Aerodynamic Design Optimization of a Non-Lifting Strut

by

Justin G. Veenendaal

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

University of Manitoba

Winnipeg

Abstract

Aerodynamic Design Optimization of a Non-Lifting Strut

By

Justin G. Veenendaal

A design engineer has a desire to obtain the best possible design configuration producing the most desirable result. This is especially true in designs involving aerodynamics. This thesis presents a way to design the optimum airfoil for a non-lifting strut-like application. This is achieved by combining the governing laws of aerodynamics with appropriate numerical models to simulate an inputted steady flow regime. By using a robust yet simple parameterization method to represent airfoils and by implementing a genetic algorithm, optimization is achieved and occurs in a timely manner. Performing the optimization across a range of flow fields and for struts in different applications also allows some trends to be deduced, thus providing valuable knowledge to design engineers.

Acknowledgements

The work done in this thesis was not completed without sweat and tears. A big thank-you to my wife Annette who provided these, who bore with me this past year and a half and who moved across the world so that I could pursue my studies, I can never thank-you enough. The additional love and support that I received from my two sons Damon and Micah was more than enough that I needed for motivation to complete this work.

Thank-you also to my advisors, Robert Derksen and Tim Rogalsky for entrusting this topic into my hands, for guidance, for keeping the ship steady and for the good hearted light conversation. Thanks also to the committee members, Mark Tachie and Abba Gumel for taking the time to review and examine this work.

Table of Contents

List of Figures iv

List of Tables vii

List of Symbols ix

1 Introduction 1

1.1 Motivation 2
1.2 Literature review 3

2 Background and theory 4

2.1 Aerodynamic fundamentals 4
2.2 Boundary-layer models 9
2.3 Airfoil parameterization 15
2.4 Optimization 18

3 Solution space 23

3.1 Domain definition 24
3.2 Hard constraints 26
3.3 Limitations 27

4 Code development 29

4.1 Programming and resources 29
4.2 Penalty functions 32
4.3 Convergence 33

5 Test Input 38

5.1 Flow field variables 38
5.2 Shape constraints 38

6 Results and Discussions 42

6.1 Struts loaded axially 42
6.2 Struts loaded in bending 54

7 Conclusions and future work 69

8 Bibliography 71

Appendix A – Optimized area constrained airfoil shapes 73

Appendix B – Optimized moment of inertia constrained airfoil shapes 83

Appendix C – Fortran90 computer code (Area constrained) 95

Appendix D – Fortran90 computer code (Ixx constrained) 111

List of Figures

Figure 2.1 – Airfoil nomenclature....5

Figure 2.2 – Pressure and shear stress integration....5

Figure 2.3 – Aerodynamic force relationships....6

Figure 2.4 – Circulation around wing....7

Figure 2.5 – Flow separation over an airfoil....8

Figure 2.6 – Attached streamlines over an airfoil....9

Figure 2.7 – Boundary layer model (Cebeci, 2004)....13

Figure 2.8 – Interactive boundary-layer....13

Figure 2.9 – Flow separation bubble....15

Figure 2.10 – BP representation....16

Figure 2.11 – Mechanism of DE....21

Figure 4.1 – Overview of Code....31

Figure 4.2 - Normalized distribution of difference vectors – Generation 1....35

Figure 4.3 - Normalized distribution of difference vectors – Generation 5....36

Figure 4.4 - Normalized distribution of difference vectors – Generation 25....36

Figure 4.5 - Normalized distribution of difference vectors – Converged....37

Figure 5.1 – Method of employing shape constraint....40

Figure 6.1 – $0.05/c^2$ area constrained airfoil at Re = 150,000....42

Figure 6.2 – $0.05/c^2$ area constrained airfoil at Re = 1,000,000....42

Figure 6.3 – $0.05/c^2$ area constrained airfoil at Re = 15,000,000....42

Figure 6.4 – $0.1/c^2$ area constrained airfoil at Re = 500,000....43

Figure 6.5 – $0.1/c^2$ area constrained airfoil at Re = 15,000,000....43

Figure 6.6 – Area constrained crest lateral position trends 44

Figure 6.7 – Area constrained crest max thickness position trends 44

Figure 6.8 – Crest location versus flow behaviour change, A = 0.05/c² 45

Figure 6.9 – Crest location versus flow behaviour change, A = 0.1/c² 45

Figure 6.10 - Area constrained wedge angle trends 48

Figure 6.11 – Cost plots of optimized shapes 49

Figure 6.12 – Cost of best performing vector vs generation - A=0.05/c², Re=5e4 49

Figure 6.13 – Cost of best performing vector vs generation - A=0.05/c², Re=5e5 50

Figure 6.14 – Cost of best performing vector vs generation - A=0.05/c², Re=5e6 50

Figure 6.15 – Convergence history, A=0.05/c², Re=5e4 51

Figure 6.16 - Convergence history, A=0.05/c², Re=5e5 51

Figure 6.17 - Convergence history, A=0.05/c², Re=5e5 52

Figure 6.18 – Ixx = 1.0e-5/c^4 constrained airfoil at Re = 100,000 54

Figure 6.19 – Ixx = 1.0e-5/c^4 constrained airfoil at Re = 300,000 55

Figure 6.20 – Ixx = 1.0e-5/c^4 constrained airfoil at Re = 500,000 55

Figure 6.21 – Ixx = 1.0e-5/c^4 constrained airfoil at Re = 10,000,000 55

Figure 6.22 – Ixx = 2.5e-5/c^4 constrained airfoil at Re = 200,000 55

Figure 6.23 – Ixx = 2.5e-5/c^4 constrained airfoil at Re = 1,500,000 56

Figure 6.24 – Ixx = 1.0e-4/c^4 constrained airfoil at Re = 300,000 56

Figure 6.25 – Ixx = 1.0e-4/c^4 constrained airfoil at Re = 10,000,000 56

Figure 6.26 – Moment of inertia constrained crest lateral position trends 57

Figure 6.27 – Moment of inertia constrained crest max thickness position trends 58

Figure 6.28 – Crest location versus flow behaviour change, Ixx = 1.0e-5/c^4 58

Figure 6.29 – Crest location versus flow behaviour change, Ixx = 6.0e-5/c^4 59

Figure 6.30 – Moment of inertia constrained wedge angle trends 62

Figure 6.31 – Cost plots of moment of inertia optimized airfoils 63

Figure 6.32 – Cost of best performing vector vs generation - Ixx=1.0e-5/c^4, Re=5e4 63

Figure 6.33 – Cost of best performing vector vs generation - Ixx=1.0e-5/c^4, Re=5e5 64

Figure 6.34 – Cost of best performing vector vs generation - Ixx=1.0e-5/c^4, Re=5e5 64

Figure 6.35 – Convergence history, Ixx=1.0e-5/c^4, Re=5e4 65

Figure 6.36 – Convergence history, Ixx=1.0e-5/c^4, Re=5e5 65

Figure 6.37 – Convergence history, Ixx=1.0e-5/c^4, Re=5e6 66

List of Tables

Table 3-1 – Corresponding airfoil chord lengths for freestream conditions (air) 25

Table 3-2 – Initial population upper and lower limits on airfoil parameters 27

Table 5-1 – Test matrix of cross-sectional area constrained airfoils 41

Table 5-2 – Test matrix of second moment of inertia constrained airfoils 41

Table 6-1 – Area = 0.05/c² airfoil parameters 46

Table 6-2 – Area = 0.075/c² airfoil parameters 47

Table 6-3 – Area = 0.1/c² airfoil parameters 47

Table 6-4 – Area = 0.125/c² airfoil parameters 47

Table 6-5 – Area = 0.15/c² airfoil parameters 47

Table 6-6 – NCE's for A=0.05/c² 52

Table 6-7 – NCE's for A=0.075/c² 53

Table 6-8 – NCE's for A=0.1/c² 53

Table 6-9 – NCE's for A=0.125/c² 53

Table 6-10 – NCE's for A=0.15/c² 54

Table 6-11 – Ixx = 1.0e-5/c^4 airfoil parameters 60

Table 6-12 – Ixx = 2.5e-5/c^4 airfoil parameters 60

Table 6-13 – Ixx = 6.0e-5/c^4 airfoil parameters 61

Table 6-14 – Ixx = 1.0e-4/c^4 airfoil parameters 61

Table 6-15 – Ixx = 2.5e-4/c^4 airfoil parameters 61

Table 6-16 – NCE's for Ixx=1.0e-5/c^4 66

Table 6-17 – NCE's for Ixx=2.5e-5/c^4 67

Table 6-18 – NCE's for Ixx=6.0e-5/c^4 67

Table 6-19 – NCE's for Ixx=1.0e-4/c^4 67

Table 6-20 – NCE's for Ixx=2.5e-4/c^4 68

List of Symbols

Upper case Roman

A	axial force (Eq. (2.2))
A	cross-sectional area
A	damping length constant (Eq. (2.9))
CR	crossover control constant
D	drag force
F	scaling factor
I_{xx}	second moment of inertia about the x-axis
L	lift force
N	normal force
P	pressure
R_t	ratio of wall shear to max Reynolds shear stress
R_{xtr}	Reynolds number at turbulence transition
U	x-component of velocity
U_e	free stream velocity
U_i	velocity component (index notation)
U_∞	free stream velocity
V	y-component of velocity

Lower case Roman

b_i	Bezier control point
c	airfoil chord length

c_d	drag coefficient
c_p	pressure coefficient
dz_{te}	trailing edge thickness
l	mixing length
p	pressure
p_l	lower surface pressure
p_u	upper surface pressure
r_{le}	leading edge radius
s_l	lower surface length
s_u	upper surface length
t	time
u	x-component of fluctuating velocity
u_i	fluctuating velocity component (index notation)
u_τ	friction velocity
v	y-component of fluctuating velocity
x	Cartesian coordinate
x_t	x-coordinate of airfoil crest
y	Cartesian coordinate
y_t	y-coordinate of airfoil crest
z	Cartesian coordinate

Lower case Greek

α	angle of attack
α	outer region turbulent viscosity factor (Eq. (2.9))

β_{te}	trailing edge wedge angle
γ	intermittency factor outer region
γ_{tr}	intermittency factor transition region
δ_{ij}	Kronecker delta
θ	surface angle from horizontal
κ_t	airfoil crest curvature
λ_1	arbitrary parameter 1
λ_2	arbitrary parameter 2
λ_{amp}	amplification factor
μ	dynamic viscosity
ν	kinematic viscosity
ρ	density
τ_l	lower surface shear stress
τ_u	upper surface shear stress
τ_w	wall shear

1 Introduction

The desire for the best possible design is what inspires design optimization. Aerodynamic optimization is then simply the desire to obtain the best design considering the aerodynamic properties of the physical domain. Aerodynamic properties range extensively but the common ones which are most often studied are lift, drag, pitching moment, and centre of pressure. For instance, one can optimize the lift to drag ratio of an aircraft wing or airfoil or determine the optimum location of the centre of pressure of an aircraft thus leading to better flight stability. However in doing so, each problem may contain tens or hundreds of parameters and factors that influence these desires. Formulating all of the possible design configurations, if at all possible, can lead to tens or hundreds of thousands of tests or simulations.

Up until the last couple of decades, most aerodynamic testing of airfoils was done in wind tunnels. Optimizing a problem with a high number of configurations can be especially impractical if each design needs to be created and tested in a wind tunnel! Even with the advent of high performance computers, the time required to run flow simulations for even a hundred design configurations simply may not meet the time constraints placed on a design project. From the configurations tested, a designer is able to pick the best one but that does not give confidence that it is indeed ‘the best one’. It is obvious now that speed of design is important. Speed in optimization is required to both propose a new design and assess that design. Of equal importance, of course, is accuracy, as one must be confident that the design performs as required.

It is upon these two factors that this thesis pivots. Here the term design refers to the design of a non-lifting symmetric airfoil. The optimization in this design problem refers

to a configuration of a set of airfoil representative parameters that result in minimum drag. In order to achieve this it is important to be able to use a method that accurately defines the geometric properties of an airfoil, that accurately assess the aerodynamic properties of the airfoil, and that does so in a quick and efficient manner.

