# On the Development of an Improved Lift-Constrained Aerodynamic Optimization Algorithm

by

Laura K. Billing

A thesis submitted in conformity with the requirements
for the degree of Masters of Applied Science
Graduate Department of Aerospace Engineering
University of Toronto

## Abstract

# On the Development of an Improved Lift-Constrained Aerodynamic Optimization Algorithm

Laura K. Billing

Masters of Applied Science

Graduate Department of Aerospace Engineering

University of Toronto

2006

An improved algorithm is presented for performing lift-constrained aerodynamic optimization. The angle of attack is set in the flow solution stage, ensuring the desired lift coefficient is achieved. Changes to the flow solver are carried through to the adjoint calculation of the gradient. For this project, the drag coefficient is the objective function, but the algorithm could easily be extended to other lift-constrained optimization problems. Work on normalizing objective functions for multi-point optimization ensures that the weight given to each design point is independent of the initial objective function value, and that the thickness constraints have the same meaning for all design points.

Several design cases are examined. One case is of particular interest, a multipoint optimization with eighteen design points, which is compared to optimized airfoils for each individual design point, to examine the benefits of morphing airfoils for a realistic design case.

# Acknowledgements

I could not possibly have produced this thesis without the invaluable assistance of my supervisor, Dr. D. W. Zingg. His teaching and guidance have been indispensable to me, and I have appreciated all his insightful suggestions.

I owe a debt of thanks to all the students and staff at UTIAS, especially the students in the CFD lab. I am particularly grateful to Jon Driver for his help becoming familiar with Optima, and James McDonald, Scott Northrup, and Chad Oldfield who all kindly provided valuable help with the programming aspects of my thesis.

I am immensely grateful to my family. They have been extremely supportive of my scholastic endeavours, and I am indebted to them for all the support they have given to me over the years.

Finally, I would like to gratefully acknowledge financial support from the Natural Sciences and Engineering Research Council of Canada, and the University of Toronto.

Thank-you, all.

Laura Billing

University of Toronto Institute for Aerospace Studies
August 8, 2006

# Contents

**Abstract** iii

**Acknowledgements** v

**Contents** vii

**List of Figures** ix

**1 Introduction** 1

1.1 Motivation . . . . . 1

1.2 Existing Methods . . . . . 2

**2 Governing Equations** 9

2.1 Optimization Equations . . . . . 9

2.1.1 Problem Formulation . . . . . 9

2.1.2 Design Variables . . . . . 10

2.1.3 Objective Function . . . . . 11

2.1.4 Constraints . . . . . 11

2.2 Flow Equations . . . . . 12

2.2.1 Navier-Stokes Equations . . . . . 12

2.2.2 Turbulence Model . . . . . 14

2.2.3 Thin-Layer Approximation and Coordinate Transformation . . . . 15

**3 Algorithm** 19

3.1 Flow Solver - Approximate Factorization . . . . . 19

3.2 Flow Solver - Newton-Krylov Algorithm . . . . . 23

3.3 Optimizer Equations . . . . . 25

3.4 Other Modifications . . . . . 28

**4 Results** **31**

4.1 Flow Solver . . . 31

4.2 Gradient Calculations . . . 33

4.3 Optimization Cases . . . 33

4.3.1 Single-Point Subsonic Airfoil Design . . . 34

4.3.2 Single-Point Transonic Airfoil Design . . . 37

4.3.3 Four-Point Optimization Case . . . 39

4.3.4 Two-Point Pareto Front . . . 42

4.3.5 Eighteen-Point Optimization Case . . . 46

**5 Conclusions and Recommendations** **53**

**References** **57**

**A New Analytical Derivatives** **61**

A.1 Derivative of Coefficients with Respect to Angle of Attack . . . 61

A.2 Derivatives of Coefficients with Respect to Flow Variables . . . 62

A.2.1 Force Due to Pressure . . . 62

A.2.2 Force Due to Friction . . . 63

A.3 Derivatives of Existing Residual Equations . . . 66

A.3.1 Far-Field Boundary Residual . . . 66

A.3.2 Outflow Boundary Residual . . . 68

**B Input Files for Optimizations** **71**

B.1 Subsonic Single-Point Optimization Input File . . . 71

B.2 Transonic Single-Point Optimization Input File . . . 73

B.3 Four-Point Optimization Input Files . . . 74

B.4 Two-Point Pareto Front Input Files . . . 77

**C Input Files for Eighteen-Point Optimization** **79**

# List of Figures

2.1 NACA0012 airfoil with B-spline curve control points . . . . . . . . . . . . 10

3.1 Effect of number of iterations to beginning of angle of attack relaxation on convergence rate of approximate factorization flow-solver . . . . . . . 21

3.2 Effect of relaxation parameter on convergence rate of approximate factorization flow-solver . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

3.3 Effect of frequency of angle of attack relaxation on convergence rate of approximate factorization flow-solver . . . . . . . . . . . . . . . . . . . . 22

4.1 Performance of the new flow solver . . . . . . . . . . . . . . . . . . . . . 32

4.2 Subsonic single-point optimization, with second-difference dissipation . . 35

4.3 Subsonic single-point optimization, without second-difference dissipation 36

4.4 Transonic single-point optimization . . . . . . . . . . . . . . . . . . . . 38

4.5 Control points and design variables (shaded) for the RAE 2822 airfoil . . 39

4.6 Objective function and gradient convergence histories for single-point design 40

4.7 Single-point optimization . . . . . . . . . . . . . . . . . . . . . . . . . . 40

4.8 Objective function and gradient convergence histories for two-point design 41

4.9 Two-point optimization . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

4.10 Four-point optimization . . . . . . . . . . . . . . . . . . . . . . . . . . 43

4.11 Pareto front for two-point optimization . . . . . . . . . . . . . . . . . . 45

4.12 Control points and design variables (shaded) for the NACA 0015 airfoil . 46

4.13 Airfoil sections found using floating maximum thickness constraint . . . . 49

4.14 Expanded view of airfoil sections found using floating maximum thickness constraint . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

4.15 Airfoil sections found using fixed maximum thickness constraint . . . . . 50

4.16 Expanded view of airfoil sections found using fixed maximum thickness constraint . . . . . . . . . . 50

A.1 Boundary locations . . . . . . . . . . 67

# List of Tables

4.1 Comparison of single design variable optimization to new flow solver . . . 33
4.2 Comparison of gradients calculated with different methods . . . . . . . . 34
4.3 Comparison of lift and drag coefficients found with the new algorithm and the old algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
4.4 Comparison of time to convergence with the new algorithm and the old algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
4.5 Thickness constraints for transonic single-point optimization . . . . . . . 37
4.6 Comparison of results for transonic single-point optimization . . . . . . . 38
4.7 Thickness constraints for Pareto front case . . . . . . . . . . . . . . . . . 44
4.8 Drag coefficients for different weights . . . . . . . . . . . . . . . . . . . . 44
4.9 Optimization points . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
4.10 Design performance of the multipoint airfoil . . . . . . . . . . . . . . . . 51
4.11 Design performance of the multipoint airfoil . . . . . . . . . . . . . . . . 52
4.12 Comparison of drag coefficients of the single-point designs with those of the baseline multipoint design . . . . . . . . . . . . . . . . . . . . . . . . 52

# Chapter 1

# Introduction

## 1.1 Motivation

Historically, aircraft were developed using the trial-and-error method of design. A new design would be developed, then tested either in a wind tunnel or in flight [18]. This was an expensive design process, as a model had to be built for each design possibility, and this did not encourage the development of radically different designs. Instead, existing designs were carefully tuned.

Computational fluid dynamics began being used for aerodynamic analysis and design starting in the mid-1960s [13]. However, limitations in computational power and the need for algorithm development meant that computational aerodynamics was not a practical design tool until significantly later. As computational power increased and optimization algorithms were improved, the possibility of performing numerical optimization became more realistic [14]. Numerical optimization represents a powerful tool that can be used in aircraft design. There are four broad categories of optimization algorithms that can be used: direct search methods, stochastic methods, gradient-based methods, and fully-coupled methods. This project presents changes to an existing gradient-based algorithm. Gradient-based algorithms use the gradient of the objective function with respect to the design variables to move towards the optimal design. They can be efficient, since relatively few iterations are required to achieve significant improvements in the objective function. Gradient-based optimizers are generally less successful in noisy design spaces, and converge to a local optimum. Also, they have the disadvantage that, unless expensive finite-difference gradient calculations are made, modifications to the flow solver must be extended into the gradient calculation [29].

Most early cases of numerical optimization investigated inverse design problems [9, 11]. Inverse design requires the user to specify a target pressure distribution for the surface of the airfoil, and then the optimizer will change the shape of the airfoil to achieve this goal. Unfortunately, the selection of an appropriate target pressure distribution requires significant experience and knowledge on the part of the user. Other possible goals to be considered when conducting airfoil optimization include endurance factor maximization, lift-to-drag ratio maximization, drag-constrained lift maximization, and lift-constrained drag minimization. These all require less experience on the part of the user, and are practical optimization problems. Lift-constrained drag minimization is quite common in airfoil design, as the lift required for an aircraft is often known (from the estimated weight of the aircraft) and the designers wish to minimize drag to reduce the operating cost of the aircraft. There are several methods that currently exist for performing lift-constrained drag minimization. The objective of this project is to modify an existing code to allow it to perform lift-constrained drag minimization with fewer user-generated parameters.

## 1.2 Existing Methods for Lift-Constrained Aerodynamic Optimization

The code being modified for this project is an airfoil optimization program called Optima2D. This program uses a Newton-Krylov method for two-dimensional shape optimization. The design variables in Optima2D for lift-constrained drag minimization are the shape of the airfoil, which is described using a B-spline, and its angle of attack. Using the quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, an objective function is minimized. When lift-constrained drag minimization is performed in Optima2D, the following objective function is used.

$$\mathcal{J} = \begin{cases} \omega_L \left(1 - \frac{C_l}{C_l^*}\right)^2 + \omega_D \left(1 - \frac{C_d}{C_d^*}\right)^2 & \text{if } C_d > C_d^* \\ \omega_L \left(1 - \frac{C_l}{C_l^*}\right)^2 & \text{otherwise} \end{cases} \tag{1.1}$$

where $C_l^*$ is the target lift coefficient, $C_d^*$ is the target drag coefficient, $\omega_L$ is the weight for the lift coefficient optimization, and $\omega_D$ is the weight for the drag coefficient optimization [20]. $C_d^*$ is often selected low enough to be considered unattainable.

This objective function consists of two competing objectives, namely attaining the target lift and minimizing drag. Achieving both simultaneously is difficult, since a decrease in drag typically corresponds to a decrease in lift [22]. As the lift is a constraint on the design, it is important to ensure that the desired coefficient of lift is reached. To do this, either the weight on the lift coefficient objective must be very high (which will reduce the amount by which drag is decreased), or the target lift entered into the optimization program must be a carefully selected amount greater than the actual target lift. To determine how much greater the target lift coefficient given to the program should be than the actual desired lift coefficient requires a great deal of experience. Some level of experience is also required to successfully select a value for $C_d^*$, and selecting appropriate parameters can take several trials. Achieving lift-constrained drag minimization with this method can therefore be difficult and time consuming.

Variations on the use of a penalty for the lift constraint have been implemented by several other researchers. Nielsen and Anderson [24] employ a method similar to the one described above. They use an objective function defined as

$$\mathcal{J} = \omega_1(C_l - C_l^*)^2 + \omega_2(C_d - C_d^*)^2 \tag{1.2}$$

where $C_l^*$ and $C_d^*$ are the target lift and drag once again. The weights $\omega_1$ and $\omega_2$ are initially chosen based on the lift and drag coefficients.

$$\frac{\omega_2}{\omega_1} = \frac{C_l}{C_d} \tag{1.3}$$

However, as in Optima2D, as the optimization process continues it becomes necessary to adjust the weights to ensure that lift is in fact constrained, so this method has flaws very similar to those of the algorithm being modified, as it also uses objectives for lift and drag that are competing [25]. This objective function is combined with a gradient-based optimizer using the Davidon-Fletcher-Powell quasi-Newton method [24].

Previous to Nielsen's work, the weights had been chosen based on the relative importance (as determined by the user) of maintaining lift as compared to minimizing drag [2].

$$\mathcal{J} = \frac{1}{2}(C_l - C_l^*)^2 + 10 \times \frac{1}{2}(C_d - C_d^*)^2 \tag{1.4}$$

This results in a lift coefficient that is close to, rather than equal to, the target lift coefficient. Anderson and Bonhaus [1] used a similar approach. Their objective function was

$$\mathcal{J} = \omega_1(C_l - C_l^*)^2 + \omega_2 C_d^2 \tag{1.5}$$

This eliminates the need for a target drag coefficient to be selected, but still requires some basis for the assignment of weights. For the case studied, $\omega_1 = 1$ and $\omega_2 = 25$, with the weight on the drag set higher so that each term would have an approximately equal contribution to the objective function. Tse and Chan [33] use a similar objective function, but without squaring the $C_d$ term.

Mohammadi [17] also uses a penalty method for lift-constrained drag minimization. In this method however, the penalty to the objective function is:

$$\mathcal{J}_{penalty} = |C_l - C_l^0| \tag{1.6}$$

where $C_l$ is the actual lift coefficient and $C_l^0$ is the initial lift coefficient. It is worth noting that this assumes that the initial lift coefficient is the target lift coefficient, but it would likely be a trivial matter to use the same penalty method with a fixed target lift coefficient replacing the initial lift coefficient in the given equation.

Dadone and Grossman [3] use a similar method to perform drag-constrained lift maximization, using a penalty for drag in the objective function to ensure the drag constraint is met.

$$\mathcal{J} = -C_l + R_1\delta_1 \left(\frac{C_d}{C_{d,max}}\right)^2 \tag{1.7}$$

where $C_l$ and $C_d$ are the current lift and drag coefficients, respectively, $R_1$ is the penalty constant, $C_{d,max}$ is the maximum allowable drag coefficient, and $\delta_1$ is 0 when $C_d \leq C_{d,max}$ and 1 otherwise. If desired, the approach developed in this project could be extended to drag-constrained lift maximization.

Kim et al. [16] proposed a different approach to including lift as a penalty. This can be developed by describing the target drag, $C_d^*$, by the equation

$$C_d^* = C_d + \frac{\partial C_d}{\partial \alpha}\Delta\alpha \tag{1.8}$$

where $C_d$ is the current drag, and $\Delta\alpha$ is the change in angle of attack required to reach the desired lift coefficient, $C_l^*$. The target lift coefficient can be similarly described as

$$C_l^* = C_l + \frac{\partial C_l}{\partial \alpha}\Delta\alpha \tag{1.9}$$

where $C_l$ is the current lift coefficient. Equations 1.8 and 1.9 can be combined to give the equation for the new objective function.

$$\mathcal{J} = C_d - \frac{\left(\frac{\partial C_d}{\partial \alpha}\right)}{\left(\frac{\partial C_l}{\partial \alpha}\right)}(C_l - C_l^*) \tag{1.10}$$

The first part of this equation is the true objective, namely minimizing drag, and the second part is a penalty given to attain the desired lift coefficient. Kim et al. [15] combine this objective function with a flow solver that gradually changes angle of attack to ensure the lift coefficient satisfies the following equation:

$$C_l^* \leq C_l \leq 1.003 C_l^* \tag{1.11}$$

The lift constraint may also be included in a Lagrangian function, and then the goal of the optimization is to minimize the Lagrangian function. This approach has been taken by Drela [4] and Zhang and Lum [34].

Soemarwoto and Labrujère [32] include the lift constraint in the gradient calculation. They take the fact that the lift coefficient should not change as you move through the design space to give

$$\left(\frac{dC_l}{d\alpha}\right)\delta\alpha + \left(\frac{dC_l}{dX}\right)\cdot\delta X = 0 \tag{1.12}$$

Then, setting the angle of attack, $\alpha$, as a dependent variable, the change in angle of attack can be calculated as

$$\delta\alpha = -\left(\frac{dC_l}{d\alpha}\right)^{-1}\left(\frac{dC_l}{dX}\right)\cdot\delta X \tag{1.13}$$

This can then be included in the calculation of the gradient of the objective function with respect to the geometric design variables with fixed lift to give

$$\left(\frac{d\mathcal{J}}{dX}\right)_{C_l} = \frac{d\mathcal{J}}{dX} - \left(\frac{d\mathcal{J}}{d\alpha}\right)\left(\frac{dC_l}{d\alpha}\right)^{-1}\left(\frac{dC_l}{dX}\right) \tag{1.14}$$

This ensures that the desired lift coefficient is reached.

Elliott and Peraire [8] discuss a variety of different possible approaches to implementing the lift constraint. They propose that the constraints could be included in the Lagrangian describing the optimization problem. The drawback of this approach is the complexity of implementation. They also discuss the possibility of finding the angle of attack for the desired coefficient of lift in the flow solver (the method being explored in this project). This option is attractive, but was rejected due to the difficulty of adding the lift constraint to the flow solver. The constraints may also be implemented using the quadratic penalty method. They mention that this has the advantage of simplicity of implementation, as it is simply a factor added to the objective function, but has the disadvantage that ill-conditioning is inevitable when the scalar penalty parameter is large.

Penalty terms may also be added to the Lagrangian, which eliminates ill-conditioning for large scalar penalty parameters. For more details on methods for implementing constraints, see [10].

The final option discussed is implementing the constraints in the optimization stage, rather than in the gradient calculation [8]. This is done by partially solving a sequence of linearly constrained subproblems [7]. The constraints are modelled as being linear for these subproblems as follows

$$\tilde{c}_k = c_k + A_k X \tag{1.15}$$

where $c_k$ is the constraint, $A_k$ is a matrix which has the gradients of the constraint as its rows, and $X$ is the vector of the design variables. The matrix $A_k$ is used in the calculation of the search direction in the constrained line search, for the steepest descent method. Additionally, a compensation for the nonlinearity of the constraint must be included when calculating the next set of design variables. When using the quasi-Newton BFGS method, the calculation of the Hessian approximation is also affected by the change. Adding the target lift as a constraint in this manner increases the complexity of implementation, although it does eliminate the need for the various user-selected parameters discussed above.

Jameson [14] sets the coefficient of lift as an input to the flow solver to ensure that the lift constraint is consistently satisfied. Jameson's flow solution algorithm uses an explicit multi-stage time-marching method with multigrid [12]. Because it is explicit, a small time step is required, so it is possible to use angle of attack relaxation in all parts of the flow solver. Angle of attack relaxation involves adjusting the angle of attack during the flow solution based on the difference between the current coefficient of lift and the desired coefficient of lift.

Keeping the coefficient of lift constant using angle of attack relaxation also affects the gradient calculation, as described by Reuther et al. [30]. For Jameson's approach, the objective function is set to the coefficient of drag and the gradient is calculated with the continuous adjoint equation. To include the fixed lift coefficient in the gradient calculation, we consider the fact that a change in the shape of the airfoil will cause a change in the objective function.

$$\delta \mathcal{J} = \tilde{\delta} C_d + \frac{\partial C_d}{\partial \alpha} \delta \alpha \tag{1.16}$$

The first term, $\tilde{\delta} C_d$, is the change in the coefficient of drag that occurs due to the change in shape with the angle of attack held constant. The change in angle of attack in the second term, $\delta\alpha$, occurs because the flow solver modifies the angle of attack to ensure that lift is consistently achieved. The change in angle of attack can be found based on the fact that we wish for the coefficient of lift to be held constant, namely

$$\delta C_l = 0 \tag{1.17}$$

$$\tilde{\delta} C_l + \frac{\partial C_l}{\partial \alpha}\delta\alpha = 0 \tag{1.18}$$

where $\tilde{\delta} C_l$ is the change in the coefficient of lift that occurs due to the change in shape with the angle of attack held constant and $\delta\alpha$ is the same change in angle of attack as used in Equation 1.16. Equations 1.16 and 1.18 can then be combined to give

$$\delta\mathcal{J} = \tilde{\delta} C_d - \frac{\left(\frac{\partial C_d}{\partial \alpha}\right)}{\left(\frac{\partial C_l}{\partial \alpha}\right)}\tilde{\delta} C_l \tag{1.19}$$

This gradient accounts for changes in drag due to shape change and due to the change in angle of attack required to reach the desired coefficient of lift with the new shape, and is calculated using the continuous adjoint method.

The goal of this project is to implement the lift constraint in the flow solution algorithm of Optima2D, and to modify the objective function gradient calculation accordingly. The flow solver in Optima2D uses implicit Euler time-marching with approximate factorization in the first stage of convergence [26], where angle of attack relaxation can be used, and Newton's method to drive the residual to zero in the second stage [19]. Because the second stage is fully implicit, an equation involving the desired coefficient of lift must be included in the residual calculations. Gradient calculation is performed using the discrete adjoint method, and a method of including the effect of the fixed lift coefficient on the gradient is developed.

# Chapter 2

# Governing Equations

Optima2D is the optimization program that will be modified for this project. In Optima2D, objective function minimization is achieved using the quasi-Newton BFGS optimization method with a backtracking line search, where gradients are calculated using the adjoint method [20]. The calculation of the gradient has been modified, as the modifications to the flow solver change the residual calculations. There are two parts to the flow solver that was modified for this project. The first part is the CYCLONE program, which uses an approximate factorization method to solve the Navier-Stokes equations. The second part is the PROBE program, which uses a Newton-Krylov method to solve the equations. Both use the Spalart-Allmaras turbulence model [19].

