADJ has made use of LU-SGS and point Jacobi solver for an implicit solution in order to have faster convergence with MPI (Message Passing Interface) parallelization using domain decomposition.

The parametrization of design variables is an important step for the optimization algorithm. Present work uses Non Uniform Rational B-spline (NURBS) for the representation of the surface of the aircraft wing. A C++ code was developed for the purpose of getting the design/control variable vector to use in the optimization cycle. The implementation of NURBS and its use are explained. In an optimization cycle new shapes are formed and these have to be re-gridded to get the flow solutions. The cost of re-gridding can be reduced with the use of mesh deformation algorithms. A C++ code was written for mesh deformation using radial basis functions (RBF) involving powers of the radial coordinates.

A rectangular wing with NACA 0012 airfoil with propeller mounted upstream was considered as the control wing for the aerodynamic shape optimization. The wing has an aspect ratio of 12. The optimization was done for the minimization of Induced Drag for a constant lift coefficient $C_L = 0.4$, with fixed semi-span and aspect ratio. The drag reduction achieved was 8.2 counts with mesh size of 10^5, for a varying chord distribution with the wing thickness to chord ratio held to be constant. Other cases, varying thickness with holding the wing chord constant have also been investigated. This verified the the optimum chord distribution is shorter behind the propeller and longer outboard, which was obtained by Rakshith *et al.* (2011) for large aspect ratio wings. Continuing further present work also investigates a wing shape for a shorter aspect ratio wing for the turbo-prop aircraft wing with tractor configuration, here control wing of taper ratio of 0.5 with NACA0012 airfoil was used.

Contents

Chapter 1

Introduction

Although optimization techniques have been in use for about a century, increasing computational power and the need to constantly improve design have given a boost to the development of more sophisticated tools for optimization. Demanding the best possible solutions for a function of interest within suitable constraints can give a good idea of the optimization problem. In complex engineering systems, a small change can often lead to significant benefits. For example, a small reduction in the drag of an aircraft translates into higher top speed, quicker acceleration, shorter take-off distance and lower direct operating cost in the form of fuel savings. One way to identify such small changes is by a traditional trial and error approach, but a better approach can be by solving an inverse mathematical problem for an optimum solution which may involve nonlinear constraints.

In engineering applications, solution of nonlinear constraint optimization problems involves the search for a minimum of a nonlinear objective function subjected to nonlinear constraints. It is common for such optimization problems to have multiple extrema. Due to this difficulty two different approaches have emerged in the area of nonlinear constrained optimization: local and global methods. Local methods aim to obtain a local minimum in the neighborhood of an input configuration or design. In general they do not guarantee the minimum so obtained is an absolute global minimum. These methods usually use the gradients or Hessians of the objective function, and the constraints and the gradients are used to obtain the search direction to march towards the optimized solution. On the other hand global methods aim to obtain the absolute minimum of the function. They do not need gradient information, and are mostly based on stochastic procedures which include many functional calculations. Although local methods do not aim for the global optimum, several approaches can be used to continue searching for a global optimum. More detailed classification, implementation of algorithms and comparison of different algorithm can be found in Rao (1996).

As a special field of optimization subject to partial differential equation (PDEs), shape optimization and flow control have seen steady research interests over the past several decades. Especially in aerodynamic design, the transition from simulation alone to a coupled simulation and optimization approach is progressing continuously. Advances in Computational Fluid Dynamics (CFD) give a good base for optimization in a frame where fluid flows are considered. There have been many such optimizations carried out in the past. One of the first in aerodynamics was Prandtl's demonstration that an elliptic load distribution gives the lowest induced drag for a large aspect ratio wing. Lighthill (1945) designed airfoils for a prescribed pressure distribution in incompressible flows; This was later extended to compressible flow by McFadden (1979). Hicks & Henne (1978) designed a wing in a potential flow using a conjugate gradient method to minimize the objective function. Here the gradients were computed using simple finite differences. With the advent of high speed computing and the availability of efficient analysis algorithms, automatic design procedures have become possible. This is due to the coupling of CFD solution algorithms with different optimization algorithms.

CFD has been used widely as an analysis tool in order to aid the design process. Combining CFD with optimization algorithms is a nontrivial task. In the present study a gradient search (or local) method is used for the optimization algorithm to find the optimum shape of a wing for a turboprop aircraft. This optimization cycle is shown in chapter 2. The shape is parameterized in design or control variables. These variables are given as initial input to the optimization problem with a defined objective function. The optimization algorithm gives as output new values for the design variables that give the minimum value of the objective function. The governing equation for the flow is put in as a constraint to the optimization problem with such other nonlinear constraints like aspect ratio, lift coefficient etc. as considered appropriate for the problem. In the gradient search method the gradients in the control/design variable space are used to provide a search direction leading towards the optimum. There are methods which use the Hessian or an approximation to it to get better convergence towards the optimum.

A simple way to calculate gradients is by using finite differences, but the computational cost is proportional to the number of design variables and can become very high if the number of design variables is large. An alternative is the adjoint method which calculates the gradients as explained in chapter 4. Jameson (1990) first applied the adjoint method in the design of airfoils in transonic inviscid flow. This optimization was done to reduce the pressure drag at Mach 0.73 of the RAE2822 airfoil, and a reduction in drag coefficient was demonstrated. Subsequently, it was extended to design using the Navier-Stokes equations. The adjoint method is very useful when the number of design variables is large as the objective function is made non sensitive by a large number of design or control variables. In this method a dual problem is solved to calculate the gradients.

Combining CFD with the optimization algorithms has been a very efficient tool for design. In the present study, this combination is used to reduce the induced drag of a wing with propeller for a turboprop aircraft with tractor configuration. Turboprops are often designed by treating propeller and wing separately, although the aerodynamic influence of propeller on the wing has been studied for a long time. Prandtl (1921) was the first to study propeller-wing interactions with experimental investigations in a wind tunnel. For a tractor configuration Prandtl observed an increase in drag for propeller axis below the wing, and a decrease in drag for propeller axis above the wing. More recently this problem has been studied with different objectives and different theoretical frameworks with the propeller modeled as an actuator disk, and also with experimental investigations. Kroo (1986) and Veldhuis & Heyma (2000) studied the optimization of wing for a propeller driven aircraft. Kroo (1986) followed a variational calculus approach for optimization. Veldhuis (2005) used modified lifting line theory to study propeller slipstream effects on the wing. The influence of swirl was also considered. Wing twist distribution was carried out to get lower drag. He concluded that the chord distribution obtained by the optimization process was unrealistic. Rakshith *et al.* (2011) presented new wing shapes for turboprops in tractor configuration. They modified the lifting line theory to include the propeller effects and studied the influence of propeller slipstream on wing characteristics. The optimization was carried out for minimum induced or total drag at given lift coefficient (usually 0.4) and fixed aspect ratio of 12. They found reasonable wing shapes with 9.15% reduction in the drag, which was validated using CFD tools and wind tunnel tests.

We present here a relatively higher fidelity optimization procedure for wing shapes for minimum induced drag with constant lift coefficient of 0.4 and aspect ratio of 12 and 6. The details are discussed in chapter 5. We have considered a rectangular wing with NACA0012 airfoil section with propeller as a control wing for optimizing the shape for minimum induced drag. Coupling the Euler Equations with a propeller solved using Blade Element Theory was used to calculate the induced drag on the wing as a part of the optimization cycle. Implementation of SIMD, open-MP, MPI domain decomposition was done for quicker flow solutions. The shape was parameterized using Non Uniform Rational B-spline (NURBS) and the gradients obtained using the adjoint method. The shape optimization was carried out by an in-house written code PROP-OPT, which uses the open source optimization library NLOPT (Non Linear Optimization).

The thesis is organized as follows:

- Chapter 2: We state the optimization problem for minimum drag with geometrical and aerodynamic constraints. This chapter also shows the flow chart of the optimization cycle and briefly discusses each part.

- Chapter 3: The numerical details of the optimization problem are described here. Algorithms such as shape parameterization, flow solutions by Euler equation with a source term, and a brief discussion on gradients are presented. The use of open source library NLOPT is also explained. The validation of each part is established.

- Chapter 4: This chapter presents the use of the adjoint equation to calculate the gradients. Other methods such as a finite difference scheme to solve the equations are also discussed. The validation of the gradient calculation in the adjoint framework is presented.

- Chapter 5: Different cases for optimization of wing with propeller are discussed in this chapter. A rectangular wing with NACA0012 airfoil section was taken as an input control wing in most of the cases. The results of all the cases are also explained.
- Chapter 6: Shape optimization of a small aspect ratio wing for propeller-driven aircraft with tractor configuration is performed in this chapter. The wing of the present generation turbo-prop aircraft with reduced aspect ratio is considered as an control wing for the optimization. The optimal shape is presented and discussed.
- Chapter 7: Summary and conclusion are discussed with suggestions for possible future work.
- Appendix

Chapter 2

Formulation of the Optimization Problem

As with most engineering applications, the aerodynamic shape optimization problem is a nonlinear constrained optimization problem. The goal is to find the best possible shape under certain given constraints. This chapter focuses on formulating a general optimization problem, specifically using gradient-based methods, and then coupling the optimization technique with the CFD algorithms used in the present study. A brief discussion of the code PROP-OPT developed in this thesis for carrying out the optimization is presented in section 2.2.

In Section 2.1 a general nonlinear constrained problem is mathematically formulated and all its various elements are described. A description follows of the local optimality conditions for the general nonlinear constraint problem and also a general discussion of how such problems are solved. The coupling of the CFD algorithms and the optimization techniques is discussed in section 2.2. A flowchart explaining the optimization cycle is also shown. Formulation of the appropriate cost/objective function of this study is also described, and cases selected for the present study are addressed.

2.1 Mathematical Formulation

The objective of optimal shape design is to alter either a part or the whole of a given initial surface to achieve a desired objective. The objective could be to achieve a shape having least resistance or drag or highest lift , or a shape that delays transition to turbulent flows. The objective is posed in the form of a cost function. A general nonlinear constrained minimization problem can be formulated as follows:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h_i(\mathbf{x}) = 0, \qquad i = 1, \ldots, p. \\ & g_j(\mathbf{x}) \leq 0, \qquad j = 1, \ldots, q. \end{aligned} \tag{2.1}$$

where $\mathbf{x}$ is the design variable vector (also called control variable vector sometimes), $f(\mathbf{x})$ is the cost function and $h_i(\mathbf{x})$ is the equality constraint function on $\mathbf{x}$, $g_j(\mathbf{x})$ is the inequality constraint function on $\mathbf{x}$, and p and q are the number of equality and inequality constraints respectively. It is assumed that $f(\mathbf{x}), h_i(\mathbf{x}), g_j(\mathbf{x})$ are nonlinear functions and have continuous first and second derivatives.

The design variable vector must satisfy both inequality and equality constraints. Hence a design space $\mathbf{R}$ is defined as

$$\mathbf{R} = \{\mathbf{x} : h_i(\mathbf{x}) = 0 \text{ for } i = 1, 2, \ldots, p \text{ and } g_j(\mathbf{x}) \leq 0 \text{ for } j = 1, 2, \ldots, q\} \tag{2.2}$$

A point $\mathbf{x}$ in R, $\mathbf{x} \in \mathbf{R}$, is considered a 'feasible' point. In a constrained optimization problem the minimum of the function must lie inside the feasible region R. The inequality constraints are a set of equations that impose certain bounds on the design variable vector. Inequality constraints can be active or inactive. For a feasible point $\mathbf{x}$, if $g_k(\mathbf{x}) = 0$ for any k, the inequality constraint is considered to be active. The feasible point $\mathbf{x}$ satisfying an active constraint is at a limit of the design space and all its neighboring points are not necessarily in the feasible region. On the other hand, for a feasible point $\mathbf{x}$, $g_k(\mathbf{x}) \neq 0$, say $g_k(\mathbf{x}) < 0$, then the inequality constraint is said to be inactive. In this case, all its neighboring points are feasible and this inequality constraint does not need to be considered when looking for a new design point from $\mathbf{x}$.

To know whether an optimum has been reached locally, a design variable $\mathbf{x}^*$ needs to satisfy the Karush-Kuhn-Tucker (KKT) conditions. The detailed proof is found in Rao (1996), however it is outlined as follows

If $\mathbf{x}^*$ is a local minimizer of optimization problem 2.1 and the gradient of all active constraints at this point are linearly independent, then the following relations hold:

$$\begin{aligned}
&h_i(\mathbf{x}^*) = 0 \qquad \text{for } i = 1, \ldots, p\\
&g_j(\mathbf{x}^*) \leq 0 \qquad \text{for } j = 1, \ldots, q\\
&\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = f(\mathbf{x}^*) + \sum_{i=1}^{p} \lambda_i^* \nabla_x h_i^*(\mathbf{x}^*) + \sum_{i=1}^{q} \mu_i^* \nabla_x g_i^*(\mathbf{x}^*) = 0\\
&\lambda_i^* h_i^*(\mathbf{x}^*) = 0 \qquad \text{for } i = 1, \ldots, p\\
&\mu_j^* g_j^*(\mathbf{x}^*) = 0 \qquad \text{for } j = 1, \ldots, q
\end{aligned} \tag{2.3}$$

where

$$\mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = f(\boldsymbol{x}^*) + \sum_{i=1}^{p} \lambda_i^* h_i^*(\mathbf{x}^*) + \sum_{i=1}^{q} \mu_i^* g_i^*(\mathbf{x}^*),$$

$\mathcal{L}$ is the Lagrangian, and $\nabla_{\boldsymbol{x}}\mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^* \boldsymbol{\mu}^*)$, is the gradient of the Lagrangian with respect to $\mathbf{x}$, λ_i^* for $i = 1, \ldots, p$ and μ_i^* for $i = 1, \ldots, q$ are the Lagrange multipliers associated with the local optimum. Notice that for unconstrained optimization problems, the KKT conditions become a single condition, the gradient of the objective function at the minimum must be zero. The KKT conditions must be satisfied for a point to be a local minimum.

However, this does not guarantee that the point is an optimum. To guarantee that a point give minimum functional value, another condition must be added to the KKT condition. The sufficient condition is

A point $\mathbf{x}^*$ is a local minimizer of the problem 2.1 if it satisfies the Karush-Kuhn-Tucker conditions and the following relation holds:

$$\mathbf{N}^T(\mathbf{x}^*)\nabla_x^2\mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\mathbf{N}(\mathbf{x}^*) \tag{2.4}$$

is positive definite, where $\mathbf{N}^T(\mathbf{x}^*)$ is a matrix whose columns are the basis of the subspace $\mathbf{N}$, where $\mathbf{N}$ is the null space of the space whose basis is formed by the gradient of all active constraints.

2.2 Coupling of CFD and optimization algorithm

The previous section discussed a general optimization problem with nonlinear constraints. In aerodynamic shape optimization the nonlinear constraint often includes the flow governing equation, Euler equation in the present study. Thus the cost/objective function is now a function of a state flow variable and the design variable. For a cost function $J(\mathbf{U}, \boldsymbol{\beta})$ the minimization problem now can be formulated as

$$\begin{array}{lr} \underset{\boldsymbol{U},\boldsymbol{\beta}}{\text{minimize}} & J(\boldsymbol{U}, \boldsymbol{\beta}) \\ \text{PDE constraint} & R(\boldsymbol{U}) = 0, \\ \text{Geometric constraint} & g(\boldsymbol{\beta}) = 0, \\ \text{Aerodynamic constraint} & h(\boldsymbol{U}, \boldsymbol{\beta}) = 0 \end{array} \tag{2.5}$$

where $\boldsymbol{U}$ is the state flow variable, $\boldsymbol{\beta}$ is the design variable vector.

The present study includes the flow solution by the Euler equation which solves for the flow with a modeled propeller slipstream. The aim of the optimization is to reduce the induced drag of a wing-propeller configuration. The coefficient of drag is calculated by calculating the pressure force on the wing. The component of this force in the direction of flow is taken as drag while that normal to the flow is taken as lift. The coefficients of lift and drag are given as:

$$C_D = \frac{D}{q_\infty S} \quad \text{and} \quad C_L = \frac{L}{q_\infty S} \quad \text{where} \quad q_\infty = \frac{1}{2}\rho_\infty U_\infty^2, \tag{2.6}$$

C_L and C_D are the coefficient of lift L and drag D force respectively, q_∞ is the dynamic pressure, ρ_∞, U_∞ are density and velocity respectively subscript ∞ denotes conditions for upstream and S is the reference wing area. When only vortex drag is required, lifting line theory gives the coefficient of drag as

$$C_D = \frac{C_L^2}{\pi \mathit{AR} e} \tag{2.7}$$

where $\mathit{AR} = b^2/S = b/\bar{c}$ is the aspect ratio of the wing, b is the span $\bar{c}$ is the mean wing chord, and e is the span efficiency. The drag prediction by lifting line theory suggests that at the minimum induced drag the aerodynamic efficiency becomes close to one. The span efficiency depends on geometry. However, it may be a function of C_L but often a weak one (Kroo 2001), so it remains useful for comparing the induced drag of different configurations. This suggests that for the uniqueness of the solution to the optimization problem 2.5 constraining either C_L or L any two of the three parameters b, S and AR is sufficient, see (Hicken & Zigg 2010). The reference area is the planform area of the wing geometry(i.e the projected area of the geometry on the plane whose normal is in thickness direction).