1.1 Motivation

Non-lifting airfoils as the name suggests are airfoils not designed for lift applications. Rather the primary purpose of these symmetric airfoils is to streamline the flow while producing minimal disruptions. Hence the primary aerodynamic property that is important here is drag. Drag is the force that directly opposes the motion of the flow. There are many applications of non-lifting airfoils in aerodynamic designs. One example is in the design of struts positioned in a flow. These struts are used in the landing gear of an airplane, wing struts on a biplane, or struts in an F1 race car suspension system. These applications demand that in addition to being aerodynamic, they also have to be structurally sound. Another application can be a filtering or grating system upstream of a hydroelectric dam, or other protective equipment in turbo machinery. In these cases the purpose is to keep unwanted objects from entering fan blades or turbines and causing damage. While achieving this it is also important to minimize the impact on fluid flow as an inefficient grating design would negate precious upstream momentum.

Considering the vast number of uses of non-lifting airfoils in designs and the importance of their applications it is desirable for a given flow field to implement the best possible aerodynamic shape while maintaining crucial structural requirements. The aim of this research is to be able to obtain exactly that and to achieve this for axially loaded struts as well as struts loaded in bending.

1.2 Literature review

Optimization of airfoils has been around since the Wright brothers and likely even earlier. Since the First World War the National Advisory Committee of Aeronautics (NACA) started to develop and test airfoils with the goal of increasing lift. Although many of the first tests were trial and error, as research in this area evolved further mathematics were incorporated into the designs. Methods today use various mathematical models to generate shapes and computer-simulated boundary layer analyses to grade them. Intelligent schemes such as genetic algorithms are also employed in order to generate, locate, and enhance optimal configurations.

Genetic algorithms have previously been employed in aerodynamic design to obtain optimization. Both the shape of a fan blade (Rogalsky, Derksen, & Kocabiyik, 2000) and the spacing of fan blades (Rogalsky, Derksen, & Kocabiyik, 1999) have successfully used such algorithms in their design. These optimizations as well as the optimization presented in this thesis are for steady uniform operating conditions. The operating conditions in flight, however, cannot be assumed to be steady but rather are typically unsteady and somewhat unpredictable. While there has been work done in this area (Huyse & Lewis, 2001), significant computational effort was required and there the goal of optimization was a set percentage improvement from an existing design.

2 Background and theory

This chapter details the four main topics that comprise the objective of this thesis. The basic fundamentals of aerodynamics are presented first (§2.1) as this introduces the bottom-line variables where optimization is achieved. The viscous nature of these variables requires knowledge of boundary-layer theory and mathematical models for airfoil aerodynamics, and these are presented in §2.2. Since many airfoil shapes used in design are common or prescribed, altering these shapes is not possible without a technique that parameterizes these shapes. Section 2.3 summarizes one such method where this can be achieved. Having accumulated all of the necessary variables and parameters that describe a flow over an airfoil for a single case, an optimization strategy can now be incorporated. Section 2.4 details a strategy that can be used to evaluate many variations of the flow domain in an effort to optimize the shape.

2.1 Aerodynamic fundamentals

The study of aerodynamics is the study of fluid mechanics as it pertains to flow in air. Most recognizably this study applies to aircraft, flight and airfoils. An airfoil is any structure where its surface is designed to promote aerodynamic properties within a flow. Generically the most common surface is that of a wing that is long compared with its width and a two dimensional curve outlining the surface through the cross section describes an airfoil. The fundamentals of flow over airfoils can be found in many intermediate aerodynamics textbooks. The fundamentals as described in this section are summarized from *Fundamentals of Aerodynamics 5th edition* by Anderson (2011).

Referring to Figure 2.1, an airfoil is comprised of an upper surface and a lower surface. The intersections of these surfaces are denoted the leading and trailing edges

defined by the flow direction, that is, from leading edge to trailing edge. The straight line that joins these two points defines the chord of the airfoil. In aerodynamics it is often useful to use non-dimensional forms of parameters. Parameters involving a length are normalized by the chord length and parameters involving a velocity are normalized by the freestream velocity.

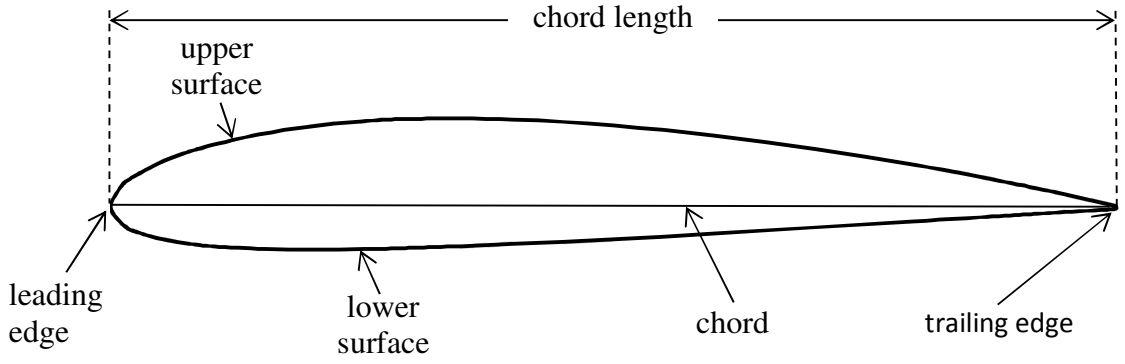


Figure 2.1 – Airfoil nomenclature

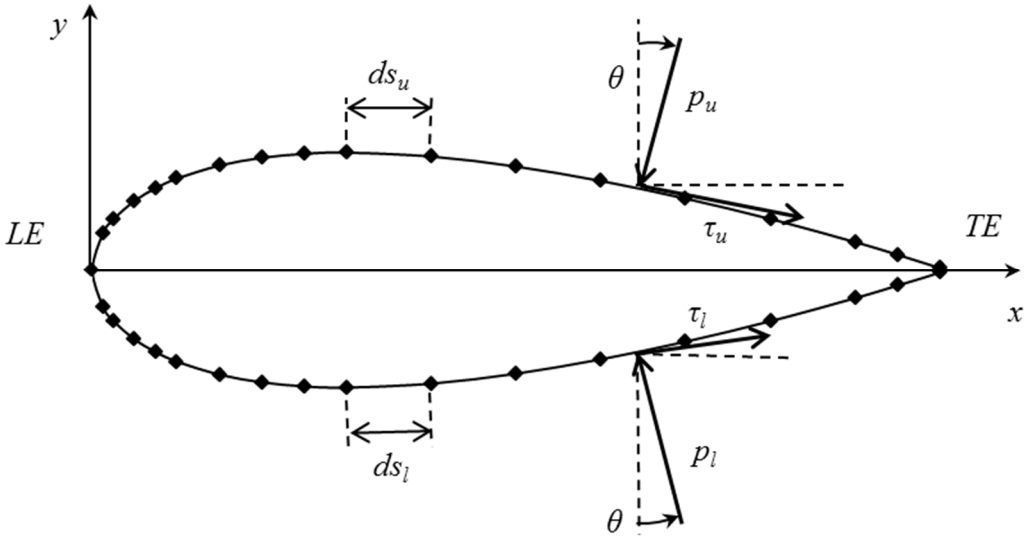


Figure 2.2 – Pressure and shear stress integration

An airfoil in a freestream causes a pressure distribution to develop on the upper and lower surfaces. When integrated over their respective surfaces, the resulting difference generates a net force with a component normal to the chord and an axial component parallel to the chord.

Correspondingly due to the viscous nature of the freestream at a molecular level, a shear stress distribution also develops over the surfaces. When integrated along the surface, the resulting additional forces contribute to the net normal and axial forces. Figure 2.2 illustrates this behaviour. Mathematically the normal (N) and axial (A) components can then be written as:

$$N = -\int_{LE}^{TE} \left(p_u \cos\theta + \tau_u \sin\theta\right) ds_u + \int_{LE}^{TE} \left(p_l \cos\theta - \tau_l \sin\theta\right) ds_l \qquad (2.1)$$

$$A = \int_{LE}^{TE} \left(-p_u \sin\theta + \tau_u \cos\theta\right) ds_u + \int_{LE}^{TE} \left(p_l \sin\theta + \tau_l \cos\theta\right) ds_l \qquad (2.2)$$

Once determined, these forces can also be subject to a transformation by rotating the airfoil by a specified angle of attack. From this the vertical and horizontal forces can be resolved to obtain the lift (L) and drag (D) forces respectively. This relationship is shown in the following figure and set of equations.

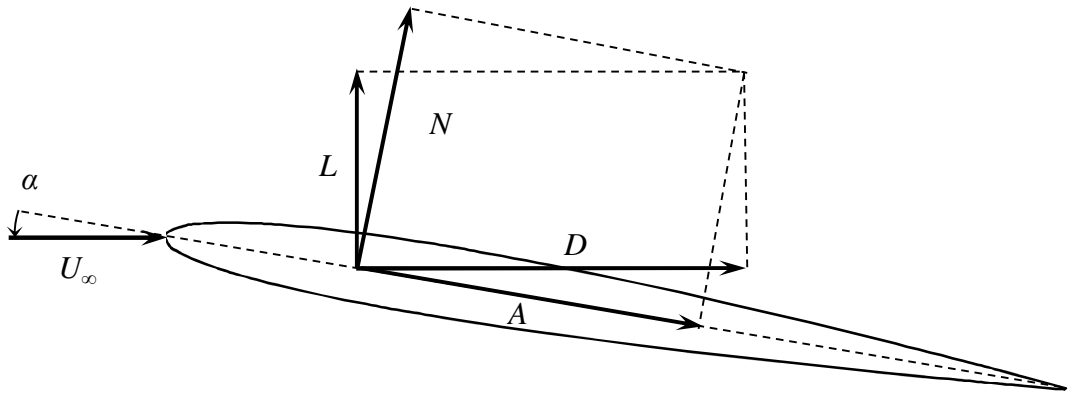


Figure 2.3 – Aerodynamic force relationships

$$L = N\cos\alpha - A\sin\alpha \qquad (2.3)$$

$$D = N\sin\alpha + A\cos\alpha \qquad (2.4)$$

As suggested by Figure 2.2, the procedure to determine the pressure and shear stress distributions over the surfaces of the airfoil is to subdivide the surface into short length segments or panels. In an inviscid flow this is done by assigning a vortex of unknown strength to each panel, applying the Kutta condition to the trailing edge and then solving for the unknown strengths. Obtaining this allows the calculation of the total circulation (Γ) or vorticity. The Kutta—Joukowski theorem then states that this total circulation produces lift.

The circulation around an airfoil is generated by the fluid's viscosity:

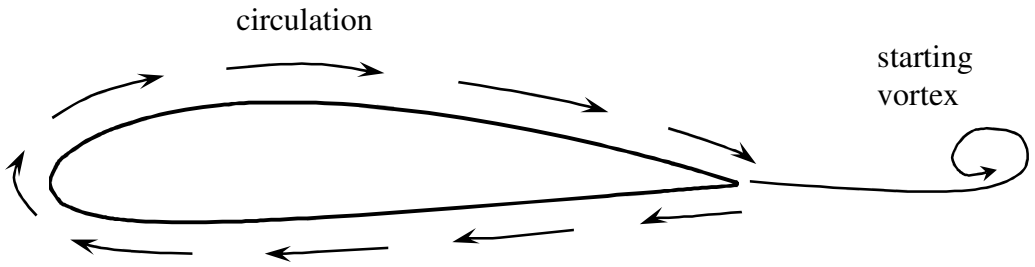


Figure 2.4 – Circulation around wing

The viscosity leads to the formation of the starting vortex, which according to the balanced Navier—Stokes equation gives this circulation. Figure 2.4 shows the two vortices formed in this flow acting in their respective opposite directions. Superimposing this velocity field with the freestream velocity field reveals that the average surface velocity beneath the wing becomes less than the surface velocity above the wing, thus

generating the necessary pressure differential to cause lift. Thus it is the fluid's viscosity that generates lift. Coincidentally it is also the fluid's viscosity that generates drag.