## 2.1 Optimization Equations

### 2.1.1 Problem Formulation

The optimization problem can be described as the minimization of the objective function $\mathcal{J}$ with respect to the vector of design variables $X$ subject to constraints $C_j$. Mathematically, this can be written as

$$\min_{X} \mathcal{J}(Q, X) \tag{2.1}$$

where the following constraints are satisfied:

$$C_j(Q, X) \leq 0, \quad j = 1, 2 \ldots N_c \tag{2.2}$$

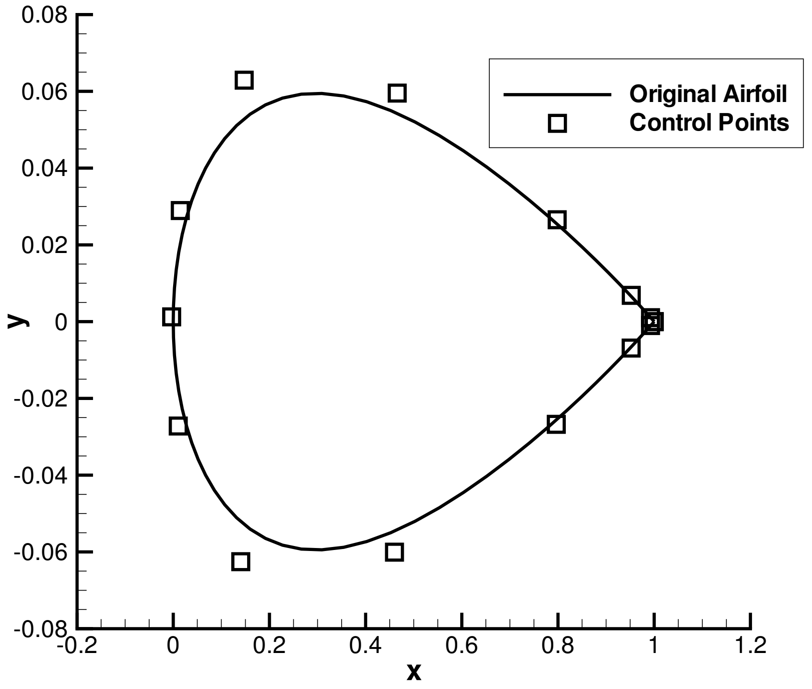


Figure 2.1: NACA0012 airfoil with B-spline curve control points

In these equations, $Q$ is the vector of flow variables, and $N_c$ is the number of constraints. The flow variables can be found based on the relation

$$R(Q, X) = 0 \tag{2.3}$$

where $R$ is the residual of the flow equations, and as such the flow variables are a function of the design variables.

### 2.1.2 Design Variables

The airfoil shape is described by a B-spline with a user determined number of control points (see Figure 2.1). The design variables are then chosen from these control points. The angle of attack can also be a design variable. For the new method, however, the angle of attack acts as a flow variable in the flow solution, so it is no longer used as a design variable.

### 2.1.3 Objective Function

For this project, the objective function being minimized is the drag coefficient normalized by its initial value:

$$\mathcal{J} = \frac{C_d}{C_{d,0}} \tag{2.4}$$

where $\mathcal{J}$ is the objective function, $C_d$ is the drag coefficient, and $C_{d,0}$ is the drag coefficient at the first flow solution in the optimization process. Both drag coefficients are taken at the end of flow solution steps, so they are the drag reached at the target lift coefficient.

### 2.1.4 Constraints

To ensure that the airfoil is realistic and that it meets any construction or structural requirements in terms of thickness, thickness constraints are imposed on the design problem. They are imposed using the penalty method, so that

$$\mathcal{J} = \mathcal{J}_O + \mathcal{J}_C \tag{2.5}$$

where $\mathcal{J}_O$ is the original objective function as outlined above, and $\mathcal{J}_C$ is the penalty due to the thickness constraints.

Thickness constraints can be specified in three different ways: specific locations along the airfoil that must be a specified thickness, a global or floating thickness constraint that specifies the minimum thickness of the thickest point of the airfoil, and an area constraint that limits the total area of the airfoil.

For the specific thickness constraints, there are $N_C$ thickness constraints. Then the penalty is calculated as

$$\mathcal{J}_C = \sum_{i=0}^{N_C} \begin{cases} \omega_C (1 - \frac{h_i}{h_i^*})^2 & \text{if } h_i < h_i^* \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

where $\omega_C$ is the weight given to the penalty due to the fixed thickness constraints, $h_i$ is the thickness at location $i$, and $h_i^*$ is the minimum thickness at location $i$. Specific thickness constraints are often used to prevent the cross-over of the upper and lower surfaces of the airfoil by ensuring positive thickness.

For a floating thickness constraint, the user selects a number of locations along the airfoil at which to check the thickness. At each calculation of the penalty due to the floating thickness constraint, the program finds the location among those selected by the user that is the thickest, records this thickness, $h_n$, and compares it to the floating thickness $h_{min}$. The penalty is then

$$\mathcal{J}_C = \begin{cases} \omega_{tc}\left(1 - \frac{h_n}{h_{min}}\right)^2 & \text{if } h_n < h_{min} \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

where $\omega_{tc}$ is the weight given to the penalty due to the floating thickness constraint.

For an area based constraint, the program calculates the initial area of the airfoil $A_0$, and then the penalty is based on the current area of the airfoil $A_n$. The penalty applied is

$$\mathcal{J}_C = \begin{cases} \omega_{AC}\left(1 - \frac{A_n}{A_{frac}A_0}\right)^2 & \text{if } A_n < A_{frac}A_0 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

where $\omega_{AC}$ is the weight given to the penalty due to the area constraint, and $A_{frac}$ is the fraction of the area of the initial airfoil that is desired to be the minimum airfoil area.

## 2.2 Flow Equations

### 2.2.1 Navier-Stokes Equations

The governing flow equations are the compressible Navier-Stokes equations, used in conjunction with the Spalart-Allmaras turbulence model. For the two-dimensional flows being studied here, the Navier-Stokes equations are given by

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = \mathrm{Re}^{-1}\left(\frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y}\right) \tag{2.9}$$

where

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}$$

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{bmatrix} \quad F = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 \\ v(e+p) \end{bmatrix} \tag{2.10}$$

$$E_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \varphi_1 \end{bmatrix} \quad F_v = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \varphi_2 \end{bmatrix}$$

$Q$ is the vector of flow variables, $E$ and $F$ are the convective flux vectors, and $E_v$ and $F_v$ are the viscous flux vectors. The variables in the viscous flux vectors are defined as

$$\tau_{xx} = (\mu + \mu_t)(4u_x - 2v_y)/3 \tag{2.11}$$

$$\tau_{xy} = (\mu + \mu_t)(u_y + v_x) \tag{2.12}$$

$$\tau_{yy} = (\mu + \mu_t)(-2u_x + 4v_y)/3 \tag{2.13}$$

$$\varphi_1 = u\tau_{xx} + v\tau_{xy} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma - 1)^{-1}\partial_x(a^2) \tag{2.14}$$

$$\varphi_2 = u\tau_{xy} + v\tau_{yy} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma - 1)^{-1}\partial_y(a^2) \tag{2.15}$$

The pressure, $p$, and speed of sound, $a$, can be determined from the flow variables in the vector $Q$ as follows

$$p = (\gamma - 1)\left[e - \frac{1}{2}\rho(u^2 + v^2)\right] \tag{2.16}$$

$$a = \sqrt{\frac{\gamma p}{\rho}} \tag{2.17}$$

All variables are used in non-dimensional form. These are found by scaling the dimensional variables as shown here.

$$\begin{array}{lllllllll} \rho & = & \frac{\bar{\rho}}{\rho_\infty} & u & = & \frac{\bar{u}}{a_\infty} & v & = & \frac{\bar{v}}{a_\infty} \\ e & = & \frac{\bar{e}}{\rho_\infty a_\infty^2} & x & = & \frac{\bar{x}}{c} & y & = & \frac{\bar{y}}{c} \\ t & = & \frac{\bar{t} a_\infty}{c} & & & & & & \end{array} \tag{2.18}$$

The values with a bar represent the dimensional variables, and those with the subscript $\infty$ are freestream values. $c$ is the chord length of the airfoil.

### 2.2.2 Turbulence Model

These equations are combined with the Spalart-Allmaras turbulence model. This turbulence model can be used to calculate the dynamic eddy viscosity, $\mu_t$, by solving the following one-equation transport model for $\tilde{\nu}$, which is the non-dimensional working variable:

$$\frac{\partial\tilde{\nu}}{\partial t}+u\frac{\partial\tilde{\nu}}{\partial x}+v\frac{\partial\tilde{\nu}}{\partial y}=\frac{c_{b1}}{\mathrm{Re}}\left(1-f_{t2}\right)\tilde{S}\tilde{\nu}+\frac{1}{\sigma\mathrm{Re}}\left\{\left(1+c_{b2}\right)\bigtriangledown\cdot\left[\left(\nu+\tilde{\nu}\right)\bigtriangledown\tilde{\nu}\right]-c_{b2}\left(\nu+\tilde{\nu}\right)\bigtriangledown^2\tilde{\nu}\right\}$$
$$-\frac{1}{\mathrm{Re}}\left(c_{w1}f_w-\frac{c_{b1}}{\kappa^2}f_{t2}\right)\left(\frac{\tilde{\nu}}{d_w}\right)^2+\mathrm{Re}f_{t1}\Delta U^2 \quad (2.19)$$

$\tilde{\nu}$ is non-dimensionalized using the freestream kinematic laminar viscosity.

$$\tilde{\nu}=\frac{\bar{\tilde{\nu}}}{\nu_\infty}=\bar{\tilde{\nu}}\frac{\rho_\infty}{\mu_\infty} \quad (2.20)$$

From the working variable, the kinematic eddy viscosity, $\nu_t=\mu_t/\rho$ can be calculated as

$$\nu_t=\tilde{\nu}f_{v1} \quad (2.21)$$

where

$$f_{v1}=\frac{\chi^3}{\chi^3+c_{v1}^3} \quad (2.22)$$

with $\chi$ defined as

$$\chi=\frac{\tilde{\nu}}{\nu} \quad (2.23)$$

Other terms in the transport model are defined as follows. The production term, $\tilde{S}$ is defined as

$$\tilde{S}=S\mathrm{Re}+\frac{\tilde{\nu}}{\kappa^2d_w^2}f_{v2} \quad (2.24)$$

where $d_w$ is the distance to the closest wall, the magnitude of the vorticity, $S$ is defined as

$$S=\left|\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right| \quad (2.25)$$

and

$$f_{v2}=1-\frac{\chi}{1+\chi f_{v1}} \quad (2.26)$$

The destruction function, $f_w$ is defined as

$$f_w = g \left[ \frac{1 + c_{w3}^3}{g^6 + c_{w3}^6} \right]^{\frac{1}{6}} \tag{2.27}$$

with

$$g = r + c_{w2}(r^6 - r) \tag{2.28}$$

and

$$r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d_w^2} \tag{2.29}$$

The functions $f_{t1}$ and $f_{t2}$ control the location of the transition from laminar to turbulent flow. Although some work has been done studying the effect of determining the laminar to turbulent transition on the optimization [5, 6], for this project the flow is assumed to be fully turbulent. Because of this, the functions $f_{t1}$ and $f_{t2}$ have been set to zero. All other parameters are constants, as shown below.

$$\begin{array}{rclrcl} c_{b1} & = & 0.1355 & c_{b2} & = & 0.622 \\ \kappa & = & 0.41 & \sigma & = & \frac{2}{3} \\ c_{w1} & = & c_{b1}/\kappa^2 + (1 + c_{b2})/\sigma & c_{w2} & = & 0.3 \\ c_{w3} & = & 2.0 & c_{v1} & = & 7.1 \end{array} \tag{2.30}$$

### 2.2.3 Thin-Layer Approximation and Coordinate Transformation

In the existing algorithm, the Navier-Stokes equations are transformed from the physical domain of Cartesian coordinates to the computational domain of general curvilinear coordinates where

$$\tau = t \tag{2.31}$$

$$\xi = \xi(x, y, t) \tag{2.32}$$

$$\eta = \eta(x, y, t) \tag{2.33}$$

The curvilinear coordinates are selected such that grid lines are straight, parallel, and evenly spaced by one unit. This forms a rectangular domain in $\xi$ and $\eta$ which facilitates the use of finite differences in the discretization of the flow equations [28].

The thin-layer Navier-Stokes equations in curvilinear coordinates are given as

$$\frac{\partial \widehat{Q}}{\partial \tau} + \frac{\partial \widehat{E}}{\partial \xi} + \frac{\partial \widehat{F}}{\partial \eta} = \mathrm{Re}^{-1} \frac{\partial \widehat{S}}{\partial \eta} \tag{2.34}$$

where $\widehat{Q}$ is defined as

$$\widehat{Q} = J^{-1} Q = J^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \tag{2.35}$$

with $J$ the metric Jacobian of the coordinate transformation

$$J^{-1} = (x_\xi y_\eta - x_\eta y_\xi) \tag{2.36}$$

The convective flux vectors are defined as

$$\widehat{E} = J^{-1} \begin{bmatrix} \rho U \\ \rho U u + \xi_x p \\ \rho U v + \xi_y p \\ (e+p)U - \xi_t p \end{bmatrix} \quad \widehat{F} = J^{-1} \begin{bmatrix} \rho V \\ \rho V u + \eta_x p \\ \rho V v + \eta_y p \\ (e+p)V - \eta_t p \end{bmatrix} \tag{2.37}$$

where the contravariant velocities are

$$U = \xi_t + \xi_x u + \xi_y v \quad V = \eta_t + \eta_x u + \eta_y v \tag{2.38}$$

The viscous flux vector is given by

$$\widehat{S} = J^{-1} \begin{bmatrix} 0 \\ \eta_x m_1 + \eta_y m_2 \\ \eta_x m_2 + \eta_y m_3 \\ \eta_x (u m_1 + v m_3 + m_4) + \eta_y (u m_2 + v m_3 + m_5) \end{bmatrix} \tag{2.39}$$

with

$$m_1 = (\mu + \mu_t)(4\eta_x u_\eta - 2\eta_y v_\eta)/3 \tag{2.40}$$

$$m_2 = (\mu + \mu_t)(\eta_y u_\eta + \eta_x v_\eta) \tag{2.41}$$

$$m_3 = (\mu + \mu_t)(-2\eta_x u_\eta + 4\eta_y v_\eta)/3 \tag{2.42}$$

$$m_4 = (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma - 1)^{-1} \eta_x \partial_\eta (a^2) \tag{2.43}$$

$$m_5 = (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma - 1)^{-1} \eta_y \partial_\eta (a^2) \tag{2.44}$$

The equation for the Spalart-Allmaras turbulence model must also be transformed into the curvilinear coordinate system. It is then

$$\frac{\partial\tilde{\nu}}{\partial\tau} + U\frac{\partial\tilde{\nu}}{\partial\xi} + V\frac{\partial\tilde{\nu}}{\partial\eta} = \frac{1}{\mathrm{Re}}\left\{c_{b1}\tilde{S}\tilde{\nu} - c_{w1}f_w\left(\frac{\tilde{\nu}}{d_w}\right)^2 + \frac{1}{\sigma}[(1+c_{b2})T_1 - c_{b2}T_2\right\} \tag{2.45}$$

with

$$\begin{aligned} T_1 &= \xi_x\frac{\partial}{\partial\xi}\left[(\nu+\tilde{\nu})\xi_x\frac{\partial\tilde{\nu}}{\partial\xi}\right] + \eta_x\frac{\partial}{\partial\eta}\left[(\nu+\tilde{\nu})\eta_x\frac{\partial\tilde{\nu}}{\partial\eta}\right] \\ &+\xi_y\frac{\partial}{\partial\xi}\left[(\nu+\tilde{\nu})\xi_y\frac{\partial\tilde{\nu}}{\partial\xi}\right] + \eta_y\frac{\partial}{\partial\eta}\left[(\nu+\tilde{\nu})\eta_y\frac{\partial\tilde{\nu}}{\partial\eta}\right] \end{aligned} \tag{2.46}$$

and

$$T_2 = (\nu+\tilde{\nu})\left[\xi_x\frac{\partial}{\partial\xi}\left(\xi_x\frac{\partial\tilde{\nu}}{\partial\xi}\right) + \eta_x\frac{\partial}{\partial\eta}\left(\eta_x\frac{\partial\tilde{\nu}}{\partial\eta}\right) + \xi_y\frac{\partial}{\partial\xi}\left(\xi_t\frac{\partial\tilde{\nu}}{\partial\xi}\right) + \eta_y\frac{\partial}{\partial\eta}\left(\eta_t\frac{\partial\tilde{\nu}}{\partial\eta}\right)\right] \tag{2.47}$$

These equations are discretized in time to move towards a steady-state solution.

# Chapter 3

# Algorithm

This project investigates the effect of setting the target lift coefficient as an input parameter to the flow solver, and having the appropriate angle of attack for the given lift coefficient determined in the flow solver at each iteration in the optimization process. The objective function for lift-constrained drag minimization then depends only on drag, using the following equation

$$\mathcal{J} = \frac{C_d}{C_{d,0}} \tag{3.1}$$

where $C_d$ is the drag coefficient and $C_{d,0}$ is the initial drag coefficient. Minimizing this objective function will minimize the drag coefficient.

## 3.1 Flow Solver - Approximate Factorization

The first step in the implementation of this method is to enable the flow solver to have the coefficient of lift as an input and to then output the angle of attack required to achieve this lift. In the previously existing code, the input parameters to the flow solver are Mach number, Reynolds number, angle of attack, and airfoil shape, and the output includes the coefficients of lift, drag, and moment. This is changed so that the input parameters to the flow solver become Mach number, Reynolds number, coefficient of lift, and airfoil shape, and the output becomes angle of attack and the coefficients of drag and moment.

To move towards a steady-state solution, we apply first-order implicit time marching to Equation 2.34. This gives

$$\widehat{Q}^{n+1} - \widehat{Q}^{n} + \Delta t \left( \widehat{E}_{\xi}^{n+1} + \widehat{F}_{\eta}^{n+1} - \mathrm{Re}^{-1} \widehat{S}_{\eta}^{n+1} \right) = 0 \tag{3.2}$$

Linearizing the flux vectors allows us to solve the above equation for the step in the solution

$$\Delta\widehat{Q}^n = \widehat{Q}^{n+1} - \widehat{Q}^n \tag{3.3}$$

For more details on the approximate factorization flow solver, see [28].

In the approximate factorization flow solver, the change in the flow variables in one iteration is quite small, and the residual is only slightly reduced at each iteration. Because of this, the relation between the coefficient of lift and the angle of attack does not need to be solved for explicitly. Instead, the method of angle of attack relaxation, which was recommended by the originators of the ARC2D code and is already a part of the flow solver, was tested [28]. In this method, the change in angle in attack that is applied is a correction based on the difference between the current coefficient of lift and the target coefficient of lift:

$$\Delta\alpha = -\beta_\alpha(C_l - C_l^*) \tag{3.4}$$

where $\Delta\alpha$ is the correction to be made to the angle of attack, $\beta_\alpha$ is a relaxation parameter, $C_l$ is the current coefficient of lift, and $C_l^*$ is the target lift coefficient. The relaxation parameter can be approximated initially from the derivative of the coefficient of lift with respect to angle of attack, which is theoretically $2\pi$ per radian for thin airfoils in incompressible, inviscid, irrotational flow. As the angle of attack in our code is given in degrees, we can approximate the relaxation parameter as shown below.

$$\Delta C_l \approx 2\pi \text{ per radian } \Delta\alpha \tag{3.5}$$

$$\Delta C_l \approx 0.1097 \text{ per degree } \Delta\alpha \tag{3.6}$$

$$\Delta\alpha \approx 9.12^\circ\ \Delta C_l \tag{3.7}$$

This suggests that a value of less than 9 should be used for the relaxation coefficient. The developers of the original approximate factorization code recommend that a value of 2 be used for this parameter [28]. To determine the best value to use for the relaxation parameter, a parametric study was done. Two other parameters were also studied. Both of these parameters are related to the fact that angle of attack relaxation does not need to be performed at each iteration in the flow solution, nor does it need to be performed right from the beginning of the solution. It has been found to work well if there are a certain number of iterations before the first use of angle of attack relaxation (iclstart) and if there

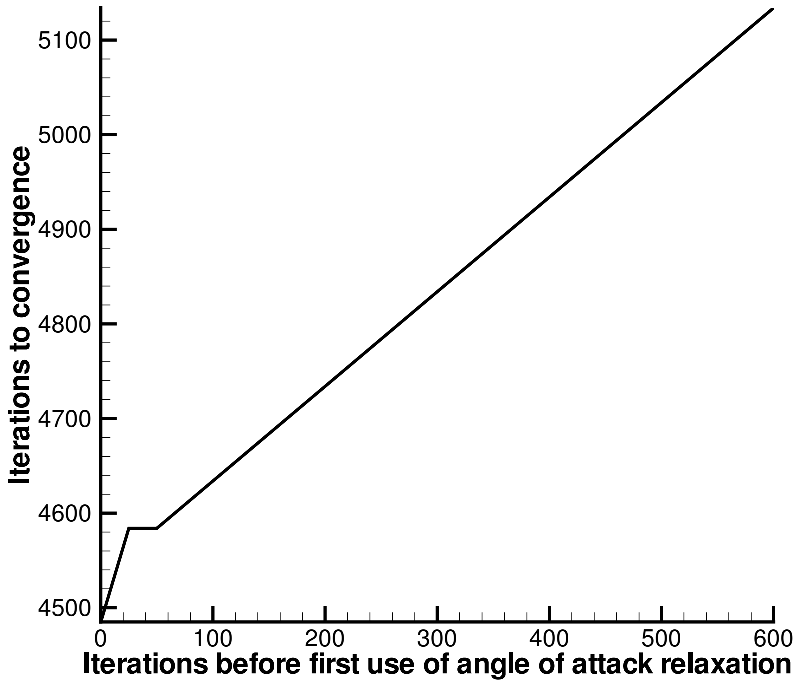


Figure 3.1: Effect of number of iterations to beginning of angle of attack relaxation on convergence rate of approximate factorization flow-solver

are a certain number of iterations between each use of angle of attack relaxation (iclfreq). A parametric study was performed to determine which values of these parameters would allow the algorithm to converge rapidly while remaining stable for a number of cases.