We take the aerodynamic constraint as a specified value of the lift coefficient. In order to reduce the computational cost for calculating the gradients of C_L, it is augmented to the cost function by a penalty which is a function of C_L. The modified cost function is then given as

$$J(\boldsymbol{U}, \boldsymbol{\beta}) = K_1 \frac{C_D}{C_{D0}} + K_2 (C_L - C_{L0})^2 \tag{2.8}$$

where K_1, K_2 are the constants, which can be used to control the strength of the constraints; C_{L0} and C_{D0} are the constant values for coefficient of lift and drag respectively, usually the value being that for the control wing.

To summarize, in the present work the minimization of the induced drag is considered with the coefficient of lift held to be constant, with the two geometric constraints, reference area S and the span b, being fixed. The span is kept constant by changing the control variable only in the chord direction while keeping the thickness to chord ratio constant. A C++ code PROP-OPT was developed for the optimization of the shape for the minimum induced drag. PROP-OPT uses an open source optimization library NLOPT which implements sophisticated algorithms for constrained optimization techniques. A flow chart for the algorithm is given in fig. 2.1. We start with a control wing configuration, which is parametrized using NURBS to get the control points which represent the initial shape of wing by using a basis function, as will be discussed in chapter 3 in detail. The state variable value is calculated over the wing using PROP-EULER++ code which is an Euler equation solver. The gradient is then calculated by the adjoint method. The adjoint solver needs the converged state flow variable as an input. The gradient information, the converged flow solution and the geometric constraints with their gradient information are given as inputs to the optimizer. The optimizer uses the subroutines for constrained optimization algorithm and gives a new design variable as an output. This new design

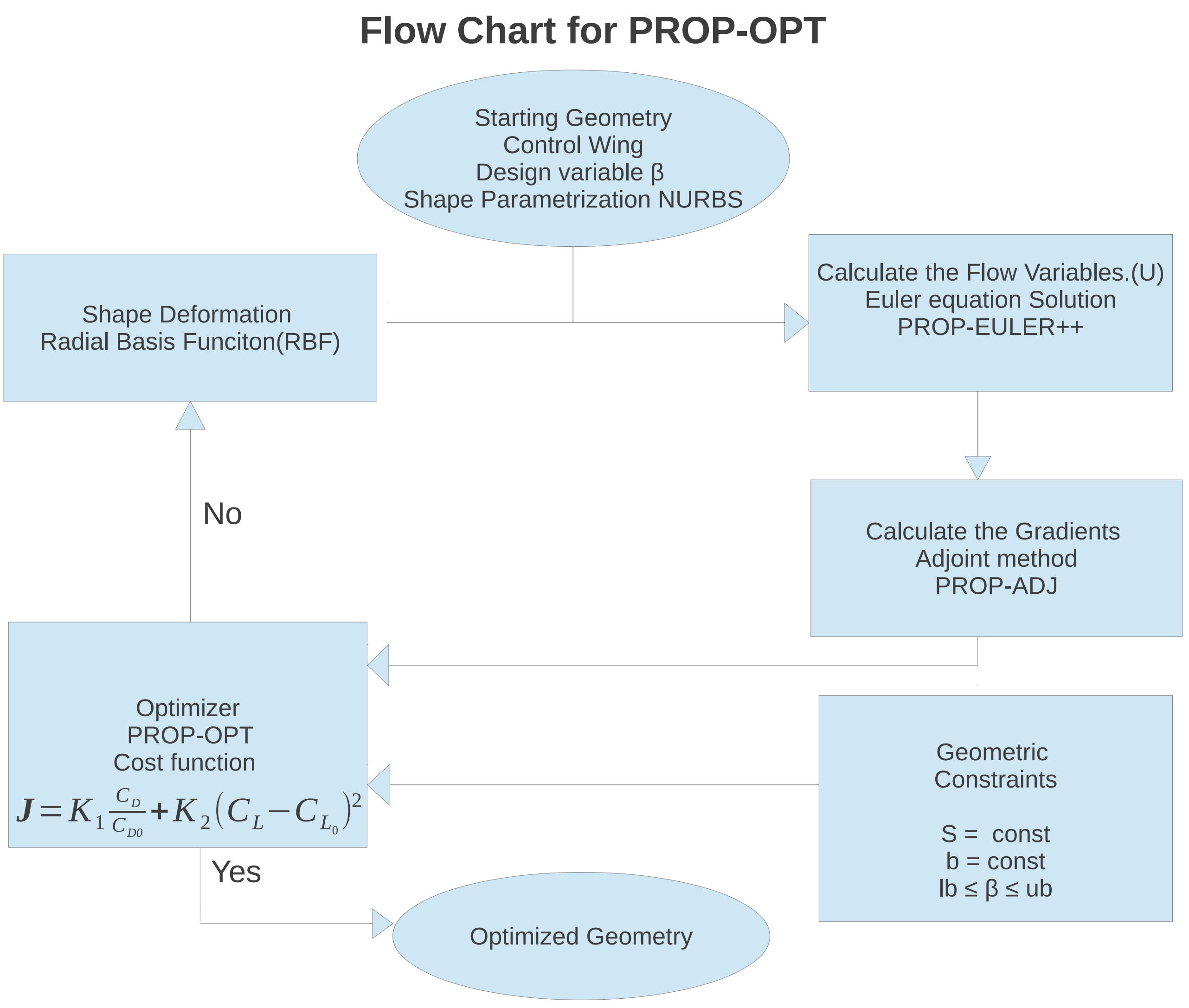


Figure 2.1: Flowchart describing the optimization cycle

variable gives the new shape, and the cycle is completed until the KKT conditions are reached in the optimizer, after which we get the optimized design variable vector.

Chapter 3

Optimal Shape Design- Numerical Details

In this chapter, the numerical details of the various parts of the shape optimization cycle are explained. The part concerning shape parametrization techniques using NURBS is explained in section 3.1. A C^{++} code was written for fitting the mesh points to a NURBS surface to get the control points. The flow solution using the Euler Equation with the propeller source term in the PROP-EULER code of Rakshith (2013) is explained briefly and a C++ implementation of this code was also written for optimizing the code with respect to computing time. Use of the open source optimization library NLOPT is explained in section 3.5. The library is a Non linear Optimization library consisting of various numerical algorithms for various optimization routines available on Internet featuring non linear equality as well as inequality constraints. Hence it gives us a choice in selecting the most suitable optimization method. The present study uses the Sequential Least Square Programing algorithm (SLSQP).

3.1 Shape Parametrization

In order to optimize any shape , it is necessary to express it with a finite number of design variables, preferably as few as possible to minimize the computation cost during the optimization. Shape Parametrization is an essential part of any shape optimization. In this Section we describe the use of Non Uniform Rational B-spline (NURBS) for the parametrization in the frame of Aerodynamic Shape Optimization. In Computer Aided Design, geometric representation methods commonly use Bezier curves, B-splines and NURBS. These representations use control points with defined basis functions to create the surface points. This makes it suitable for getting smooth shapes with desired continuity of surface, and hence very useful in shape optimization. The Bezier curve is the first interpolation method to make use of control points (http://www.tsplines.com). Furthermore, the method is simple to implement, and moving the control points allows for easy and visual shape modification. Nevertheless, Bezier curves have some disadvantages. They cannot represent conics exactly, the parametrization is global (which means that if a control point is moved the whole curve is modified), and increase in the number of control points increases the degree and order of the curve or surfaces, which might not be needed.

One of the main advantages of using the B-spline is having any number of control points with any degree and order of curves and surfaces. B-spline also uses the frame of control points, and the basis function is more complex, but with the advantage that local characteristics such as displacement of a curve or surface induces only local modification. More details on the implementation are found in Shene (2008). Though conics cannot be represented exactly using B-splines, the method is often found to be very useful. Present work uses NURBS, which are an extension of the B-spline, using the same basis function but rationalizing it with weighted coefficients assigned to each control point. This gives another degree of freedom, and the family of curves is much wider than with B-splines or Bezier curves. The algorithm, explained in the next section, is easy to implement with a definition introduced by Cox (1972) and Boor (1978).

3.1.1 Definition of NURBS curves and surfaces

B-splines are constructed using a polynomial function called B-spline basis function. Let $\hat{n}, \hat{m}$ be the number of points in 2 direction along the surface and suppose there are two piecewise polynomial functions of order p, q. Then the product of these two piecewise defined polynomial function is the finite polynomial B-spline surface. Let these functions be N^p, N^q and consider knot vectors $\zeta = (\zeta_1, \zeta_2, .., \zeta_{\hat{n}+p})$ and $\nu = (\nu_1, \nu_2, ..., \nu_{\hat{m}+q})$, defined as follows,

$$\zeta_i = \begin{cases} 0 & \text{if } i < p \;, \\ \frac{i-p}{\hat{n}-p} & \text{if } p \leq i \leq \hat{n} \quad \text{ for } i = 1, ..., \hat{n}+p \;, \\ 1, & i > \hat{n}. \end{cases} \tag{3.1}$$

The B-Spline basis function is computed piecewise and recursively with Cox-deBoor recurrence (Shene 2008). More details can also be found in Becker & Jameson (2011). Introducing

$$\begin{aligned} N_i^1 &= \begin{cases} 1 & \text{if } \zeta_i \leq u < \zeta_{i+1} \\ 0 & \text{otherwise} \;, \end{cases} \\ N_i^p(u) &= \frac{u - \zeta_i}{\zeta_{i+p} - \zeta_i} N_i^{p-1}(u) + \frac{\zeta_{i+p+1} - u}{\zeta_{i+p+1} - \zeta_{i+1}} N_{i+1}^{p-1}(u) \end{aligned} \tag{3.2}$$

The general B- spline surface is given as

$$S(u, v) = \sum_{i=1}^{\hat{n}} \sum_{j=1}^{\hat{m}} N_i^p(u) N_j^q(v) P_{i,j}, \quad u, v \in [0, 1], \tag{3.3}$$

where, $P_{i,j}$ denotes the $(i, j)^t h$ entry of the control points $P \in \mathbb{R}^{\hat{n}} \times \mathbb{R}^{\hat{m}} \times \mathbb{R}^3$. The general

NURBS surface is a B-spline surface with a nonuniform knot vector and is defined as ,

$$S(u,v) = \frac{\sum_{i=1}^{\hat{n}} \sum_{j=1}^{\hat{m}} w_{i,j} P_{i,j} N_i^p(u) N_j^q(v)}{\sum_{i=1}^{\hat{n}} \sum_{j=1}^{\hat{m}} w_{i,j} N_i^p(u) N_j^q(v)}, \quad u,v \in [0,1] \tag{3.4}$$

where $w_{i,j}$ are the weights associated with each control point.

In grid based shape optimization there can be a need to calculate an approximation to the surface grid points which represent the optimized shape. Fitting to a NURBS surface is a powerful technique to change grid point positions, and therefore the shape of the objects, by adjusting the control points; more details of NURBS fitting are found in Becker & Jameson (2011). Structured and unstructured grids are handled differently. Usually unstructured grids are challenging to parametrize as they involve explicitly defined cells with their connectivity information and do not form any structure for the positions of the cells; they can be seen as an arbitrary cloud of points. A structured quadrilateral surface mesh can be easier. However, in unstructured grids the parameters u, v are calculated by projecting all the grid points on a mean plane, and knowing the neighboring elements of the point on the mean plane the grid points can be deformed and spaced in a $[0, 1][1, 0]$ frame. The computational domain mesh in the present work is an unstructured tetrahedral mesh. The knot vector calculation is common to both types of grids, hence can used to calculate the basis functions, and a linear regression problem is solved for the overdetermined system of equations in order to get the control points of the surface grid, which is used as a design variable in the shape optimization.

There are many CAD software packages which use the NURBS surface for having control over the smoothness of the surface, but there are fewer open source code available to fit a NURBS surface from the mesh points. Hence a C++ code was written in-house to get the control points from a grid. The code facilitates both types of grids, unstructured as well as structured. The grid generation was done using GAMBIT (GAMBIT 2004).

3.2 Flow Solutions

Shape optimization for the minimum induced drag requires the state variable to have the constraint of the flow governing equation. Hence there is a need to solve the flow equation at every optimization cycle, which makes the whole process computationally very costly. In the present work the in-house code PROP-EULER^{++} was optimized with respect to computational time with the collaboration of Intel labs, Bangalore. The code is a C++ implementation of PROP-EULER of Rakshith (2013). The code solves the Euler Equation in an unstructured cell centered finite volume method framework for the flow over a wing with propeller. The propeller is modeled by the blade element theory (henceforth called BET) module. This module is used to provide the source distributions of momentum and

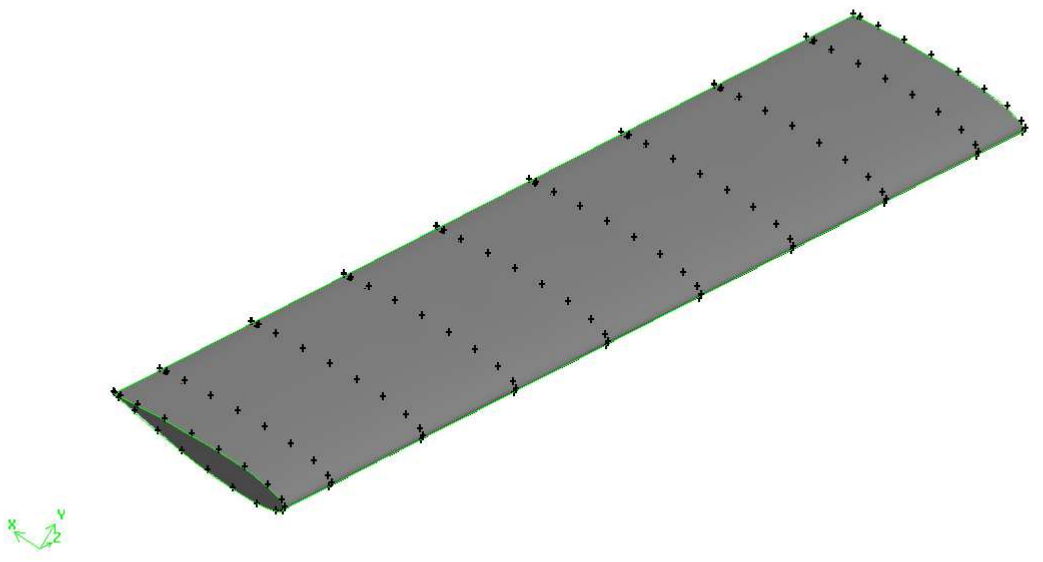


Figure 3.1: NURBS surface of a rectangular wing, the black points indicates Control points and gray shade is the surface, 210 control points with 21 along chord and 10 along span.

energy on an infinitesimally thin actuator disk that represents a propeller in the PROP-EULER^{++} code. The details of the PROP-EULER^{++} code and results of simulation using it are shown subsequently.

The code solves the 3-D inviscid equations with a source term,

$$\frac{\partial U}{\partial t} + \frac{\partial}{\partial x}(G_x) + \frac{\partial}{\partial y}(G_y) + \frac{\partial}{\partial z}(G_z) = S \tag{3.5}$$

where U is the vector of conserved variables , G_x, G_y and G_z are the flux vectors along the coordinate directions x, y and z respectively, and S is the vector of Source terms, thus

$$U = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho u_z \\ \rho e \end{bmatrix} \quad G_x = \begin{bmatrix} \rho u_x \\ p + \rho u_x^2 \\ \rho u_x u_y \\ \rho u_x u_z \\ (p + \rho e)\, u_x \end{bmatrix} \quad G_y = \begin{bmatrix} \rho u_y \\ \rho u_x u_y \\ p + \rho u_y^2 \\ \rho u_y u_z \\ (p + \rho e)\, u_y \end{bmatrix} \quad G_z = \begin{bmatrix} \rho u_z \\ \rho u_x u_z \\ \rho u_y u_z \\ p + \rho u_z^2 \\ (p + \rho e)\, u_z \end{bmatrix}, \quad S = \begin{bmatrix} 0 \\ S_x \\ S_y \\ S_z \\ S_e \end{bmatrix} \tag{3.6}$$

Here, ρ is density, u_x, u_y, u_z are components of the fluid velocity along the x, y, z directions respectively, p is pressure and e is the total energy per unit mass, given by

$$e = \frac{p}{\rho(\gamma - 1)} + \frac{1}{2}\left(u_x^2 + u_y^2 + u_z^2\right) \tag{3.7}$$

The first entry in S represents the mass source (zero strength), S_x, S_y, S_z are the momentum sources in x, y and z directions respectively and S_e is the energy source given by

$$S_e = u_x S_x + u_y S_y + u_z S_z \tag{3.8}$$

The momentum sources are calculated using BET. The details of the BET can be found in Rakshith (2013), however we will breifly state the equations.

If we consider the disk approach to model the propeller, then the total thrust on the disk is given by

$$T = \int_0^R \int_0^{2\pi} \sigma(r)\, r \,\mathrm{d}\theta \,\mathrm{d}r = \int_0^R \sigma(r)\, 2\pi r \;\mathrm{d}r \tag{3.9}$$

$$Q = \int_0^R \int_0^{2\pi} \tau(r)\, r \,\mathrm{d}\theta \,\mathrm{d}r = \int_0^R \tau(r)\, 2\pi r \;\mathrm{d}r \tag{3.10}$$

where σ is the thrust density and is the torque density.