The airfoil's profile drag is the total drag produced; combinations of form drag and skin friction drag. Form drag is a function of how sleek the shape of the airfoil is, or how well the streamlines of the flow are attached to the airfoil. Regions of separation as shown in Figure 2.5 are where the streamlines become detached and pressure distributions become disruptive resulting in large increases of form drag. Skin friction on the other hand arises from the interaction of the airfoil and the fluid surrounding the airfoil. The no-slip boundary condition requires that the fluid molecules immediately adjacent to the airfoil have zero relative velocity with the airfoil. This implies that large velocity gradients are present near the airfoil surface. This velocity gradient combined with the fluid's dynamic viscosity produces a shear stress which becomes balanced by the skin friction.

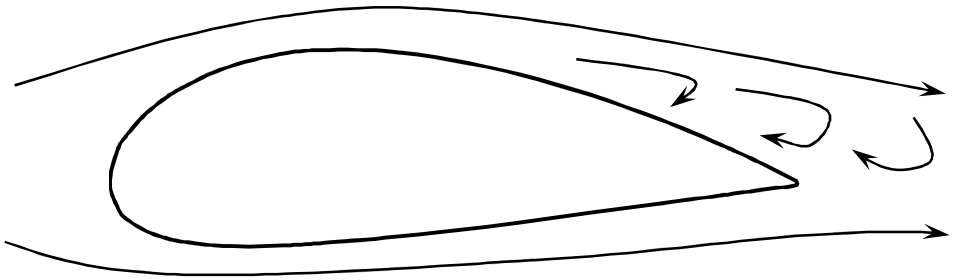


Figure 2.5 – Flow separation over an airfoil

For shapes with fully attached streamlines as shown in Figure 2.6, the profile drag is due purely to skin friction drag. As separation occurs in a flow, the presence of form drag starts to appear thus creating the combination in profile drag. It is important to note

that skin friction terms are not present in the separated regions and as separation becomes severe, the profile drag is predominantly due to form drag.

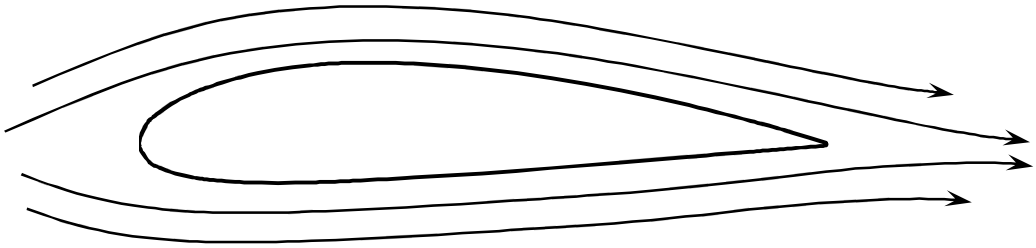


Figure 2.6 – Attached streamlines over an airfoil

As described above, aerodynamic lift can be estimated by using a vortex panel method. To add an estimation of drag to the method, the viscous effects need to be added in such a way to account for both the form drag and the skin friction drag. This requires a combination of boundary-layer models.

2.2 Boundary-layer models

For total laminar flow, the Navier—Stokes equations could easily be employed and solved using a finite difference or finite volume numerical simulation. However most flows over airfoils result in turbulence and a transition from laminar flow to turbulent flow. For turbulent flows, the Navier—Stokes equations still apply but this complicates the solution significantly. Turbulence is chaotic in nature and extremely sensitive to initial and boundary conditions. Various methods can be employed to solve the Navier—Stokes equations such as using direct numerical simulation (DNS) or turbulent-viscosity models. While the accuracy of DNS is desirable, the computational power required to achieve this accuracy particularly in an optimization problem is too expensive. Using a

turbulent-viscosity model allows for a much quicker solution but is done while sacrificing some accuracy. These models involve solving the Navier—Stokes equations for average flow field properties rather than instantaneous ones and replacing turbulent stresses with appropriate models. These Reynolds stress terms as they appear in the Reynolds Averaged Navier—Stokes (RANS) equations, as shown in (2.5), increase the number of unknowns from four (3 velocity and 1 pressure scalar) to ten (6 added stress terms) thus creating a closure problem. Some simplifications however can be made as they relate to flow over an airfoil.

$$\rho \frac{\overline{D}\langle U_j \rangle}{\overline{D}t} = \frac{\partial}{\partial x_i}\left[\mu\left(\frac{\partial \langle U_i \rangle}{\partial x_j} + \frac{\partial \langle U_j \rangle}{\partial x_i} \right) - \langle p \rangle \delta_{ij} - \rho \langle u_i u_j \rangle \right] \qquad (2.5)$$

The x-direction momentum equation for a boundary layer flow assuming time independence and neglecting variations in the z-direction is:

$$U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} = -\frac{1}{\rho}\frac{dP}{dx} + \nu\frac{\partial^2 U}{\partial y^2} - \frac{\partial}{\partial y}(uv) \qquad (2.6)$$

The y-direction momentum equation integrates to the Bernoulli equation thus the pressure gradient term can be written in terms of the freestream velocity.

$$-\frac{1}{\rho}\frac{dP}{dx} = U_\infty \frac{dU_\infty}{dx} \qquad (2.7)$$

It is important to note here that these equations are for a flat-plate boundary layer flow. While an airfoil is not flat, a piecewise panel method is and can be used along with a coordinate transformation relating adjacent panels. Also, the curvature of the airfoil leads to flow acceleration on convex surfaces and deceleration on concave surfaces. This behaviour will later become important in defining the limits of the solution space.

To solve the continuity and *x*-direction momentum equations, a closure method must be introduced for the Reynolds shear stress term. Methods of doing so involve using a turbulent-viscosity model and relating the Reynolds shear stress components to the mean velocity gradients. The relationship in (2.8) includes the eddy or turbulent viscosity which is a property of the flow rather than the fluid,

$$uv = \nu_T \frac{\partial U}{\partial y} \qquad (2.8)$$

The turbulent viscosity is then a function of a timescale and a lengthscale and is somewhat arbitrary. Various models use different scales and acceptance is based on the accuracy of the results obtained. The popular k-ε is a two-equation model that relates the scales to the kinetic energy and rate of energy dissipation of the flow. The model used in this problem uses the zero-equation Cebeci—Smith (CS) model.

The CS model developed by Cebeci (2004) is a two layer model suitable for high speed flows with boundary layers which are thin and attached. The model consists of an inner layer and an outer layer where a turbulent viscosity is defined for each of the layers. The interface location, where the turbulent viscosity of the inner layer equals that of the outer layer, is denoted y_c. The following equations provide the descriptions of the turbulent viscosities of the two layers.

$$\nu_{inner} = l^2 \left| \frac{\partial U}{\partial y} \right| \gamma_{tr} \qquad (2.9)$$

$$\nu_{outer} = \alpha \int_0^{\delta} (U_e - U) dy \cdot \gamma_{tr} \gamma$$

$$l = 0.40 \cdot y\left[1 - \exp\left(-\frac{y}{A}\right)\right]$$

$$A = 26\frac{\nu}{N}u_\tau^{-1},\ N = \left(1 - 11.8\frac{\nu u_e}{u_\tau^{\ 3}}\frac{du_e}{dx}\right)$$

$$\gamma_{tr} = 1 - \exp\left[-G(x - x_{tr})\int_{x_{tr}}^{x}\frac{dx}{u_e}\right]$$

$$G = \frac{3}{C^2}\frac{u_e^{\ 3}}{\nu^2}R_{x_{tr}}^{\ -1.34},\ C^2 = 213\left(\log R_{x_{tr}} - 4.7323\right)$$

$$\alpha = \frac{0.0168}{\left[1 - \beta\left(\frac{\partial u}{\partial x}/\frac{\partial u}{\partial y}\right)\right]_m^{1.5}}$$

$$\beta = \begin{cases} \dfrac{6}{1 + 2R_t(2 - R_t)} & R_t \le 1.0 \\ \dfrac{1 + R_t}{R_t} & R_t \ge 1.0 \end{cases}$$

$$R_t = \frac{\tau_w}{\left(-\rho(uv)\right)_m}$$

So far the equations that define the boundary layer have been for an ideal case. These apply for boundary layers that are subjected to high Reynolds numbers and are thin and attached to the body. However, airfoils that have high cambers or steep thickness profiles can cause the boundary layer to separate from the body, even at low angles of attack and particularly for high Reynolds numbers. At points of separation the equations described so far produce a singularity, and thus are ineffective. Being able to determine the external velocity and pressure distribution at this point or region requires the involvement of the inverse method. The inverse method prescribes as boundary conditions in addition to the CS boundary conditions the wall shear or displacement thickness. This approach creates additional unknowns, so an iterative sweeping approach

is used. This approach simulates a boundary layer by imposing a blowing velocity in the interaction region. The total effect of the viscous flow together with the flow separation is shown in Figure 2.7. In (a) the representation of the flow due to viscous effects is shown. Flow separation (b) is included into the calculation by implementing a fictitious blowing velocity (c) thus enabling the prediction of the external velocity. Details of the mathematics that simulate the blowing velocity can be found in the text by Cebeci (2004).

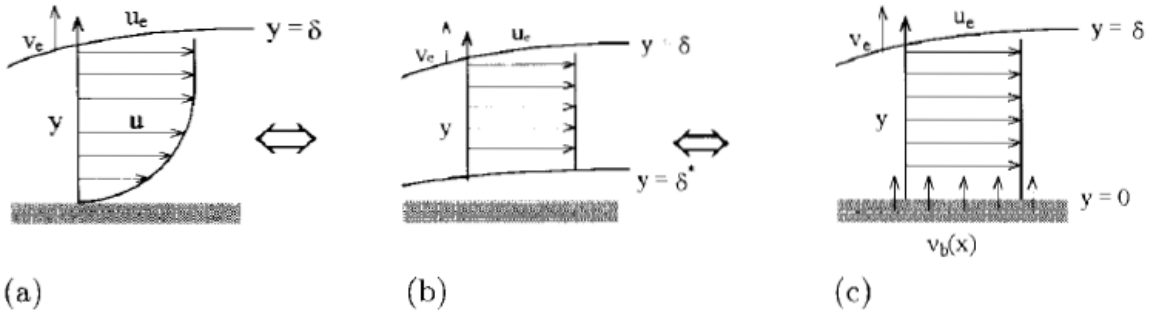


Figure 2.7 – Boundary layer model (Cebeci, 2004)

Combining the methods introduced creates a panel method with added viscous effects as well as flow separation. As shown in Figure 2.8, this is an iterative cycle that is solved until a convergence test is met.

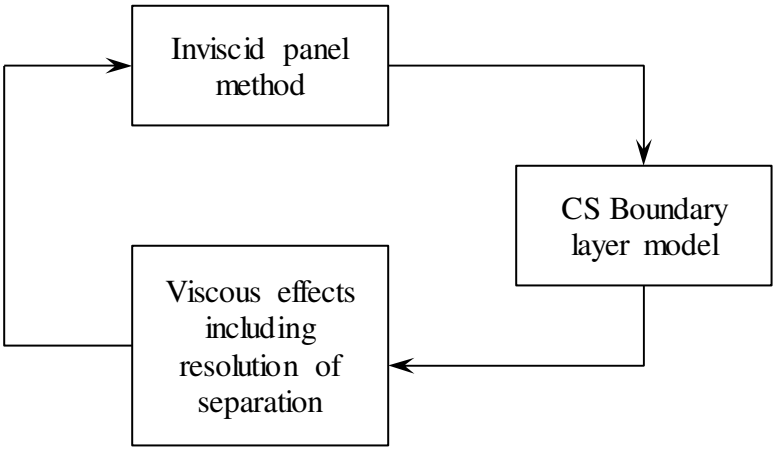


Figure 2.8 – Interactive boundary-layer

Like all simplified turbulent models, the CS and inverse models were based on assumptions. It is important to understand these assumptions as they help to define the solution space. These models have assumed that the boundary layers are thin. Also the inverse method that resolves separation has assumed that the blowing velocity acts in a vertical direction. This assumption is basically that of thin airfoil theory, thus flow separation on airfoils with large cambers or bluff body portions are not accurately captured. This aspect is critical to keep in mind when generating airfoil shapes using a parameterization method.