The figures shown here were obtained from a subsonic case with flow conditions $\text{Ma}_\infty = 0.25$, $C_l^* = 0.9$, and $\text{Re} = 2.88 \times 10^6$, on a grid 264 by 65. Similar results for the parameters were obtained for a transonic test case with $\text{Ma}_\infty = 0.7$, $C_l^* = 0.2$, and $\text{Re} = 9 \times 10^6$. Test case residual converged to $10^{-7}$ in all cases shown here. As can be seen from Figure 3.1, the number of iterations prior to using angle of attack relaxation is linearly related to the number of iterations before the algorithm converges. This parameter is not important to the convergence of the algorithm, and was set to zero for all cases. For the other two parameters, namely the frequency of the angle of attack relaxation and the relaxation parameter, there is a trade-off between speed and stability (see Figures 3.2 and 3.3). For the relaxation parameter (Figure 3.2), the algorithm runs more quickly as the value increases up to a certain point, after which the time to

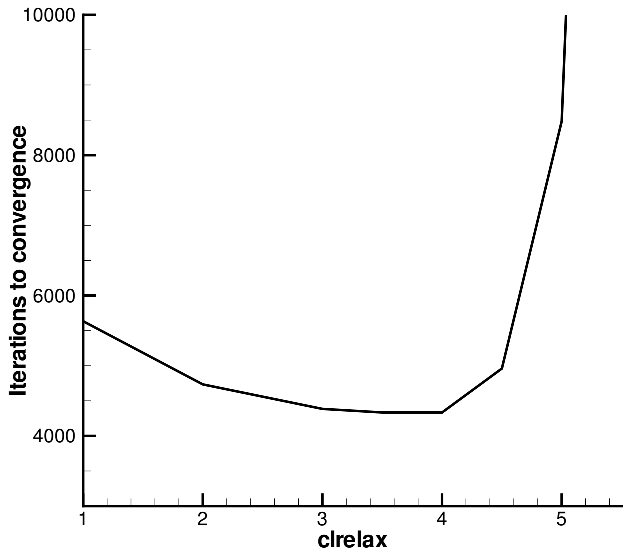


Figure 3.2: Effect of relaxation parameter on convergence rate of approximate factorization flow-solver

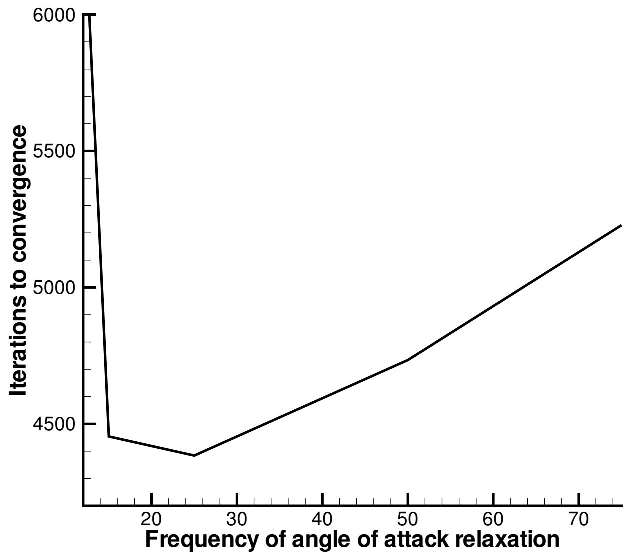


Figure 3.3: Effect of frequency of angle of attack relaxation on convergence rate of approximate factorization flow-solver

converge increases rapidly until the algorithm no longer converges. As a good balance between speed and stability, a value of 3 was chosen for the relaxation parameter. For the frequency of performing angle of attack relaxation, it can be seen in Figure 3.3 that the more frequently it is performed, the more rapidly the algorithm converges, until it is performed approximately every 15 iterations, after which the algorithm begins to have convergence problems. The algorithm was therefore run every 50 iterations to ensure stability.

## 3.2 Flow Solver - Newton-Krylov Algorithm

Unlike the approximate factorization solver, the steps taken by the Newton-Krylov algorithm are quite large. Because of this, the relationship between the coefficient of lift and the angle of attack must be more accurately computed and included in the calculations.

In the Newton-Krylov method, for flow solution with no variation in angle of attack, the governing equations, once spatially discretized, lead to a nonlinear system of equations

$$R(\widehat{Q}, X) = 0 \tag{3.8}$$

where $R$ is the residual vector, $\widehat{Q}$ is the vector of flow variables scaled by the Jacobian found in spatial discretization, and $X$ is the vector of design variables. This equation is then solved using Newton's method. To do this, the following linear system of equations must be solved.

$$A^{(n)} \Delta \widehat{Q}^{(n)} = R^{(n)} \tag{3.9}$$

where $A$ is the flow Jacobian at iteration $n$, defined as

$$A = \frac{\partial R}{\partial \widehat{Q}} \tag{3.10}$$

To solve Equation 3.9, the generalized minimal residual (GMRES) Krylov subspace method [31] is used.

To ensure that the desired lift coefficient is achieved when the algorithm converges, a new variable is added to the flow variable vector, $\widehat{Q}$. The new variable is the angle of attack, and the new flow vector is

$$\widehat{\mathcal{Q}} = \widehat{Q}, \widehat{\alpha} \tag{3.11}$$

The variable $\widehat{\alpha}$ is the angle of attack, in degrees, scaled by 100. This ensures that new terms are of a similar order of magnitude to previously existing terms. This scaling is used throughout the Newton-Krylov flow solver and the adjoint gradient calculation, but will not be carried through here for clarity. To allow the angle of attack to be found, a new equation is added to the residual vector. This equation is

$$R_{C_l} = C_{l,\text{calculated}} - C_{l,\text{target}} = 0 \tag{3.12}$$

It was found in the process of implementing the code that this new equation was a different order of magnitude than other residual equations, so to allow the code to converge, this equation was scaled to ensure the new diagonal element of the flow Jacobian is 1. This scaling is propagated throughout the code, but will not be included in the expansion of equations for simplicity.

Adding this equation to the initial residual vector gives the expanded residual vector

$$\mathcal{R} = R(\widehat{\mathcal{Q}}, X), R_{C_l} \tag{3.13}$$

The modified flow solver now solves the nonlinear system of equations

$$\mathcal{R}(\widehat{\mathcal{Q}}, X) = 0 \tag{3.14}$$

Using Newton's method, then, the system of linear equations to be solved then becomes

$$\mathcal{A}^{(n)} \Delta \widehat{\mathcal{Q}}^{(n)} = \mathcal{R}^{(n)} \tag{3.15}$$

where $\mathcal{A}$ is the modified flow Jacobian, defined as $\frac{\partial \mathcal{R}}{\partial \widehat{\mathcal{Q}}}$. This is the same as the existing flow Jacobian $\frac{\partial R}{\partial \widehat{Q}}$, but with an additional row and column that are related to the new residual equation and the inclusion of the angle of attack as a flow variable. This means that the new row contains the terms of $\frac{\partial R_{C_l}}{\partial \widehat{Q}}$. The new column contains the terms of $\frac{\partial R}{\partial \alpha}$. The new diagonal term belongs to both of these and is the partial derivative of the new residual equation with respect to angle of attack.

The linear system of equations is solved using GMRES with right preconditioning, as for the flow solver without a variable angle of attack. For the flow solver, the full second-order Jacobian does not need to be determined explicitly. This is because in applying Jacobian-free GMRES the matrix $\mathcal{A}$ is not required, but the matrix vector product $\mathcal{A}v$ is. In finding this, the derivative properties of the matrix $\mathcal{A}$ are used. We know that

$$\mathcal{A} = \frac{\partial \mathcal{R}}{\partial \widehat{\mathcal{Q}}} \tag{3.16}$$

and so a forward difference approximation is used when the vector $\mathcal{A}v$ is required in GMRES, as shown below [27]:

$$\mathcal{A}v = \frac{\mathcal{R}(\widehat{\mathcal{Q}} + \epsilon v) - \mathcal{R}(\widehat{\mathcal{Q}})}{\epsilon} \tag{3.17}$$

As mentioned earlier, GMRES is used with a preconditioner. This is done to accelerate convergence. For this preconditioner, the approximate Jacobian, which contains only nearest neighbour terms, is used. Therefore, only the new diagonal term, namely the partial derivative of the new residual equation with respect to the angle of attack, is added to the existing preconditioner. In effect, this is the partial derivative of coefficient of lift with respect to angle of attack. This term is calculated analytically, as the angle of attack is used in calculating the effect of the normal and chord directed forces on lift and drag (see Appendix A).

## 3.3 Optimizer Equations

The effect of having a variable angle of attack in the flow solver must also be taken into account in the optimization stage. The method currently used in Optima2D is the BFGS quasi-Newton method to calculate an approximation to the inverse Hessian, combined with a backtracking line search. Once the approximate inverse Hessian is calculated, it is used to find a search direction. The search direction is found by solving the equation

$$s_p = -H_p \mathcal{G}_p \tag{3.18}$$

where $s_p$ is the new search direction to be used in the backtracking line search, $H_p$ is the approximate inverse Hessian, and $\mathcal{G}_p$ is the gradient of the objective function with respect to the design variables. The value of the approximate inverse Hessian is calculated using BFGS as follows.

$$H_{p+1} = \left[ I - \frac{s_p y_p^T}{s_p^T y_p} \right] H_p \left[ I - \frac{y_p s_p^T}{s_p^T y_p} \right] + \frac{s_p s_p^T}{s_p^T y_p} \tag{3.19}$$

where $p$ is the current iteration, $y_p$ is the change in the gradient since the previous iteration, that is $y_p = \mathcal{G}_p - \mathcal{G}_{p-1}$.

The gradient is the derivative of the objective function with respect to the design variables, $\frac{d\mathcal{J}}{dX}$, where $\mathcal{J}$ is the objective function value and $X$ is the vector of design variables, of size $N_D$, the number of design variables. The objective function being studied depends on the design variables directly, as they affect the integration of the pressure along the surface of the airfoil, and also on the flow variables $\mathcal{Q}$, a vector of size $N_Q$, which in turn also depend on the design variables. Both of these effects must be accounted for in calculating the derivative of the objective function with respect to the design variables. There are four methods for finding the derivative that will be examined in this project: finite differences, flow sensitivities, matrix-free flow sensitivities, and the adjoint method.

When finding the gradient by finite differences, centred differences are used for higher accuracy than would be obtained with forward or backward differences. To obtain the derivative of the objective function with respect to the design variable $X_i$, the following calculation is performed.

$$\frac{d\mathcal{J}}{dX_i} = \frac{\mathcal{J}(\mathcal{Q}(X_i+\epsilon), X_i+\epsilon) - \mathcal{J}(\mathcal{Q}(X_i-\epsilon), X_i-\epsilon)}{2\epsilon} \tag{3.20}$$

This is done for $i = 1 to N_D$. As the values for the objective function in this equation are obtained by performing a full flow solution, the cost of this method is very dependent on the number of design variables; each gradient calculation requires $2N_D$ flow solutions. For high numbers of design variables, this method becomes prohibitively expensive, so investigation of other methods is necessary.

These other methods are obtained beginning from the chain rule expansion of the derivative of the objective function with respect to the design variables, namely,

$$\mathcal{G} = \frac{d\mathcal{J}}{dX_i} = \frac{\partial \mathcal{J}}{\partial X_i} + \frac{\partial \mathcal{J}}{\partial \mathcal{Q}}\frac{d\mathcal{Q}}{dX_i} \tag{3.21}$$

for $i = 1 to N_D$. However, we know that the residual is a function of the flow variables and the turbulence variables. We also know that for any design variables the residual must be zero, so

$$\frac{d\mathcal{R}}{dX_i} = \frac{\partial \mathcal{R}}{\partial X_i} + \frac{\partial \mathcal{R}}{\partial \mathcal{Q}}\frac{d\mathcal{Q}}{dX_i} = 0 \tag{3.22}$$

$$\frac{d\mathcal{Q}}{dX_i} = -\left(\frac{\partial \mathcal{R}}{\partial \mathcal{Q}}\right)^{-1}\frac{\partial \mathcal{R}}{\partial X_i} \tag{3.23}$$

Solving Equation 3.23 for $\frac{d\mathcal{Q}}{dX_i}$ gives the direct, or flow-sensitivity method. This is done using the same generalized minimal residual solver as is used for the Newton-Krylov flow solver. The results for $\frac{d\mathcal{Q}}{dX_i}$ can then be used in Equation 3.21 to solve for the gradient. The disadvantage of using this method is that Equation 3.23 must be solved for each design variable, which can increase the cost of this method. Equation 3.23 can either be solved using the full second-order flow Jacobian, leading to the flow-sensitivity method of gradient calculation, or it can be solved using finite differences, as for the flow solver. This method is called the matrix-free sensitivity method. For the matrix-free sensitivity method, the finite differences used in GMRES are slightly different from those used in the flow solver, as centred differences are used to achieve higher accuracy. So the matrix-vector product $\mathcal{A}v$ is calculated as

$$\mathcal{A}v = \frac{\mathcal{R}(\mathcal{Q}+\epsilon v) - \mathcal{R}(\mathcal{Q}-\epsilon v)}{2\epsilon} \tag{3.24}$$

The adjoint method can be obtained by combining Equation 3.23 with Equation 3.21 to give

$$\frac{d\mathcal{J}}{dX_i} = \frac{\partial \mathcal{J}}{\partial X_i} - \frac{\partial \mathcal{J}}{\partial \mathcal{Q}} \left(\frac{\partial \mathcal{R}}{\partial \mathcal{Q}}\right)^{-1} \frac{\partial R}{\partial X_i} \tag{3.25}$$

$$\frac{d\mathcal{J}}{dX_i} = \frac{\partial \mathcal{J}}{\partial X_i} - \psi^T \frac{\partial \mathcal{R}}{\partial X_i} \tag{3.26}$$

where

$$\frac{\partial \mathcal{R}}{\partial \mathcal{Q}}^T \psi = \frac{\partial \mathcal{J}}{\partial \mathcal{Q}}^T \tag{3.27}$$

$\psi$ is known as the adjoint variable, and in Optima2D the GMRES method is used to solve Equation 3.27 [19]. This method is lower in cost than the other methods outlined above, as the adjoint equation (Equation 3.27) is independent of the number of design variables, and so only needs to be solved once for each gradient calculation, rather than $N_D$ times as in the sensitivity methods. Due to the transpose on the left hand side of the equation, the finite difference method used in the matrix-free sensitivity method can not be used here, so $\frac{\partial \mathcal{R}}{\partial \mathcal{Q}}$ must be the full second-order Jacobian [19]. In this case, differing from the Newton-Krylov flow solver, all new components in the flow Jacobian, as discussed in the section on modifications of the Newton-Krylov flow solver, must be included in the flow Jacobian.

The partial derivative of the new residual equation

$$R_{C_l} = C_l - C_l^* = 0 \tag{3.28}$$

with respect to the flow variables is just the partial derivative of the coefficient of lift with respect to the flow variables. The coefficient of lift consists of two parts. The inviscid component of the coefficient of lift depends only on the flow variables for the nodes on the airfoil surface. The viscous component also depends on the flow variables for these nodes, but also on the two layers of nodes out from the airfoil. This means that the derivative is non-zero for only these nodes. These derivatives have been determined analytically (see Appendix A).

The partial derivative of the previously existing residual equations with respect to angle of attack is non-zero only for the nodes on far-field and outflow boundaries of the grid. This is because the boundary condition set for the outer boundary of the grid depends on the $x$- and $y$-components of freestream flow velocity, which in turn depend on angle of attack. These derivatives have also been determined analytically (see Appendix A).

The partial derivative of the coefficient of lift with respect to the angle of attack was determined analytically, as the angle of attack simply affects how the normal and chord-directed forces are resolved to give lift and drag (see Appendix A).

Generally, for the optimizer, the adjoint method is used because finding the adjoint variable is independent of the number of design variables. Changing the flow solver to have $C_l$ as an input and $\alpha$ as an output decreases the number of design variables $N_D$ (since $\alpha$ is currently a design variable), but increases $N_Q$, the number of unknown variables, by one, the angle of attack. The finite difference and flow-sensitivity methods have been implemented only for testing purposes.

## 3.4 Other Modifications

One of the main goals of this project was to reduce the need to tune algorithm parameters to make the code more user friendly. However, it was found that for more realistic design cases with a number of different operating points with different objectives, the weights on different operating points are very challenging to set as different objective functions can have different orders of magnitude. Additionally, when using different objective functions the weight given to thickness constraints has a different meaning for different points, depending on the actual value of the objective function and gradient. Because of this, the calculation of the objective function has been modified to be normalized by the initial value of the objective function.

$$\mathcal{J}_{\text{new}} = \frac{\mathcal{J}_{\text{old}}}{\mathcal{J}_{\text{old,initial}}} \tag{3.29}$$

where $\mathcal{J}_{\text{old}}$, $\mathcal{J}_{\text{old,initial}}$ and $\mathcal{J}_{\text{new}}$ are all objective functions as calculated before any thickness constraint penalties are added. If the initial objective function value $\mathcal{J}_{\text{old,initial}}$ is zero, normalization is not used.

For example, looking at the objective function that has previously been used for lift-constrained drag-minimization, we have an objective that attempts to reach a target lift, $C_l^*$, and a target drag, $C_d^*$ with the objective function

$$\mathcal{J} = \begin{cases} \omega_L \left(1 - \frac{C_l}{C_l^*}\right)^2 + \omega_D \left(1 - \frac{C_d}{C_d^*}\right)^2 + t.c. & \text{if } C_d > C_d^* \\ \omega_L \left(1 - \frac{C_l}{C_l^*}\right)^2 + t.c. & \text{otherwise} \end{cases} \tag{3.30}$$

where $\omega_L$ is the weight for the lift coefficient optimization, $\omega_D$ is the weight for the drag coefficient optimization, and $t.c.$ is the penalty added due to thickness constraints [19]. The new objective function, if $C_{d,0}$ were greater than $C_d^*$, would be

$$\mathcal{J} = \begin{cases} \frac{\omega_L \left(1 - \frac{C_l}{C_l^*}\right)^2 + \omega_D \left(1 - \frac{C_d}{C_d^*}\right)^2}{\omega_L \left(1 - \frac{C_{l,0}}{C_l^*}\right)^2 + \omega_D \left(1 - \frac{C_{d,0}}{C_d^*}\right)^2} + t.c. & \text{if } C_d > C_d^* \\ \frac{\omega_L \left(1 - \frac{C_l}{C_l^*}\right)^2}{\omega_L \left(1 - \frac{C_{l,0}}{C_l^*}\right)^2 + \omega_D \left(1 - \frac{C_{d,0}}{C_d^*}\right)^2} + t.c. & \text{otherwise} \end{cases} \tag{3.31}$$

where $C_{l,0}$ is the initial lift coefficient and $C_{d,0}$ is the initial drag coefficient. If $C_{d,0}$ were less than $C_d^*$, the drag term would not be included in the denominator.

It is important to note that while the objective functions that we wish to minimize are normalized by their initial value, the thickness constraints are not included in the normalization. This means that the thickness constraint weight will have the same meaning regardless of the objective function to which it is being applied. If it were scaled by the initial objective function value, its meaning would depend on that initial value, and that is not desirable.

# Chapter 4

# Results

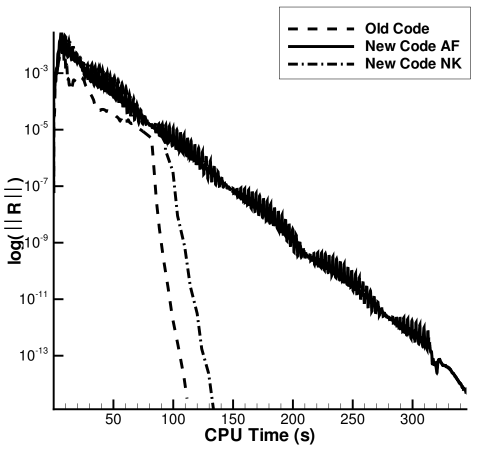
## 4.1 Flow Solver

Two cases were studied to demonstrate the performance of the new flow solver, both with just approximate factorization and with Newton-Krylov after using approximate factorization for start-up. The flow cases were first run with the existing code, and then run with a desired lift coefficient set to the final lift coefficient achieved by running the previously existing code. The same grids were used for comparisons between the two different versions of the code. Both cases have previously been run by Nemec [19] to demonstrate flow solver performance. For all cases studied in this project, the circulation correction was not used.

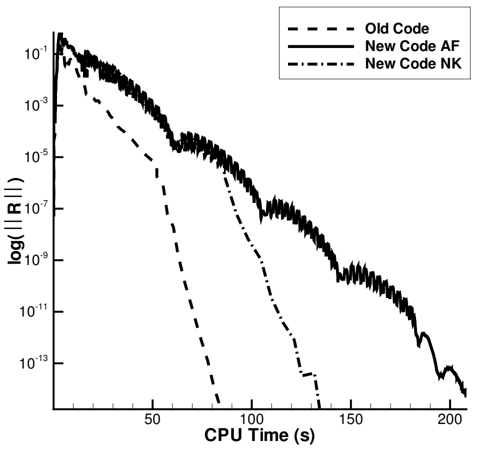
Case 1 was run for the NACA 0012 airfoil at $\text{Ma}_\infty = 0.3$ and $\text{Re} = 2.88 \times 10^6$. For the previously existing code, the angle of attack is 6°. This gave a coefficient of lift of $C_l = 0.669221$, which was then set as the target coefficient of lift for the runs with the flow solver with the variable angle of attack, starting from the initial angle of attack of 6°. The grid has $265 \times 53$ nodes.

Case 2 was run for the RAE 2822 airfoil on a grid with $257 \times 57$ nodes. The flow conditions are: $\text{Ma}_\infty = 0.729$ and $\text{Re} = 6.5 \times 10^6$. The initial angle attack for all cases is 2.31°, which gives a coefficient of lift of $C_l = 0.67953$, which is used as the target lift coefficient in the cases run with a variable angle of attack.

Results shown in Figure 4.1 show that the Newton-Krylov flow solver (denoted as NK) combined with the approximate factorization flow solver (denoted as AF) for start-up is approximately 1.5 to 2.5 times faster than using only the approximate factorization flow solver. The new flow solver is approximately 20 to 50% slower than the previous



(a) Case 1



(b) Case 2

Figure 4.1: Performance of the new flow solver

algorithm, but the increased time required for flow solution does not prevent the new algorithm from being a success, as the new algorithm eliminates the need for repeated tests varying parameters in the optimization stage.