Expression for the thrust density is given by

$$\sigma(r) = \frac{N\, 0.5\, \rho\, V_R^2\, c\, (c_l \cos\phi - c_d \sin\phi)}{2\pi r} \tag{3.11}$$

and for the torque density τ

$$\tau(r) = \frac{N\, 0.5\, \rho\, V_R^2\, c\, r\, (c_l \sin\phi + c_d \cos\phi)}{2\pi r} \tag{3.12}$$

where N is the number of blades, ρ is the density, V_R is the velocity of air relative to the airfoil section of the blade at r, defined as the vector sum of the axial component $V(1+a)$ and the circumferential component $r\Omega(1-a^{'})$ with Ω, representing the angular velocity, a and $a^{'}$ representing the axial and rotational inflow factors respectively, and V the flight speed, c_l and c_d are the lift and drag coefficients of the propeller blade at radius r and ϕ is the angle of advance.

With the torque and thrust densities at any given radius, calculated using eqs. 3.11 and 3.12, they are then integrated over the area of the surface of the finite volumes abutting the disk to obtain the surface force and hence the momentum sources in r, θ, z coordinates. These are then transformed to Cartesian coordinates x, y, z to obtain S_x, S_y, S_z.

While carrying out the numerical simulation,

- c_l and c_d values for airfoil sections at prescribed radii are computed for a range of angles of attack by prescribing the local flow conditions (i.e. sum of flight speed (V) and tangential velocity ($r\Omega$)) and given as an input to the blade element module. These coefficients are computed here using XFOIL (Drela & Giles 1987);

- aerodynamic coefficients for an airfoil at any radius in between two discrete stations (considered in previous step) are computed using the coefficients of adjacent stations by linear interpolation;

- aerodynamic coefficients for an airfoil at any angle of attack in between two angles of attack considered are computed using the coefficients of adjacent angles of attack by spline interpolation;

Eq. (3.5) is solved numerically using a second order positivity-preserving KFVS (Kinetic Flux Vector Splitting) scheme (Ghosh *et al.* 1998), which employs reconstruction of the entropy variables called q-variables (Deshpande 1986). It has been shown that the use of q-variables gives a computationally efficient code that yields smooth solutions (Ghosh *et al.* 1998).

Unstructured grids were used to discretized the domain, and were generated using GAMBIT (GAMBIT 2004).For faster convergence implicit time stepping based on point Jacobi and LUSGS (Lower Upper Symmetric Gauss Siedel) was used (Jameson & Yoon 1986). The code was optimized for computational time in collaboration with Intel labs, Bangalore for implementation of Single Instruction Multiple Data (SIMD) for use in the optimization routines.

The propeller is modeled as an infinitesimally thin rotating actuator disk which not only produces a pressure jump across it and accelerates the flow along the axis of the propeller but also imparts a swirl to the flow (hence the word "rotating") (Rajagopalan 1989; Lötstedt 1995). This is accomplished by considering the forces that the propeller imposes on the fluid as source terms along all the three directions in the momentum equations and in the energy equation. If x is along the propeller axis, then the sources in y and z directions impart an angular and radial momentum to the fluid passing through the disk. The source terms are zero everywhere except in the region where the actuator disk is present. To estimate the source strengths, a distribution of force densities on its surface in all the three directions is required, which when integrated over the surface area of the finite volume abutting the disk, gives the average force acting on that particular element of area. These forces, converted to force densities, act as source terms in eq. (3.5). More Details on implementation of propeller module is found in Rakshith (2013).

3.2.1 Validation of code

Transonic Flow past the Onera M6 wing was chosen as the test case for the validation of PROP-EULER^{++}. As the equations contain a source vector we now keep it zero for validating it for the wing without propeller. This is a general validation test case for the performance of numerical schemes and the correctness of the code. The free stream was at a Mach number of 0.8395 and angle of attack of 3.06°. The arrow in fig. 3.2 shows the direction of flow. The λ-shock structure on the suction surface of the wing (seen in the experiments) can be identified by plotting pressure contours obtained from the Euler code as shown in fig. 3.2. The coordinate system of the wing is shown on the top right of fig. 3.2 where x is along the chord, y is along the thickness and z is along the span of the wing. Quantitative surface pressure comparisons at 44% and 65% of the semispan are shown in figures 3.3, 3.4.

PROP-EULER^{++} is a C++ implementation of PROP-EULER hence we use the same validation as was done by Rakshith (2013). Experimental results from Hartman & Biermann (1938) on a propeller having R. A. F. 6 aerofoil sections were used for validating the propeller module. All geometric details needed to model the propeller as an actuator disk in the Euler code were available in the report. The case chosen has the following details;

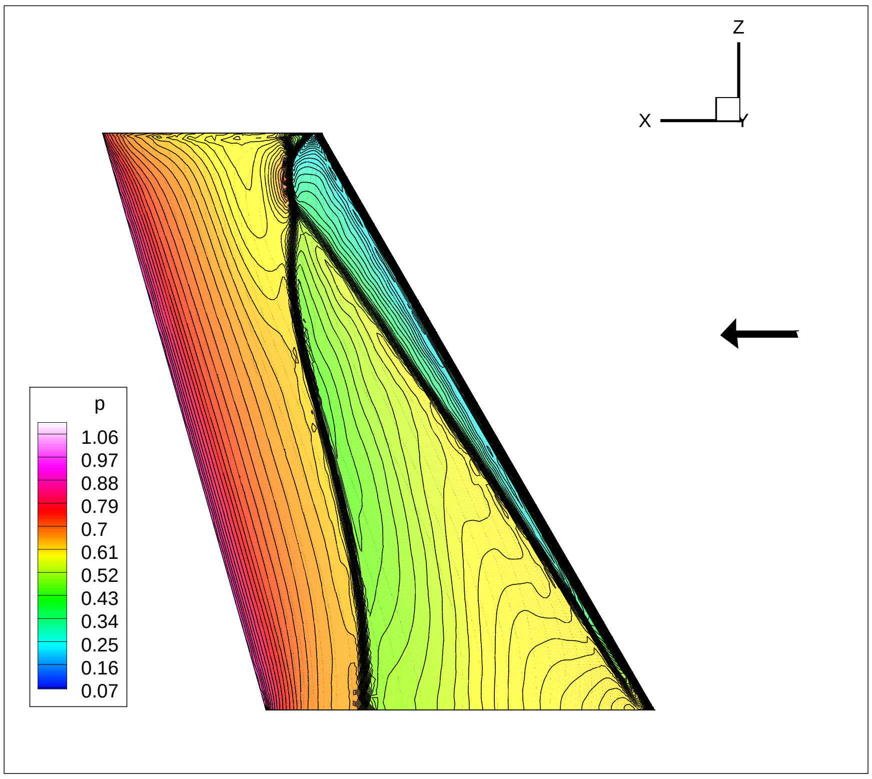
- Diameter: 10 ft
- Number of blades: 4
- Blade aerofoil section: R. A. F. 6
- Propeller blade geometry as shown in fig. 3.5. Pitch setting of 25° at 75% radius was considered.
- Propeller speed: 1000 rpm

Using the chord, pitch and aerofoil thickness variation along the radius of the propeller given in fig. 3.5, aerofoils were constructed and their corresponding drag polars were computed using XFOIL. These act as an input to the propeller module.

CFD simulations were carried out for advance ratios 1, 1.1, 1.2 and 1.3 and for each case the thrust coefficient was computed. Fig. 3.6 shows the variation of thrust coefficient with advance ratio. The continuous line represents experimental results from Hartman & Biermann (1938) drawn through the points obtained after digitizing fig. 3.7. It can be seen in fig. 3.6 that there is close agreement between the numerical simulations and experimental data. For advance ratios below 1, the free stream mach numbers were becoming very low (< 0.1) which posed difficulties in running the compresible code. Therefore simulations for advance ratios greater than or equal to 1 were carried out.

3.3 Gradient Calculation

As is well known, in gradient-based shape optimization technique the converged flow solution and gradient evaluation with respect to each design variable are required for each optimization cycle. In the present work a C++ code was written to calculate the gradients of the cost function, which solves the adjoint equations as explained in detail in the



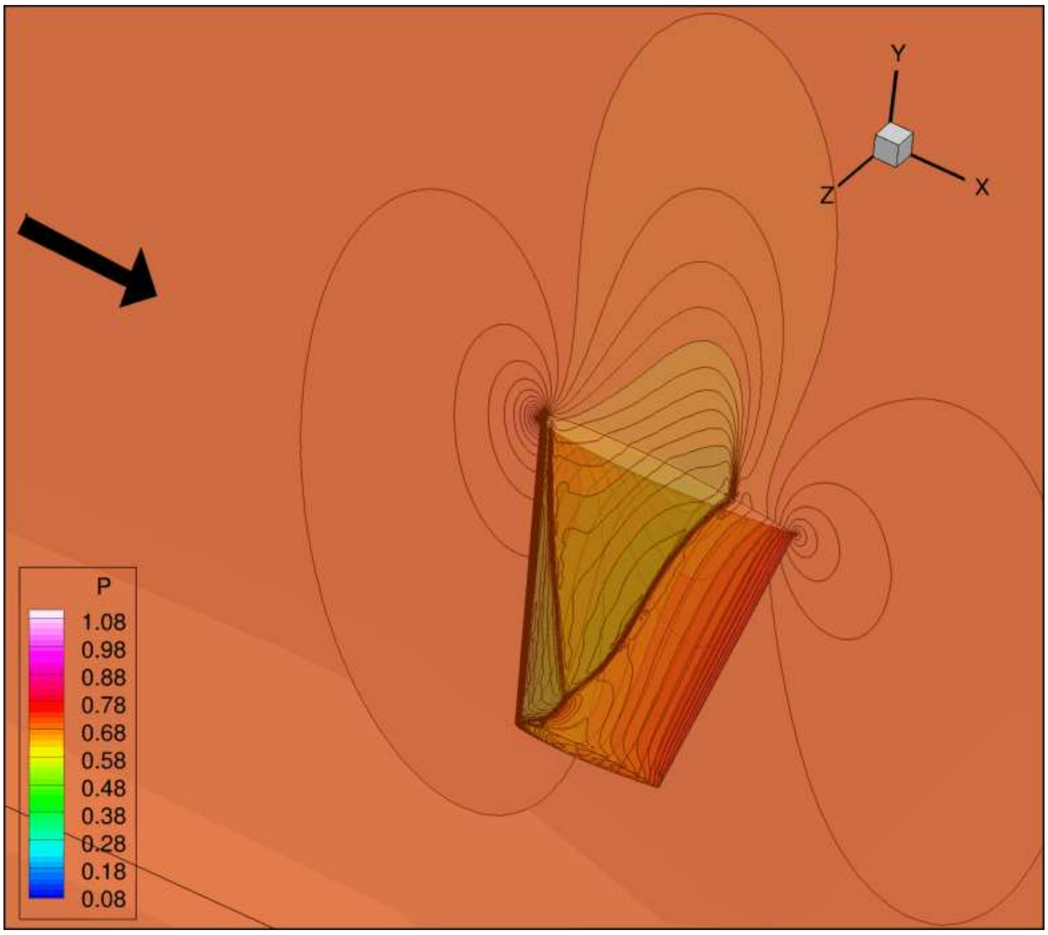


Figure 3.2: Pressure contours on the surface of Onera M6 wing obtained using PROP-EULER^{++} code. λ-shock structure is seen on the suction side of the wing.

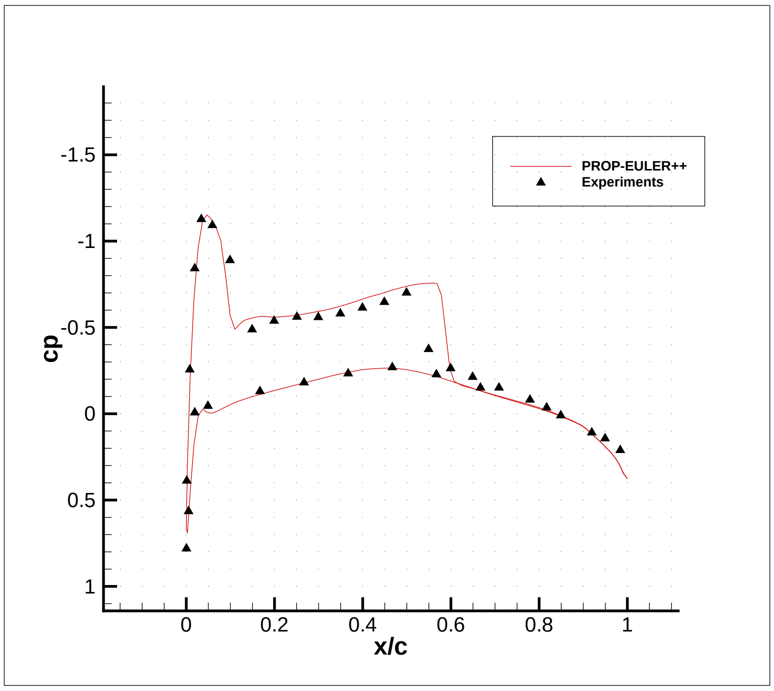


Figure 3.3: Surface pressure coefficient at 44% semi-span

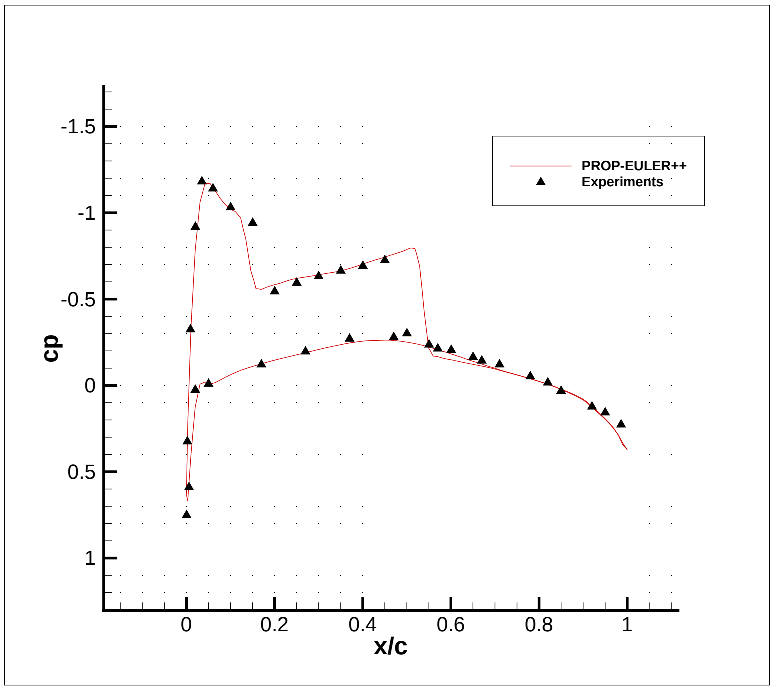


Figure 3.4: Surface pressure coefficient at 65% semi-span

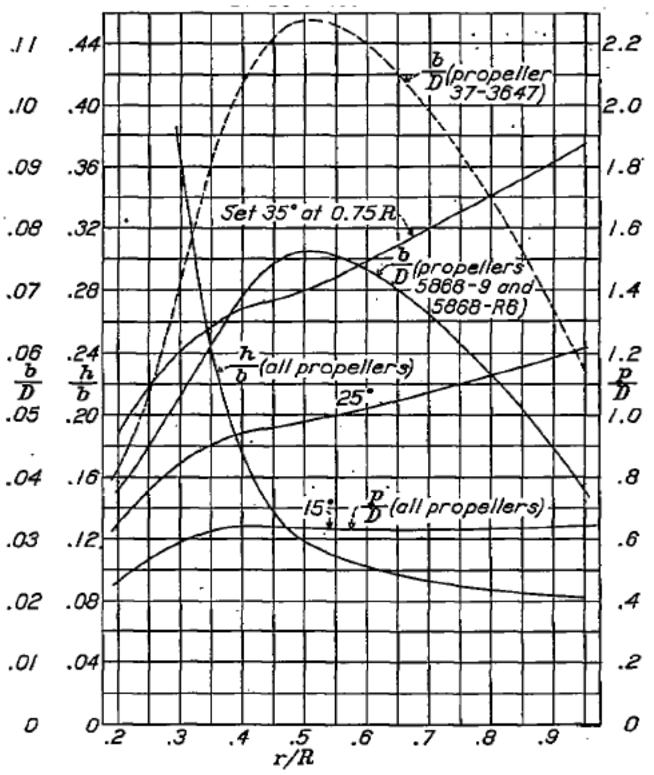


Figure 3.5: Description of propeller blade geometry used for validation of blade element module (figure taken from Hartman & Biermann (1938))

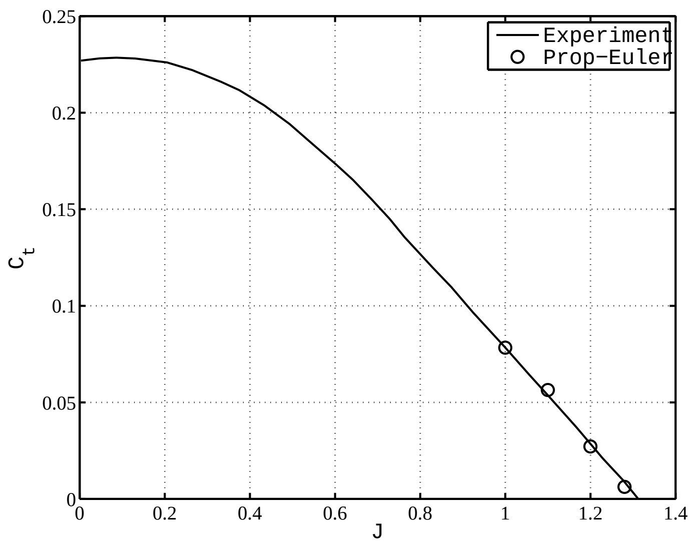


Figure 3.6: Plot of thrust coefficient versus advance ratio showing the validation of PROP-EULER code (Rakshith 2013)