Before proceeding it is necessary to review the transition from laminar flow to turbulent flow. While each type of flow produces drag from the molecular level, turbulent flow produces additional drag resulting from Reynolds shear stresses. It is this additional drag that greatly increases the net drag so one can see that minimizing this effect can produce desired consequences. For most airfoils there is a transition point where the laminar flow over the leading edge becomes unstable and transitions to turbulent flow. The point where this occurs is referred to as the transition point and is rarely clearly defined. For low Reynolds numbers (<200,000-300,000) this occurs at the trailing edge but for higher Reynolds numbers this occurs somewhere along the surfaces of the airfoil. While the previously mentioned models can predict this transition based on a local Reynolds number there is a common phenomenon that complicates things. A flow separation bubble occurs when laminar flow over an airfoil detaches from the surface and then reattaches in a turbulent manner. This type of transition is much more difficult to model and occurs at Reynolds numbers near 300,000. This behaviour is shown graphically in Figure 2.9.

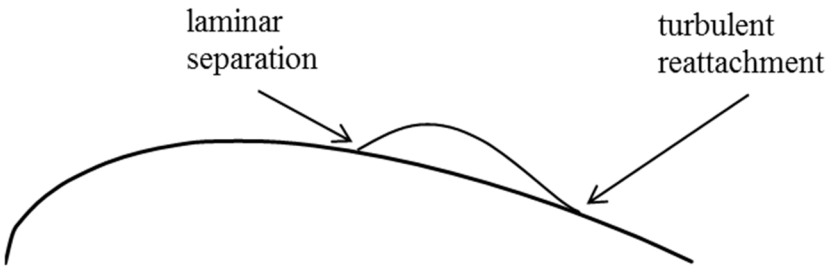


Figure 2.9 – Flow separation bubble

2.3 Airfoil parameterization

Mathematical equations must be used to represent an aerodynamic shape in order to obtain a computational design. For instance, NACA airfoils can be modelled using a system of equations that define its shape. Systems of equations have proven to be a successful way of capturing the geometric properties of an airfoil. The use of shape functions (Chang, Torres, & Tung, 1995), Legendre polynomials (Coiro & Nicolosi, 1995), Bezier polynomials (Venkataraman, 1995), and PARSEC parameters (Sobieczky, 1999) have all been used to represent existing airfoils and generate new ones. There is no perfect way to represent an airfoil as there are always limitations. Some methods have the unwanted ability to generate non-aerodynamic or abstract shapes. While this is needed to test the boundaries of the solution space, on the other hand shapes produced must be aerodynamic enough that they can be evaluated accurately. Other representation methods are so restrictive that many aerodynamic shapes are excluded and boundaries may not be explored. Methods can also exhibit peculiar or unstable behaviours when combined with optimization strategies. It is therefore desirable that a method be used that can, as much as possible, represent a complete solution space of airfoils. It is also desirable to use a method that is stable in optimizing with parameters that are well-defined, interpreted, and controlled. Bezier—PARSEC (BP) is a method that has been successfully demonstrated

to be capable of capturing aerodynamic shapes within a set solution space (Derksen & Rogalsky, 2009).

The BP method is a combination of Bezier curves and PARSEC variables. Bezier curves are defined by a number of control points. The first and last control points define the ends of the curve while any intermediate points pull the curve in a weighted direction. Points far from the curve exhibit more pull than a point that lies close to the curve. In the development of BP, Bezier curves as shown in equation (2.10) were found to be effective in airfoil representation. The control points (x_n, y_n) are then expressed in terms of PARSEC or similar type variables that have aerodynamic and physical meaning.

$$\begin{aligned} x(u) &= x_0(1-u)^3 + 3x_1u(1-u)^2 + 3x_2u^2(1-u) + x_3u^3 \\ y(u) &= y_0(1-u)^3 + 3y_1u(1-u)^2 + 3y_2u^2(1-u) + y_3u^3 \end{aligned} \qquad (2.10)$$

PARSEC parameterization involves generating a series of shape functions about physical geometric descriptions similar to the BP method. There are six PARSEC parameters that relate to the thickness profile. They are: the radius of the leading edge r_{le}, the location of the thickness crest (x_t, y_t), the curvature of the thickness crest κ_t, the wedge angle at the trailing edge β_{te}, and the thickness of the trailing edge dz_{te}.

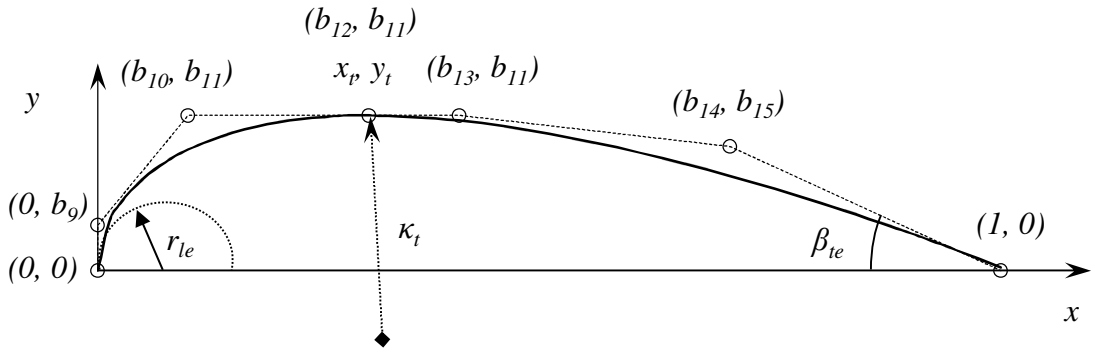


Figure 2.10 – BP representation

The thickness of the trailing edge from a calculation perspective is optimized when it is zero as this allows the Kutta condition to be easily satisfied when a panel method is used. However, in reality the thickness if the trailing edge cannot be zero as this provides no local structural strength in that area. In the work presented in this thesis this concept will be kept in mind while fixing the thickness of the trailing edge to be zero.

BP uses four curves to represent an airfoil; two for the thickness profile and two for the camber profile. These when combined then produce the curves for the upper and lower surfaces. For the specific form of BP shown in equation (2.10), BP 3333, the four curves are all cubic. The work described in this thesis focuses on symmetric airfoils thus only the thickness profile representations are needed. Figure 2.10 illustrates the thickness representation by Bezier control points (in brackets) and PARSEC parameters. BP constrains the Bezier control points in terms of the PARSEC parameters by implementing conditions on the first and second derivatives for the two curves. Hard constraints are also imposed to prevent negative thickness and any looping back upon itself. The result of parameterization was the following:

$$b_9 = \frac{3}{2}\kappa_t\left(x_t - b_{10}\right)^2 + y^t$$

b_{10} is the lowest real root of the quartic :

$$\frac{27}{4}\kappa_t{}^2\left(b_{10}\right)^4 - 27\kappa_t{}^2 x_t\left(b_{10}\right)^3 + \left(9\kappa_t y_t + \frac{81}{2}\kappa_t{}^2 x_t{}^2\right)\left(b_{10}\right)^2$$

$$+\left(2r_{le} - 18\kappa_t x_t y_t - 27\kappa_t{}^2 x_t{}^3\right)\left(b_{10}\right) + \left(3y_t{}^2 + 9\kappa_t x_t{}^2 y_t + \frac{27}{4}\kappa_t{}^2 x_t{}^4\right) = 0$$

wi thin the bounds :

$$\max\left\{0, x_t - \sqrt{\frac{-2y_t}{3\kappa_t}}\right\} < b_{10} < x_t \qquad (2.11)$$

$$b_{11} = y_t$$

$$b_{12} = x_t$$

$$b_{13} = 2x_t - b_{10}$$

$$b_{14} = 1 + \left[dz_{te} - \left(\frac{3}{2}\kappa_t\left(x_t - b_{10}\right)^2 + y_t\right)\right]\cot\left(\beta_{te}\right)$$

$$b_{15} = \frac{3}{2}\kappa_t\left(x_t - b_{10}\right)^2 + y_t$$

2.4 Optimization

There are many methods that can be employed when optimizing a set of parameters. But a method of optimizing that produces the global minimum within a solution space is not quite so simple. Issues such as local minima or premature convergence are factors that make finding the global minimum difficult. It is also necessary to have a method that finds all solutions provided that they exist. An important note to mention here is that these types of optimization strategies may not be perfect since there are not unique solutions to these strategies and there is no guarantee that a solution even exists. The strategy will also always be a unique function of the problem at hand. Thus it is important to carefully assess the results of the strategy. If a solution is indeed a true optimum, then the result will be the optimal independent of the strategy used to

obtain the solution. A demonstrated successful strategy that has been used in aerodynamic optimization is a relatively new scheme known as differential evolution (DE) (Price & Storn, 1997).

DE is actually quite simple. The first step of DE is to generate an initial population of parameter containing vectors with values chosen at random within the full solution space. Each vector contains the genes or DNA of that population member – in the case of the work presented in this thesis, the parameters that fully describe the shape of an airfoil. Each vector is scored using a cost function. Here the cost function is the Interactive Boundary-Layer (IBL) solver of Figure 2.8. Once complete, a new set of population members or trial vectors is generated. These 'child' vectors are the result of mutation and crossover or recombination. Once these vectors have all been created, they are scored in the same manner as their 'parents' or target vectors and if the score or cost of the child results in a value lower than that of the parent, the child replaces the parent in the next generation. If on the other hand the child fails to score lower than the parent, it is the parent that continues on to the next generation.

The terms mutation and crossover have been introduced and together they play an integral part in keeping the population stable and directed towards a converged optimized result. Mutation is making small perturbations to a population member based on the magnitude of the global difference in population members. Therefore in an initial population the degree of mutation is significantly more profound than in a generation that is approaching convergence. Mutating is the necessary means to exploration and changing a population, but by scaling these mutations a population is able to remain stable. There are different ways to mutate a vector; equations (2.12) and (2.13)

describe the operation as it is performed on a gene or parameter within a population vector for the work of this thesis. The scaling factor *F* is arbitrary but the optimal values of this for most functions lie in the range of 0.3 to 1.0.

Crossover on the other hand is a means of insuring that good genes are being passed on in the process of building child or trial vectors. In a manner similar to that of biological reproduction, this enables the search for the optimum to focus on the tried,

$$X_c' = X_c + F(X_a - X_b) \quad (2.12)$$

$$X_e' = X_e + F(X_{best} - X_e) + F(X_a + X_b - X_c - X_d) \quad (2.13)$$

tested, and true. In DE, a parent vector is combined with the mutated vector to form the final child or trial vector. There are different methods to perform this crossover, either exponential or binary. The technique used here, binary crossover, is based on a random value being generated for each parameter of every vector. If the random number is less than a pre-determined crossover constant (*CR*), than the mutated vector parameter becomes the trial vector parameter, otherwise the parent vector parameter becomes the trial vector parameter. The value *CR* is taken in the range $0 \leq CR \leq 1$. If $CR = 1$, this indicates that every trial vector parameter will come from the mutated vector. If on the other hand $CR = 0$, this indicates that the trial vector will exactly replicate the parent vector. This is undesirable and if this is the case, then to ensure that the trial vector differs from the parent vector by at least one parameter, the last parameter is taken to be that of the mutated vector.

The description of the optimization strategy is represented by *DE/X/Y/Z*. The *DE* stands for differential evolution; the *X* denotes the type of vector to be mutated, the *Y*

denotes the number of difference vectors considered for mutation and the *Z* denotes either an exponential or binary type of crossover. Thus combining strategies (2.12) and (2.13) with binary crossover, these can be labelled as DE/rand/1/bin and DE/rand-to-best/2/bin respectively. Figure 2.11 summarizes and illustrates the DE mechanism DE/rand/1/bin where *A, B, C, D* represent a set of target vectors and *A', B', C', D'* represent trial vectors. Further details of the DE mechanism can be readily found in Price (1999) or (Price & Storn, 1997).