The flow solution with a variable angle of attack can also be compared to an optimization with the angle of attack as the only design variable. This has been done for both Cases 1 and 2. The initial angle of attack for Case 1 was set to 5°, for both the flow solution and the optimization, to prevent the initial point of the single design variable case being the optimum. For Case 2, the initial angle of attack was 3°. The convergence tolerance for the optimization was for the gradient to be below $10^{-6}$ for Case 1 and $10^{-5}$ for Case 2. Comparing the new flow solver to the single design variable optimization shows a decrease of approximately 70% in the time required for the desired lift coefficient to be reached (see Table 4.1). Also, it can be seen that the lift coefficient is closer to the desired coefficient of lift for the new flow solver than for the optimization. While this could be improved by lowering the tolerance on the optimization convergence, this will also increase the total time required for the optimization to occur. This application of the flow solver is useful for finding flow characteristics at fixed lift over a range of Mach numbers. This has previously been done by single design variable optimization, but can now be done with the new flow solver.

| Case | Method | $C_l^*$ | $C_{l,final}$ | $C_{d,final}$ | Final $\alpha$ | CPU Time (s) |
|---|---|---|---|---|---|---|
| 1 | Optimization | 0.669221 | 0.669220895 | 0.014933215 | 6.00 | 466.1 |
| 1 | Flow Solve | 0.669221 | 0.669221000 | 0.014933221 | 6.00 | 125.9 |
| 2 | Optimization | 0.679530 | 0.679530007 | 0.015043231 | 2.31 | 563.7 |
| 2 | Flow Solve | 0.679530 | 0.679530000 | 0.015043279 | 2.31 | 125.3 |

Table 4.1: Comparison of single design variable optimization to new flow solver

## 4.2 Gradient Calculations

The case studied to examine the accuracy found with the new objective function is a single-point optimization case, with 10 design variables. The initial airfoil is the NACA0012 airfoil, and there are 15 points used to define the airfoil. The point at the leading edge and the four points near the trailing edge are not used as design variables. The flow conditions are $\text{Ma}_\infty = 0.7$, $C_l^* = 0.4728$, and $\text{Re} = 9 \times 10^6$, on a grid 264 by 65. This case has been previously studied by Nemec [19, 20]. No thickness constraints were imposed. The gradient for the ten design variables was calculated using the three different methods described above: finite-difference, flow-sensitivity, and adjoint (see Table 4.2). The finite-difference method is used as the basis for comparison of the other three methods. In all cases the flow was fully converged to a tolerance of $5 \times 10^{-15}$ prior to the calculation of the gradient.

These results show that the gradients found by the different methods are extremely close in value, and therefore that the gradient calculated using the adjoint method is valid.

## 4.3 Optimization Cases

All optimization cases were run, as mentioned in Section 4.1, without the circulation correction. In the optimization algorithm, all design variables were scaled by their initial values, and the initial inverse Hessian estimate is formed using the scaled variables, Hessian A as described by Zingg et. al [36]. Unless otherwise stated, all cases were run without second-difference dissipation.

| Control Point | Finite Diff | Flow-Sensit Grad | Flow-Sensit (% Diff) | Adjoint Grad | Adjoint (% Diff) |
|---|---|---|---|---|---|
| 3 | -14.8567 | -14.8564 | 0.001822 | -14.8564 | 0.001832 |
| 4 | -9.30289 | -9.30280 | 0.000953 | -9.30280 | 0.000944 |
| 5 | -6.02968 | -6.02967 | 0.000297 | -6.02966 | 0.000312 |
| 6 | -3.35382 | -3.35378 | 0.001214 | -3.35378 | 0.001220 |
| 7 | -0.889360 | -0.889174 | 0.02081 | -0.889173 | 0.02098 |
| 9 | -12.5174 | -12.5172 | 0.001395 | -12.5172 | 0.001382 |
| 10 | 25.0431 | 25.0430 | 0.000151 | 25.0430 | 0.000145 |
| 11 | 17.0894 | 17.0892 | 0.001523 | 17.0892 | 0.001525 |
| 12 | -24.4999 | -24.4997 | 0.000593 | -24.4997 | 0.000593 |
| 13 | -7.34901 | -7.34894 | 0.001018 | -7.34893 | 0.001041 |

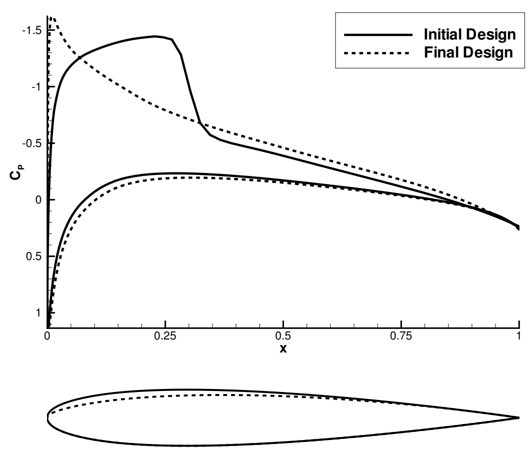
Table 4.2: Comparison of gradients calculated with different methods

### 4.3.1 Single-Point Subsonic Airfoil Design

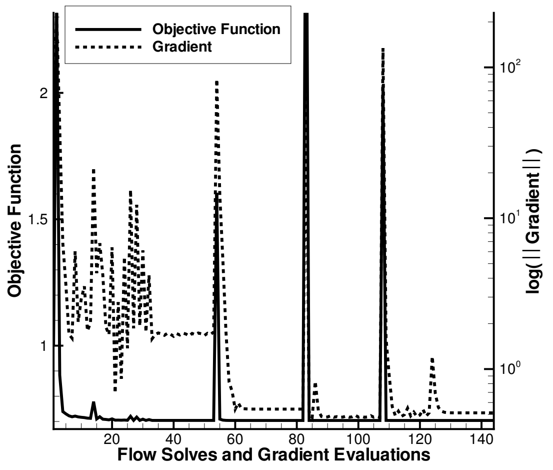
This case is the case examined in Section 4.2. In the optimization case, however, only four design variables, control points 9 to 12, were used to allow comparison to work done by Nemec [19] and Nemec and Zingg [20, 21]. Nemec performed this optimization using the previously existing lift-constrained drag minimization code. No thickness constraints were imposed. The case was run both with second-difference dissipation ($\kappa_2 = 1.0$) and without second-difference dissipation ($\kappa_2 = 0.0$). See Appendix B for the input file. All subsequent cases were run without second-difference dissipation.

The resulting airfoils and pressure distributions are shown in Figures 4.2(a) and 4.3(a), for the case with and without second-difference dissipation respectively. The convergence of the objective function and gradient are shown in Figures 4.2(b) and 4.3(b). This shows that the gradient converges less consistently with second-difference dissipation than without, due to the pressure switch [19]. The three spikes in the gradient and objective function when second-difference dissipation is used occur when the optimization is restarted from the steepest descent method after the objective function stalls during a line search.

The reference case was also run using the normalized code but with the previously



(a) Pressure distributions and airfoil shapes

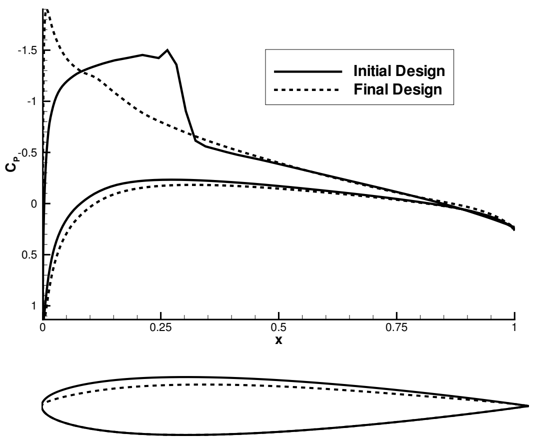


(b) Optimization convergence history

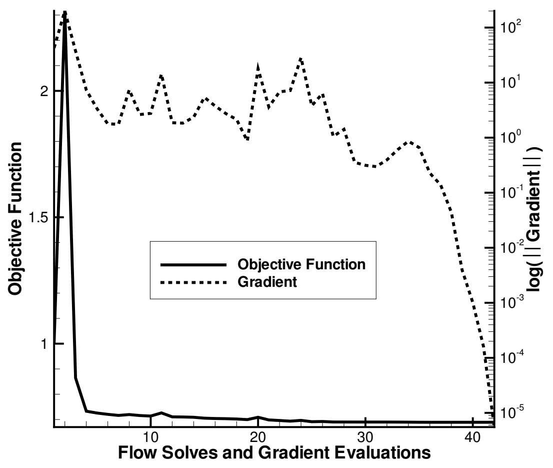
Figure 4.2: Subsonic single-point optimization, with second-difference dissipation

existing objective function without second-difference dissipation. The target lift was 0.4728, the target drag was 0.0112, and the weights are $\omega_L = 2.0$ and $\omega_D = 1.0$, as in the reference case. This case converged to a significantly higher drag coefficient than was reached with the new optimization algorithm. The reason for this is that the drag achieved with the previously existing algorithm is already lower than the specified target drag, the reference case here is not truly representative of lift-constrained drag minimization. Because of this, another test case was run with the previously existing algorithm, but with the target drag set to 0.01. This case has a lower drag coefficient, but the desired lift is not reached. This shows some of the difficulties encountered in using the previous algorithm for lift-constrained drag minimization. The final lift and drag coefficients for all cases studied are shown in Table 4.3.

This demonstrates that the desired lift coefficient is achieved, with no significant change in the final drag coefficient, when compared to the reference case run with second-difference dissipation. When compared to the reference cases run without second-difference dissipation, the airfoil achieved with the new algorithm shows clear benefits over both airfoils achieved with the previously existing algorithm, namely lower drag at the target lift. This case shows the difficulties that may be encountered in using the previous algorithm for lift-constrained drag minimization, as the values given to target lift and drag have a substantial effect on the final coefficients reached. The total time



(a) Pressure distributions and airfoil shapes



(b) Optimization convergence history

Figure 4.3: Subsonic single-point optimization, without second-difference dissipation

| Version | $C_l^*$ | $C_d^*$ | $C_l$ | $C_d$ |
|---|---|---|---|---|
| Old, with dissipation | 0.4728 | 0.0112 | 0.4726 | 0.01123 |
| New, with dissipation | - | - | 0.4728 | 0.0112309 |
| Old, no dissipation | 0.4728 | 0.0112 | 0.4728 | 0.0110896 |
| Old, no dissipation | 0.4728 | 0.01 | 0.4623 | 0.0109583 |
| New, no dissipation | - | - | 0.4728 | 0.0108270 |

Table 4.3: Comparison of lift and drag coefficients found with the new algorithm and the old algorithm

required for the cases without second-difference dissipation are shown in Table 4.4. We can see that the case run with the new algorithm takes significantly longer than either case run with the previous algorithm. This is due to the increased time required for the flow solver to converge, as the gradient calculation requires approximately the same amount of time in all three trials. However, to achieve true lift-constrained drag minimization with the previous algorithm, several more tests varying the target lift and drag coefficients would likely need to be run, so the total time required for lift-constrained drag-minimization is less with the new algorithm.

| Version | $C_l^*$ | $C_d^*$ | Time (s) | Flow Solves and Gradient Evaluations |
|---|---|---|---|---|
| Old, no dissipation | 0.4728 | 0.0112 | 1160.27 | 13 |
| Old, no dissipation | 0.4728 | 0.01 | 3458.24 | 58 |
| New, no dissipation | - | - | 6597.52 | 42 |

Table 4.4: Comparison of time to convergence with the new algorithm and the old algorithm

### 4.3.2 Single-Point Transonic Airfoil Design

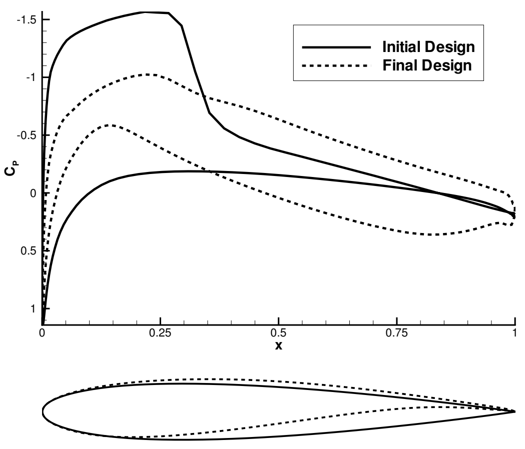
The second case examined is a transonic lift-constrained drag minimization case. The same case was also examined by Nemec [19] and by Nemec, Zingg and Pulliam [22, 23]. They have varied the weights in the two competing parts of the previously existing lift-constrained drag minimization objective function to obtain a Pareto front showing the trade-off between satisfying the lift constraint and minimizing drag. We will compare results found in the Pareto front to results for lift-constrained drag minimization found with the new algorithm.

The initial airfoil is the NACA 0012 airfoil. The grid consists of $201 \times 45$ nodes. The flow conditions are $Ma_\infty = 0.7$, $C_l^* = 0.55$, and $Re = 9 \times 10^6$. The airfoil is described by 15 B-spline control points, 10 of which are used as design variables. There are three fixed thickness constraints, as shown in Table 4.5, together with final thicknesses achieved. See Appendix B for input file.

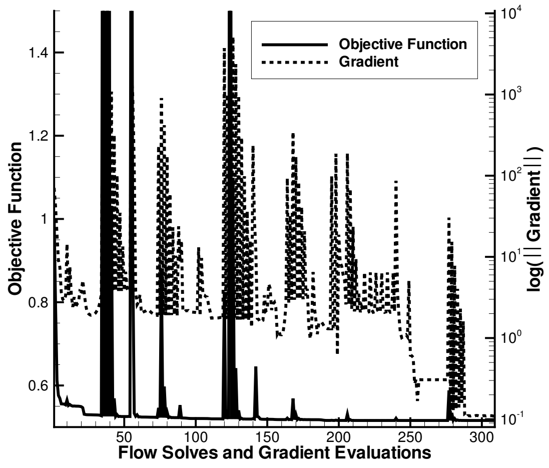
| Constraint Number | 1 | 2 | 3 |
|---|---|---|---|
| Location (% c) | 25 | 92 | 99 |
| Desired Thickness (% c) | 11.8 | 0.9 | 0.2 |
| Final Thickness (% c) | 11.780879 | 0.89995105 | 0.19999537 |

Table 4.5: Thickness constraints for transonic single-point optimization

Figure 4.4(a) shows the initial and final pressure distributions and airfoil shapes. The convergence history of the objective function and gradient are shown in Figure 4.4(b). The optimization was restarted three times from the steepest descent method when the objective function stalled during a line search. It is important to note that although over 300 design iterations were required for the gradient to decrease by over two orders of



(a) Pressure distributions and airfoil shapes



(b) Optimization convergence history

Figure 4.4: Transonic single-point optimization

magnitude, the objective function is within 2% of its final value after 35 design iterations.

The final drag coefficient is 48.4% less than the initial drag coefficient. A comparison to results found by Nemec, Zingg and Pulliam [22, 23] with the previous objective function for lift-constrained drag minimization is shown in Table 4.6. This demonstrates that the drag coefficient achieved with the new algorithm is the same as the drag coefficient achieved with the previous algorithm when the weight on the lift component of the objective function is high. The desired lift coefficient is also reached with the new algorithm, whereas the results found using the previous algorithm simply approach the desired lift coefficient.

| Case | $\omega_L$ | $\omega_D$ | $C_l$ | $C_d$ |
|---|---|---|---|---|
| New Case | - | - | 0.5500 | 0.01211 |
| Nemec & Zingg Case 1 | 0.99 | 0.01 | 0.5494 | 0.01211 |
| Nemec & Zingg Case 2 | 0.90 | 0.1 | 0.5439 | 0.01203 |

Table 4.6: Comparison of results for transonic single-point optimization

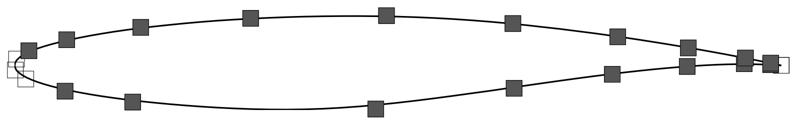


Figure 4.5: Control points and design variables (shaded) for the RAE 2822 airfoil

### 4.3.3 Four-Point Optimization Case

This case examines the effect of optimization at varying Mach numbers on the drag profile of the airfoil. This case has been previously examined by Nemec [19], and by Nemec, Zingg and Pulliam [22, 23]. It is based on one of the cases studied by Drela [4]. The initial airfoil is the RAE2822 airfoil. The airfoil shape is described by 25 B-spline control points, 19 of which are used as design variables, as shown in Figure 4.5. The grid has $257 \times 57$ nodes. The target lift at all operating points is 0.733. The Reynolds number is $2.7 \times 10^6$. Three fixed thickness constraints are imposed: $t/c \geq 0.1204$ at $x/c = 0.35$, $t/c \geq 0.005$ at $x/c = 0.96$, and $t/c \geq 0.0012$ at $x/c = 0.99$. The first optimization is a single-point optimization, performed at $\mathrm{Ma}_\infty = 0.74$. See Appendix B for input files.

Figure 4.6 shows the convergence of the objective function and gradient. Figure 4.7(a) shows the initial and final pressure distributions at the design Mach number, and airfoil shapes. The change in the performance of the airfoil over a range of Mach numbers can be seen in Figure 4.7(b). The optimization results in a 36.4% decrease in drag, from 0.02247 to 0.01423, at the design point. The same reduction in drag was achieved by Nemec [19] when performing the same optimization with the previously existing algorithm. The resulting airfoil is not a practical design, however, as for Mach numbers lower than the design point, there is an increase in drag. Because of this, multipoint optimizations are attempted. The multipoint optimizations are performed using a weighted objective function

$$\mathcal{J}_m = \sum_{i=0}^{N_m} w_i \mathcal{J}_i \tag{4.1}$$

where $\mathcal{J}_m$ is the objective function for the multipoint optimization, $N_m$ is the number of design points, $w_i$ is the user assigned weight for design point $i$, and $\mathcal{J}_i$ is the objective function for design point $i$. A two-point optimization is performed with design Mach numbers 0.68 and 0.74, with weights 1.0 and 2.0, respectively. Figure 4.8 shows the convergence of the objective function and gradient for the two-point optimization. Note

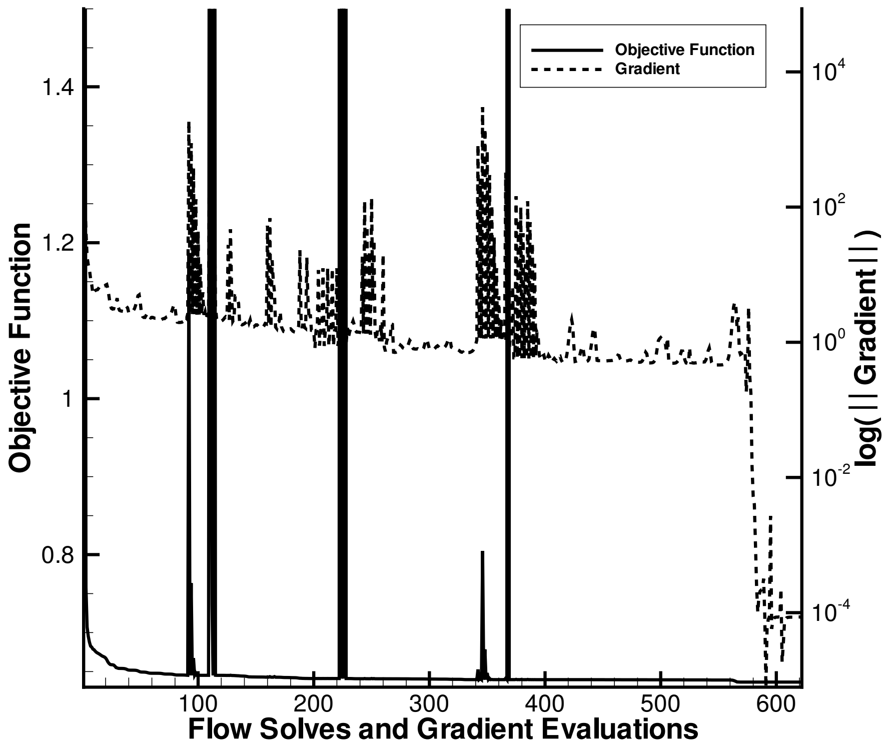
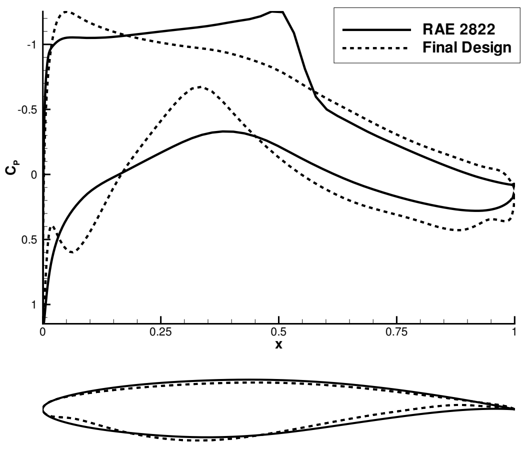
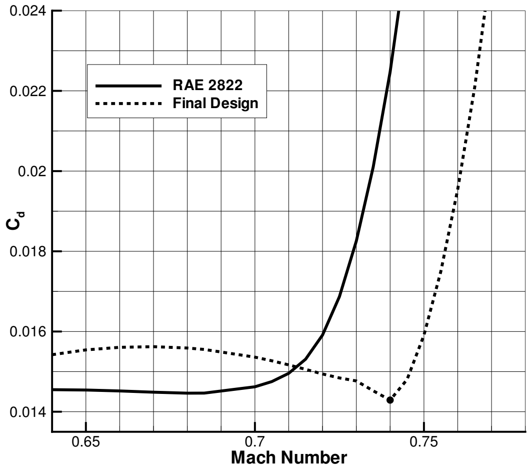


Figure 4.6: Objective function and gradient convergence histories for single-point design



(a) Pressure distributions and airfoil shapes ($\mathrm{Ma}_\infty = 0.74$)



(b) Drag coefficient at $C_l = 0.733$

Figure 4.7: Single-point optimization

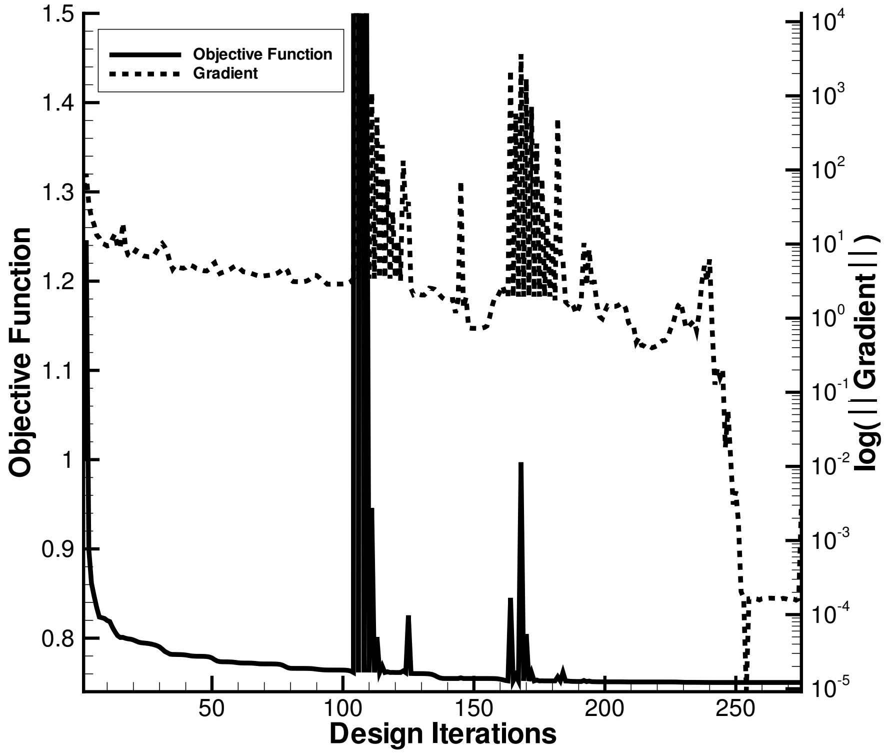


Figure 4.8: Objective function and gradient convergence histories for two-point design

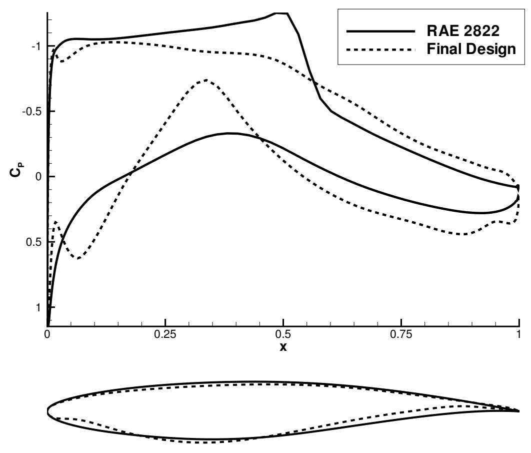
that for each design iteration, two flow solves and gradient evaluations are performed. Figure 4.9(a) shows the initial and final pressure distributions at Mach number 0.74, and the initial and final airfoil shapes. Figure 4.9(b) shows the performance of the airfoil over a range of Mach numbers. This shows significant improvement at drag below the design Mach number, without much increase in drag at the design Mach number; the design drag coefficient is reduced by 35.9%, which is not significantly less than the decrease achieved in the single-point optimization.