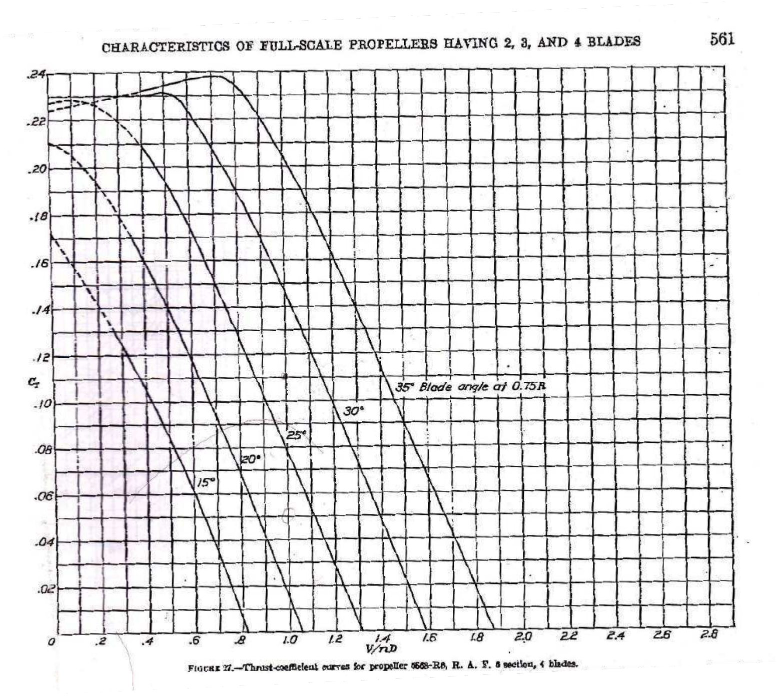


Figure 3.7: Plot of thrust coefficient versus advance ratio for different blade settings, taken from Hartman & Biermann (1938)

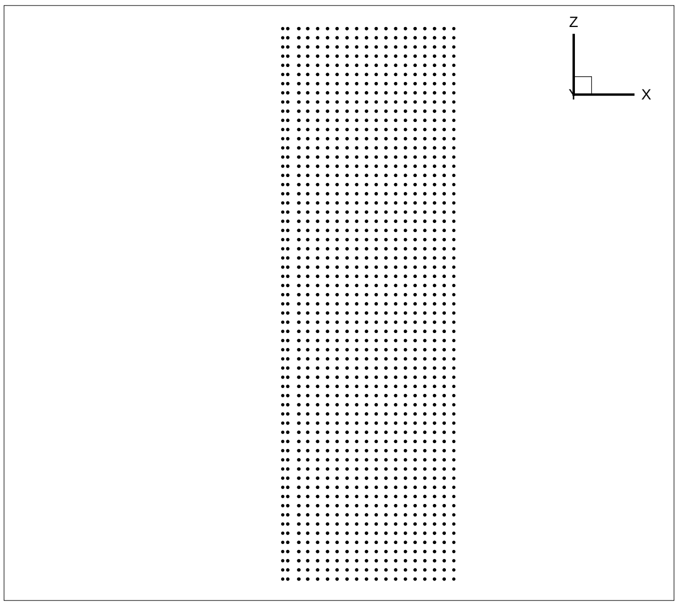
next chapter, section 4.3. A direct way of calculating the gradient is also discussed, making use of finite difference methods, along with the use of the adjoint equation solution to reduce the computational cost. The code is coupled with the automatic differentiation subroutines to calculate the adjoint variable, which enables calculation of the total derivative.

3.4 Mesh deformation

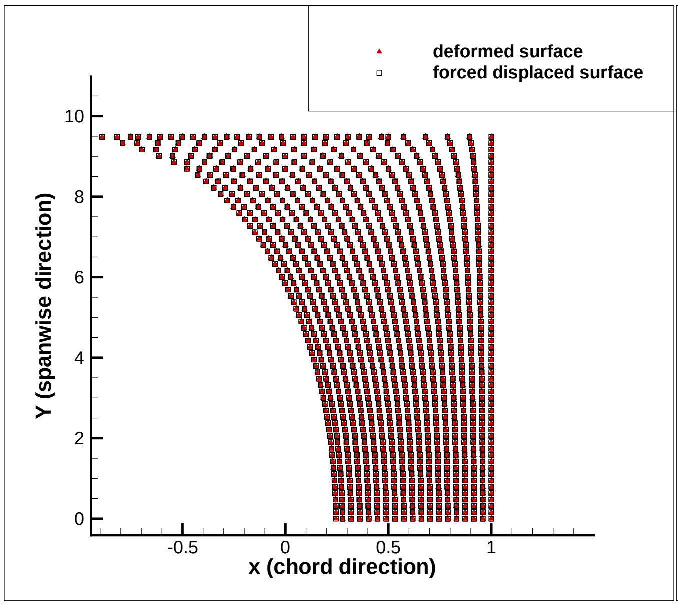
In each optimization cycle a new set of design variables is created, which represents the new wing shape. To solve the flow governing equation on this new shape we need to have a new domain grid which fits the new wing shape. This can be done by re-gridd the whole domain again, but this is a computationally costly operation. In the present work we use a deformation technique which, instead of re-gridding the whole domain, uses the old mesh and deforms it towards the new shape. The displacement in the shape of the wing is extrapolated to the mesh points on whole domain. To do so, we have used radial basis functions. A C++ code was written for this purpose to calculate the new deformed grid as an output for a given input mesh with the surface displacements. The code also recalculates all the cell property information. The present work uses radial powers as the radial basis function; a detailed description of the implementation and the algorithms for radial basis functions can be found in Jakobsson & Amoignon (2006). Fig.3.8 shows the deformation done for a toy surface of a rectangular wing with NACA0012 airfoil, with a coarse grid of 30,000 cells in the domain and 800 nodes on the surface. Fig. (a) shows the initial grid. In order to check the deformation a forced displacement is given to the surface node points, and then the whole grid is deformed for this new forced displacement on the surface. The wall points on the deformed grid are shown in comparison with the 2 different initial given displacements to points on the surface in figures (b) and (c).

3.5 Shape optimization

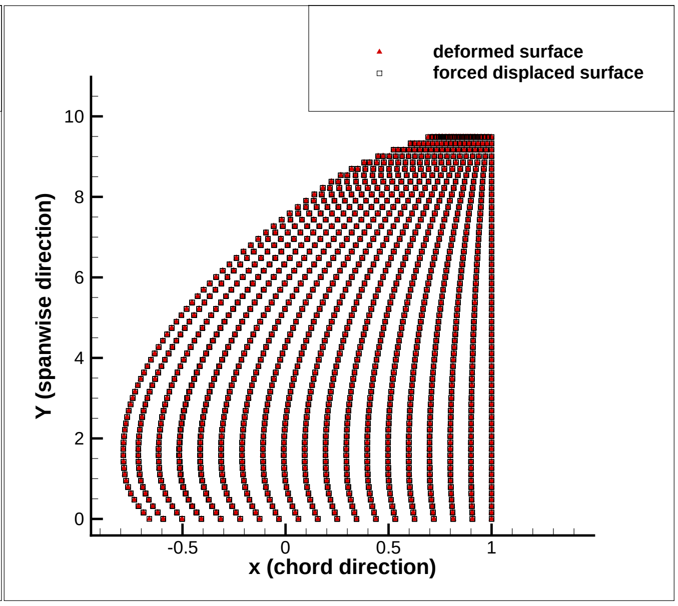
In the present study, a software package PROP-OPT was developed which uses NLOPT (http://ab initio.mit.edu/wiki/index.php/NLopt) for the present nonlinear constraint shape optimization. NLOPT features C++ library with a common interface for a number of different free optimization routines available on Internet. It also has good support for large scale optimization algorithms, which give us the choice to use a suitable optimization algorithm depending on the problem on hand. Although NLOPT has features for gradient-based as well as gradient-free methods, present study uses Sequential Least Square Quadratic Programing Algorithm (SLSQP) for nonlinear constrained gradient-based optimization, which supports inequality as well as equality constraints, and is based on the implementation by Kraft (1988). PROP-OPT coupled the NLOPT with the Euler Equation solver and the adjoint equation solver for gradients. The control points of NURBS are used as optimization variables.



(a) initial wing surface nodes



(b)



(c)

Figure 3.8: comparison of the forced deformation and the surface points of the deformed grid

Chapter 4
Sensitivity Analysis and Adjoint Method

We consider in this chapter the methods used for analyzing the sensitivities, which is performed after the flow solution has been obtained. The purpose is to calculate the sensitivity of cost functions with respect to the control parameters defined by NURBS. The main question of interest is, what is the perturbation in the cost function due to a perturbation in the geometry? In the present work two sensitivity analysis methods are considered. The first one can be solved directly by using the finite difference and complex steps method, the implicit gradient method etc. The second one solves the dual problem, which is also referred as the adjoint problem.As will be shown below, although the present work focuses mainly on the adjoint problem, the direct method is a necessary intermediate step, which is required for the purpose of developing and testing the adjoint formulation. It will also be discussed in the case of gradient based shape optimization, where there are more control points. Adjoint methods can reduce the computational cost very efficiently.

4.1 Sensitivity Gradient

We now present a way to calculate the sensitivity gradient, which is necessary for determining the direction of descent or ascent towards the optimum. Consider a scalar cost function J, which is to be minimized in the inverse design, for given design variables β and U, where β is the control parameter based on the NURBS parameterization, and U is the state flow variable vector which satisfies the Euler Equations. The parameters U and β can be interpreted as the input to the system whereas the output must be the cost function J in a control theory approach. The minimization of the cost function is subject to the constraint that the discrete flow equation and boundary conditions are all satisfied. This is expressed as

$$R(U, \beta) = 0. \tag{4.1}$$

As the cost function is a function of U and β, its total derivative is given by,

$$\frac{dJ(U, \beta)}{d\beta} = \frac{\partial J}{\partial \beta} + \sum_{i=0}^{nt} \frac{\partial J}{\partial U_i} \frac{\partial U_i}{\partial \beta} \tag{4.2}$$

where nt is the number of cells in the computational domain.

The above equation contains the flow sensitivity term ($\partial U_i/\partial\beta$) which is subjected to the constraint of the governing equation, which in the present case are the Euler equations. A direct way of solving the equation for the gradients is by use of the finite difference method, which is elaborated in the following section, but we will see that these methods are computationally costly as the number of design variables β increases. One way of solving the equation is by augmenting the cost function by the product of the flow residual and a new variable called the adjoint variable, then solving a dual problem to calculate the gradient. This method is called the adjoint method, which we will discuss section 4.3.

4.2 Direct Differentiation Methods

One of the earliest methods of finding the sensitivity gradients is by the use of the finite difference method. The gradients are calculated by perturbing the design parameters and then solving the flow equations for a new perturbed conserved vector U which results in the perturbed cost functions $I(\beta \pm \Delta\beta)$. A central difference formula for a gradient approximation is then given by,

$$\frac{\partial J}{\partial\beta} \cong \frac{J(\beta+\Delta\beta) - J(\beta-\Delta\beta)}{2\Delta\beta} + O(\Delta\beta)^2. \tag{4.3}$$

In the above equation the leading term in the truncation error clearly shows that the gradient is second order accurate. The truncation error can be controlled by choosing the values of $\Delta\beta$. An interesting point to observe is that each of the sensitivity gradients requires two flow solutions. The major disadvantage of this method is that it is computationally expensive, if there are a large number of control variables. The central differencing method requires $2N$ flow solutions in order to find the sensitivity gradients with respect to N control variables, Hence this method is better when the number of design variables is small.

Another method of calculating the gradients is the implicit gradients method, in which the variation in the cost function is given by

$$\delta J = \frac{\delta J}{\delta U}\delta U + \frac{\delta J}{\delta\beta}\delta\beta \tag{4.4}$$

This equation shows the variation of cost function due to variation in the state variable δU and the design variable $\delta\beta$. Here the perturbation δU can be calculated by considering the linear variation in the governing steady flow equations and solving a system of linear equations given by

$$\frac{\partial R}{\partial U}\delta U = -\frac{\partial R}{\partial\beta}\delta\beta \tag{4.5}$$

The sensitivities can be calculated by back substituting for the δU in equation 4.4. For each control variable β a linear system of equations is solved, which makes this method also computationally very expensive.

4.3 Discrete Adjoint Approach

The previous section showed the direct way of computing the sensitivities where at least one flow solution was required to calculate the gradient after perturbing each design variable. Now here using control theory, the sensitivity of the cost function is being made independent of the flow variable δU by augmenting the cost functions. This makes the flow calculation only once for any number of design variables, hence giving an advantage of much less computational cost.

Consider the discrete state flow equation,

$$R(U, \beta) = 0. \tag{4.6}$$

The gradient of the above equation with respect to the state variable U and the design variable β is given by

$$\frac{dR}{d\beta} = \sum_{i=0}^{nt} \frac{\partial R_i}{\partial U_i} \frac{\partial U_i}{\partial \beta} + \frac{\partial R_i}{\partial \beta} \tag{4.7}$$

Let ψ be the Lagrange multiplier or adjoint variable. Using eq. (4.6) we can augment the cost function and then calculate the gradient as,

$$\frac{dJ}{d\beta} = \frac{\partial J}{\partial \beta} + \sum_{i=0}^{nt} \left(\frac{\partial J}{\partial U_i} \frac{\partial U_i}{\partial \beta} + \psi_i^T \frac{\partial R_i}{\partial U_i} \frac{\partial U_i}{\partial \beta} + \psi_i^T \frac{\partial R_i}{\partial \beta} \right) \tag{4.8}$$

Now the above equation shows that the term $\partial U_i / \partial \beta$ can be taken as a common factor, and then we can find a suitable value of ψ_i such that the coefficient of $\partial U_i / \partial \beta$ is zero. Now we see that remaining terms of eq. (4.8) do not have a sensitivity to the state variable.

Collecting all the terms that are coefficients of the $\partial U_i / \partial \beta$ and equating them to zero we get the adjoint system of equations.

$$\psi_i \left(\frac{\partial R_i}{\partial U_i} \right)^T + \left(\frac{\partial J}{\partial U_i} \right)^T = R_i^* = 0 \tag{4.9}$$

Now we see that the adjoint equations are linear in ψ_i. The equations are solved for the adjoint variables ψ_i and they are then substituted in eq. (4.8) to get the total gradient of the cost function, which is then given by

$$\frac{dJ}{d\beta} = \frac{\partial J}{\partial \beta} + \sum_{i=0}^{nt} \psi_i^T \frac{\partial R_i}{\partial \beta} \tag{4.10}$$

The main advantage of this formulation is that the variation in cost function is independent of the number of design variables. As two equation for the gradient 4.9 and 4.10 are solved, this approach is also known as duality formulation by Giles & Pierce (2000)

Two approaches are known for the formulation of the adjoint equation, continuous adjoint method and discrete adjoint method. The present work focuses on the discrete adjoint approach. In the continuous adjoint method, the governing state partial differential equations are first linearized and then combined with the first variation in the cost function using adjoint variables. More details can be seen in Jameson (1988). In the discrete adjoint approach, developed by Elliot (1998), Neilson (1998), Mohammadi & Pironneau (1999) and H. J. Kim & Nakahashi (2000), the governing equations are first discretized, then linearized and then combined with the discrete form by using the adjoint variables. A detailed comparison of the two approaches can be found in Nadarajah & Jameson (2000).

The adjoint equation 4.9 is usually very large and is solved iteratively by introducing a pseudo time derivative,

$$V_i \frac{d\psi_i}{dt_i} + R_i^* = 0, \tag{4.11}$$

where R_i^* is the adjoint residual defined by eq. (4.9), and V_i is the volume of the control volume considered. The above equation is solved by local time stepping combined with the implicit scheme for convergence acceleration. Note that the adjoint equation (4.9) does not have any derivatives with respect to the design variables.

As the adjoint equation is solved iteratively by introducing the time derivative, its state update is given as

$$\psi_i^{n+1} = \psi_i^n - \frac{\Delta t}{V}\left(\psi_i^n \frac{\partial R_i}{\partial U_i}^T + \frac{\partial J}{\partial U_i}^T\right). \tag{4.12}$$

Here R_i is the flux residual of the Euler equations. If F is the flux of the conserved variable U then the residual for a control volume can be given as

$$R_i = \sum_{i=0}^{noe} \mathbf{F}.\hat{\mathbf{n}} dA \tag{4.13}$$

where noe is the number of faces of a control volume, dA is the area of the face and $\hat{n}$ is the unit vector normal to the face.

As we have seen in chapter 4 the propeller is modeled by considering the forces that it imparts on the fluid as a source term in the Euler equations. The residual R_i in the above equation is calculated considering the Euler equations.

To achieve higher convergence rates we have used an implicit scheme to solve the discrete adjoint equations using a matrix free Lower Upper Symmetric Gauss Seidel (LUSGS) method (Jameson & Yoon 1986). The detailed derivation can be found in Anil (2008), so we shall give only a brief account here

The state update of the adjoint equation can be written as

$$V_i \frac{\psi_i^{n+1} - \psi_i^n}{\Delta t_i} + R^*(\psi_i^{n+1}) = 0 \tag{4.14}$$

Now the above equation is rearranged, and using the kinetic flux vector splitting scheme (Deshpande 1986) for the flux and using Jameson's spectral approximations to make it matrix free, we get the equation

$$\left[\frac{V_i}{\Delta t_i} + \frac{1}{2}\sum_{i=0}^{nbh(i)} \rho_i n_{ij} dA_{ij}\right]\Delta\psi_i^n + \left[\frac{1}{2}\sum_{j=0}^{nbh(i)} (A - \rho I)_i^T n_{ji} dA_{ij}\right]\Delta V_j^n = -R_i^*(\psi^n) \tag{4.15}$$

where ρ is the spectral radius given by Jameson , I is the Identity Matrix, A is the full flux Jacobian , n_{ji} is the unit normal to a face and n is the time level and nbh is the number of neighboring cells of the cell i.