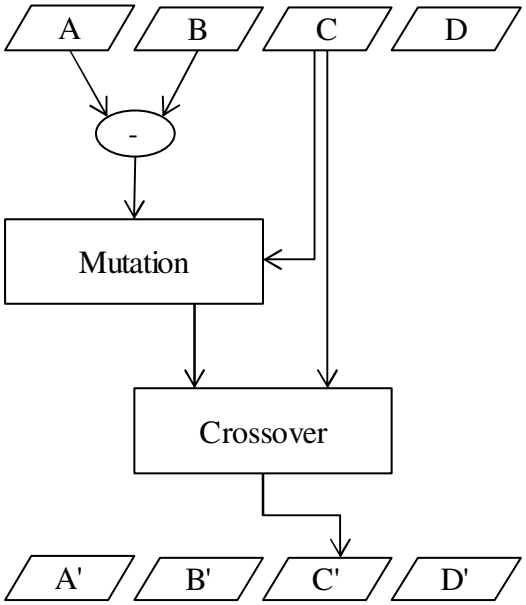


Figure 2.11 – Mechanism of DE

With this mechanism, a few issues arise that need to be addressed. Mutation can often result in parameter values that lie outside the solution space boundaries. When this happens it may be desirable to explore the out-of-bounds range if physically possible. Normally, however, that parameter is redirected back inside the solution space. This can be done either by setting the value of the parameter either to the nearest boundary, or to half-way in between the target vector parameter and the nearest boundary. The latter is

the approach that is taken in this work. Also, ill-defined solution variables can lead to premature convergence or lack thereof. Experimentation is necessary to determine suitable values of the DE constants, as well as population size. Price (1999) suggests a population size of 20 times the number of parameters, a scaling factor $0.5 \leq F \leq 0.9$, and a crossover constant $0.8 \leq CR \leq 1.0$ as stable yet efficient values in obtaining optimization. As mentioned previously, the true global optimum is independent of these values.

The final aspect of DE that needs attention is that of convergence. Various forms of criteria have been used in DE with the most common being that of closeness to a benchmark. A benchmark is a desired value and if known, an acceptable tolerance can be defined. If the lowest-cost trial vector in a population generation is within that tolerance, then convergence is reached. The work presented in this thesis has no workable way of setting a benchmark since the optimal solutions do not yet exist thus making it difficult to establish such a convergence criterion. An alternative method that has been demonstrated successfully in a power grid optimization problem is a criterion based on the maximum distance between all trial vectors in the current population (Zielinski, Weitkemper, Laur, & Kammeyer, 2006). Setting this distance to an acceptably low value allows any single vector in the final population to be a solution. Of course this also inhibits the possibility that all of the trial vectors are forced to converge to the same vector when in fact other optimal locations exist. This behaviour can cause unwanted results when using a multimodal objective function. Other methods such as monitoring the percent change of the optimum vector have proven to work successfully in applications but preliminary work suggested that premature convergence was obtained.

3 Solution space

From the theory and concepts introduced in §2 the complete solution space of the optimization problem can be narrowed down and formed. Each of the previous subsections has implicitly set their own physical bounds thus requiring a concise domain to be well defined and understood as well as some hard constraints to be set.

Before continuing it is necessary to describe the units of the variables that define the solution space. Aerodynamic flows are most often described in dimensionless terms. Physical space variables are non-dimensionalized by the airfoil's chord length thus airfoil coordinates or locations are described as a fraction of its chord length. Two non-dimensional numbers are needed to fully describe the flow field or environment of the solution space. The Mach number is a ratio of the freestream flow velocity to the speed of sound in the freestream conditions as shown in equation (3.1). The Reynolds number is a ratio of the flows' inertial force to the fluids' viscous force and is increased by an increase in the inertial force, either the freestream velocity or the chord length and is shown in equation (3.2). The fluid kinematic viscosity is generally constant as it is a property of the fluid rather than the flow field noting that viscous force can also be dependent on the presence of turbulence within the flow.

$$\mathrm{Ma} = \frac{U_\infty}{a} \qquad (3.1)$$

$$\mathrm{Re}_c = \frac{U_\infty c}{\nu} \qquad (3.2)$$

3.1 Domain definition

A shape optimization problem starts with complete physical space. The problem at hand considers just two dimensions of this space as many aerodynamic flows over airfoils can be assumed to behave in this way since the chord length or its thickness is significantly less than its depth. The next assumption is that the solution to the optimization problem of minimum drag is an airfoil. Obviously it is known that the shape with the least drag resistance is no shape at all, i.e. a cylinder with a zero diameter or a flat plate with zero thickness. But for a size constrained problem (more about this constraint later) this can be easily validated by the aerodynamic fundamentals presented in §2.1. The tear drop shape is one where the shape can remain streamlined with the flow field at higher speeds and higher Reynolds numbers as shown in Figure 2.6. Where a shape can remain streamlined with the flow and avoid large regions of flow separation, the drag on the shape will be mostly due to skin friction drag. Form drag on the other hand is a much more adverse drag that is produced in the flow separated regions. Thus without progressing into the optimization method, a niche within the complete physical domain can be defined where to a high degree of confidence, it is known that the optimal solution lies. This tear drop shaped solution can then be described in this space in terms of the previously mentioned dimensionless coordinates.

Two other dimensionless numbers define the rest of the solution domain. The Reynolds number as mentioned previously is a ratio of the flow field inertial forces to the fluid molecular viscous forces. Thus the range of possible values can cover the highly viscous region (Re values close to zero) to the inviscid range (Re values approaching infinity). The work described in this thesis considers aerodynamic shape optimization for

application in a useful range for most engineering tasks. This useful range depends on the kinematic viscosity of the fluid (1.5e-5 m²/s for air) and what are generally encountered freestream velocities. Thus most airfoils for engineering applications lie within a range of Re = 50,000 to Re = 15,000,000. These Reynolds number can be made up many various ways and as shown in Table 3-1 together span a very large domain. It is important to note here that this ignores applications in high speed flight which would generate higher Reynolds numbers yet. However in this range it is necessary to start looking at the other dimensionless number – the Mach number more closely.

		Chord length (m)	
		Re = 50000	Re = 15000000
Freestream velocity (m/s)	0.01	75	22500
	0.1	7.5	2250
	1	0.75	225
	10	0.075	22.5
	100	0.0075	2.25

Table 3-1 – Corresponding airfoil chord lengths for freestream conditions (air)

As mentioned earlier, the Mach number is a ratio of the freestream velocity to the speed of sound within the freestream. So far the theory and models presented have been based on incompressible flow field assumptions. Most sources agree that a flow field can be deemed to be incompressible if the Mach number is less than 0.3 (Anderson, 2011). Values above this warrant looking at the compressible effects of the flow. At these high Mach numbers it is necessary to include density variation and thermal effects as part of the aerodynamic property predictions. Since the Cebeci—Smith model used to evaluate the resulting drag on an aerodynamic shape does not include these compressible effects, the flow field domain is limited to Mach numbers less than 0.3. Also since the freestream velocity appears in both the Reynolds number and Mach number and since there is

limitation to the incompressible flow region, the Mach number becomes arbitrary. While the Mach number can be taken to be arbitrary and only adds some correction to the aerodynamic coefficients within the boundary layer solver, the same cannot be said for the Reynolds number.

The boundary layer thickness of steady state flow over an airfoil increases as the Reynolds number decreases. As mentioned in §2.2, the Cebeci—Smith boundary layer model is sufficient for evaluating thin airfoils. So a big question that has to be answered is when does an airfoil qualify as being thin enough or when is an airfoil too thick to evaluate correctly using a thin airfoil boundary layer solver? The thick and thin description does not apply to just the physical shape of the airfoil. In evaluating an airfoil, the thickness of an airfoil is that of its physical thickness plus the thickness of the boundary layer. Thus it is important to pay particular attention to thicker airfoils at low Reynolds numbers as these shapes will tend to push and test the limits of the boundary layer model. Since the boundary layer solver is an iterative process with an internal convergence check, these airfoils may return a null value thus aiding this acceptability check. Other than this premise, there are no real guidelines as to the acceptance of the boundary layer model and a final verdict can only be made by an inspection of the results.

3.2 Hard constraints

With limits already imposed on the Mach number and Reynolds number, some of the other limits that apply to the solution space are now described. Recall from §2.4 that a symmetric airfoil can be represented using six PARSEC parameters. These parameters control eight Bezier control points which then generate the thickness profile curves of the airfoil. Looking at Figure 2.10 the following observations can be made which then serve

as hard constraints for the parameter sets. A maximum value of -0.001 is imposed on the leading edge radius as well as the crest curvature as these curves must remain negative or convex. Also since the airfoil must have positive thickness it is necessary to constrain the y-coordinate of the crest to be greater than zero as well as specifying the wedge angle to be greater than zero. An upper limit on the wedge angle can be defined to be 90° (or 1.57 radians) since for values larger than this the curve will not behave in a one-to-one relationship. In addition to this relationship a monotonic behaviour is also imposed on the leading edge. Together these constraints ensure that the curve does not come back on itself or allow the generation of non-aerodynamic shapes. With this solution space clearly defined and the constraints that apply to this space imposed, a code can be developed that encompasses all of these previous aspects and models. This behaviour adds computational complexity to the loop shown in Figure 2.8. The following table shows the constraints that were made to the initial population. The limits were set large enough to ensure that the complete solution space was equally represented with favoritism given to no particular area.

	min	max
r_{le}	-0.10	-0.001
k_t	-25.0	-0.01
β_{te}	0.001	1.57
x_t	0.15	0.70
y_t	0.05	0.25

Table 3-2 – Initial population upper and lower limits on airfoil parameters

3.3 Limitations

In optimizing, this validity criteria combined with a random shape generator poses the possibility of some real serious problems. Simply put, it is possible that a poor

aerodynamic shape in theory gets wrongly evaluated such that a low cost or drag is predicted. During early code development, some of these false trends developed and were able to be resolved using penalty functions within the boundary layer solver. This will be discussed in more detail in §4.2.

Another potential limitation is that the parameterization method of §2.3 excludes the optimal solution or is simply unable to parameterize the potential solution. The BP 3333 method as shown by Rogalsky (2004) was able to replicate most of the airfoils studied and the ones where the method was unable were shapes with large cambers and shapes with cusps, both of which are not applicable in this application. Thus a high level of confidence is obtained by using the BP 3333 parameterization method as a shape generator covering a large percentage of the significant solution space. Within the parameterization method, care also needs to be taken in the event that generated solutions do not fall under the heading of conventional airfoils. That is, although the BP 3333 parameterization method has shown itself to be very successful for these airfoils not much evidence in its ability to replicate non-conventional airfoils is available.

The most significant limitation in using a combination of codes resulting in an accumulation of convergence criteria is that of numerical accuracy. It is hypothesized that some of the airfoil representation parameters will have a more profound effect on drag prediction than others. That is, some of the parameters may only contribute a small effect, likely outside the accuracy range of the solver. The result of this may be that although a solid understanding of the major parameters can be obtained; there is less hope for concise understanding of these lesser parameters.

4 Code development

By piecing together the theoretical portions of §2 and understanding the domain of applicability as well as the limitations presented in §3, an optimization program can be constructed in order to achieve the objectives. The information presented in this section describes how this is achieved in the work presented in this thesis. Code development, issues that arose and how they were handled, as well as convergence are described in detail.

4.1 Programming and resources

The code that was developed to solve the optimization problem was written using Fortran 90. Fortran 90 is a general purpose programming language excellent at crunching numbers both scientifically and numerically. This language was chosen for this reason and due to the boundary layer solver code being available in this language.

Briefly, Fortran 90 is a source program and when compiled creates an object program. This is linked with the any libraries and subprograms that were utilized in the source program, and together this forms an executable program. The executable program can be ran on a computer with the same operating system that was used to create it. For the development of the code in this thesis, a PC with an Intel Core 2 Quad CPU 2.4GHz and 4GB memory with a Windows 7 operating system was used. This allowed for reasonable runtimes, which will be discussed later on.