A four-point optimization is also performed, with design Mach numbers 0.68, 0.71, 0.74, and 0.76, with weights 1.0, 1.0, 2.0, and 3.0 respectively. The resulting airfoil and pressure distributions at $\mathrm{Ma}_\infty = 0.74$ are shown in Figure 4.10(a). The performance of the airfoil over a range of Mach numbers is shown in Figure 4.10(b). The drag coefficient in this case has been decreased by 33.7% from the drag coefficient for the RAE 2822 airfoil at $\mathrm{Ma}_\infty = 0.74$. This is a slightly smaller reduction in drag than was seen by Nemec [19], but the difference is due to the fact that objective functions here have been normalized, and so the initial function value does not affect the effective weight given to each design point. When the normalization is not used, the higher initial drag coefficients at $\mathrm{Ma}_\infty = 0.74$ and $\mathrm{Ma}_\infty = 0.76$ artificially increase the weight given to these points, resulting in a greater decrease in drag. The drag coefficient found here for $\mathrm{Ma}_\infty = 0.74$ is higher than the drag coefficients found for both the single- and two-point optimizations, but the drag coefficient diverges at $\mathrm{Ma}_\infty = 0.76$ rather than $\mathrm{Ma}_\infty = 0.74$. This is desirable if the airfoil will be operating at $\mathrm{Ma}_\infty = 0.74$, as slightly off-design conditions will not lead to a dramatic increase in drag.

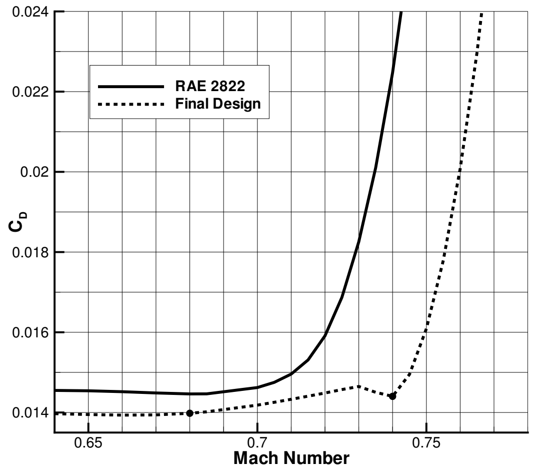
### 4.3.4 Two-Point Pareto Front

This case examines the trade-off in trying to minimize drag at two different design points, with a fixed lift. The initial airfoil is the RAE 2822 airfoil. There are 15 B-spline control points, 10 of which are used as design variables. The grid has $257 \times 57$ nodes. Three thickness constraints are imposed, as shown in Table 4.7. The Reynolds number at both operating points is $9 \times 10^6$, and the two Mach numbers being studied are 0.68 and 0.75. The target lift coefficient at both operating points is 0.715. See Appendix B for the input file. The objective function being evaluated is a weighted sum of the objective functions at each of the design points:

$$\mathcal{J} = \omega \mathcal{J}_{\mathrm{Ma}_\infty = 0.75} + (1-\omega)\mathcal{J}_{\mathrm{Ma}_\infty = 0.68} \tag{4.2}$$

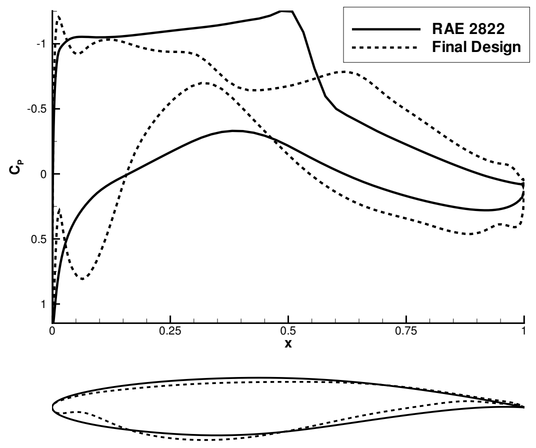


(a) Pressure distributions and airfoil shapes ($\mathrm{Ma}_\infty = 0.74$)

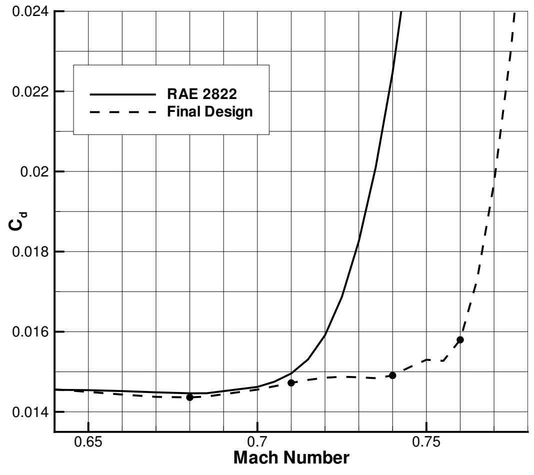


(b) Drag coefficient at $C_l = 0.733$

Figure 4.9: Two-point optimization



(a) Pressure distributions and airfoil shapes ($\mathrm{Ma}_\infty = 0.74$)



(b) Drag coefficient at $C_l = 0.733$

Figure 4.10: Four-point optimization

This case has been previously studied by Zingg and Elias [35], and results are compared.

| Constraint Number | 1 | 2 | 3 |
|---|---|---|---|
| Location (% c) | 1 | 25 | 99 |
| Desired Thickness (% c) | 2.53 | 12.1 | 0.2 |

Table 4.7: Thickness constraints for Pareto front case

Figure 4.11 shows the trade-off between minimizing drag at the two design points, given different weights, and the value of the drag coefficients at each point is shown in Table 4.8. Comparing to results found by Zingg and Elias, the new code allows a greater decrease in the drag coefficient when the Mach number is 0.68, and a smaller decrease in the drag coefficient for $\text{Ma}_\infty = 0.75$. This is because of the normalization of the objective function. For the previous code, the initial drag coefficient for the higher Mach number creates a greater effective weight for the minimization of this drag. Similar results could be found by increasing the weight on the design point with Mach number 0.75 if this is a more frequently used operating point, or if lower drag is required for some reason.

| $\omega$ | $C_d$ at $\text{Ma}_\infty = 0.75$ | $C_d$ at $\text{Ma}_\infty = 0.68$ |
|---|---|---|
| 0.10 | 0.0157727 | 0.0135602 |
| 0.30 | 0.0143473 | 0.0137382 |
| 0.40 | 0.0142410 | 0.0137385 |
| 0.50 | 0.0142096 | 0.0137725 |
| 0.60 | 0.0141918 | 0.0138080 |
| 0.70 | 0.0141581 | 0.0138224 |
| 0.75 | 0.0141443 | 0.0138442 |
| 0.80 | 0.0141295 | 0.0138755 |
| 0.85 | 0.0141250 | 0.0139199 |
| 0.90 | 0.0140909 | 0.0140275 |

Table 4.8: Drag coefficients for different weights

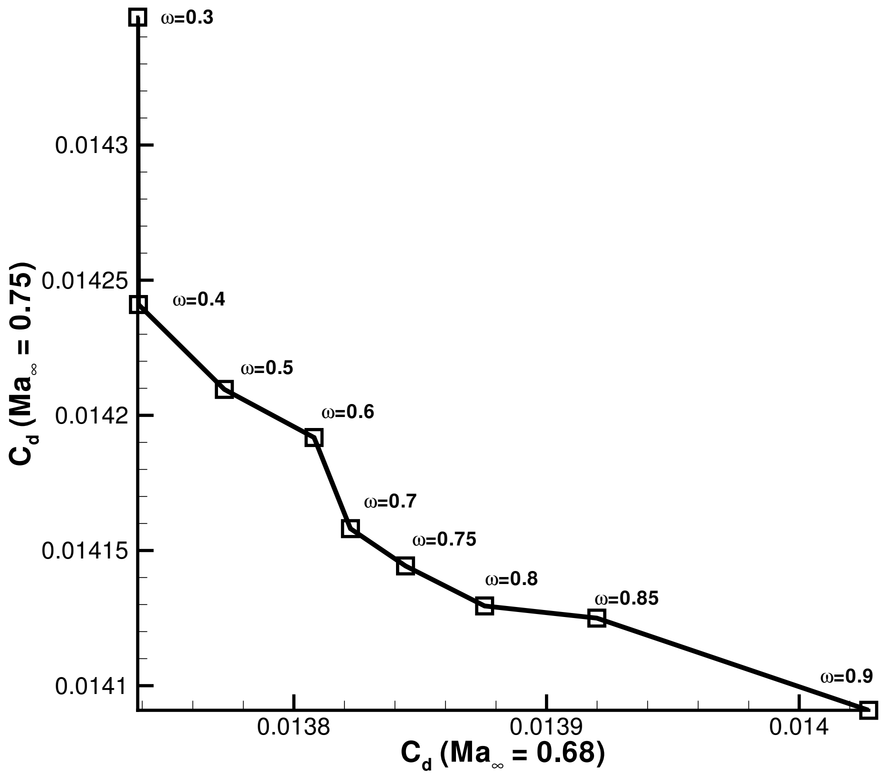


Figure 4.11: Pareto front for two-point optimization

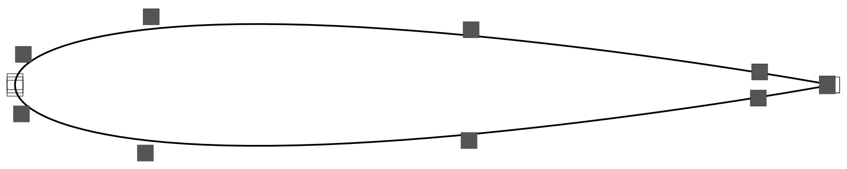


Figure 4.12: Control points and design variables (shaded) for the NACA 0015 airfoil

### 4.3.5 Eighteen-Point Optimization Case

The objective of this optimization case is to develop a single airfoil that satisfies the list of optimization criteria in Table 4.9, as well as having a maximum thickness of at least 14% chord. The first four operating points (A-D) in Table 4.9 represent cruise conditions at different altitudes with different aircraft weights. The objective for these design conditions is to minimize drag while maintaining the specified lift coefficient. The next four operating points (E-H) represent long-range cruise, at various altitudes and aircraft weights. The goal at these operating conditions is to minimized the product of the Mach number and the lift-to-drag ratio, but since the Mach number and lift coefficient are fixed, this corresponds to minimizing drag. The next eight operating points (I-P) represent dive conditions. These are constraints on the airfoil, and require shock strengths to be modest, namely with the upstream Mach number at all shocks less than 1.35. The final two operating points (Q and R) correspond to low-speed operations. The objective for these operating conditions is to ensure that the maximum lift coefficient is at least 1.6. To ensure the maximum thickness is at least 14%, a floating thickness constraint of 14.2% thickness is applied. In addition to the floating thickness constraint, two thickness constraints are imposed on the trailing edge of the airfoil to prevent crossover. The airfoil must be at least 1% thick at $x = 95\%$ of the chord length, and at least 0.2% thick at $x = 99\%$ of the chord length. The initial airfoil selected for this optimization is the NACA 0015 airfoil, since it does not violate the floating thickness constraint. The grid contains $289 \times 65$ nodes. The airfoil is described by 15 B-spline control points, with 10 as design variables, as shown in Figure 4.12.

Work has been performed on this optimization using the previously existing algorithm to achieve lift-constrained drag minimization. For this case, the new algorithm will be used for design points A to P, but the previous objective function will be used for points

| Point | Re | $Ma_\infty$ | Objective | $C_l$ Constraint |
|---|---|---|---|---|
| A | 27.32M | 0.72 | Minimize Drag | 0.17 |
| B | 27.32M | 0.72 | Minimize Drag | 0.28 |
| C | 18.57M | 0.72 | Minimize Drag | 0.27 |
| D | 18.57M | 0.72 | Minimize Drag | 0.45 |
| E | 24.22M | 0.64 | Minimize Drag | 0.21 |
| F | 24.22M | 0.64 | Minimize Drag | 0.36 |
| G | 16.46M | 0.64 | Minimize Drag | 0.34 |
| H | 16.46M | 0.64 | Minimize Drag | 0.57 |
| I | 28.88M | 0.76 | Minimize Shock (Ma $<$ 1.35) | 0.28 |
| J | 28.88M | 0.76 | Minimize Shock (Ma $<$ 1.35) | 0.15 |
| K | 28.88M | 0.76 | Minimize Shock (Ma $<$ 1.35) | 0.46 |
| L | 28.88M | 0.76 | Minimize Shock (Ma $<$ 1.35) | 0.25 |
| M | 19.62M | 0.76 | Minimize Shock (Ma $<$ 1.35) | 0.45 |
| N | 19.62M | 0.76 | Minimize Shock (Ma $<$ 1.35) | 0.24 |
| O | 19.62M | 0.76 | Minimize Shock (Ma $<$ 1.35) | 0.74 |
| P | 19.62M | 0.76 | Minimize Shock (Ma $<$ 1.35) | 0.40 |
| Q | 11.8M | 0.16 | Maximize Lift | - |
| R | 15.0M | 0.20 | Maximize Lift | - |

Table 4.9: Optimization points

Q and R as no target lift has been given. Using the previous algorithm, several steps were required to reach a final optimized airfoil. The operation began by optimizing for points A to H, and on subsequent iterations the various parameters in the objective function were varied, and additional design points were added. In total, seventeen optimization operations were required to go from the initial airfoil to the final optimized airfoil.

In testing the new algorithm, the goal is to reach the final optimized airfoil in as few optimization steps as possible. Ideally, we would like the user to simply enter the design points they have selected and run the optimization algorithm, but in actuality it is more complicated. For the initial attempt at optimization, each point was given the same weight. This optimization was not successful, but from the results of this run, some design points were found to have easily reduced objective functions, while others had objective functions that increased. To ensure that all points reached their objectives, the weights given to the more difficult to optimize points were increased, while the other points continued to be given a weight of one.

Changes were also made in the course of the optimization process to design points Q and R. The objective for these two points is to maximize the lift, and no constraint is given for the drag. Because of this, the following objective function is used:

$$\mathcal{J} = \left(1 - \frac{C_l}{C_l^*}\right)^2 + t.c. \tag{4.3}$$

where $t.c.$ is the penalty due to the thickness constraints. The initial airfoil gives lift coefficients of 0.83 and 0.84 at design points Q and R respectively. In the first attempt at optimization, the target lift for these points was set to 1.2, and this was increased in subsequent attempts.

In three optimization stages, by varying weights and parameters for design points Q and R (see Appendix C), an optimized airfoil was achieved. The constraints given in points I to R are satisfied, and the drag at the eight design points is significantly reduced (see Table 4.10). The final thickness constraint values are shown in Table 4.11, with the location and thickness of the floating thickness constraint given. At some of the off-design points, treated as constraints, the performance exceeds the requirements. This may indicate that the weight on the operating point is higher than it need be, or it may indicate that the point is not critical, namely that the constraint would be satisfied even if it was not included in the optimization.

To consider the possible benefits of adaptive airfoils, single-point optimizations were also performed at the cruise operating conditions in Table 4.9. The grids, design points and thickness constraints were all as given for the eighteen-point optimization problem. This resulted in significantly different airfoil shapes, shown in Figures 4.13 and 4.14. To constrain the shape of the airfoil further, single-point optimizations were run for the cruise operating conditions starting with the multipoint airfoil as the initial airfoil, and with the maximum thickness constraint set as a fixed thickness constraint to ensure the maximum thickness was at the same location as the maximum thickness of the multipoint airfoil. The new thickness constraint requires $y/c = 0.142$ at $x/c = 0.36666667$. The airfoil shapes achieved in these optimizations are shown in Figures 4.15 and 4.16. Table 4.12 gives the resulting airfoil performance for both cases.

The airfoils found with a fixed maximum thickness constraint have higher drag coefficients than those found with the floating maximum thickness constraint, but significantly

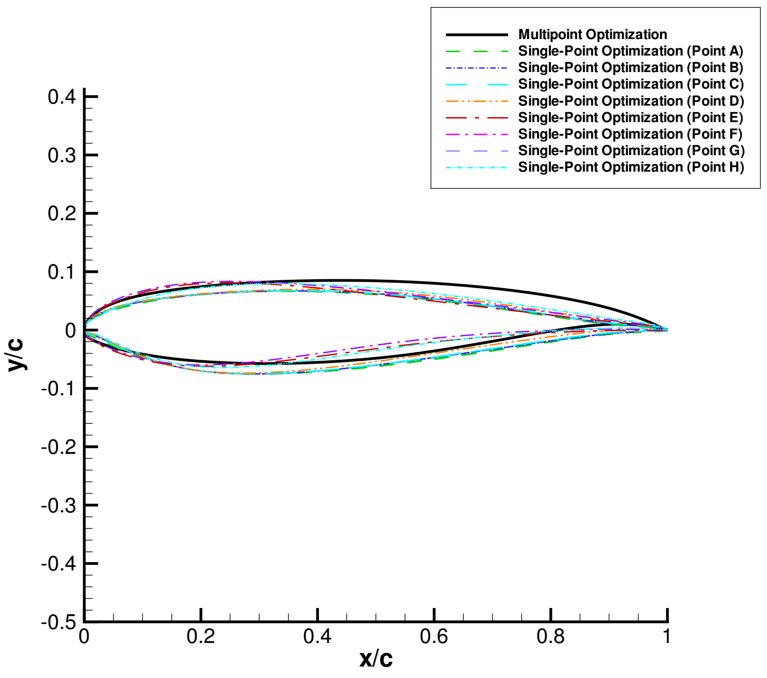


Figure 4.13: Airfoil sections found using floating maximum thickness constraint

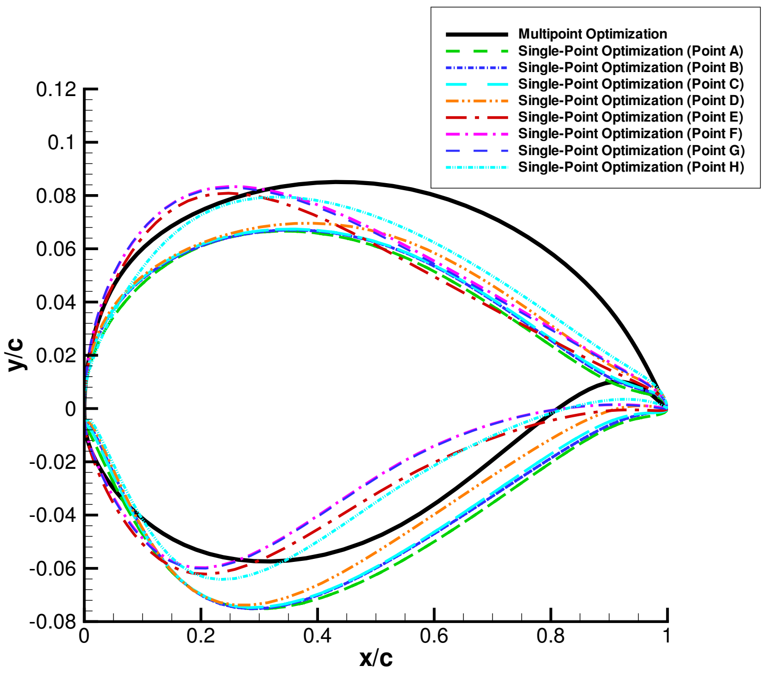


Figure 4.14: Expanded view of airfoil sections found using floating maximum thickness constraint