Using the identity $\sum_{j=0}^{nbh(i)} n_{ij} dA = 0$ the above equation is written in matrix form as

$$M_{ij}\Delta\psi_i^n = -R_i^*(\psi_i^n) \tag{4.16}$$

where M_{ij} is defined as

$$\begin{aligned} M_{ij} &= -\frac{1}{2}\sum_{j=0}^{nbh(i)} (A - \rho I)_i^T n_{ij} dA, \quad \text{for } i \neq j \\ &= \frac{V_i}{\Delta t_i} + \frac{1}{2}\sum_{j=0}^{nbh(i)} \rho_i n_{ij} dA, \quad \text{for } i = j \end{aligned} \tag{4.17}$$

Now the solution of eq. (4.15) requires calculation of the inverse of the matrix M_{ij}. Using implicit LUSGS method the matrix M is factorized into three parts, respectively lower, symmetrical and upper triangular matrices, and the matrix systems can be solved in two steps given as follows

$$D^{-1}(D + U)\Delta\psi^n = \Delta\psi^{*n} \tag{4.18}$$

$$(L + D)\Delta\psi^{*n} = -R^*(\psi^n). \tag{4.19}$$

After further simplifications we get

$$\Delta\psi_i^{*n} = -D_i^{-1}R_i^* - D_i^{-1}L_i\Delta\psi_j^{*n} \tag{4.20}$$

$$\Delta\psi_i^n = \Delta\psi_i^{*n} - D_i^{-1} U_i \Delta\psi_i^n \tag{4.21}$$

$$D_i = \frac{V_i}{\Delta t_i} + \frac{1}{2}\sum_{j=0}^{nbh(i)} \rho_i n_{ij} dA \tag{4.22}$$

$$L_i \Delta\psi_j^* = -\left[\cdot\frac{1}{2} \sum_{j=0}^{nbh(i)} \left\{ \left(\frac{\partial F_i}{\partial U_i}\right)^T \Delta\psi_j^{*n} - \rho_i \Delta\psi_j^{*n} \right\} n_{ij} dA \right] \quad \text{for } j < i \tag{4.23}$$

$$U_i \Delta\psi_j^n = -\left[\frac{1}{2} \sum_{j=0}^{nbh(i)} \left\{ \left(\frac{\partial F_i}{\partial U_i}\right)^T \Delta\psi_j^{n} - \rho_i \Delta\psi_j^{*n} \right\} n_{ij} dA \right] \quad \text{for } j > i \tag{4.24}$$

Here Δt_i is the time step, and D, U, and L are the diagonal, upper and lower triangle matrices formed from the matrix M.

We see that the sensitivity and the adjoint equations contain the derivatives of cost function J and the flow flux residual R. These derivatives can be evaluated in several ways. One of the classic ways is to manually hand differentiate it, in which the analytical expression for the derivative of a given function is derived and subroutines are created. Often this method turns out to be laborious, time consuming and error prone because of the complexity of the given function. Another way, which was also mentioned in section 4.2, is the finite difference method. We can use the central differencing formula and calculate the derivatives with second order accuracy. However there is another method to calculate derivatives, which is known as Automatic Differentiation and is used in the present work. In this method, differentiation of subroutines is performed by a computer tool called TAPENADE, which is developed by INRIA (http://tapenade.inria.fr:8080/tapenade). We can find other several softwares available which can also perform Automatic Differentiation (AD) like ADIFOR, ADOLC, TAMC,TAF, etc; more details can be found on http://www.autodiff.org.

TAPENADE consists of elementary functions whose derivatives are known and uses these in a form of chain rule to calculate the desired derivatives. The programming language used in present work is C, and for a given subroutine which calculates a function, TAPENADE gives a subroutine as an output which calculates the derivative of the function. It is to be noted that AD tools rely on the assumption that the function which is calculated by input subroutines is piecewise differentiable. AD does not incur truncation error and gives exact values for the derivatives if the floating point operations are performed with infinite precision arithmetic. More details of AD are presented in the Appendix.

In general AD can be carried out in two modes, forward and reverse. These modes are distinguished by the way the chain rule of differentiation is applied to a sequence of elementary operations. In the present work, since there is a need of calculating the transpose of the gradient, the reverse mode is more suitable in the adjoint equation solver. More details can be found in Praveen (2006).

The Algorithm for solving the Adjoint Equations can be stated in steps as follows,

1. Give the converged flow solution as an input parameter with the flow conditions and the grid files.

2. Initialize the state variables to converged flow solutions and adjoint variables to zero. item Calculate the Gradients $\partial J/\partial U$, $\partial R/\partial U$ at each cells using AD subroutines, created by TAPENADE.

3. Calculate the adjoint variable ψ at next time level using the state update eq.(4.12)

4. Repeat step 3 until convergence.

Earlier in this section the Adjoint equation was solved using a pseudo time derivative in an iterative manner. However, as the adjoint equation is linear in the adjoint variable, it can be solved by using a linear algebra package for system of linear equations, for example GMRES with suitable preconditions.

The state update can be done by implicit methods, in the present work the LUSGS (Lower Upper Symmetric Gauss Seidel) and Point Jacobii methods were implemented in adjoint solver for better convergence. The code developed for this purpose, named PROP-ADJ, used MPI subroutines to run on parallel machines. After the adjoint calculation the total gradients are given as

$$\frac{dJ}{d\beta} = \frac{\partial J}{\partial \beta} + \sum_{i=0}^{nt} \psi_i^T \frac{\partial R}{\partial U_i} \tag{4.25}$$

Here again the partial derivative terms, $\partial J/\partial \beta$ and $\partial R_i/\partial U_i$, are calculated by the above mentioned methods.

The adjoint solution can be indirectly validated by checking the accuracy of the gradients. This has been done by comparing the gradients with the finite difference (FD) approximations. We used OneraM6 wing for the validation. As there is a need of converged flow solution twice to calculate the gradient for each of the control points, a very coarse grid is used. The number of control points used was 36; these had 4 component directions x , y , z *and* w each being in the coordinate directions and w as the weight associated to each control points. The finite difference approximation will have discretization and round off error, which largely depends on the step size used in finite difference operations, a large step may lead to more discretization error whereas a very small step

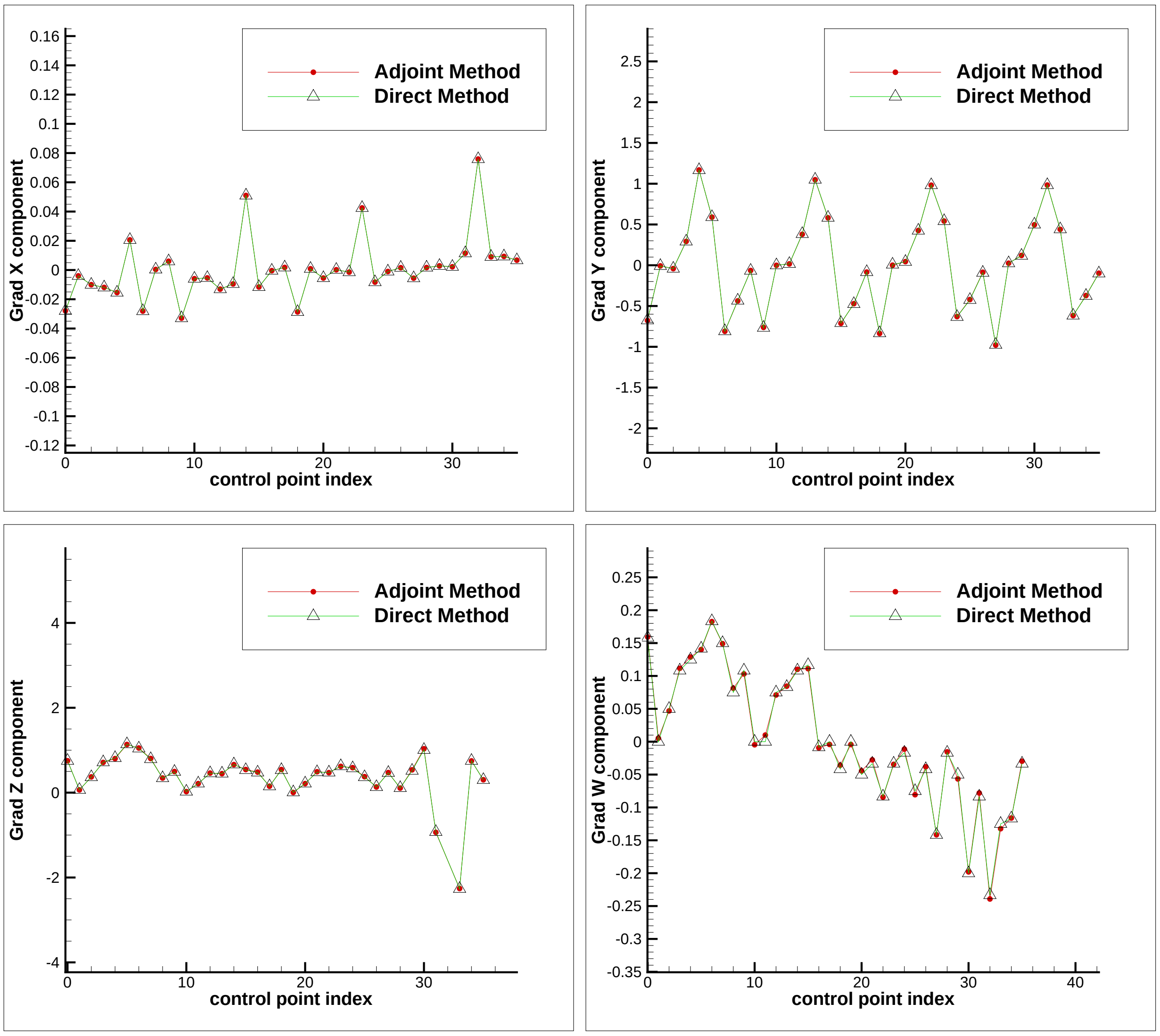


Figure 4.1: Comparision of the Finite diffrence gradients and the Gradients obtained from Adjoint Solver for 36 control points for course grid of 2717 tetrahedral cells

can lead to round off errors, hence there should be an optimum step size. But this is difficult to determine because of its dependency on many factors like complexity of the objective/cost function.

Chapter 5

Two Case Studies

We now present the results and validation for aerodynamic shape optimization for a wing carried out using the code PROP-OPT in the presence of a propeller. We consider a rectangular wing with NACA0012 airfoil as control wing for all the cases. The aspect ratio of the wing is 12 with mean chord of 1 and the taper ratio 1. We consider level cruise flight conditions at lift coefficient of 0.4. The slipstream needed was calculated using PROP-EULER++ code (described in chapter 3). The propeller, mentioned in Hartman & Biermann (1938), is chosen for all the cases in the present optimization studies. This particular propeller was chosen because the entire information needed to carry out the numerical simulation is available in the public domain, and this propeller is typical of the kind used in turbo-props. The design parameters were the control points of the NURBS surface. Here we have two cases. In the first case the thickness to chord ratio is kept constant as the chord varies along the span, and in the second case the chord is fixed and the thickness is allowed to vary. In the first case we get the chord distribution by movement of the control points along the chord of any section of the wing i.e in the spanwise direction. The leading edge in all the cases is always fixed, allowing only the trailing edge to vary. In the second case, the chord is kept constant and the thickness is allowed to vary along the span. This is achieved by allowing the control points to move in thickness direction while keeping fixed in other direction. The taper ratio is kept 1 in case 1 and case 2. The cost function used for minimization is

$$J(U,\beta) = K_1 \frac{C_D}{C_{D0}} + K_2(C_L - C_{L0})^2 \tag{5.1}$$

where K_1 and K_2 are constants, and present case we choose the values as $K_1 = 1.0$, $K_2 = 100.0$.

5.1 Case 1 with constant thickness to chord ratio

Keeping the thickness to chord ratio constant in this case we use the same airfoil section over the span and varying only the chord with a fixed span. The wing area was kept constant during optimization. The present work uses the Euler equation as a constraint for the optimization. There were 4 equidistant sections taken along the span with each section having 20 control points along the chord. C^2 continuity of the surface is maintained using NURBS.

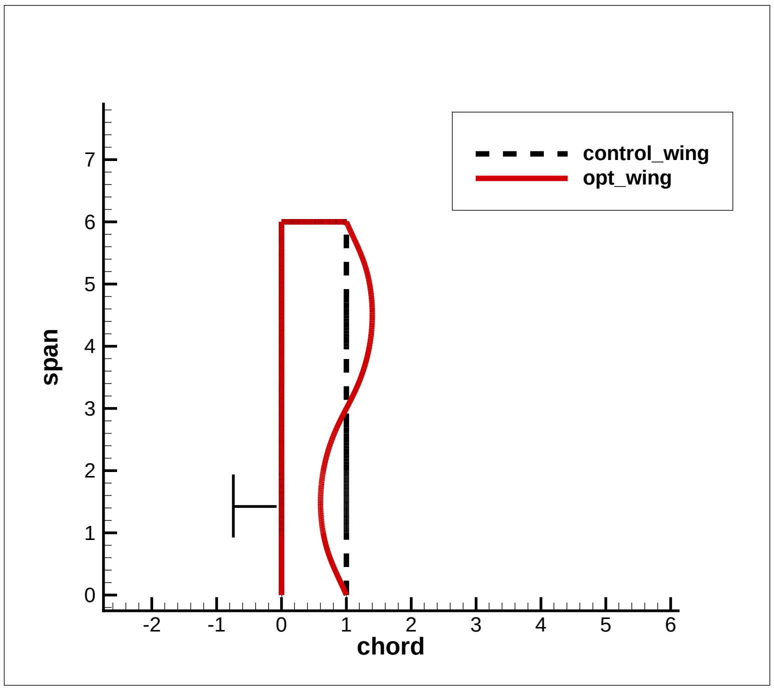


Figure 5.1: Chord distribution comparison with the control wing

We consider zero twist distribution in the wing. Fig.5.1 shows a comparison of the optimal chord distribution with the control wing. Geometric constraints given were span, root and tip chord, all of which were fixed; the planform area is also constant. We get 8.2 counts of reduction in the drag compared with the control wing. The cost function history with number of optimization cycle is shown in fig. 5.3. As the thickness to chord ratio is held constant the thickness varies along the span like the chord. This is shown in fig.5.2. The number of grid cells used was 3×10^5. The drag reduction in present work is lower than the reduction which Rakshith (2013) had obtained. One of the reasons might be that the present work includes compressibility effects while Rakshith (2013) used lifting line theory to calculate drag which doesn't account for compressibility. To demonstrate grid independence it is necessary to use very fine meshes, but this has not been possible here due to high computational costs. The present work includes the development of PROP-OPT described in section 2.2 which can be used for finer grids. However, Rakshith (2013) has observed that, for an optimal wing with propeller, the chord behind the propeller is shorter than on either side. Here a similar conclusion has been reached by using a higher fidelity optimization. This validates PROP-OPT as well as the results of Rakshith (2013).