The mainline of the program contains six subroutines. The subroutines are PARAMS, INIPOP, GEOM, PANEL_IBL, RESID and POSTPRO. The range of values along with the physical hard constraints are specified in subroutine PARAMS. Subroutine INIPOP simply assigns a parameter a random value within the associated range. This

subroutine contains a loop cycling through the total number of parameters. Subroutine GEOM contains subroutine ROOTS which is a quartic root solver as well as a choice of either function X_AREA or IXX depending on the application. Function X_AREA calculates the cross sectional area of the airfoil while function IXX calculates the airfoil's second moment of inertia. The purpose of these subroutines is to perform the arithmetic presented in §2.3. Subroutine PANEL_IBL is the boundary layer solver initially introduced in §2.2 and contains many other subroutines created by Cebeci (2004). The convergence of the program is managed by subroutine RESID. The final subroutine POSTPRO manages the export of data, some of the data processed and other data organized for further processing.

Within the code's mainline there are three primary loops. The first loop contains the generation of the initial population (subroutines INIPOP, GEOM, and PANEL_IBL). The loop cycles through all of the population members and works to ensure that all available member spots are filled with valid shapes. These shapes are then scored and their values stored within an array. The other two loops are nested. The outer loop is active until convergence is reached or the maximum number of allowable generations is exceeded. For each generation (inner loop), every new population member is chosen through the differential evolution (DE) method. Trial vectors are created, checked for shape validity and evaluated. Trial vectors then replace the target vectors if they outperform them. At the end of each generation, convergence is checked, and if achieved then post processing begins. A schematic of these loops is presented in Figure 4.1.

The specification of the DE control variables is done within the mainline. The results obtained by the code for the optimization were obtained using a population size of

100, a scaling factor of F = 0.3, and a crossover control constant of CR = 0.8. The maximum number of allowable generations was set at 500. Confidence in these variables can be established by altering them and then ensuring that the same solution is obtained. Although some combinations of these variables lead to faster solution convergence than others, it was not the exercise of this thesis to find these optimum combinations but rather to ensure that premature convergence did not occur.

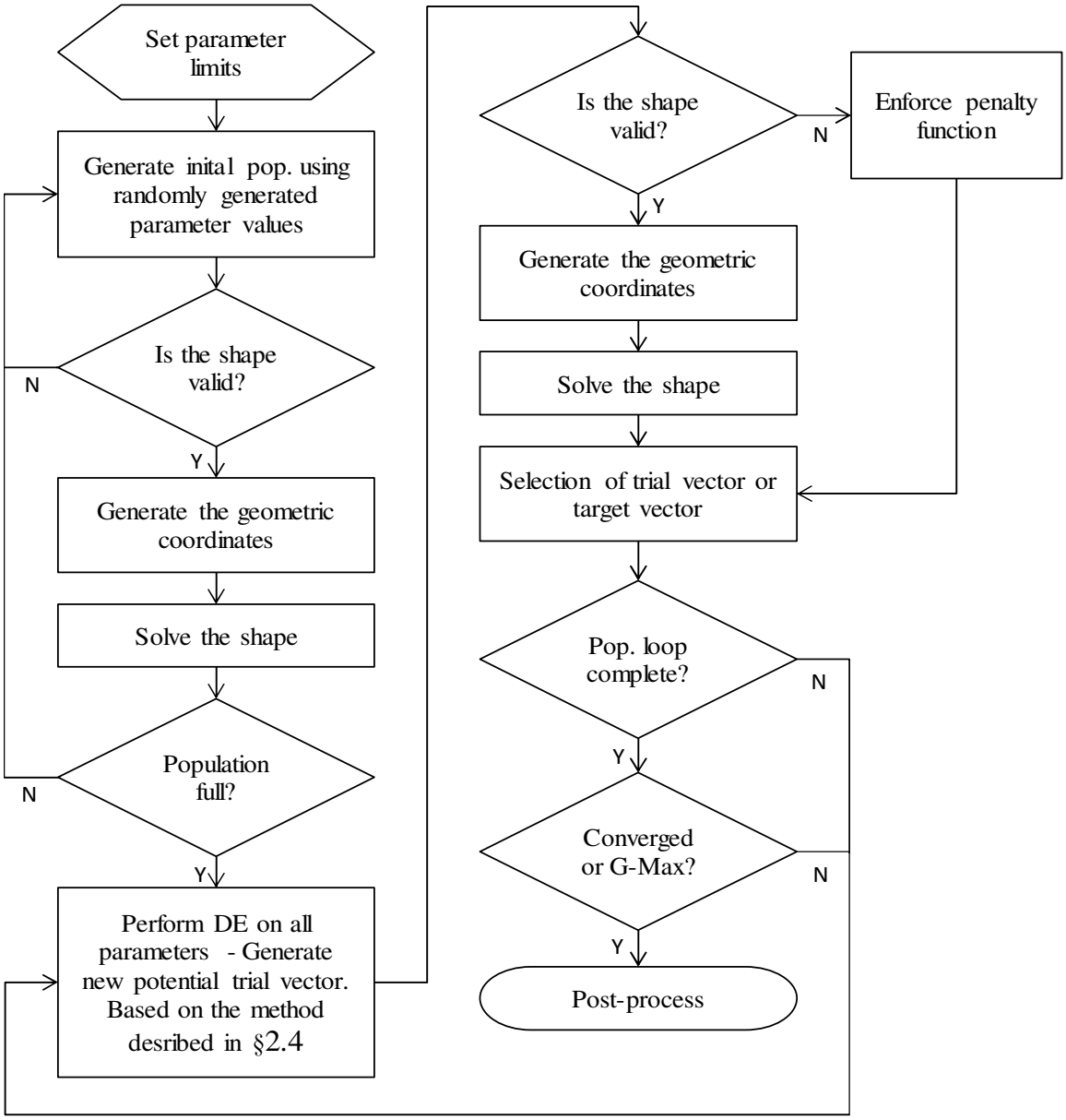


Figure 4.1 – Overview of Code

4.2 Penalty functions

Looking back at Figure 4.1 one of the actions is to enforce the penalty function. This block occurs if a shape fails a validity test. Since invalid shapes are so commonly produced in the initial population, for that population only, invalid shapes are simply rejected. However, it is less likely to produce an invalid shape in subsequent generations. Here the shape may be nearly valid, and in order to explore the solution space boundaries a penalty is enforced.

There are three serious cases that arise that require penalty functions. These penalty functions or soft constraints penalize a shape thereby coaxing it out of the evolution algorithm. One case where penalty functions are required as mentioned above is when invalid shapes are produced. The other cases that were determined are when the internal convergence of the boundary layer does not occur and when separation occurs at the trailing edge. Each is handled in different ways and is handled with care since they can occur late in the optimization and therefore close to the solution. Simply terminating the population member may not lead to the most desirable outcomes.

The case where an invalid shape is produced almost never occurs beyond the third generation. Simply applying a large cost to this population member thereby terminating it is an acceptable method of handling this violation. Also where the boundary layer solver fails to obtain convergence has to be terminated without a much more in depth study of the boundary layer solver. Such instances are evident for thicker airfoils at lower Reynolds number where shapes exhibit larger curvature and thus the growth rates of the boundary layers become more difficult for the boundary layer solver to compute.

The third instance where a penalty function was necessary was when flow separation occurred at the trailing edge. Here the boundary layer solver did not accurately predict the pressure drag that resulted from this separation but simply stated that since there was separated flow then skin friction was zero and thus drag was zero. The result of this is that large wedge angles are permitted and thus fat trailing edges to be produced. It cannot be accurately determined what the correct effect on drag this would be but nor can it be allowed to happen. The approach taken in this thesis is to impose a weighted penalty on the separation. That is, small separation would result in a small penalty and large separation would result in a large penalty. The penalty is then calculated on the local pressure acting in the separated zone. This added pressure drag is then added to the calculated drag coefficient as described in the following equation. This is summed over every panel where separation occurs starting at the trailing edge and working towards the leading edge.

$$c_d = c_d + c_p \cdot dy \qquad (4.1)$$

4.3 Convergence

The importance of convergence and the method of obtaining convergence was introduced in §2.4. Fundamentally convergence is obtained when ongoing change or evolution in a population ceases. To extract this from a population poses the challenge mostly because the behaviour of an evolving population varies from one optimization problem to another. The most common problem faced in establishing a convergence criterion is that of premature convergence. Terminating the solution early may simply have resulted in a good solution being obtained, but assurance that the optimal set of

parameters has been located cannot be given. On the flip side setting a convergence criteria too strictly can result in wasted time and energy.

In an application where numerical solving techniques are used such as the panel method boundary layer solver used in this problem, the discussion of accuracy and precision arises. Running applications with single precision real numbers leads to solutions with up to seven decimal places. For the numerical type problem considered here, the level of precision obtained simply far exceeds the accuracy of these values. Errors arise from the assumptions made in solving the Navier—Stokes equations of the flow field, from modelling the turbulence and from rounding errors to name a few. Thus it is important to understand that optimal configurations may be overridden by some other slightly less optimal ones. There is no easy way to avoid this error other than to reduce it by performing the experiment as many times as possible in order to obtain a large enough number of data points. It is this approach that is employed in the work of this thesis to address the issue of accuracy. The precision of calculated values is then still used to determine if one configuration outperforms another.

In order to establish a convergence and stopping criterion it is important to observe the behaviour of a population set. Figure 4.2 - Figure 4.5 illustrates a typical example of a converging population of a two-dimensional problem with parameters λ_1 and λ_2. While the present problem contains more degrees of freedom the behaviour is representative of the current optimization problem considering the x-coordinate of the crest and the wedge angle; the two dominant parameters. Figure 4.2 shows a typical difference vector distribution in the initial population where the difference vector is measured from the normalized origin to the data point. The behaviour shows the distribution of difference

vectors trending about a main minimum and other local minima which is observed in early generations less than 10 as seen in Figure 4.3. The number of other local minima varies between zero and three from one simulation to the next. As a population advances (Figure 4.4) the area of exploration tends to decrease further and further until the point of convergence is reached as seen in Figure 4.5. An important characteristic to note is that the final solution converges to a single point. Mathematically this is explained by equation (2.12) and thus any further mutations have no effect on generating a new population.