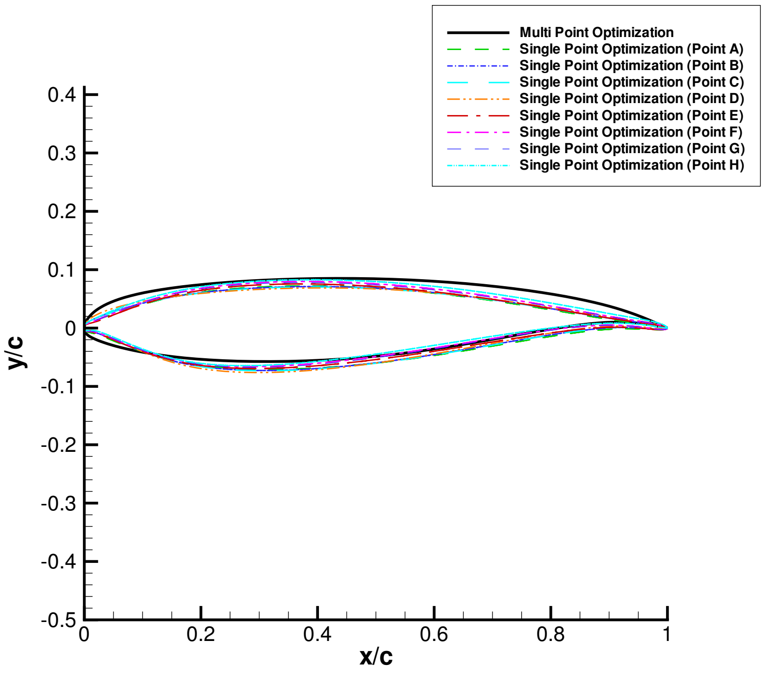


Figure 4.15: Airfoil sections found using fixed maximum thickness constraint

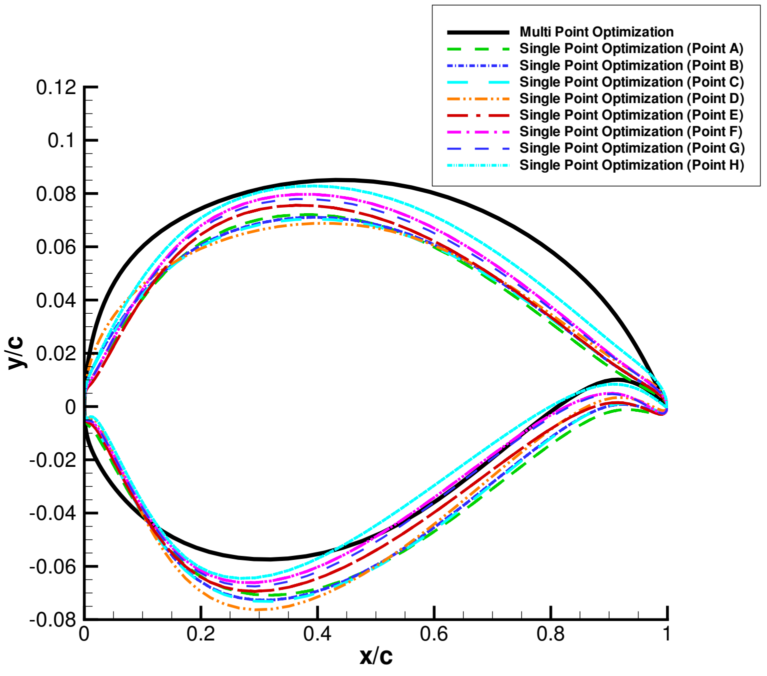


Figure 4.16: Expanded view of airfoil sections found using fixed maximum thickness constraint

| Point | $C_l$ | $C_d$ | Maximum Mach Number | $C_{l,\max}$ |
|---|---|---|---|---|
| A | 0.17 | 0.0125 | - | - |
| B | 0.28 | 0.0126 | - | - |
| C | 0.27 | 0.0128 | - | - |
| D | 0.45 | 0.0134 | - | - |
| E | 0.21 | 0.0122 | - | - |
| F | 0.36 | 0.0125 | - | - |
| G | 0.34 | 0.0126 | - | - |
| H | 0.57 | 0.0136 | - | - |
| I | 0.28 | 0.0139 | 1.19 | - |
| J | 0.15 | 0.0143 | 1.26 | - |
| K | 0.46 | 0.0171 | 1.28 | - |
| L | 0.25 | 0.0138 | 1.17 | - |
| M | 0.45 | 0.0170 | 1.27 | - |
| N | 0.24 | 0.0139 | 1.17 | - |
| O | 0.74 | 0.0562 | 1.34 | - |
| P | 0.40 | 0.0158 | 1.25 | - |
| Q | - | - | - | 1.77 |
| R | - | - | - | 1.78 |

Table 4.10: Design performance of the multipoint airfoil

smaller shape changes are required. The fixed maximum thickness is more practical from a design point of view, as a spar could be placed at the point of maximum thickness. The airfoils with the floating thickness constraints show that larger drag reductions are attainable, if it is possible to generate these airfoil shapes. The second set of airfoils with a fixed thickness constraint particularly demonstrate that a morphing airfoil with relatively small shape changes will allow a significant decrease in drag at cruise speeds.

| Constraint number | 1 | 2 | 3 |
|---|---|---|---|
| Location (% c) | 0.36666667 | 0.95 | 0.99 |
| Desired thickness (%c) | 0.142 | 0.01 | 0.002 |
| Final thickness (%c) | 0.14126 | 0.012165 | 0.0027402 |

Table 4.11: Design performance of the multipoint airfoil

| Point | $C_d$ (multipoint) | $C_d$ (single-point) floating t.c. | % Reduction | $C_d$ (single-point) fixed t.c. | % Reduction |
|---|---|---|---|---|---|
| A | 0.0125 | 0.01061 | 14.8 | 0.01103 | 11.4 |
| B | 0.0126 | 0.01089 | 13.6 | 0.01123 | 11.0 |
| C | 0.0128 | 0.01094 | 14.3 | 0.01135 | 11.2 |
| D | 0.0134 | 0.01169 | 13.0 | 0.01199 | 10.8 |
| E | 0.0122 | 0.01067 | 12.4 | 0.01107 | 9.1 |
| F | 0.0125 | 0.01108 | 11.1 | 0.01140 | 8.6 |
| G | 0.0126 | 0.01107 | 11.9 | 0.01145 | 8.9 |
| H | 0.0136 | 0.01217 | 10.9 | 0.01245 | 8.4 |

Table 4.12: Comparison of drag coefficients of the single-point designs with those of the baseline multipoint design

# Chapter 5

# Conclusions and Recommendations

An improved algorithm has been developed for lift-constrained drag minimization in Optima2D. Both stages of the flow solver have been modified to allow the angle of attack to change during flow solution to ensure that the desired lift coefficient is achieved. In the approximate factorization stage of the flow solver, angle of attack relaxation has been used. This takes the difference between the current coefficient of lift and the desired coefficient of lift and changes the angle of attack by some multiple of this amount. The Newton-Krylov flow solver has been modified to include a new residual equation that is the difference between the current coefficient of lift and the target coefficient of lift. This residual equation has been driven to zero by adding the angle of attack to the vector of flow variables. The new residual equation has also been included in the calculation of the gradient using the adjoint method.

The capability to combine different optimization goals in one multipoint optimization has also been added to Optima2D. To allow weights to be assigned to different design points more easily, and to allow the weight given to thickness constraints to have the same meaning for different design points, objective functions have been normalized by their initial value before thickness constraint penalties have been added.

The new algorithm was compared to the previously existing algorithm used for lift-constrained drag minimization and the following conclusions were reached:

1. The flow solution algorithm ensures that the desired lift coefficient is reached. This requires 20 to 50% more time than for the flow solver to converge without variable angle of attack.

2. The new flow solver is useful when performing lift-constrained Mach number sweeps.

These were previously achieved by performing an optimization with the angle of attack as the only design variable, but using the variable angle of attack flow solver requires approximately 70% less time.

3. The new algorithm consistently performs lift-constrained drag minimization, as compared to the previous algorithm, which minimized a compound objective function that could approximate lift-constrained drag minimization. As such, lengthy and expensive tests to tune the parameters in the previous objective function are not required.

4. The new optimization algorithm successfully converges to similar results as the previous optimization algorithm reaches when a high weight is given to the lift portion of the objective function.

5. The total time required for the optimization to converge with the new algorithm is greater than required for the previously existing algorithm, due to the increased time required for each flow solution.

Future work includes the following:

- The angle of attack is consistently in degrees in Optima2D. Converting this to radians would be beneficial, and may eliminate the need to scale the angle of attack when it is used as a variable in the modified flow solver.

- This method currently does not include the circulation correction. Allowing the circulation correction to be used may improve the performance of the algorithm.

- Extend this algorithm into three dimensions. One challenge in doing this may be the storage of the new equation in the flow Jacobian, as the adjoint equation is solved in parallel.

- If multipoint cases with a large number of design points will be studied in the future, parallelizing the code to allow the flow solutions and gradient calculations for different operating points to be performed on different processors will allow the cases to be run in significantly less time.

- Extending work by Zingg and Elias [35] to automate weight selection in multi-point cases could also decrease the time required for these cases. It will remain difficult, however, to automate weight selection for constraints on local maximum Mach number, as this value is not determined until after the optimization is finished. Including the calculation of local Mach number in Optima2D and using the maximum local Mach number as the objective function to minimize may be useful if this will be a common goal.

# References

[1] W. K. Anderson and D. L. Bonhaus. Airfoil design on unstructured grids for turbulent flows. *AIAA Journal*, 37(2):185–191, February 1999.

[2] W. K. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. AIAA Report 97-0643, AIAA, 1997.

[3] A. Dadone and B. Grossman. Fast convergence of viscous airfoil design problems. *AIAAJ*, 40(10):1997–2005, 1997.

[4] M. Drela. Pros & cons of airfoil optimization. In D. A. Caughey and M. M. Hafez, editors, *Frontiers of Computational Fluid Dynamics 1998*, pages 363–381, Singapore, 1998. World Scientific.

[5] J. Driver. Optimal aerodynamic shape design with transition prediction. Master's thesis, University of Toronto, 2005.

[6] J. Driver and D. W. Zingg. Optimized natural-laminar-flow airfoils. AIAA Report 2006-247, AIAA, 2006.

[7] J. Elliott. *Aerodynamic Optimization Based on the Euler and Navier-Stokes Equations Using Unstructured Grids*. PhD thesis, Massachusetts Institute of Technology, 1998.

[8] J. Elliott and J. Peraire. Constrained, multipoint shape optimisation for complex 3D configurations. *The Aeronautical Journal*, 102(1017):365–376, 1998.

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

[10] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, London, 1981.

[11] A. Iollo and M. D. Salas. Optimum transonic airfoils based on the Euler equations. *Computers & Fluids*, 28:653–674, 1999.

[12] A. Jameson. Solution of the Euler equations by a multigrid method. *Applied Mathematics and Computation*, 13:327–356, 1983.

[13] A. Jameson. A perspective on computational algorithms for aerodynamic analysis and design. *Progress in Aerospace Sciences*, 37(2):197–243, 2001.

[14] A. Jameson. Efficient aerodynamic shape optimization. AIAA Report 2004-4369, AIAA, 2004.

[15] H.-J. Kim, S. Obayashi, and K. Nakahashi. Flap-deflection optimization for transonic cruise performance improvement of supersonic transport wing. *Journal of Aircraft*, 38(4):709–717, 2001.

[16] H.-J. Kim, D. Sasaki, S. Obayashi, and K. Nakahashi. Aerodynamic optimization of supersonic transport wing using unstructured adjoint method. *AIAA Journal*, 39(6):1011–1020, June 2001.

[17] B. Mohammadi. A new optimal shape design procedure for inviscid and viscous turbulent flows. *International Journal for Numerical Methods in Fluids*, 25:183–203, 1997.

[18] T. E. Nelson and D. W. Zingg. Fifty years of aerodynamics: Successes, challenges, and opportunities. *CAS Journal*, 50(1):61–84, March 2004.

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

[20] M. Nemec and D. W. Zingg. Towards efficient aerodynamic shape optimization based on the Navier-Stokes equations. AIAA Report 2001-2532, AIAA, 2001.

[21] M. Nemec and D. W. Zingg. Newton-Krylov algorithm for aerodynamic design using the Navier-Stokes equations. *AIAA Journal*, 40(6):1146–1154, June 2002.

[22] M. Nemec, D. W. Zingg, and T. Pulliam. Multi-point and multi-objective aerodynamic shape optimization. AIAA Report 2003-5548, AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, June 2003.

[23] M. Nemec, D. W. Zingg, and T. H. Pulliam. Multipoint and multi-objective aerodynamic shape optimization. *AIAA Journal*, 42(6):1057–1065, June 2004.

[24] E. Nielsen. *Aerodynamic Design Sensitivities on and Unstructured Mesh Using the Navier-Stokes Equations and a Discrete Adjoint Formulation.* PhD thesis, Virginia Polytechnic Institute and State University, 1998.

[25] E. J. Nielsen and W. K. Anderson. Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations. Paper 98-4809, AIAA, September 1998.

[26] A. Pueyo. *An Efficient Newton-Krylov Method for the Euler and Navier-Stokes Equations.* PhD thesis, University of Toronto, 1998.

[27] A. Pueyo and D. W. Zingg. Efficient Newton-Krylov solver for aerodynamic computations. *AIAA Journal*, 36(11):1991–1997, November 1998.

[28] T. H. Pulliam. Efficient solution methods for the Navier-Stokes equations. In *Lecture Notes for the Von Karman Institute for Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings, January 20–24, 1986.* Von Karman Institute for Fluid Dynamics, Brussels, Belgium, 1986.

[29] T. H. Pulliam, M. Nemec, T. Holst, and D. W. Zingg. Comparison of evolutionary (genetic) algorithm and adjoint methods for multi-objective viscous airfoil optimizations. AIAA Report 2003-0298, AIAA, 2003.

[30] J. J. Reuther, A. Jameson, J. J. Alonso, M. J. Rimlinger, and D. Saunders. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1. *Journal of Aircraft*, 26(1):51–60, 1999.

[31] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[32] B. I. Soemarwoto and Th. E. Labrujère. Airfoil design and optimization methods: Recent progress at NLR. *International Journal for Numerical Methods in Fluids*, 25:183–203, 1997.

[33] D. Tse and L. Chan. Transonic airfoil design optimization using soft computing methods. *CAS Journal*, 46(2):65–73, June 2000.

[34] Z. Zhang and K.-Y. Lum. Airfoil optimization design of drag minimization with lift constraint using adjoint equation method. AIAA Report 2006-55, AIAA, January 2006.

[35] D. W. Zingg and S. Elias. On aerodynamic optimization under a range of operating conditions. AIAA Report 2006-1053, AIAA, 2006.

[36] D. W. Zingg, T. M. Leung, L. Diosady, A. H. Truong, and S. Elias. Improvements to a Newton-Krylov adjoint algorithm for aerodynamic optimization. AIAA Report 2005-4857, AIAA, 2005.

# Appendix A

# New Analytical Derivatives

Three new derivatives were required in the formation of the new flow Jacobian $\frac{\partial \mathcal{R}}{\partial \mathcal{Q}}$. These are the derivative of the coefficient of lift with respect to the angle of attack, the derivative of the coefficient of lift with respect to the flow variables, and the derivative of the previously existing residual equations with respect to angle of attack.

The derivatives of the drag coefficient are included also, as these are used in the calculation of the derivative of the objective function with respect to the flow variables.

## A.1 Partial Derivative of Lift and Drag Coefficients with Respect to Angle of Attack

The coefficient of lift consists of two parts; the first is the vertical component of the pressure force on the airfoil and the second comes from the viscous terms. For both parts, a chord directed force coefficient, $C_C$, and a normal force coefficient, $C_N$, are calculated. These are then resolved into lift and drag coefficients:

$$C_l = C_N \cos(\alpha) - C_C \sin(\alpha) \tag{A.1}$$

$$C_d = C_N \sin(\alpha) + C_C \cos(\alpha) \tag{A.2}$$

The derivative of the coefficient of lift with respect to angle of attack is then

$$\frac{\partial C_l}{\partial \alpha} = -C_N \sin(\alpha) - C_C \cos(\alpha) \tag{A.3}$$

and

$$\frac{\partial C_d}{\partial \alpha} = C_N \cos(\alpha) - C_C \sin(\alpha) \tag{A.4}$$

## A.2 Partial Derivatives of Lift and Drag Coefficients with Respect to Flow Variables

### A.2.1 Force Due to Pressure

To find the derivative of the lift due to pressure with respect to the flow variables, first the derivative of the pressure with respect to the flow variables must be found. Pressure is calculated based on the flow variables as

$$p = (\gamma - 1)\left(e - \frac{1}{2}\frac{(\rho u)^2 + (\rho v)^2}{\rho}\right) \tag{A.5}$$

The partial derivatives of pressure with respect to each of the flow variables are then:

$$\frac{\partial p}{\partial \rho} = 0.5(\gamma - 1)\frac{(\rho u)^2 + (\rho v)^2}{\rho^2} \tag{A.6}$$

$$\frac{\partial p}{\partial (\rho u)} = -(\gamma - 1)\frac{\rho u}{\rho} \tag{A.7}$$

$$\frac{\partial p}{\partial (\rho v)} = -(\gamma - 1)\frac{\rho v}{\rho} \tag{A.8}$$

$$\frac{\partial p}{\partial e} = (\gamma - 1) \tag{A.9}$$

The lift coefficient is calculated from the normal and chord directed force components. When we substitute the derivatives for the pressure, we can develop the derivatives of these force components as:

$$\frac{\partial C_N}{\partial q_i} = -\frac{\frac{\partial p}{\partial q_i}\left[(x(j,1) - x(j-1,1)) + (x(j+1,1) - x(j,1))\right]}{\mathrm{Ma}_\infty^2} \tag{A.10}$$

$$\frac{\partial C_C}{\partial q_i} = \frac{\frac{\partial p}{\partial q_i}\left[(y(j,1) - y(j-1,1)) + (y(j+1,1) - y(j,1))\right]}{\mathrm{Ma}_\infty^2} \tag{A.11}$$

with $i = 1, 4$, and

$$q_1 = \rho \tag{A.12}$$

$$q_2 = \rho u \tag{A.13}$$

$$q_3 = \rho v \tag{A.14}$$

$$q_4 = e \tag{A.15}$$

and where $x(j,k)$ and $y(j,k)$ are the $x$ and $y$ positions of the node $j,k$.

Then,

$$\frac{\partial C_l}{\partial q_i} = \frac{\partial C_N}{\partial q_i}\cos(\alpha) - \frac{\partial C_C}{\partial q_i}\sin(\alpha) \tag{A.16}$$

and

$$\frac{\partial C_d}{\partial q_i} = \frac{\partial C_N}{\partial q_i}\sin(\alpha) + \frac{\partial C_C}{\partial q_i}\cos(\alpha) \tag{A.17}$$

with $i = 1, 4$ and $q_i$ as defined above.

### A.2.2 Force Due to Friction

A proportion of the lift is generated due to friction in viscous flow. This is caused by the shear forces:

$$C_f = \frac{\tau_w}{0.5\rho_\infty u_\infty^2} \tag{A.18}$$

where

$$\tau_w = \mu\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) \tag{A.19}$$

The coefficient of viscosity is calculated based on the Reynolds number, since

$$\mathrm{Re} = \frac{\rho_\infty u_\infty C}{\mu_\infty} \tag{A.20}$$

where $C$ is the chord length of the airfoil, $\rho_\infty$ is the freestream density, $u_\infty$ is the freestream velocity, and $\mu_\infty$ is the freestream viscosity.

Since the flow variables are stored in curvilinear coordinates, the calculation of $\tau_w$ is expanded using the chain rule to give:

$$\tau_w = \mu_\infty\left[\left(\frac{\partial u}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial u}{\partial \eta}\frac{\partial \eta}{\partial y}\right) - \left(\frac{\partial v}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial v}{\partial \eta}\frac{\partial \eta}{\partial x}\right)\right] \tag{A.21}$$

The partial derivatives of flow velocities with respect to $\xi$ and $\eta$ are calculated using finite differences:

$$\frac{\partial u}{\partial \xi} = \frac{1}{2}\left(\frac{(\rho u)_{j+1,k}}{\rho_{j+1,k}} - \frac{(\rho u)_{j-1,k}}{\rho_{j-1,k}}\right) \tag{A.22}$$

$$\frac{\partial v}{\partial \xi} = \frac{1}{2}\left(\frac{(\rho v)_{j+1,k}}{\rho_{j+1,k}} - \frac{(\rho v)_{j-1,k}}{\rho_{j-1,k}}\right) \tag{A.23}$$

$$\frac{\partial u}{\partial \eta} = -\frac{3}{2}\frac{(\rho u)_{j,k}}{rho_{j,k}} + 2\frac{(\rho u)_{j,k+1}}{\rho_{j,k+1}} - \frac{1}{2}\frac{(\rho u)_{j,k+2}}{\rho_{j,k+2}} \tag{A.24}$$

$$\frac{\partial v}{\partial \eta} = -\frac{3}{2}\frac{(\rho v)_{j,k}}{rho_{j,k}} + 2\frac{(\rho v)_{j,k+1}}{\rho_{j,k+1}} - \frac{1}{2}\frac{(\rho v)_{j,k+2}}{\rho_{j,k+2}} \tag{A.25}$$

When these derivatives have been calculated, the coefficient of friction at each point is calculated, and then the average over two adjacent points in the $\xi$ direction is taken.

$$C_{f,j} = \frac{\tau_{w,j}}{0.5\rho_\infty u_\infty^2} \tag{A.26}$$

$$C_{f,j,av} = \frac{C_{f,j} + C_{f,j-1}}{2} \tag{A.27}$$

The average coefficients of friction are integrated in the $x$- and $y$-directions to give the chord directed and normal forces:

$$C_N = -\sum_{j=jtail1}^{jtail2} C_{f,j,av}(x_j - x_{j-1}) \tag{A.28}$$

$$C_C = \sum_{j=jtail1}^{jtail2} C_{f,j,av}(y_j - y_{j-1}) \tag{A.29}$$

The normal and chord directed forces are then resolved into the vertical and horizontal directions to give lift and drag:

$$C_l = C_N \cos(\alpha) - C_C \sin(\alpha) \tag{A.30}$$

$$C_d = C_N \sin(\alpha) + C_C \cos(\alpha) \tag{A.31}$$

The angle of attack is not directly affected by the flow variables, so we know that

$$\frac{\partial C_l}{\partial q_i} = \frac{\partial C_N}{\partial q_i}\cos(\alpha) - \frac{\partial C_C}{\partial q_i}\sin(\alpha) \tag{A.32}$$

and

$$\frac{\partial C_d}{\partial q_i} = \frac{\partial C_N}{\partial q_i}\sin(\alpha) + \frac{\partial C_C}{\partial q_i}\cos(\alpha) \tag{A.33}$$

with $q_i$ as defined in section A.2.1.