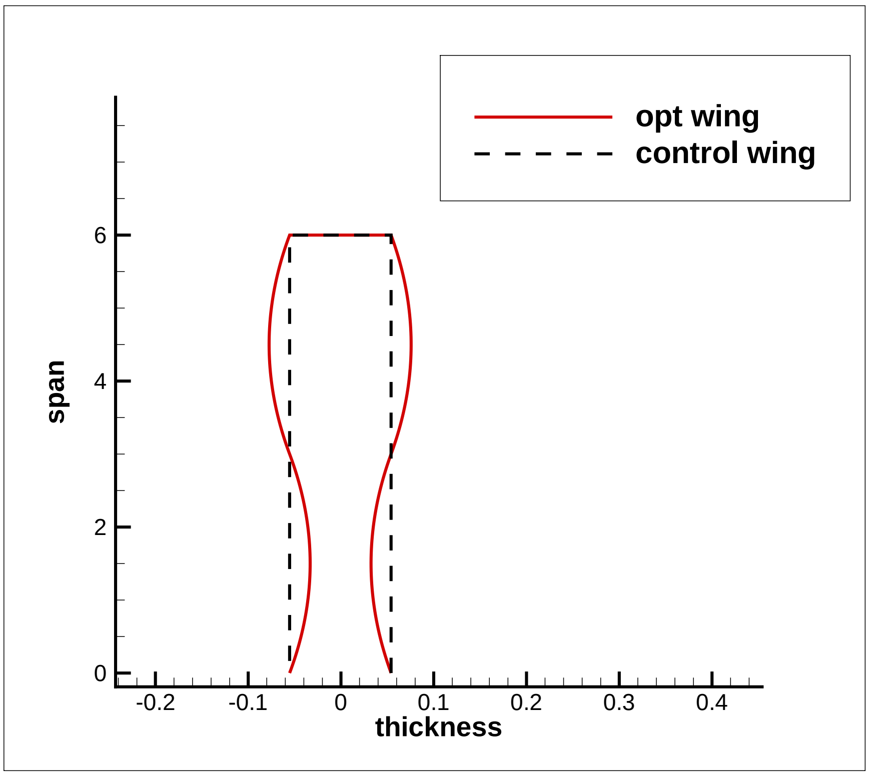


Figure 5.2: Thickness distribution comparison with the control wing, section taken at 0.4% of the root chord

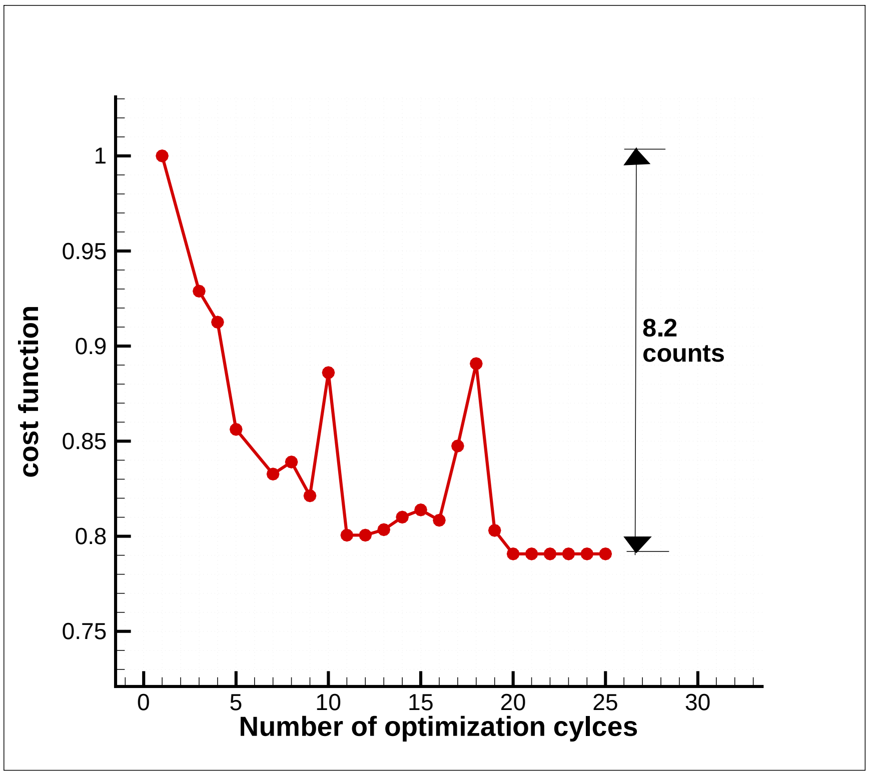


Figure 5.3: Cost function history for chord varying optimization

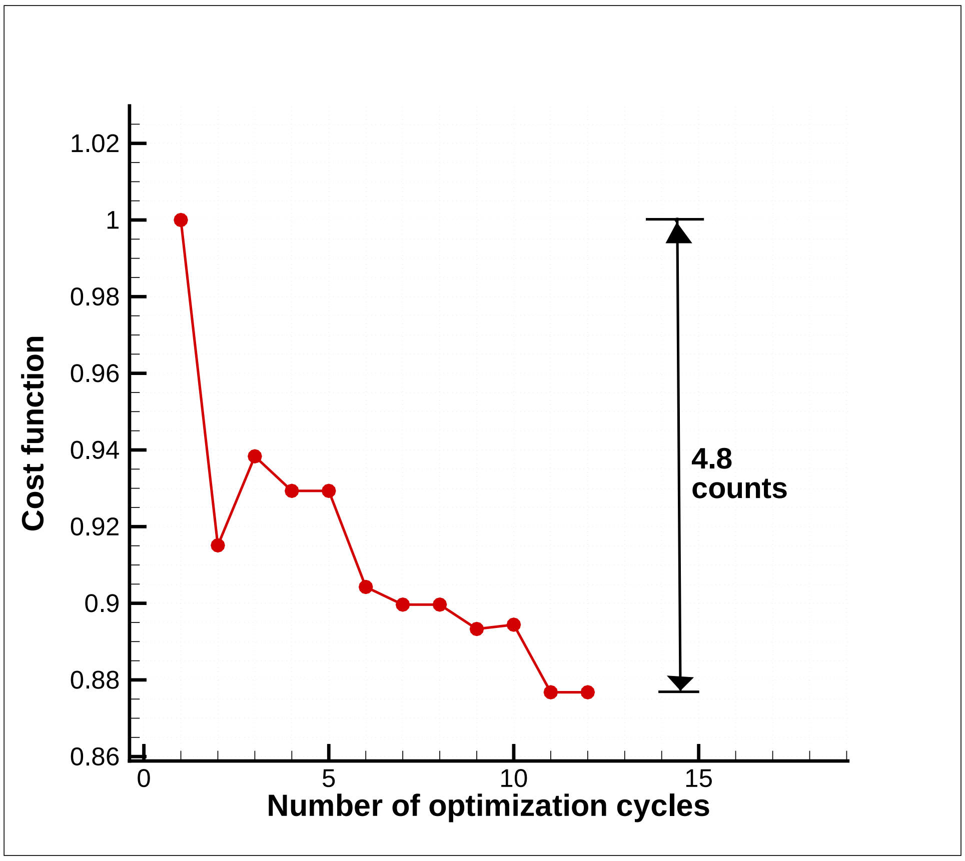
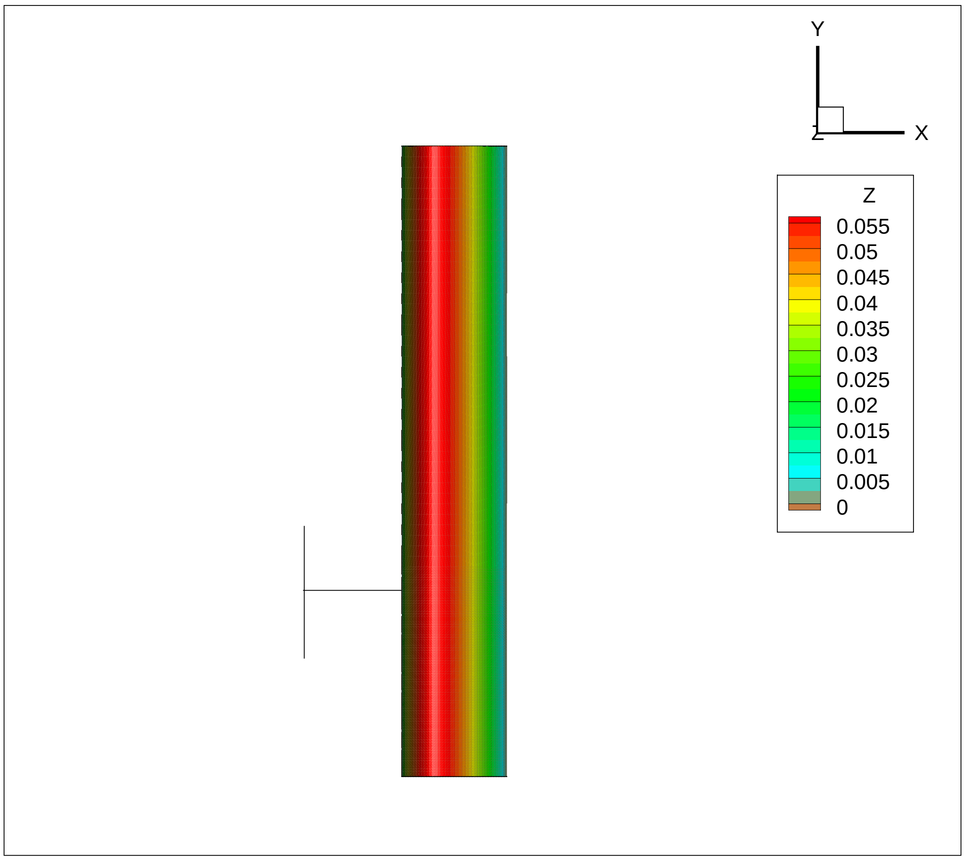


Figure 5.4: Cost function history for thickness varying optimization

5.2 Case 2: with constant chord and variation in thickness

This case considers the chord to be constant and we try to get the optimal thickness distribution over the wing span. Fig.5.5 shows the thickness contours compared with the control wing. The surface area is kept the same as that of the control wing. Fig.5.6 shows the various sections taken along the span of the wing and compares the optimal with the control wing. The planform remains the same as there is no chord variation. As the wing has high aspect ratio, the sensitivity of the drag coefficient to airfoil thickness is comparatively less, hence the drag reduction is expected to be less when compared with case 1. In any case the drag of the optimal wing is 4.8 counts less than the control wing. This is shown in 5.4. The grid size used was 3×10^5. In case 1 we observed that chord behind the propeller is less than at other locations, in the present case we have a similar observation that the maximum thickness of airfoil sections considered along the span was lower behind the propeller and greater outboard.



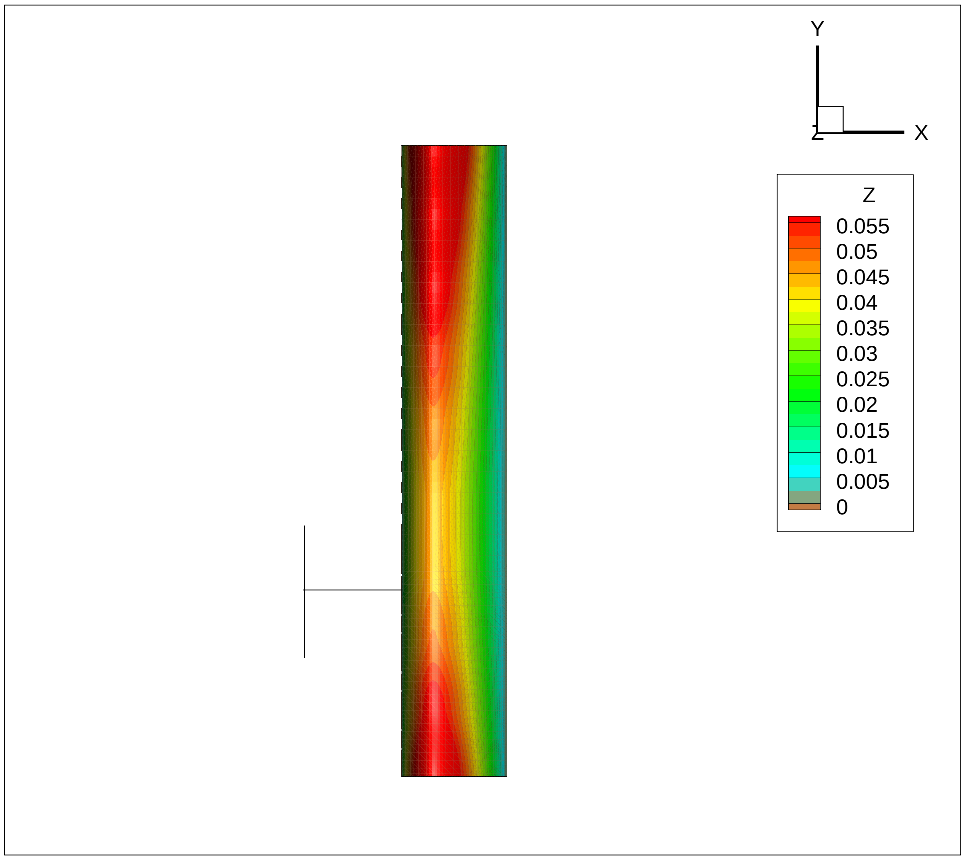


Figure 5.5: thickness contours for control and optimal wing. AR = 12, cl = 0.4, control points = 20 x 4

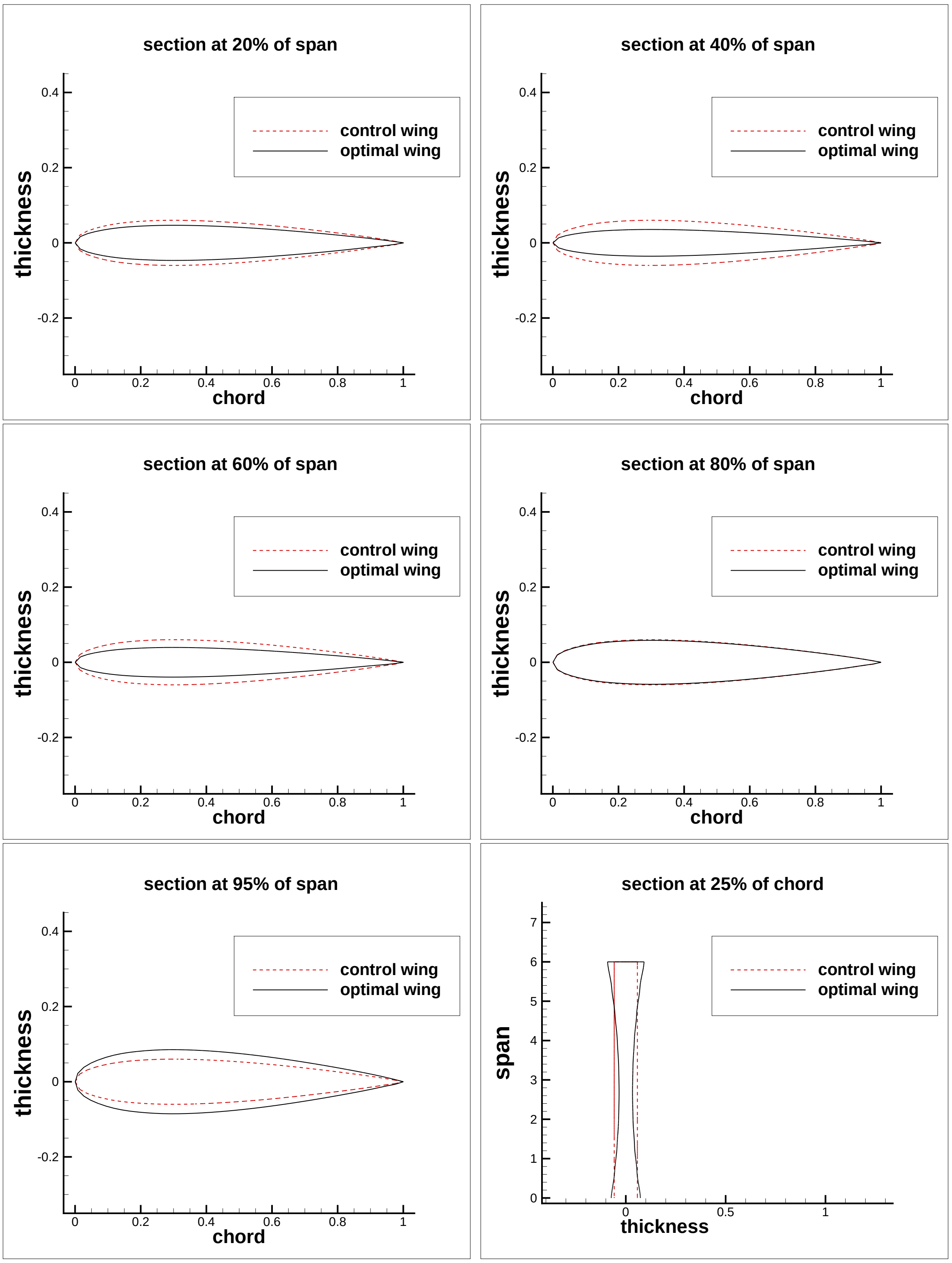


Figure 5.6: Thickness distribution comparison with the control wing, section taken at 0.4% of the root chord

Chapter 6

Optimization of small aspect ratio wings for turbo-prop aircrafts

In this chapter we present results for the aerodynamic shape optimization carried out for small aspect ratio wings of turbo-prop aircraft with tractor configuration. Rakshith *et al.* (2011) found optimal planforms using lifting line theory for wings of aspect ratio of 12. One of the limitations of lifting line theory is that it is not valid for low aspect ratio wings. Here we exploit the Euler code to optimize shapes for lower drag of low aspect ratio wings for turbo-prop aircraft. Most of the current generation turbo-prop aircraft have high aspect ratio wings. In the present work we take the wing shape of current generation turbo-prop aircraft and change its aspect ratio to 6, and use this shape as a control wing for the optimizer. We take a taper ratio of 0.5. The leading edge, span and wing area are held fixed. We also consider the linear twist distribution with a washout of $3.0°$ for the control wing. Hence for the minimization of induced drag the design variable/control variable vector also includes the local angle of twist for each chosen section along the span. The chord, thickness and the span is considered to be along x, y and z direction respectively. To parameterize the shape we use the control points of a NURBS surface. These control points are used to define a design variable/ control variable vector. We take equidistant sections across the span along which the control points of the NURBS are defined. This is shown in fig. 6.2. For each section we take a constant multiplier to the x coordinate of the control points along the section. This multiplier variable when multiplied gives chord variation as the multiplier variable changes for various sections. We also define an angle of twist for each section in order to get the twist distribution of the wing. We use the same multiplier for the y component (thickness direction) of the control points for each section along the span, which ensures the thickness to chord ratio is always constant. These multiplier along with the angle of twist for each section are used as a design variable in the optimization procedure. The optimization is carried out in grid size of 5×10^5. The optimization cycle with fine grid becomes computationally very costly and hence we confirm the results by running a fine grid flow solutions both on control and optimal wing. The propeller was considered the same as in case 1 in chapter 5. The twist distribution over a semi span is given in Fig.6.3.