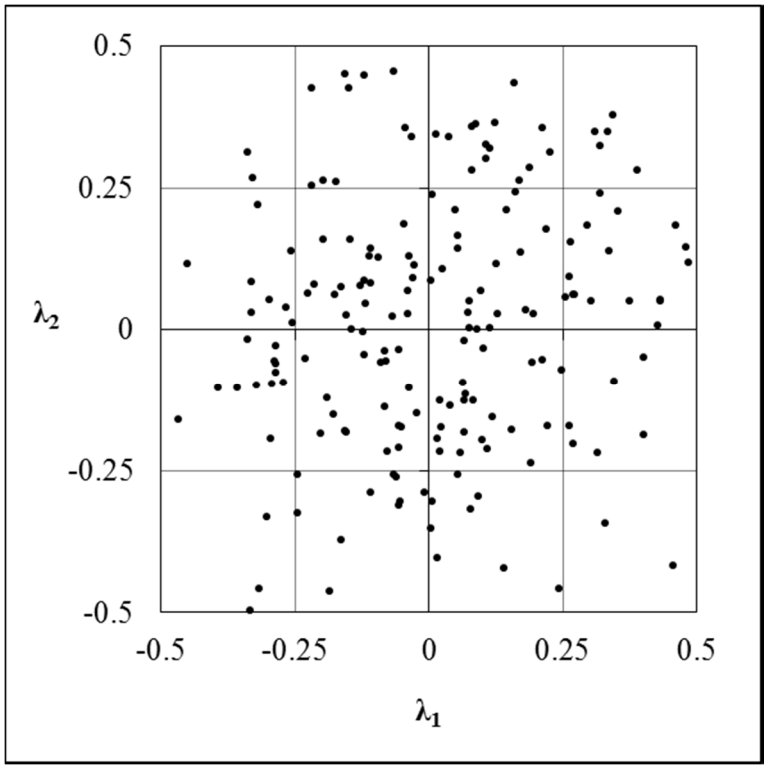


Figure 4.2 - Normalized distribution of difference vectors – Generation 1

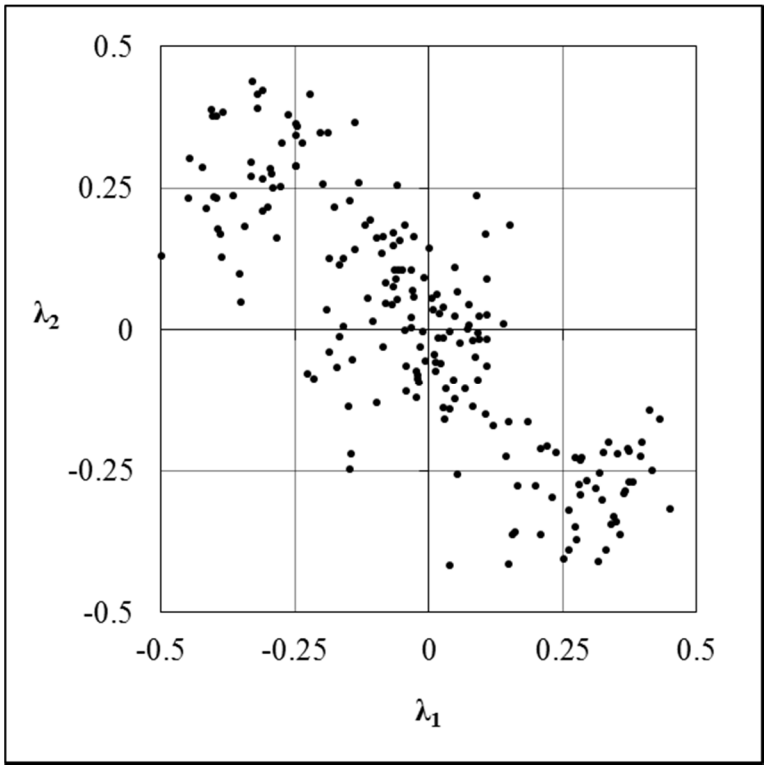


Figure 4.3 - Normalized distribution of difference vectors – Generation 5

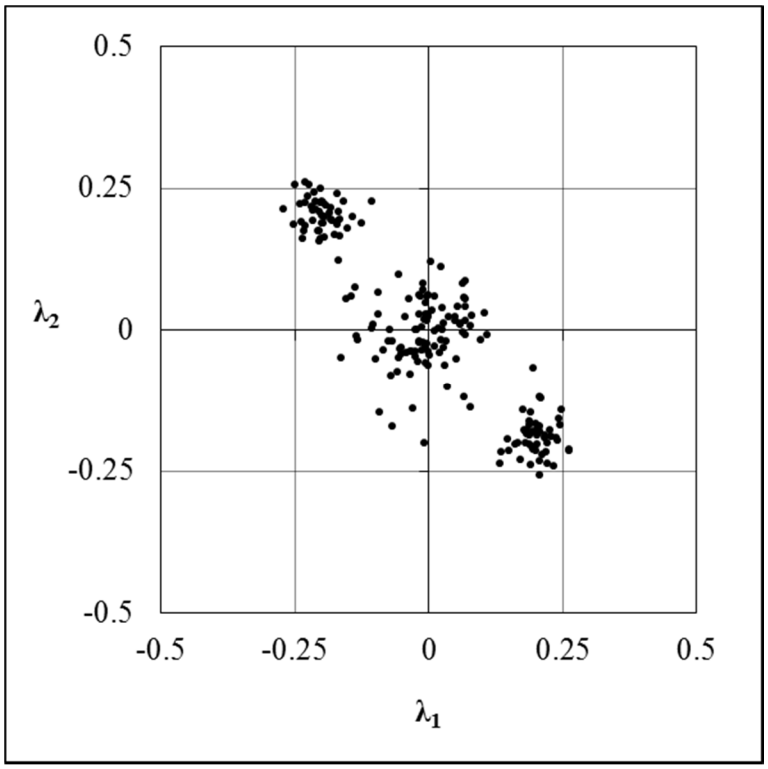


Figure 4.4 - Normalized distribution of difference vectors – Generation 25

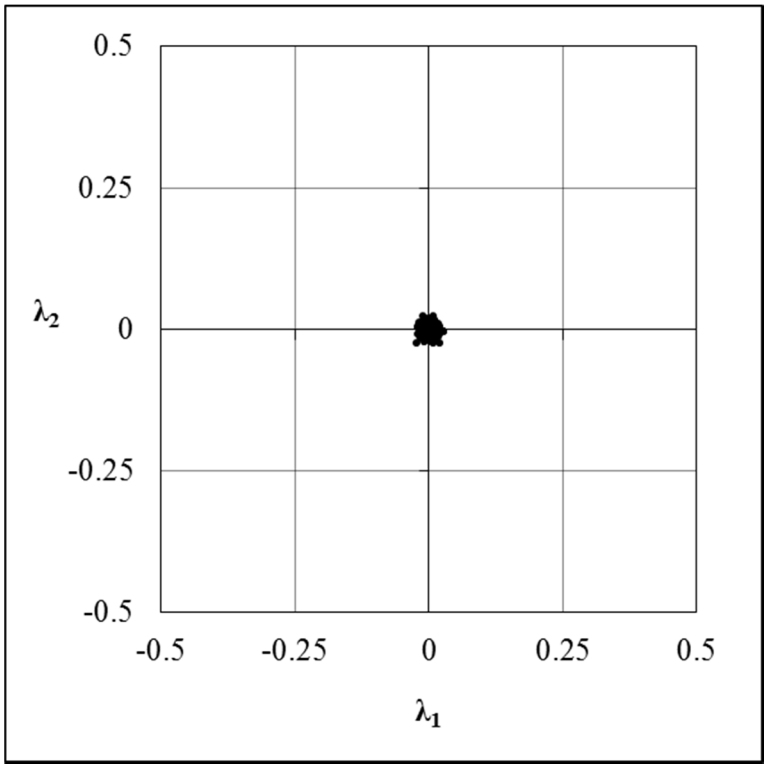


Figure 4.5 - Normalized distribution of difference vectors – Converged

Setting a stopping criterion based on precision involves specifying the maximum length of a difference vector between the highest cost vector and the lowest cost vector. This value can be determined by experimenting with a population in a guess and check manner. The optimization performed in this thesis used an absolute value of 1e-06 for all scenarios. The acceptance of this criterion is then established by inspecting the final population and noting that any further advancement in evolution will only result in negligible improvements.

Other convergence checks were explored such as monitoring the best performing vector in a population for each generation. However in this case no new improvement could be made to a vector for many generations before a sudden jump determined a new minimum. Thus using this criterion proved to be unpredictable and thus could not confidently rule out premature convergence.

5 Test Input

In order to develop an optimized airfoil it is necessary to initialize all the input variables for the space in which the airfoil will be place in. This involves setting the flow field parameters as well as setting a shape constraint.

5.1 Flow field variables

There are three variables that need to be set to completely define a flow field for flow over an airfoil. They are the Reynolds number, the Mach number and the angle of attack or the alignment of the airfoil to the flow.

Since this application deals with non-lifting airfoils and by nature these airfoils are symmetric, this implies that the angle of attack is zero degrees. The Mach number is an arbitrary number as both the Reynolds number and the Mach number account for the freestream velocity of the flow. Thus the specification of the flow field velocity can be left up to the Reynolds number definition and specification of the Mach number can be ignored. It is important to be reminded that incompressible flow conditions can be assumed for flows with Mach numbers less than 0.3. The Mach number that was chosen for the flow simulations in this optimization problem was 0.1.

Reynolds numbers were selected in the low speed to mid speed range to match that of the appropriate physical domain described in §3.1. 15 chosen values were taken in the range of Re = 50,000 to Re = 15,000,000 roughly equally spaced on a logarithmic scale.

5.2 Shape constraints

The use of shape constraints is necessary to ensure that the desired geometric properties of one airfoil are the same as that of another so that apples are compared with

apples. There are two types of geometric constraint problems described in the thesis; an area constraint and a second moment of inertia constraint.

Since a panel method is used to model each airfoil, it is convenient to view the airfoils in the same manner that the flow solver does. Taking them to be polygons Green's Theorem can be easily employed to determine the area of the airfoil as follows.

$$A = \frac{1}{2}\oint_C (-ydx + xdy)$$
$$A = \frac{1}{2}\left|\sum_{i=0}^{n-1}(-y_i x_{i+1} + x_i y_{i+1})\right| \qquad (5.1)$$

For the second moment of inertia a similar summation can be used. The summation shown below is simply a combination of the second moments of inertia of simple triangles – known as the method of composite shapes.

$$I_{xx} = \frac{1}{12}\sum_{i=1}^{n}(x_{i+1} - x_i)(y_{i+1} + y_i)\left(y_{i+1}^{\ 2} + y_i^{\ 2}\right) \qquad (5.2)$$

To complete the constraint it is necessary to introduce an amplification parameter. This parameter is added to the list of other airfoil parameters by which any specific airfoil shape is defined. The purpose of the amplification parameter is to modify the shape of the

$$\lambda_{Area} = \frac{A_{Desired}}{A_{Calculated}}$$
$$\lambda_{Ixx} = \sqrt[3]{\frac{I_{xx\_Desired}}{I_{xx\_Calculated}}} \qquad (5.3)$$

current test airfoil so that the calculated geometric properties, either area or second moment of inertia, are set equal to a desired value. The amplification parameter as calculated in formula (5.3) is then multiplied to the y-coordinates of the airfoil shape.

Modification can only be made to the y-coordinates since any alteration to the x-coordinates would change both the non-dimensionally chord-dependent terms as well as the Reynolds number. A summary of the complete procedure for implementing this constraint is shown in the following figure.

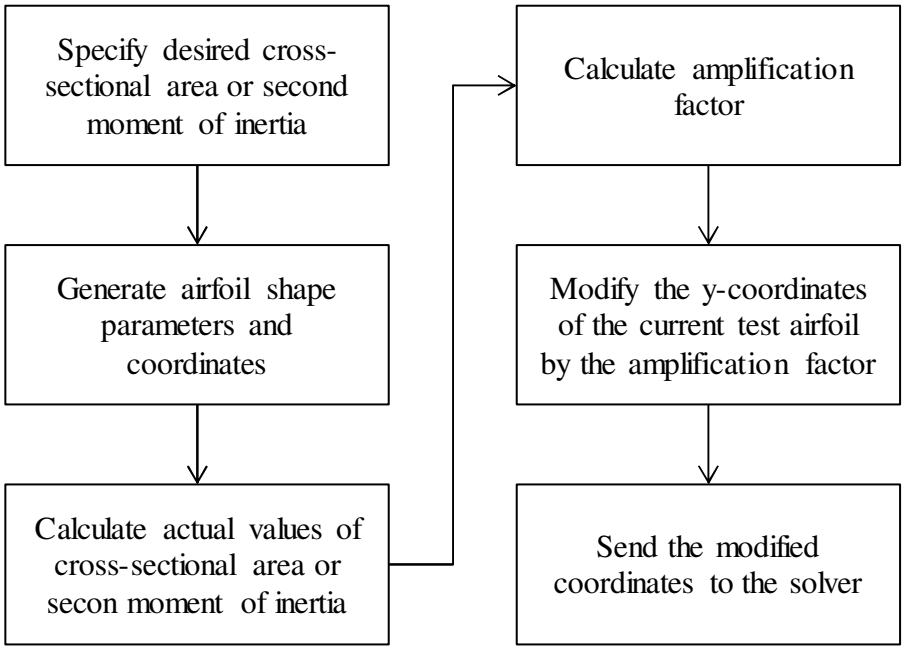


Figure 5.1 – Method of employing shape constraint

Optimized airfoil shapes for five different cross-sectional areas were used as well as for five different second moments of inertia. The following tables show the complete test matrix for the optimized results obtained in this thesis. An explanation of blanks in the matrix was explained in §4.2 of this thesis.