Due to the summation form of the calculations for $C_C$ and $C_N$, the use of finite differences means that all the different derivatives must be taken into account. The derivative of the normal force with respect to $q_i$ is:

$$\frac{\partial C_N}{\partial q_i} = -\frac{1}{\rho_\infty u_\infty^2}\left[\frac{\partial \tau_w|_{j-1,k}}{\partial q_i}(x_{j-1,k} - x_{j-2,k}) + \left(\frac{\partial \tau_w|_{j,k}}{\partial q} + \frac{\partial \tau_w|_{j-1,k}}{\partial q}\right)(x_{j,k} - x_{j-1,k}) + \left(\frac{\partial \tau_w|_{j+1,k}}{\partial q} + \frac{\partial \tau_w|_{j,k}}{\partial q}\right)(x_{j+1} - x_j) + \frac{\partial \tau_w|_{j+1,k}}{\partial q}(x_{j+2,k} - x_{j+1,k})\right] \tag{A.34}$$

and similarly, the derivative of the chord directed force is:

$$\frac{\partial C_C}{\partial q_i} = \frac{1}{\rho_\infty u_\infty^2}\left[\frac{\partial \tau_w|_{j-1,k}}{\partial q_i}(y_{j-1,k} - y_{j-2,k}) + \left(\frac{\partial \tau_w|_{j,k}}{\partial q} + \frac{\partial \tau_w|_{j-1,k}}{\partial q}\right)(y_{j,k} - y_{j-1,k}) + \left(\frac{\partial \tau_w|_{j+1,k}}{\partial q} + \frac{\partial \tau_w|_{j,k}}{\partial q}\right)(y_{j+1} - y_j) + \frac{\partial \tau_w|_{j+1,k}}{\partial q}(y_{j+2,k} - y_{j+1,k})\right] \tag{A.35}$$

where

$$\frac{\partial \tau_w|_{j-1,k}}{\partial q_i} = \mu_\infty \left[\left(\frac{\partial u_\xi|_{j-1,k}}{\partial q_i}\xi_y|_{j-1,k} + \frac{\partial u_\eta|_{j-1,k}}{\partial q_i}\eta_y|_{j-1,k}\right) - \left(\frac{\partial v_\xi|_{j-1,k}}{\partial q_i}\xi_x|_{j-1,k} + \frac{\partial v_\eta|_{j-1,k}}{\partial q_i}\eta_x|_{j-1,k}\right)\right] \tag{A.36}$$

$$\frac{\partial \tau_w|_{j,k}}{\partial q_i} = \mu_\infty \left[\left(\frac{\partial u_\xi|_{j,k}}{\partial q_i}\xi_y|_{j,k} + \frac{\partial u_\eta|_{j,k}}{\partial q_i}\eta_y|_{j,k}\right) - \left(\frac{\partial v_\xi|_{j,k}}{\partial q_i}\xi_x|_{j,k} + \frac{\partial v_\eta|_{j,k}}{\partial q_i}\eta_x|_{j,k}\right)\right] \tag{A.37}$$

$$\frac{\partial \tau_w|_{j+1,k}}{\partial q_i} = \mu_\infty \left[\left(\frac{\partial u_\xi|_{j+1,k}}{\partial q_i}\xi_y|_{j+1,k} + \frac{\partial u_\eta|_{j+1,k}}{\partial q_i}\eta_y|_{j+1,k}\right) - \left(\frac{\partial v_\xi|_{j+1,k}}{\partial q_i}\xi_x|_{j+1,k} + \frac{\partial v_\eta|_{j+1,k}}{\partial q_i}\eta_x|_{j+1,k}\right)\right] \tag{A.38}$$

$$\tag{A.39}$$

These terms are non-zero only for the nodes on the airfoil surface, where $k = 1$ and for two layers of nodes adjacent to these, with $k = 2$ and $k = 3$. At these points, the partial derivatives of the partial derivatives of the velocities with respect to the coordinate system with respect to the flow variables are as follows, with $k = 1$ except where specified:

$$\frac{\partial u_\xi|_{j-1,k}}{\partial \rho_{j,k}} = -\frac{(\rho u)_{j,k}}{2\rho_{j,k}^2} \tag{A.40}$$

$$\frac{\partial u_\xi|_{j-1,k}}{\partial (\rho u)_{j,k}} = \frac{1}{2\rho_{j,k}} \tag{A.41}$$

$$\frac{\partial u_\xi|_{j+1,k}}{\partial \rho_{j,k}} = \frac{(\rho u)_{j,k}}{2\rho_{j,k}^2} \tag{A.42}$$

$$\frac{\partial u_\xi|_{j+1,k}}{\partial (\rho u)_{j,k}} = -\frac{1}{2\rho_{j,k}} \tag{A.43}$$

$$\frac{\partial v_\xi|_{j-1,k}}{\partial \rho_{j,k}} = -\frac{(\rho v)_{j,k}}{2\rho_{j,k}^2} \tag{A.44}$$

$$\frac{\partial v_\xi|_{j-1,k}}{\partial (\rho v)_{j,k}} = \frac{1}{2\rho_{j,k}} \tag{A.45}$$

$$\frac{\partial v_\xi|_{j+1,k}}{\partial \rho_{j,k}} = \frac{(\rho v)_{j,k}}{2\rho_{j,k}^2} \tag{A.46}$$

$$\frac{\partial v_\xi|_{j+1,k}}{\partial (\rho v)_{j,k}} = -\frac{1}{2\rho_{j,k}} \tag{A.47}$$

$$\frac{\partial u_\eta|_{j,k}}{\partial \rho_{j,k}} = -\frac{3(\rho u)_{j,k}}{2\rho_{j,k}^2} \tag{A.48}$$

$$\frac{\partial u_\eta|_{j,k}}{\partial (\rho u)_{j,k}} = -\frac{3}{2\rho_{j,k}} \tag{A.49}$$

$$\frac{\partial u_\eta|_{j,k-1}}{\partial \rho_{j,k}} = \frac{2(\rho u)_{j,k}}{\rho_{j,k}^2} \quad k=2 \tag{A.50}$$

$$\frac{\partial u_\eta|_{j,k-1}}{\partial \rho_{j,k}} = \frac{2}{\rho_{j,k}} \quad k=2 \tag{A.51}$$

$$\frac{\partial u_\eta|_{j,k-2}}{\partial \rho_{j,k}} = -\frac{(\rho u)_{j,k}}{2\rho_{j,k}^2} \quad k=3 \tag{A.52}$$

$$\frac{\partial u_\eta|_{j,k-2}}{\partial (\rho u)_{j,k}} = -\frac{1}{2\rho_{j,k}} \quad k=3 \tag{A.53}$$

$$\frac{\partial v_\eta|_{j,k}}{\partial \rho_{j,k}} = -\frac{3(\rho v)_{j,k}}{2\rho_{j,k}^2} \tag{A.54}$$

$$\frac{\partial v_\eta|_{j,k}}{\partial (\rho v)_{j,k}} = -\frac{3}{2\rho_{j,k}} \tag{A.55}$$

$$\frac{\partial v_\eta|_{j,k-1}}{\partial \rho_{j,k}} = \frac{2(\rho v)_{j,k}}{\rho_{j,k}^2} \quad k=2 \tag{A.56}$$

$$\frac{\partial v_\eta|_{j,k-1}}{\partial \rho_{j,k}} = \frac{2}{\rho_{j,k}} \quad k=2 \tag{A.57}$$

$$\frac{\partial v_\eta|_{j,k-2}}{\partial \rho_{j,k}} = -\frac{(\rho v)_{j,k}}{2\rho_{j,k}^2} \quad k=3 \tag{A.58}$$

$$\frac{\partial v_\eta|_{j,k-2}}{\partial (\rho v)_{j,k}} = -\frac{1}{2\rho_{j,k}} \quad k=3 \tag{A.59}$$

## A.3 Partial Derivatives of Existing Residual Equations with Respect to Angle of Attack

The residual equations that are affected by the angle of attack occur at the boundaries of the computational domain. For viscous flow, only the far-field boundary nodes are affected, but for inviscid flow, the outflow boundary nodes are also affected. Boundaries are as shown in Figure A.1.

### A.3.1 Far-Field Boundary Residual

$$\begin{aligned} R(j,k_{max},1) &= \left((\eta_x u(j,k_{max}) + \eta_y v(j,k_{max})) - 2\frac{a(j,k_{max})}{\gamma-1}\right) \\ &\quad - \left((\eta_x u_\infty + \eta_y v_\infty) - 2\frac{a_\infty}{\gamma-1}\right) \end{aligned} \tag{A.60}$$

$$R(j,k_{max},2) = \left((\eta_x u(j,k_{max}) + \eta_y v(j,k_{max})) + 2\frac{a(j,k_{max})}{\gamma-1}\right)$$

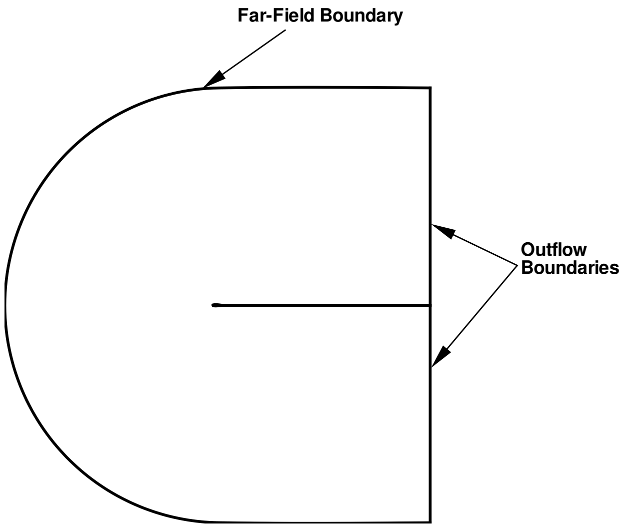


Figure A.1: Boundary locations

$$
\begin{aligned}
&-\left((\eta_x u(j,k_{max}-1)+\eta_y v(j,k_{max}-1)\right. \\
&\left.+2\frac{a(j,k_{max}-1)}{\gamma-1}\right)
\end{aligned}
\tag{A.61}
$$

The residual equations for $n=3$ and $n=4$ are dependent on whether there is inflow or outflow of fluid at the node. For inflow,

$$
R(j,k_{max},3) = \frac{\rho(j,k_{max})^\gamma}{p(j,k_{max})} - \frac{\rho_\infty^\gamma}{p_\infty} \tag{A.62}
$$

$$
R(j,k_{max},4) = (\eta_y u(j,k_{max}) - \eta_x v(j,k_{max})) - (\eta_y u_\infty - \eta_x v_\infty)) \tag{A.63}
$$

For outflow,

$$
R(j,k_{max},3) = \frac{\rho(j,k_{max}-1)^\gamma}{p(j,k_{max}-1)} - \frac{\rho(j,k_{max})^\gamma}{p(j,k_{max})} \tag{A.64}
$$

$$
\begin{aligned}
R(j,k_{max},4) &= (\eta_y u(j,k_{max}-1) - \eta_x v(j,k_{max}-1)) \\
&\quad -(\eta_y u(j,k_{max}) - \eta_x v(j,k_{max}))
\end{aligned}
\tag{A.65}
$$

These residual calculations are affected by the angle of attack since

$$
u_\infty = \mathrm{Ma}_\infty \cos\alpha \tag{A.66}
$$

$$
v_\infty = \mathrm{Ma}_\infty \sin\alpha \tag{A.67}
$$

So, taking the derivative of the residual equations with respect to angle of attack gives

$$\frac{\partial R(j,k_{max},1)}{\partial \alpha} = \eta_x \mathrm{Ma}_\infty \sin\alpha - \eta_y \mathrm{Ma}_\infty \cos\alpha \tag{A.68}$$

$$\frac{\partial R(j,k_{max},2)}{\partial \alpha} = 0 \tag{A.69}$$

$$\frac{\partial R(j,k_{max},3)}{\partial \alpha} = 0 \tag{A.70}$$

$$\frac{\partial R(j,k_{max},4)}{\partial \alpha} = \begin{cases} \eta_y \mathrm{Ma}_\infty \sin\alpha + \eta_x \mathrm{Ma}_\infty \cos\alpha & \text{if inflow} \\ 0 & \text{if outflow} \end{cases} \tag{A.71}$$

## A.3.2 Outflow Boundary Residual

$$\begin{aligned} R(1,k,1) &= \left(\xi_x u(1,k) + \xi_y v(1,k) + 2\frac{a(1,k)}{\gamma-1}\right) \\ &\quad - \left(\xi_x u_\infty + \xi_y v_\infty + 2\frac{a_\infty}{\gamma-1}\right) \end{aligned} \tag{A.72}$$

$$\begin{aligned} R(1,k,2) &= \left(\xi_x u(1,k) + \xi_y v(1,k) - 2\frac{a(1,k)}{\gamma-1}\right) \\ &\quad - \left(\xi_x u(2,k) + \xi_y v(2,k) - 2\frac{a(2,k)}{\gamma-1}\right) \end{aligned} \tag{A.73}$$

If there is inflow at the node,

$$R(1,k,3) = \frac{\rho(1,k)^\gamma}{p(1,k)} - \frac{\rho_\infty^\gamma}{p_\infty} \tag{A.74}$$

$$R(1,k,4) = (-\xi_y u(1,k) + \xi_x v(1,k)) - (-\xi_y u_\infty + \xi_x v_\infty) \tag{A.75}$$

If there is outflow at the node,

$$R(1,k,3) = \frac{\rho(1,k)^\gamma}{p(1,k)} - \frac{\rho(2,k)^\gamma}{p(2,k)} \tag{A.76}$$

$$R(1,k,4) = (-\xi_y u(1,k) + \xi_x v(1,k)) - (-\xi_y u(2,k) + \xi_x v(2,k)) \tag{A.77}$$

Then, knowing that the farfield velocities depend on the angle of attack,

$$\frac{\partial R(1,k,1)}{\partial \alpha} = \xi_x \mathrm{Ma}_\infty \sin\alpha - \xi_y \mathrm{Ma}_\infty \cos\alpha \tag{A.78}$$

$$\frac{\partial R(1,k,2)}{\partial \alpha} = 0 \tag{A.79}$$

$$\frac{\partial R(1,k,3)}{\partial \alpha} = 0 \tag{A.80}$$

$$\frac{\partial R(1,k,4)}{\partial \alpha} = \begin{cases} -\xi_y \mathrm{Ma}_\infty \sin\alpha - \xi_x \mathrm{Ma}_\infty \cos\alpha & \text{if inflow} \\ 0 & \text{if outflow} \end{cases} \tag{A.81}$$

$$\begin{aligned}
R(j_{max},k,1) &= \left(\xi_x u(j_{max},k)+\xi_y v(j_{max},k)-2\frac{a(j_{max},k)}{\gamma-1}\right) \\
&\quad -\left(\xi_x u_\infty+\xi_y v_\infty-2\frac{a_\infty}{\gamma-1}\right) \qquad (A.82)\\
R(j_{max},k,2) &= \left(\xi_x u(j_{max},k)+\xi_y v(j_{max},k)+2\frac{a(j_{max},k)}{\gamma-1}\right) \\
&\quad -\left(\xi_x u(j_{max}-1,k)+\xi_y v(j_{max}-1,k)+2\frac{a(j_{max}-1,k)}{\gamma-1}\right) \qquad (A.83)
\end{aligned}$$

The residual equations for $n=3$ and $n=4$ are dependent on whether there is inflow or outflow of fluid at the node. For inflow,

$$R(j,k_{max},3) = \frac{\rho(j_{max},k)^\gamma}{p(j_{max},k)}-\frac{\rho_\infty^\gamma}{p_\infty} \qquad (A.84)$$

$$R(j,k_{max},4) = (-\xi_y u(j_{max},k)+\xi_x v(j_{max},k))-(-\xi_y u_\infty+\xi_x v_\infty) \qquad (A.85)$$

For outflow,

$$R(j,k_{max},3) = \frac{\rho(j_{max},k)^\gamma}{p(j_{max},k)}-\frac{\rho(j_{max}-1,k)^\gamma}{p(j_{max}-1,k)} \qquad (A.86)$$

$$\begin{aligned}
R(j,k_{max},4) &= (-\xi_y u(j_{max},k)+\xi_x v(j_{max},k)) \\
&\quad -(-\xi_y u(j_{max}-1,k)+\xi_x v(j_{max}-1,k)) \qquad (A.87)
\end{aligned}$$

The derivatives are then

$$\frac{\partial R(1,k,1)}{\partial\alpha} = \xi_x \mathrm{Ma}_\infty \sin\alpha-\xi_y \mathrm{Ma}_\infty\cos\alpha \qquad (A.88)$$

$$\frac{\partial R(1,k,2)}{\partial\alpha} = 0 \qquad (A.89)$$

$$\frac{\partial R(1,k,3)}{\partial\alpha} = 0 \qquad (A.90)$$

$$\frac{\partial R(1,k,4)}{\partial\alpha} = \begin{cases} -\xi_y \mathrm{Ma}_\infty\sin\alpha-\xi_x\mathrm{Ma}_\infty\cos\alpha & \text{if inflow}\\ 0 & \text{if outflow}\end{cases} \qquad (A.91)$$

# Appendix B

# Input Files for Optimizations

## B.1 Subsonic Single-Point Optimization Input File

Input file used for case without second-difference dissipation. Input file for case including second-difference dissipation is identical but with DIS2X= 1.0 and DIS2Y= 1.0.

```
MP-OPT
  1
WEIGHT FSMACH CL_TAR WFL CD_TAR RE DVALFA ALPHA OBJ_FUNC CLALPHA CLALPHA2 CLOPT
  1.0 0.7  0.4728 1.0 0.1 9E6 FALSE 3.0 8 TRUE TRUE TRUE
&OPTIMA
   OPT_METH = 3, OPT_ITER = 500, OPT_TOL = 1.d-5,
   OPT_RESTART = FALSE, GRADIENT = 1, CDF = TRUE, FD_ETA  = 1.d-6,
   COEF_FRZ = FALSE, IFRZ_WHAT = 3,
   WFACTOR = 0.1d0, BSTEP = 1.0,
   AUTO_RESTART = TRUE, NUM_RESTARTS = 5,

   INORD = 2, IREORD = 2,
   ILU_METH = 2, LFIL = 6, PDC = 3.d0,
   IM_GMRES = 85, EPS_GMRES = 1.d-8, ITER_GMRES = 500,

   NTCON=0, WFD=0.1, WTC=50.0,
   CTX     =0.35,0.96, 0.99,
   CTY_TAR =0.1206, 0.005, 0.0012,

   WAC=0.0, AREAFAC=1.0

   NRTCON = 0, crtxl = 0.10d0, crtxt = 0.90d0, crtxn = 15,
   crthtar = 0.142d0
```

```
&END
&CYCLONE
   JMAX=257, KMAX=57, JTAIL1=29, JTAIL2=229,

   TRANSLO=0.01D0, TRANSUP=0.01D0,
   CLTOL=1.0d-11,
   RELAXCL=3.0,   ICLFREQ=50,   ICLSTRT=0,

   RESTART=FALSE, JACDT=1, IREAD=2, BCAIRF=TRUE, CIRCUL=FALSE,
   IORD=2, INTEG=2, CMESH=FALSE,

   IDMODEL=1,
   DIS2X   = 0.0,  DIS4X = 0.01,  DIS2Y = 0.0,  DIS4Y = 0.01,
   VLXI    = 0.25, VNXI  = 0.25,  VLETA = 0.25, VNETA = 0.25,
   LIMITER = 1,    EPZ   = 1.d-3, EPV   = 5.d0,

   PREC = 0, PRXI = 0.0, PRPHI = 1.0,

   TURBULNT = TRUE, ITMODEL = 2,    ISPBC = 1,     VISCEIG  = 1.d0,
   VISCOUS  = TRUE, VISETA  = TRUE, VISXI = FALSE, VISCROSS = FALSE,

   NCP = 3000, NQ = 3000,

   SNGVALTE   = FALSE, GRDSEQ_REST = FALSE, SV_GRDSEQ = FALSE,
   WRITERESID = FALSE, TIMING      = FALSE, WRITETURB = FALSE,
   FLBUD      = FALSE, PCHORD      = 0.32,  SKNFRC    = FALSE
&END
&PROBE
   NK_ITS = 40,   NK_ILU   = 2,  NK_LFIL  = 2, NK_PFRZ  = 1,
   NK_PDC = 6.d0, NK_IMGMR = 40, NK_ITGMR = 40
&END
&EXTRA
   grid_file_prefix = '257x57',
   output_file_prefix = 'test',
   restart_file_prefix = 'initial'
&END
1.E-8  | AF Convergence criteria: AF_MINR
5.E-15                        | Convergence criteria: MIN_RES
1.E-8                         | Min. res. in turb. mod. : SPMIN_RES
ISEQUAL
      1
```

```
JMXI    KMXI    IENDS    DTISEQ   DTMIS   DTOW
257     57      20000      5.d0     0.0    1.d1
```