Fig.6.4 shows the starting control wing in the optimization procedure and also the optimal wing obtained by PROP-OPT. The drag reduction from the optimization pro-

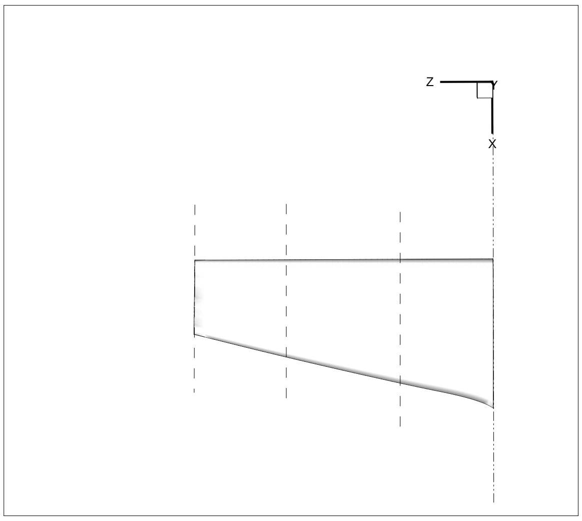


Figure 6.1: Semi-span of wing. The dashed lines mark the sections taken along the span.

cedure was 6 counts. We confirm the control wing and the optimal wing by solving the governing equation for the flow by PROP-EULER++ using 10×10^6 grid size. Although there is still need for further grid independence study. We compared the results obtained by both the grids and found that the reduction in drag is 9 counts, which confirms the optimality of the shape for induced drag reduction. The iteration history of the residual of ρ for flow calculation of the optimal wing is compared to the control wing in fig.6.5. The lift and the drag coefficients of the optimal wing compared with the control wing is also shows in fig.6.6 and fig. 6.7.

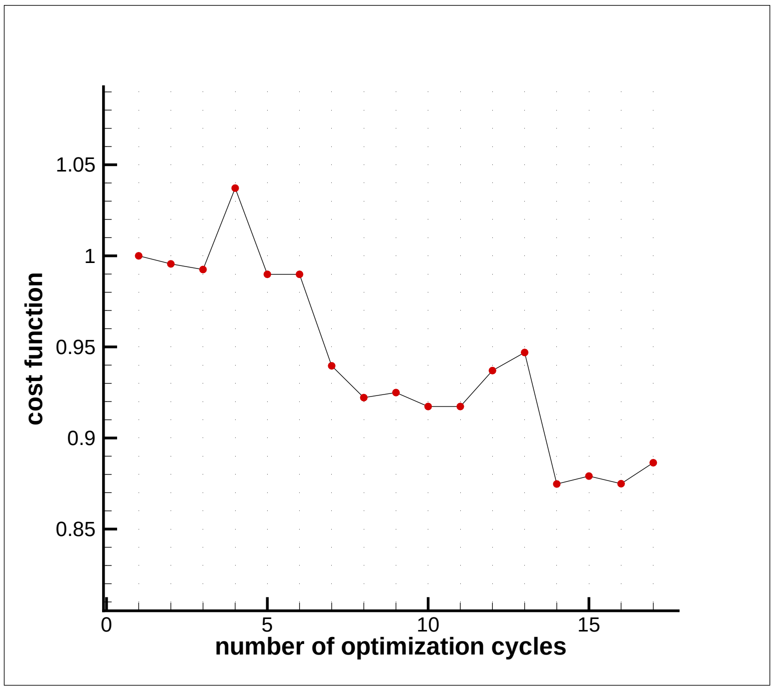


Figure 6.2: Cost function history with number of optimization cycles.

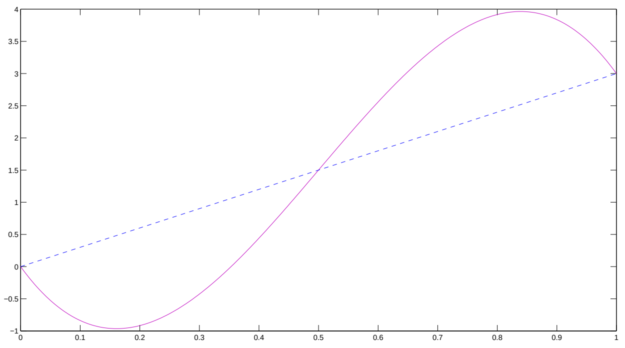
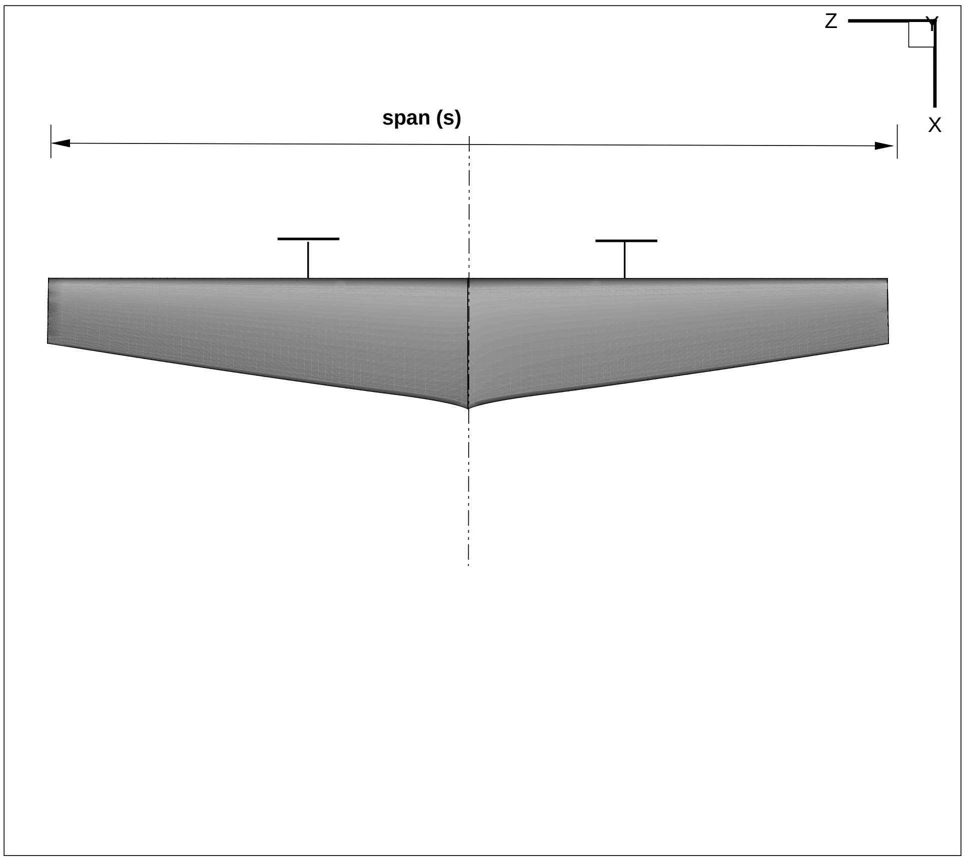


Figure 6.3: Twist distribution over a semi-span. The dashed line shows the linear washout of 3° for the control wing.



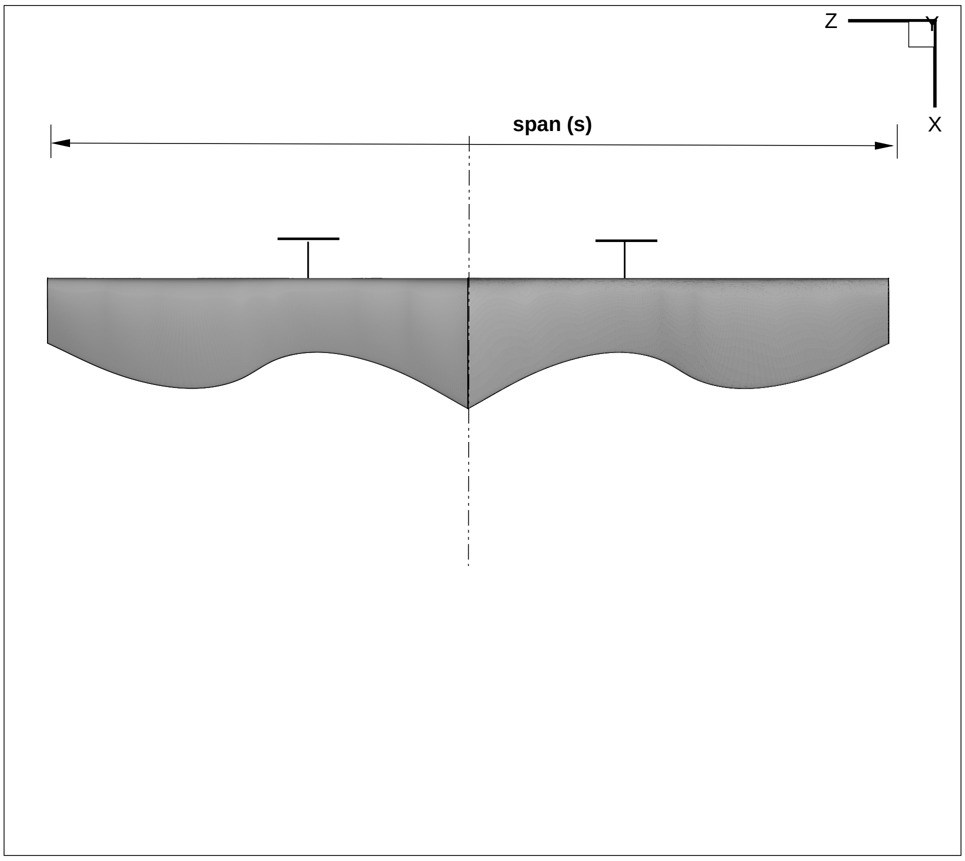


Figure 6.4: Control and optimal shapes obtained by PROP-OPT

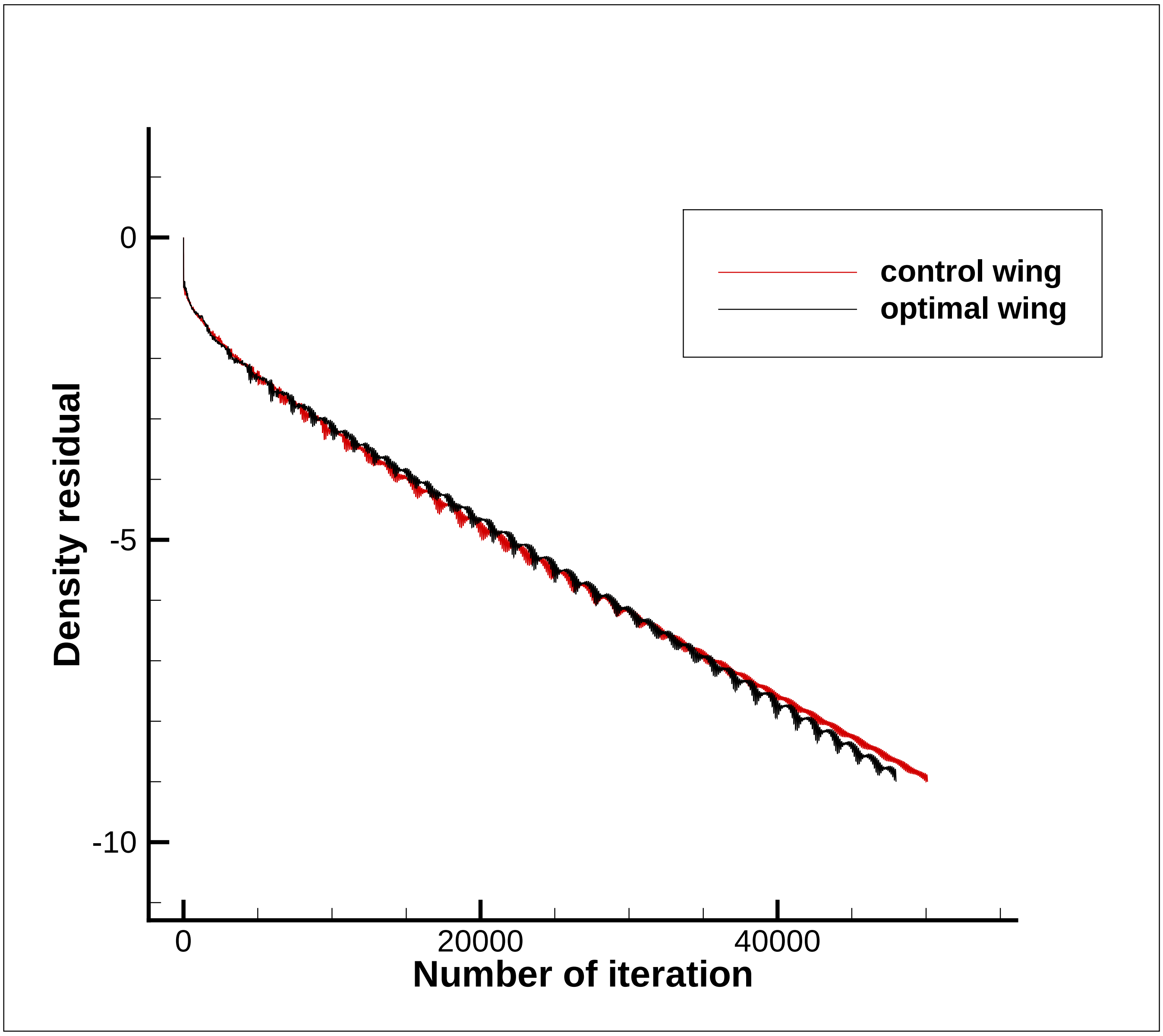


Figure 6.5: Iteration history of the rho residual for 10 million grid size

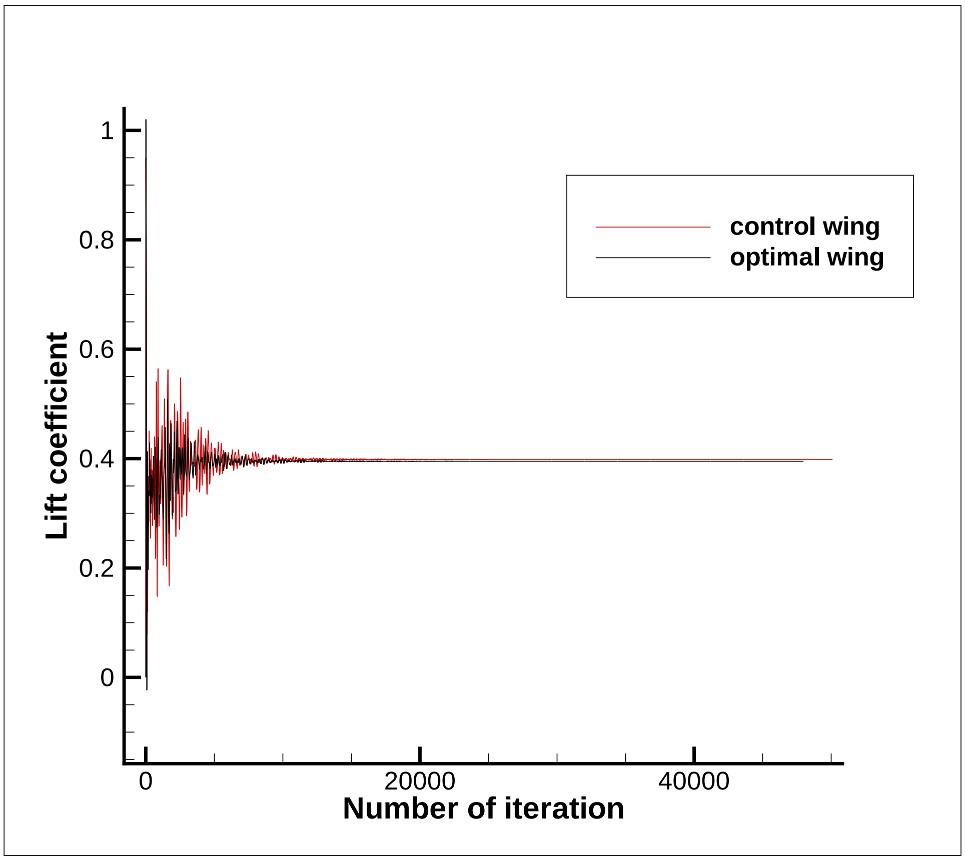


Figure 6.6: Lift coefficient history with iteration.

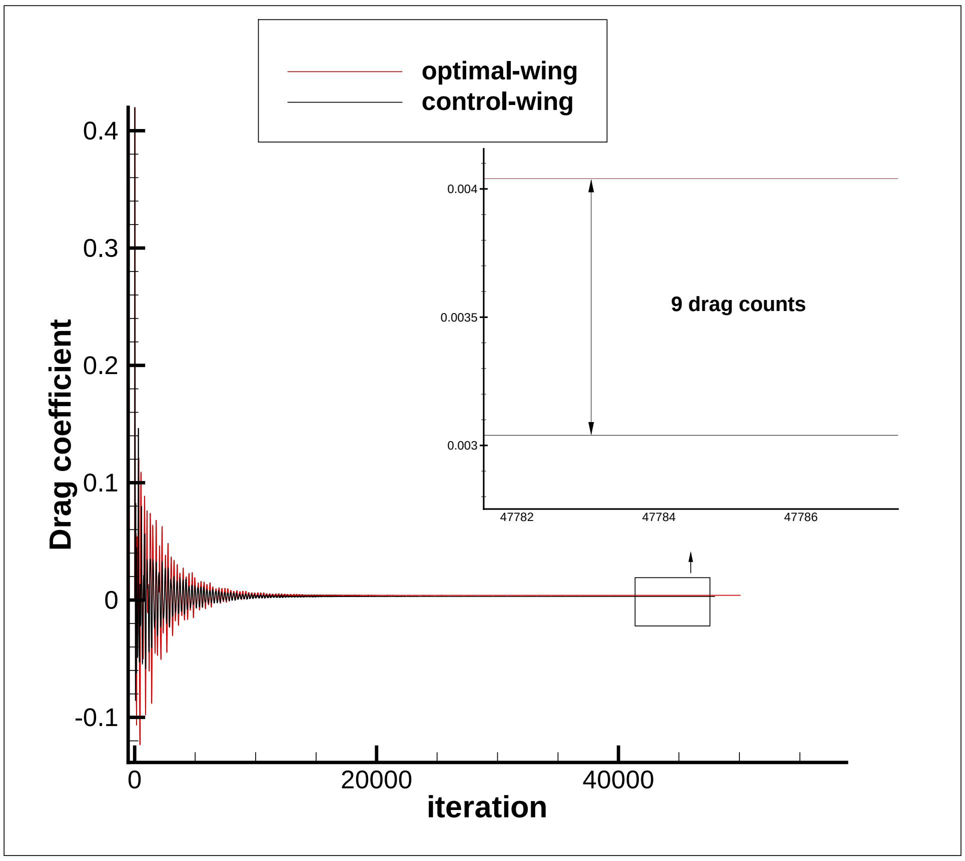


Figure 6.7: Drag coeffient history with iteration for 10 million grid size

Chapter 7
Conclusion

This thesis has focused on the development of a higher fidelity optimization code PROP-OPT for aerodynamic shape optimization of wings for propeller-driven aircraft with tractor configuration. Emphasis was on the adjoint method for the calculation of the gradients and NURBS for the parameterization of the surface. The adjoint solver, an Euler flow solver for the wing-propeller configuration and an optimizer (NLOPT library) were coupled to find an optimal shape of the wing. Herein we summarize results of the present study with some suggestions for future work.

The optimization problem was formulated in chapter 2. An in-house code PROP-OPT was developed coupling the CFD solver and the optimizer. We used a C++ library NLOPT for the various optimization techniques already implemented and available in the open source literature. The numerical details and the algorithmic implementation were reported in Chapter 3, which also includes the implementation of the parameterization of the surface shape which is to be optimized. NURBS was used to have a C^2 continuous smooth surface for whatever number of control points is or may be used. An in-house C++ code was developed for the implementation of NURBS surface fitting. The shapes were deformed using radial basis functions. The algorithm and implementation have been reported in chapter 3.

A C++ code PROP-EULER implemented by Rakshith (2013), which solved the Euler equation for the wing-propeller configuration, was optimized for time with use of SIMD, openmp and MPI domain decomposition implementations. The validation and verification of the optimized code PROP-EULER++ is reported in chapter 4. The validation was done using a standard test case of the transonic flow over ONERA M6 wing at Mach 0.83 and angle of attack 3.06 degrees.

The gradient of the cost function/ objective function was calculated using the adjoint method. An in-house code PROP-ADJ was developed for the purpose. The implementation of the algorithm and the validation are discussed in chapter 4. The validation of PROP-ADJ was done by comparing the total gradients obtained with values from the finite difference method for a toy grid. The derivatives in the PROP-ADJ were calculated using automatic differentiation techniques. This was implemented in C using the open source tool TAPENADE.

The optimization for the lowest induced drag of a wing of given area and span, with a propeller was carried out for two cases, one with chord variation and the other with thickness variation. We found that the chord distribution is similar to that obtained by Rakshith *et al.* (2011). The drag reduction in the present study was 8.2 counts, which is close to that obtained by Rakshith *et al.* (2011). They observe that the optimal wing has shorter chord behind the propeller and longer chords on either side of it, and this was also seen in the present case. Constraints on the geometry were that span, planform area and aspect ratio of 12 were fixed. Optimization was carried out for cruise conditions with lift coefficient held constant at 0.4. Another case in which the chord was kept constant and the thickness allowed to vary was considered and results have been presented in the thesis. In this case the induced drag of the optimal wing was 4.8 counts less than that of the control wing. These results were reported in chapter 5. Optimization of a small aspect ratio wing was also carried out with gird size 10×10^6, and showed a 9 count drag reduction in the new shape. These results were reported in chapter 6. The present work, together with that of Rakshith *et al.* (2011), shows that there is still considerable potential for significant improvement in turbo-prop aerodynamics.

7.0.1 Suggestions for Future study

In general turboprop aircraft have relatively high aspect ratio wings. In most cases, therefore the extended lifting line theory Rakshith *et al.* (2011) should be applicable. however the complete aircraft needs multi-disciplinary optimization, and structural considerations (root bending moments, weight, manufacturing constraints) play a strong role in determining overall aircraft performance. For example, a wing design using a main spar may demand a maximum thickness line along the spar of the wing and a wing thickness that diminishes linearly towards the wing tips to accommodate a spar whose depth varies in a similar fashion. Furthermore , as Rakshith *et al.* (2011) have shown, a planform of the kind considered here implies that, if the wing has constant t/c ratio along the span, the lower thickness in the short-chord region could result in undesirable concavities on the wing surface. If turboprop speeds were to increase (there is currently a noticeable trend in this direction) special airfoil sections and somewhat lower aspect ratios may be worth considerable. For all these reasons a surface optimization technology of the kind considered in this thesis is a useful design tool.

In all these cases the cost function would have to be modified to include other appropriate parameters in the optimization process; in addition to drag. Minimization of total drag will be need to solve the RANS equations, allowing for the turbulent flow in the boundary layer.

Present work concentrates on cruise conditions, so the multi-disciplinary optimization which mentioned earlier in this section could include design of optimized wings for phases of the flights, with suitably weighted cost function.

There is a need to do high performance computing with higher grid resolutions in order to have grid-independent values of all aerodynamic parameters.

Appendix A
Automatic Differentiation

Automatic differentiation(AD) is a set of techniques to numerically evaluate the derivative of a function specified by a computer program. AD exploits the fact that every computer program, no matter how complicated, executes a sequence of elementary arithmetic operations such as additions or elementary functions such as exp(). By applying the chain rule of derivative calculus repeatedly to these operations, derivatives of arbitrary order can be computed automatically, and accurate to working precision. There are several softwares available today which can perform AD like ADIFOR, ADOLC,TAMC, ODYSSEE,TAPENADE, etc. These tools can be found very extensively on http://www.autodiff.org. To explain this, consider the function f, given by

$$f = f_1(f_2(f_3(...(f_n(x))...))) \tag{A.1}$$

The gradient is given by

$$\frac{df}{dx} = \frac{df_1}{dx}\frac{df_2}{dx}\frac{df_2}{dx}....\frac{df_n}{dx} \tag{A.2}$$

The chain rule can be applied from left to right as well as right to left of the eq. A.2, the former one is called as reverse mode and the later as forward mode. To demonstrate further on how the AD tools perform these modes, consider a simple test function f in two variables, x_1 and x_2 as given below

$$f(x_1, x_2) = x_1x_2 + sin(x_1) + e^{x_2} \tag{A.3}$$

This test example is taken from Anil (2008), the AD tools read this function as a sequence of urinary and binary operations, given as,

$$\begin{aligned}
t_1 &= x_1\\
t_2 &= x_2\\
t_3 &= x_1x_2 = t_1t_2\\
t_4 &= sin(x_1) = sin(t_1)\\
t_5 &= e^{x_2} = e^{t_2}\\
t_6 &= t_3 + t_4\\
t_7 &= t_5 + t_6 = f
\end{aligned} \tag{A.4}$$

In forward mode the chain rule is applied from top to bottom of eqs.A.4, after applying the forward mode of AD to the function f, we get

$$\begin{aligned}
\dot{t}_1 &= \dot{x}_1 \\
\dot{t}_2 &= \dot{x}_2 \\
\dot{t}_3 &= t_1\dot{t}_2 + \dot{t}_1 t_2 \\
\dot{t}_4 &= cos(t_1)\dot{t}_1 \\
\dot{t}_5 &= e^{t_2}\dot{t}_4 \\
\dot{t}_6 &= \dot{t}_3 + \dot{t}_4 \\
\dot{t}_7 &= \dot{t}_5 + \dot{t}_6
\end{aligned} \tag{A.5}$$

here calculation of the partial derivatives $\partial f/\partial x_1$ and $\partial f/\partial x_2$ is done in two forward mode sweeps. Depending on the initialization of $\dot{t}_1$ and $\dot{t}_2$ we get either of the derivative, in this case $\dot{t}_1 = 1$ and $\dot{t}_2 = 0$ will calculate $\partial f/\partial x_1$ and vice versa.

In reverse mode the computation is in reverse order. We first define the following notations (Praveen 2006).

$$\bar{t}_k = \frac{\partial f}{\partial t_k} \text{ and } t_{i,k} = \frac{\partial t_i}{\partial t_k} \tag{A.6}$$

Using chain rule of differentiation in reverse mode, t_k can be written as

$$\bar{t}_k = \frac{\partial f}{\partial t_k} = \frac{\partial t_7}{\partial t_k} = \sum_{i \in I_k} \frac{\partial t_7}{\partial t_i}\frac{\partial t_i}{\partial t_k} = \sum_{i \in I_k} \bar{t}_i t_{i,k} \tag{A.7}$$

where index I_k is defined as

$$I_k = \{i : i > k \text{ and } t_i \text{ depends explicitly on } t_k\} \tag{A.8}$$

Reverse mode calculates all the component of the derivatives in one sweep in present case it calculates both the partial derivatives $\partial f/\partial x_1$ and $\partial f/\partial x_2$, however the intermediate derivatives are stored which requires extra memory requirements. In iterative methods only the final converged solution is of interest. If AD in reverse mode is used it cannot distinguish such situation and it will differentiate the whole iterative sequence. This leads to enormous memory requirement. In order to overcome this memory requirement AD must be applied in a piecemeal manner. The solver must be highly modular structure using subroutines and function. Then AD must be applied to those individual modules.

The test equations after applying the reverse mode of AD to the test function f are

$$\begin{aligned}
\bar{t}_7 &= \frac{\partial f}{\partial t_7} = \frac{\partial t_7}{\partial t_7} = 1 \\
\bar{t}_6 &= \frac{\partial t_7}{\partial t_6} = \bar{t}_7 t_{7,6} = 1 \\
\bar{t}_5 &= \frac{\partial t_7}{\partial t_5} = \bar{t}_7 t_{7,5} + \bar{t}_6 t_{6,5} \\
\bar{t}_4 &= \frac{\partial t_7}{\partial t_4} = \bar{t}_7 t_{7,4} + \bar{t}_6 t_{6,4} + \bar{t}_5 t_{5,4} \\
\bar{t}_3 &= \frac{\partial t_7}{\partial t_3} = \bar{t}_7 t_{7,3} + \bar{t}_6 t_{6,3} + \bar{t}_5 t_{5,3} + \bar{t}_4 t_{4,3} \\
\bar{t}_2 &= \frac{\partial t_7}{\partial t_2} = \bar{t}_7 t_{7,2} + \bar{t}_6 t_{6,2} + \bar{t}_5 t_{5,2} + \bar{t}_4 t_{4,2} + \bar{t}_3 t_{3,2} = x_1 + e^{x_2} \\
\bar{t}_1 &= \frac{\partial t_7}{\partial t_1} = \bar{t}_7 t_{7,1} + \bar{t}_6 t_{6,1} + \bar{t}_5 t_{5,1} + \bar{t}_4 t_{4,1} + \bar{t}_3 t_{3,1} + \bar{t}_2 t_{2,1} = x_2 + cos(x_1) = x_1 + e^{x_2}
\end{aligned} \tag{A.9}$$

The present work uses an open source tool called TAPENADE which was developed by Hascet and Pascual . It takes as input a computer program written in c and returns the corresponding derivative program which can be used in the solver to calculate derivatives. The details of running the tool and getting the subroutines can be found in http://tapenade.inria.fr:8080/tapenade.

References

Anil, N. 2008 Optimal control of numerical dissipation in modified kfvs (m-kfvs) using discrete adjoint method. PhD thesis, Dept. of Aerospace Engg., IISc, Bangalore, India.

Becker, G. & Jameson, M. S. A. 2011 An advanced nurbs fitting procedure for post-processing of grid-based shape optimizations. *49th AIAA Aerospace Science Meeting* .

Boor, C. D. 1978 *A Practical Guide to Splines*. Springer-Verlag.

Cox, M. G. 1972 The numerical evaluation of b-splines. *IMA journal of Applied Mathematics* .

Deshpande, S. M. 1986 A second order accurate kinetic theory based method for inviscid compressible flows. *NASA Technical Paper* .

Drela, M. & Giles, M. B. 1987 Viscous-inviscid analysis of transonic and low reynolds number airfoils. *AIAA Journal* **25**, 1347–1355.

Elliot, J. 1998 Aerodynamic optimization based on the euler and navier-stokes equations using unstructured grids. PhD thesis.

GAMBIT 2004 Gambit 2.1 documentation, user's guide. *Fluent Inc. Software* .

Ghosh, A. K., Mathur, J. S. & Deshpande, S. M. 1998 q-kfvs scheme - a new higher order kinetic method for euler equations. *16th International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics* **515**.

Giles, M. B. & Pierce, N. A. 2000 An introduction to the adjoint approach to design. *Technical Report No. NA 00/04, Oxford University Computing Laboratory* .

H. J. Kim, D. Sasaki, S. O. & Nakahashi, I. 2000 Aerodynamic optimization of supersonic transport wing using unstructured adjoint method. *Proceeding of the fist ICCFD,kyoto Japan* .

Hartman, E. & Biermann, P. D. 1938 The aerodynamic characteristics of full-scale propellers having 2, 3, and 4 blades of Clark Y and R.A.F. 6 airfoil sections. *NACA-report-640* .

Hicken, J. E. & Zigg, D. W. 2010 Induced-drag minimization of non planar geometries based on the euler equations. *AIAA Journal, Vol. 48, No. 11* .

Hicks, R. M. & Henne, P. 1978 Wing design by numerical optimization. *Joournal of Aircraft 15:407-412* .

http://tapenade.inria.fr:8080/tapenade . Inria.

http://www.autodiff.org . Automatic differentiation.

http://www.tsplines.com . Bezier curves.

http://ab initio.mit.edu/wiki/index.php/NLopt . Non linear optimization(nlopt).

Jakobsson, S. & Amoignon, O. 2006 Mesh deformation using radial basis functions for the gradient-based aerodynamic shape optimization. *Computers and fluids* .

Jameson, A. 1988 Aerodynamic design via control theory. *Journal of Scientific Computing* .

Jameson, A. 1990 Automatic design of transonic airfoils to reduce the shock induced pressure drag. *Proceedings of the 31st Israel Annual Conference on Aviation and Aeronautics* .

Jameson, A. & Yoon, S. 1986 Lower upper implicit schemes with multiple grids for the euler equations. *AIAA* .

Kraft, D. 1988 A software package for sequential quadratic programming. *Technical Report DFVLR-FB 88-28, Institut fÃ$\frac{1}{4}$r Dynamik der Flugsysteme, Oberpfaffenhofen* .

Kroo, I. 1986 Propeller-wing integration for minimum induced loss. *J. Aircraft* **23**, 561–565.

Kroo, I. 2001 Dragdue to lift:concepts for prediction and reduction. *Annu. Rev. Fluid Mech. 2001. 33:587â617* .

Lighthill, M. J. 1945 A new method of two-dimensional aerodynamic design. *Aeronautical Research council's Reports and Memoranda, Numeber 2112* .

Lötstedt, P. 1995 Accuracy of a propeller model in inviscid flow. *J. Aircraft.* **32**, 1312–1321.

McFadden, G. B. 1979 An artificial viscosity method for design of supercritical wings. PhD thesis, PHD dissertation NewYork, University.

MOHAMMADI, B. & PIRONNEAU, O. 1999 Mesh adaptation and automatic diffrentiation in a cad-free framework for optimal shape design. *Interational Journal of Numerical Methods Fluids, 30(2):127-136* .

NADARAJAH, S. & JAMESON, A. 2000 A comparison of the continuous and discreate adjoint approachto automatic aerodynamic optimization. *AIAA papare 2000-0667* .

NEILSON, J. E. 1998 Aerodynamic design sensitivities on an unstructured mesh using navier-stokes equations and a discreate adjoint formulation. PhD thesis.

PRANDTL, L. 1921 Mutual influence of wings and propeller. *NACA TN No. 74.* .

PRAVEEN, C. 2006 Adjoint code development and optimization usinf automatic diffrentiation. *National Aerospace Laboratories, India* .

RAJAGOPALAN, R. G. 1989 A numerical study of a wing and propeller in mutual interference. *AIAA* pp. 464–471.

RAKSHITH, B. R. 2013 Novel wing designs for tractor-propeller aircraft: Theory, cfd and wind tunnel test results. PhD thesis, Engineering Mechanics Unit, JNCASR, Bangalore, India.

RAKSHITH, B. R., DESHPANDE, S. M., PRAVEEN, C. & NARASIMHA, R. 2011 Assessment of lifting-line theory optimization of turboprop wings in tractor configuration using euler code. *Proc. Symp. Applied Aerodynamics and Design of Aerospace Vehicles, pp 564* .

RAO, S. S. 1996 *Engineering Optimization, Theory and Practice*. New Age International (P) Limited, Publishers.

SHENE, C.-K. 2008 B-cpline, computing coefficient.

VELDHUIS, L. L. M. 2005 Propeller wing aerodynamic interference. PhD thesis, Delft University.

VELDHUIS, L. L. M. & HEYMA, P. M. 2000 Aerodynamic optimisation of wings in multi-engined tractor propeller arrangements. *Aircraft Design* pp. 129–149.