## B.2 Transonic Single-Point Optimization Input File

```
MP-OPT
  1
WEIGHT FSMACH CL_TAR WFL CD_TAR RE DVALFA ALPHA OBJ_FUNC CLALPHA CLALPHA2 CLOPT
 1.0  0.7  0.55  1.0  0.01   9e6  FALSE  0.05 8  TRUE  TRUE  TRUE

&OPTIMA
   OPT_METH = 3,  OPT_ITER = 300, OPT_TOL = 1.d-6,
   OPT_RESTART = FALSE, GRADIENT = 1, CDF = TRUE, FD_ETA  = 1.d-6,
   COEF_FRZ = FALSE, IFRZ_WHAT = 3, WFACTOR = 0.1, BSTEP = 0.1,
   AUTO_RESTART = TRUE, NUM_RESTARTS = 5,

   INORD = 2, IREORD = 2,
   ILU_METH = 2, LFIL = 6, PDC = 3.0,
   IM_GMRES = 85, EPS_GMRES = 1.d-8, ITER_GMRES = 500,

   NTCON=3, WFD=0.5, WTC=50.0,
   CTX     =0.25,0.92,0.99,0.00,0.00,
   CTY_TAR =0.118,0.009,0.002,0.00,0.00,

   WAC=0.0, AREAFAC=1.0

   NRTCON = 0, crtxl = 0.10d0, crtxt = 0.90d0, crtxn = 15,
   crthtar = 0.142d0
&END
&CYCLONE
   JMAX=201, KMAX=45, JTAIL1=25, JTAIL2=177,

   TRANSLO=0.01, TRANSUP=0.01,
   CLINPUT= 0.55, CLTOL=1.0d-11,
   RELAXCL=3.0,   ICLFREQ=50,   ICLSTRT=0,

   RESTART=FALSE, JACDT=1, IREAD=2, BCAIRF=TRUE, CIRCUL=FALSE,
   IORD=2, INTEG=2, CMESH=FALSE,

   IDMODEL=1,
   DIS2X   = 0.0,  DIS4X = 0.01,  DIS2Y = 0.0,  DIS4Y = 0.01,
   VLXI    = 0.25, VNXI  = 0.25,  VLETA = 0.25, VNETA = 0.25,
   LIMITER = 1,    EPZ   = 1.d-3, EPV   = 5.d0,

   PREC = 0, PRXI = 0.0, PRPHI = 1.0,

   TURBULNT = TRUE, ITMODEL = 2,    ISPBC = 1,     VISCEIG  = 1.d0,
   VISCOUS  = TRUE, VISETA  = TRUE, VISXI = FALSE, VISCROSS = FALSE,

   NCP = 1000, NQ = 1000,

   SNGVALTE   = FALSE, GRDSEQ_REST = FALSE, SV_GRDSEQ = FALSE,
   WRITERESID = FALSE, TIMING      = FALSE, WRITETURB = FALSE,
   FLBUD      = FALSE, PCHORD      = 0.32,  SKNFRC    = TRUE
&END
&PROBE
   NK_ITS = 40,   NK_ILU   = 2,  NK_LFIL  = 2, NK_PFRZ  = 1,
   NK_PDC = 6.d0, NK_IMGMR = 40, NK_ITGMR = 40
&END
&EXTRA
   grid_file_prefix = 'n201x45',
   output_file_prefix = 'test',
   restart_file_prefix = 'xxx'
&END
1.E-6  | AF Convergence criteria: AF_MINR
5.E-15                         | Convergence criteria: MIN_RES
1.E-8                          | Min. res. in turb. mod. : SPMIN_RES
ISEQUAL
     1
   JMXI    KMXI    IENDS    DTISEQ   DTMIS   DTOW
   201     45      20000      5.0     0.0    1.d1
```

## B.3 Four-Point Optimization Input Files

Input file for single-point optimization:

```
MP-OPT
  1
WEIGHT FSMACH CL_TAR WFL CD_TAR RE DVALFA ALPHA
```

```
  1.0 0.74  0.733 1.0 0.1 2.7E6 FALSE 1.8281631 8 TRUE TRUE TRUE
&OPTIMA
   OPT_METH = 3, OPT_ITER = 500, OPT_TOL = 1.d-5,
   OPT_RESTART = FALSE, GRADIENT = 1, CDF = TRUE, FD_ETA  = 1.d-6,
   COEF_FRZ = FALSE, IFRZ_WHAT = 3,
   WFACTOR = 0.1d0, BSTEP = 1.0,
   AUTO_RESTART = TRUE, NUM_RESTARTS = 5,

   INORD = 2, IREORD = 2,
   ILU_METH = 2, LFIL = 6, PDC = 3.d0,
   IM_GMRES = 85, EPS_GMRES = 1.d-8, ITER_GMRES = 500,

   NTCON=3, WFD=0.1, WTC=50.0,
   CTX     =0.35,0.96, 0.99,
   CTY_TAR =0.1204, 0.005, 0.0012,

   WAC=0.0, AREAFAC=1.0

   NRTCON = 0, crtxl = 0.10d0, crtxt = 0.90d0, crtxn = 15,
   crthtar = 0.142d0
&END
&CYCLONE
   JMAX=257, KMAX=57, JTAIL1=29, JTAIL2=229,

   TRANSLO=0.01D0, TRANSUP=0.01D0,
   CLINPUT= 0.4728, CLTOL=1.0d-11,
   RELAXCL=3.0,   ICLFREQ=50,   ICLSTRT=0,

   RESTART=FALSE, JACDT=1, IREAD=2, BCAIRF=TRUE, CIRCUL=FALSE,
   IORD=2, INTEG=2, CMESH=FALSE,

   IDMODEL=1,
   DIS2X   = 0.0,  DIS4X = 0.02,  DIS2Y = 0.0,  DIS4Y = 0.02,
   VLXI    = 0.25, VNXI  = 0.25,  VLETA = 0.25, VNETA = 0.25,
   LIMITER = 1,    EPZ   = 1.d-3, EPV   = 5.d0,

   PREC = 0, PRXI = 0.0, PRPHI = 1.0,

   TURBULNT = TRUE, ITMODEL = 2,    ISPBC = 1,     VISCEIG  = 1.d0,
   VISCOUS  = TRUE, VISETA  = TRUE, VISXI = FALSE, VISCROSS = FALSE,
   NCP = 3000, NQ = 3000,

   SNGVALTE   = FALSE, GRDSEQ_REST = FALSE, SV_GRDSEQ = FALSE,
   WRITERESID = FALSE, TIMING      = FALSE, WRITETURB = FALSE,
   FLBUD      = FALSE, PCHORD      = 0.32,  SKNFRC    = FALSE
&END
&PROBE
   NK_ITS = 40,   NK_ILU   = 2,  NK_LFIL  = 2, NK_PFRZ  = 1,
   NK_PDC = 6.d0, NK_IMGMR = 40, NK_ITGMR = 40
&END
&EXTRA
   grid_file_prefix = 'rae257x57',
   output_file_prefix = 'singlept',
   restart_file_prefix = 'initial'
&END
5.E-6  | AF Convergence criteria: AF_MINR
5.E-15                          | Convergence criteria: MIN_RES
1.E-8                           | Min. res. in turb. mod. : SPMIN_RES
ISEQUAL
      1
   JMXI    KMXI    IENDS    DTISEQ   DTMIS   DTOW
   257     57      20000      5.d0     0.0    1.d1
```

Input file for two-point optimization is the same as single-point input file, except for the first lines, and the output file name. First lines of input file (up to `&OPTIMA`):

```
MP-OPT
  2
WEIGHT FSMACH CL_TAR WFL CD_TAR RE DVALFA ALPHA OBJ_FUNC CLALPHA CLALPHA2 CLOPT
  1.0 0.68  0.733 1.0 0.1 2.7E6 FALSE 1.8281631 8 TRUE TRUE TRUE
  2.0 0.74  0.733 1.0 0.1 2.7E6 FALSE 1.8281631 8 TRUE TRUE TRUE
```

Similarly, first lines for input file of four-point optimization are:

```
MP-OPT
  4
WEIGHT FSMACH CL_TAR WFL CD_TAR RE DVALFA ALPHA OBJ_FUNC CLALPHA CLALPHA2 CLOPT
  1.0 0.68  0.733 1.0 0.1 2.7E6 FALSE 1.8281631 8 TRUE TRUE TRUE
  1.0 0.71  0.733 1.0 0.1 2.7E6 FALSE 1.8281631 8 TRUE TRUE TRUE
  2.0 0.74  0.733 1.0 0.1 2.7E6 FALSE 1.8281631 8 TRUE TRUE TRUE
  3.0 0.76  0.733 1.0 0.1 2.7E6 FALSE 1.8281631 8 TRUE TRUE TRUE
```

## B.4 Two-Point Pareto Front Input Files

This input file is for the case with the weight given to the $\mathrm{Ma}_\infty = 0.75$ operating point set to 0.1. For other optimizations, the weights in the fourth and fifth lines of the input file are varied.

```
MP-OPT
  2
WEIGHT  FSMACH  CL_TAR  WFL  CD_TAR    RE      DVALFA   ALPHA
 0.9  0.68  0.715 1.0 0.01 9.0e6 FALSE 2.0 8 TRUE TRUE TRUE
 0.1  0.75  0.715 1.0 0.01 9.0e6 FALSE 2.0 8 TRUE TRUE TRUE
&OPTIMA
   OPT_METH = 3, OPT_ITER = 2000, OPT_TOL = 1.d-6,
   OPT_RESTART = FALSE, GRADIENT = 1, CDF = TRUE, FD_ETA  = 1.d-6,
   COEF_FRZ = FALSE, IFRZ_WHAT = 3, WFACTOR = 0.1d0,  BSTEP = 0.1,

   INORD = 2, IREORD = 2,
   ILU_METH = 2, LFIL = 6, PDC = 3.d0,
   IM_GMRES = 85, EPS_GMRES = 1.d-8, ITER_GMRES = 500,

   AUTO_RESTART=TRUE, NUM_RESTARTS=5,

   NTCON=3, WFD=0.01, WTC=50.0,
   CTX =     0.01,   0.25,  0.99,
   CTY_TAR = 0.0253, 0.121, 0.002
&END
&CYCLONE
   JMAX=257, KMAX=57, JTAIL1=29, JTAIL2=229,

   TRANSLO=0.01, TRANSUP=0.01,
   CLINPUT= 0.4728, CLTOL=1.0d-11,
   RELAXCL=3.0,   ICLFREQ=50,   ICLSTRT=100,

   RESTART=FALSE, JACDT=1, IREAD=2, BCAIRF=TRUE, CIRCUL=FALSE,
   IORD=2, INTEG=2, CMESH=FALSE,

   IDMODEL=1,
   DIS2X   = 0.0,  DIS4X = 0.02,  DIS2Y = 0.0,  DIS4Y = 0.02,
   VLXI    = 0.25, VNXI  = 0.25,  VLETA = 0.25, VNETA = 0.25,
   LIMITER = 1,    EPZ   = 1.d-3, EPV   = 5.d0,
   PREC = 0, PRXI = 0.0, PRPHI = 1.0,

   TURBULNT = TRUE, ITMODEL = 2,    ISPBC = 1,     VISCEIG  = 1.d0,
   VISCOUS  = TRUE, VISETA  = TRUE, VISXI = FALSE, VISCROSS = FALSE,

   NCP = 3000, NQ = 3000,

   SNGVALTE   = FALSE, GRDSEQ_REST = FALSE, SV_GRDSEQ = FALSE,
   WRITERESID = FALSE, TIMING      = FALSE, WRITETURB = FALSE,
   FLBUD      = FALSE, PCHORD      = 0.32,  SKNFRC    = FALSE
&END
&PROBE
   NK_ITS = 20,   NK_ILU   = 2,  NK_LFIL  = 2, NK_PFRZ  = 1,
   NK_PDC = 6.d0, NK_IMGMR = 40, NK_ITGMR = 40
&END
&EXTRA
   grid_file_prefix = '257x57-15raef-10dv',
   output_file_prefix = 'pm-10re',
   restart_file_prefix = 'initial'
&END
5.E-6  | AF Convergence criteria: AF_MINR
5.E-15                          | Convergence criteria: MIN_RES
1.E-8                           | Min. res. in turb. mod. : SPMIN_RES
ISEQUAL
      1
   JMXI    KMXI    IENDS    DTISEQ   DTMIS   DTOW
   257     57      20000      5.d0     0.0    1.d1
```

# Appendix C

# Input Files for Eighteen-Point Optimization

Input file for first eighteen-point optimization:

```
MP-OPT
  18
WEIGHT FSMACH CL_TAR WFL CD_TAR RE DVALFA ALPHA OBJ_FUNC CLALPHA CLALPHA2 CLOPT
 1.5    0.72  0.17  1.0  0.01   27.32e6  FALSE  0.04  8  TRUE   TRUE   TRUE
 1.5    0.72  0.28  1.0  0.01   27.32e6  FALSE  0.63  8  TRUE   TRUE   TRUE
 1.5    0.72  0.27  1.0  0.01   18.57e6  FALSE  0.59  8  TRUE   TRUE   TRUE
 2.0    0.72  0.45  1.0  0.01   18.57e6  FALSE  1.65  8  TRUE   TRUE   TRUE
 2.0    0.64  0.21  1.0  0.01   24.22e6  FALSE  1.5   8  TRUE   TRUE   TRUE
 1.5    0.64  0.36  1.0  0.01   24.22e6  FALSE  2.6   8  TRUE   TRUE   TRUE
 1.5    0.64  0.34  1.0  0.01   16.46e6  FALSE  2.5   8  TRUE   TRUE   TRUE
 1.0    0.64  0.57  1.0  0.01   16.46e6  FALSE  4.5   8  TRUE   TRUE   TRUE
 1.0    0.76  0.28  1.0  0.01   28.88e6  FALSE  1.0   8  TRUE   TRUE   TRUE
 1.0    0.76  0.15  1.0  0.01   28.88e6  FALSE  1.0   8  TRUE   TRUE   TRUE
 1.0    0.76  0.46  1.0  0.01   28.88e6  FALSE  1.0   8  TRUE   TRUE   TRUE
 1.0    0.76  0.25  1.0  0.01   28.88e6  FALSE  1.0   8  TRUE   TRUE   TRUE
 1.0    0.76  0.45  1.0  0.01   19.62e6  FALSE  1.0   8  TRUE   TRUE   TRUE
 1.0    0.76  0.24  1.0  0.01   19.62e6  FALSE  1.0   8  TRUE   TRUE   TRUE
 1.0    0.76  0.74  1.0  0.01   19.62e6  FALSE  1.0   8  TRUE   TRUE   TRUE
 1.0    0.76  0.40  1.0  0.01   19.62e6  FALSE  1.0   8  TRUE   TRUE   TRUE
 1.0  0.16  1.20  1.0  0.010   11.8e6  TRUE   8.0   6  FALSE  FALSE  FALSE
 1.0  0.20  1.20  1.0  0.010   15.0e6  TRUE   8.0   6  FALSE  FALSE  FALSE

&OPTIMA
   OPT_METH = 3,  OPT_ITER = 300, OPT_TOL = 1.d-5,
   OPT_RESTART = FALSE, GRADIENT = 1, CDF = TRUE, FD_ETA  = 1.d-6,
   COEF_FRZ = FALSE, IFRZ_WHAT = 3, WFACTOR = 0.1, BSTEP = 0.1,
   AUTO_RESTART = FALSE, NUM_RESTARTS = 5,

   INORD = 2, IREORD = 2,
   ILU_METH = 2, LFIL = 6, PDC = 3.0,
   IM_GMRES = 85, EPS_GMRES = 1.d-8, ITER_GMRES = 500,

   NTCON=2, WFD=0.0, WTC=50.0,
   CTX     =0.95,0.99,
   CTY_TAR =0.01,0.002,

   WAC=0.0, AREAFAC=1.0

   NRTCON = 1, crtxl = 0.10d0, crtxt = 0.90d0, crtxn = 15,
   crthtar = 0.142d0
&END
&CYCLONE
   JMAX=289, KMAX=65, JTAIL1=33, JTAIL2=257,

   TRANSLO=0.01, TRANSUP=0.01,
   CLINPUT= 0.17, CLTOL=1.0d-11,
   RELAXCL=3.0,   ICLFREQ=50,   ICLSTRT=100,

   RESTART=FALSE, JACDT=1, IREAD=2, BCAIRF=TRUE, CIRCUL=FALSE,
   IORD=2, INTEG=2, CMESH=FALSE,

   IDMODEL=1,
   DIS2X   = 0.0,  DIS4X = 0.01,  DIS2Y = 0.0,  DIS4Y = 0.01,
   VLXI    = 0.25, VNXI  = 0.25,  VLETA = 0.25, VNETA = 0.25,
   LIMITER = 1,    EPZ   = 1.d-3, EPV   = 5.d0,

   PREC = 0, PRXI = 0.0, PRPHI = 1.0,

   TURBULNT = TRUE, ITMODEL = 2,    ISPBC = 1,     VISCEIG  = 1.d0,
   VISCOUS  = TRUE, VISETA  = TRUE, VISXI = FALSE, VISCROSS = FALSE,

   NCP = 1000, NQ = 1000,

   SNGVALTE   = FALSE, GRDSEQ_REST = FALSE, SV_GRDSEQ = FALSE,
   WRITERESID = FALSE, TIMING      = FALSE, WRITETURB = FASLE,
   FLBUD      = FALSE, PCHORD      = 0.32,  SKNFRC    = TRUE
&END
&PROBE
   NK_ITS = 40,   NK_ILU   = 2,  NK_LFIL  = 2, NK_PFRZ  = 1,
   NK_PDC = 6.d0, NK_IMGMR = 40, NK_ITGMR = 40
&END
&EXTRA
   grid_file_prefix = 'grid-new',
   output_file_prefix = 'test',
   restart_file_prefix = 'xxx'
&END
1.E-7  | AF Convergence criteria: AF_MINR
5.E-15                        | Convergence criteria: MIN_RES
1.E-8                         | Min. res. in turb. mod. : SPMIN_RES
ISEQUAL
     1
   JMXI    KMXI    IENDS    DTISEQ   DTMIS   DTOW
   289     65      20000      5.0     0.0    1.d1
```

Input file for the second eighteen-point optimization remains the same as for the first run, except a new grid is used that is made around the final airfoil from the first optimization, the initial objective function values used in the normalization are set to the original values from the first optimization, and the weights and other parameters for some design points are changed. Weights and input details for second eighteen-point optimization, continuing from first optimized airfoil:

```
WEIGHT FSMACH CL_TAR WFL CD_TAR RE DVALFA ALPHA OBJ_FUNC CLALPHA CLALPHA2 CLOPT
 1.5    0.72  0.17  1.0  0.01   27.32e6  FALSE  0.11 8  TRUE   TRUE   TRUE
 1.5    0.72  0.28  1.0  0.01   27.32e6  FALSE  0.74 8  TRUE   TRUE   TRUE
 1.5    0.72  0.27  1.0  0.01   18.57e6  FALSE  0.73 8  TRUE   TRUE   TRUE
 3.0    0.72  0.45  1.0  0.01   18.57e6  FALSE  1.76 8  TRUE   TRUE   TRUE
 3.0    0.64  0.21  1.0  0.01   24.22e6  FALSE  0.41 8  TRUE   TRUE   TRUE
 2.5    0.64  0.36  1.0  0.01   24.22e6  FALSE  1.44 8  TRUE   TRUE   TRUE
 2.5    0.64  0.34  1.0  0.01   16.46e6  FALSE  1.34 8  TRUE   TRUE   TRUE
 1.2    0.64  0.57  1.0  0.01   16.46e6  FALSE  2.93 8  TRUE   TRUE   TRUE
 1.0    0.76  0.28  1.0  0.01   28.88e6  FALSE  0.70 8  TRUE   TRUE   TRUE
 1.0    0.76  0.15  1.0  0.01   28.88e6  FALSE  0.06 8  TRUE   TRUE   TRUE
 1.0    0.76  0.46  1.0  0.01   28.88e6  FALSE  2.07 8  TRUE   TRUE   TRUE
 1.0    0.76  0.25  1.0  0.01   28.88e6  FALSE  0.53 8  TRUE   TRUE   TRUE
 1.0    0.76  0.45  1.0  0.01   19.62e6  FALSE  2.05 8  TRUE   TRUE   TRUE
 1.0    0.76  0.24  1.0  0.01   19.62e6  FALSE  0.52 8  TRUE   TRUE   TRUE
 1.0    0.76  0.74  1.0  0.01   19.62e6  FALSE  5.00 8  TRUE   TRUE   TRUE
 1.0    0.76  0.40  1.0  0.01   19.62e6  FALSE  1.59 8  TRUE   TRUE   TRUE
 1.0  0.16  1.80  1.0  0.010   11.8e6  TRUE   8.50 6  FALSE  FALSE  FALSE
 1.0  0.20  1.80  1.0  0.010   15.0e6  TRUE   7.96 6  FALSE  FALSE  FALSE
```

Similarly, weights and input details for third eighteen-point optimization, continuing from second optimized airfoil:

```
WEIGHT FSMACH CL_TAR WFL CD_TAR RE DVALFA ALPHA OBJ_FUNC CLALPHA CLALPHA2 CLOPT
 1.5    0.72  0.17  1.0  0.01   27.32e6  FALSE  -0.4 8  TRUE   TRUE   TRUE
 1.5    0.72  0.28  1.0  0.01   27.32e6  FALSE  0.20 8  TRUE   TRUE   TRUE
 1.5    0.72  0.27  1.0  0.01   18.57e6  FALSE  0.20 8  TRUE   TRUE   TRUE
 3.0    0.72  0.45  1.0  0.01   18.57e6  FALSE  1.20 8  TRUE   TRUE   TRUE
 4.0    0.64  0.21  1.0  0.01   24.22e6  FALSE  -0.3 8  TRUE   TRUE   TRUE
 2.5    0.64  0.36  1.0  0.01   24.22e6  FALSE  0.80 8  TRUE   TRUE   TRUE
 2.5    0.64  0.34  1.0  0.01   16.46e6  FALSE  0.70 8  TRUE   TRUE   TRUE
 1.2    0.64  0.57  1.0  0.01   16.46e6  FALSE  2.20 8  TRUE   TRUE   TRUE
 1.0    0.76  0.28  1.0  0.01   28.88e6  FALSE  0.20 8  TRUE   TRUE   TRUE
 1.0    0.76  0.15  1.0  0.01   28.88e6  FALSE  -0.5 8  TRUE   TRUE   TRUE
 1.0    0.76  0.46  1.0  0.01   28.88e6  FALSE  1.50 8  TRUE   TRUE   TRUE
 1.0    0.76  0.25  1.0  0.01   28.88e6  FALSE  0.00 8  TRUE   TRUE   TRUE
 1.0    0.76  0.45  1.0  0.01   19.62e6  FALSE  1.50 8  TRUE   TRUE   TRUE
 1.0    0.76  0.24  1.0  0.01   19.62e6  FALSE  0.60 8  TRUE   TRUE   TRUE
 8.0    0.76  0.74  1.0  0.01   19.62e6  FALSE  0.00 8  TRUE   TRUE   TRUE
 1.0    0.76  0.40  1.0  0.01   19.62e6  FALSE  1.59 8  TRUE   TRUE   TRUE
 5.0  0.16  1.90  1.0  0.010   11.8e6  TRUE   16.1 6  FALSE  FALSE  FALSE
 5.0  0.20  1.90  1.0  0.010   15.0e6  TRUE   15.8 6  FALSE  FALSE  FALSE
```