		Cross-sectional area (%-c²)				
		5	7.5	10	12.5	15
Reynolds Number	50,000	√				
	75,000	√				
	100,000	√				
	150,000	√				
	200,000	√	√			
	300,000	√	√			
	500,000	√	√	√		
	700,000	√	√	√		
	1,000,000	√	√	√		
	1,500,000	√	√	√		
	2,000,000	√	√	√		
	3,000,000	√	√	√	√	
	5,000,000	√	√	√	√	√
	10,000,000	√	√	√	√	√
	15,000,000	√	√	√	√	√

Table 5-1 – Test matrix of cross-sectional area constrained airfoils

		Moment of Inertia (%-c^4)				
		1.0E-05	2.5E-05	6.0E-05	1.0E-04	2.5E-04
Reynolds Number	50,000	√				
	75,000	√				
	100,000	√				
	150,000	√				
	200,000	√	√			
	300,000	√	√	√	√	
	500,000	√	√	√	√	
	700,000	√	√	√	√	
	1,000,000	√	√	√	√	
	1,500,000	√	√	√	√	
	2,000,000	√	√	√	√	√
	3,000,000	√	√	√	√	√
	5,000,000	√	√	√	√	√
	10,000,000	√	√	√	√	√
	15,000,000	√	√	√	√	√

Table 5-2 – Test matrix of second moment of inertia constrained airfoils

6 Results and Discussions

6.1 Struts loaded axially

Optimized airfoil shapes for axially loaded struts were produced according to the test matrix shown in Table 5-1. The following figures show some of the shapes that were obtained. A complete list of optimized shapes can be found in Appendix A of this thesis.

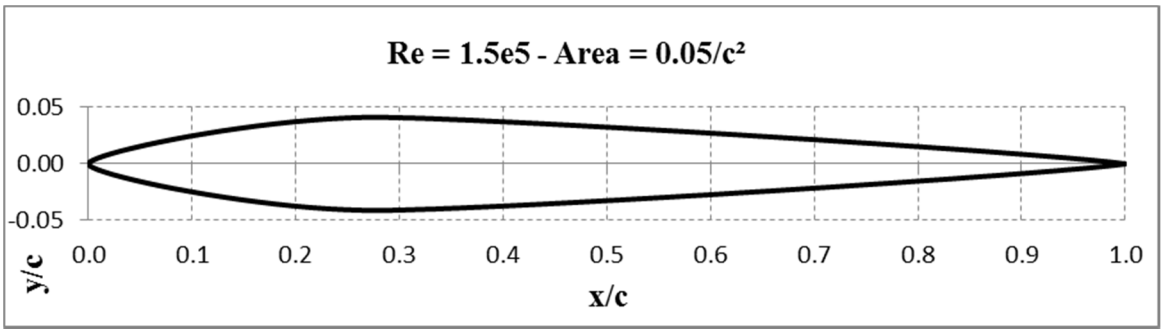


Figure 6.1 – 0.05/c² area constrained airfoil at Re = 150,000

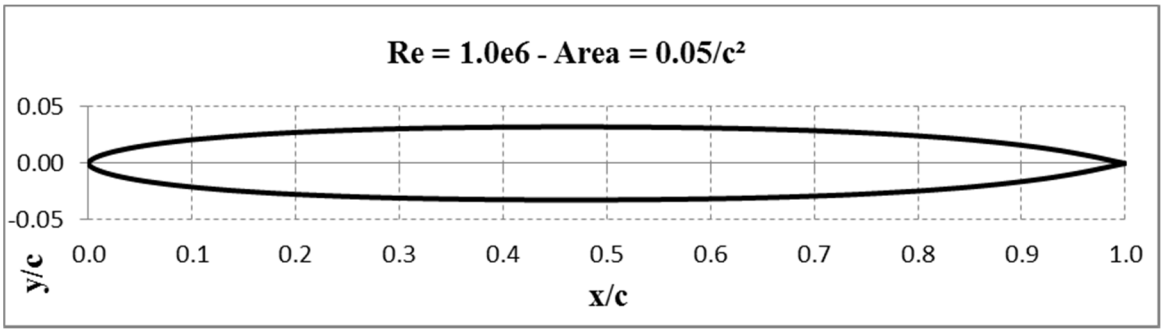


Figure 6.2 – 0.05/c² area constrained airfoil at Re = 1,000,000

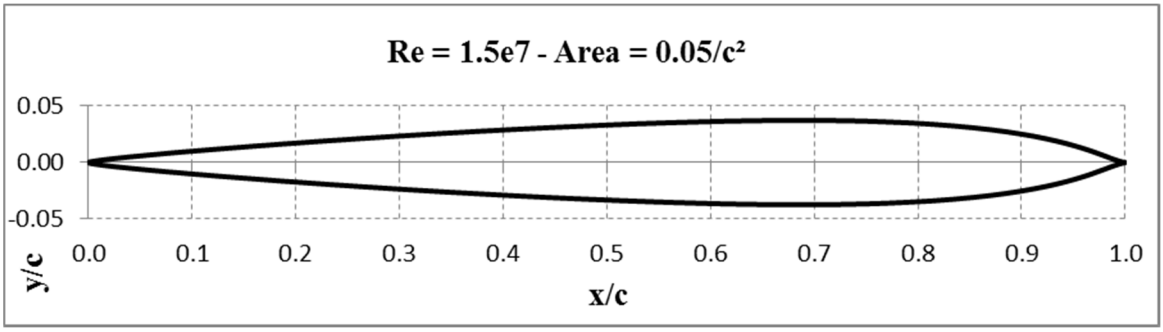


Figure 6.3 – 0.05/c² area constrained airfoil at Re = 15,000,000

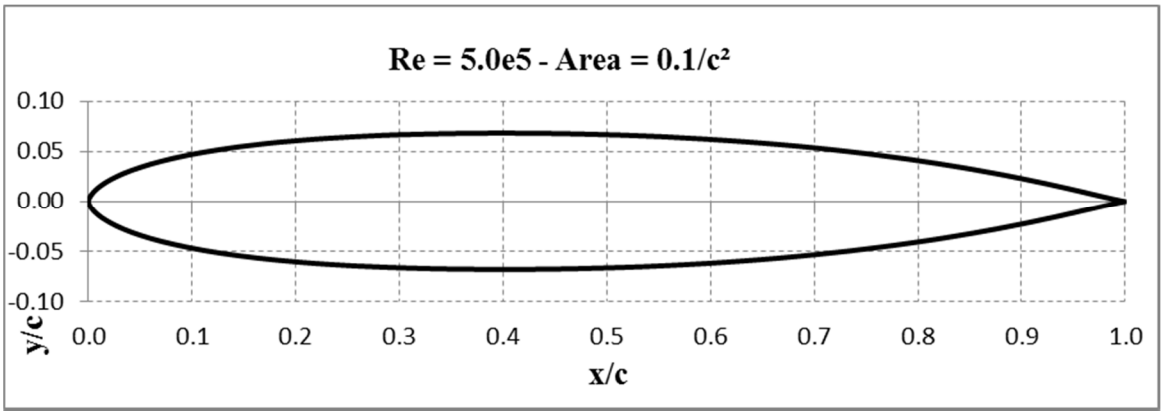


Figure 6.4 – 0.1/c² area constrained airfoil at Re = 500,000

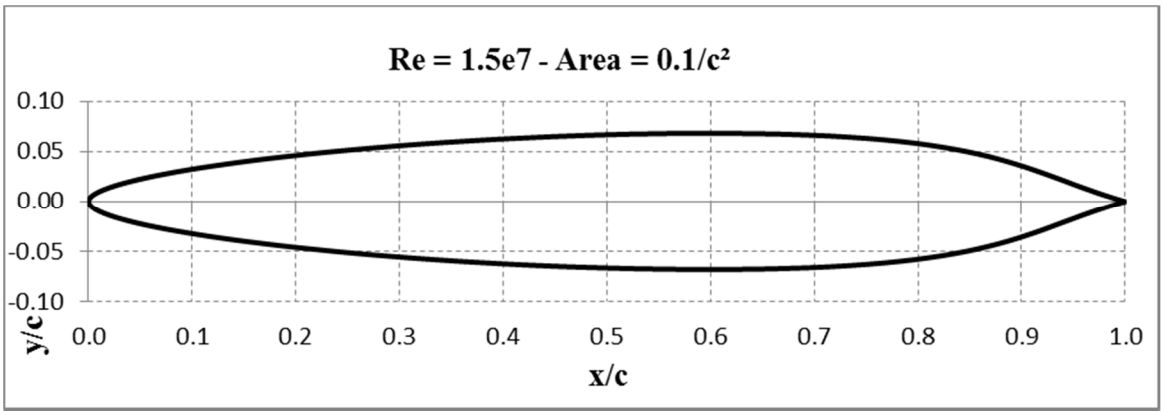


Figure 6.5 – 0.1/c² area constrained airfoil at Re = 15,000,000

The crux of the results is the behaviour of the airfoil's crest, predominantly its lateral position along the chord length. For low Reynolds numbers this appears close to the leading edge and as the Reynolds number is increased this gets pushed back towards the trailing edge. This behaviour has a direct effect on both the radius of the leading edge as well as the trailing edge wedge angle, that is, the position of the crest determines the size of these geometric properties. It can be seen in Figure 6.7 that the crest behaviour acts in a regular manner in the thickness direction with only a slight variation present. Lateral positioning of the thickness crest on the other hand as shown in Figure 6.6 increases in a linear manner for Reynolds numbers above 500,000 noting a sudden jump in the vicinity

of Reynolds numbers 300,000-500,000. For low Reynolds numbers (<300,000) for the limited data points, the crest location shows minimal variation. One further observation from Figure 6.6 is that the lateral crest location decreases with a proportional increase in area size. This is due to the thicker airfoils producing too bluff of a trailing edge resulting in flow separation and to counter this effect, the crest position is moved forward.

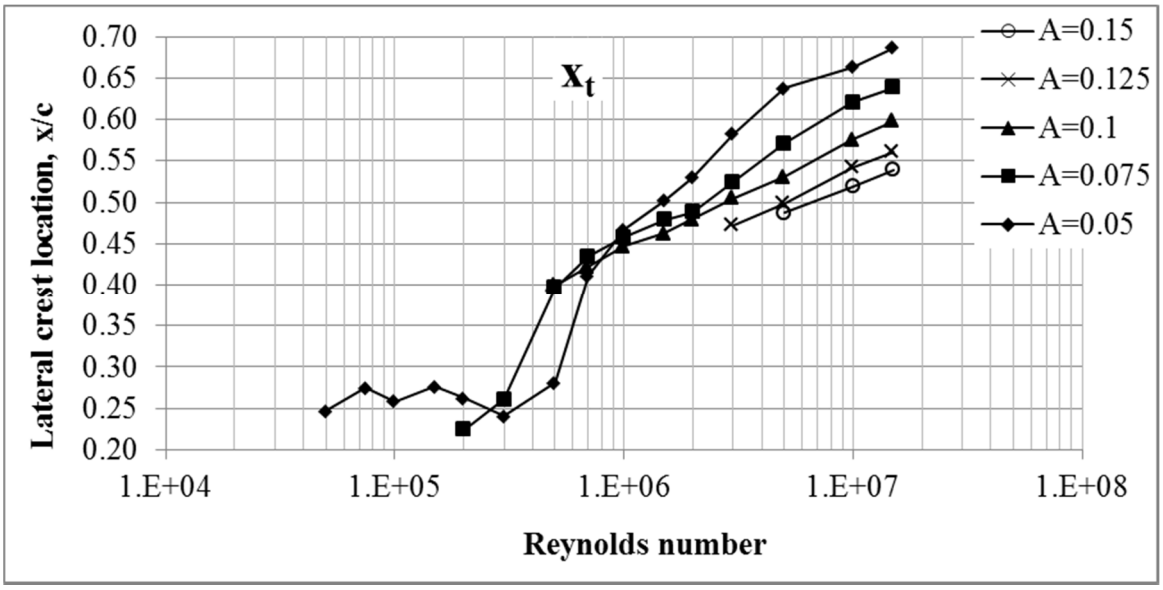


Figure 6.6 – Area constrained crest lateral position trends

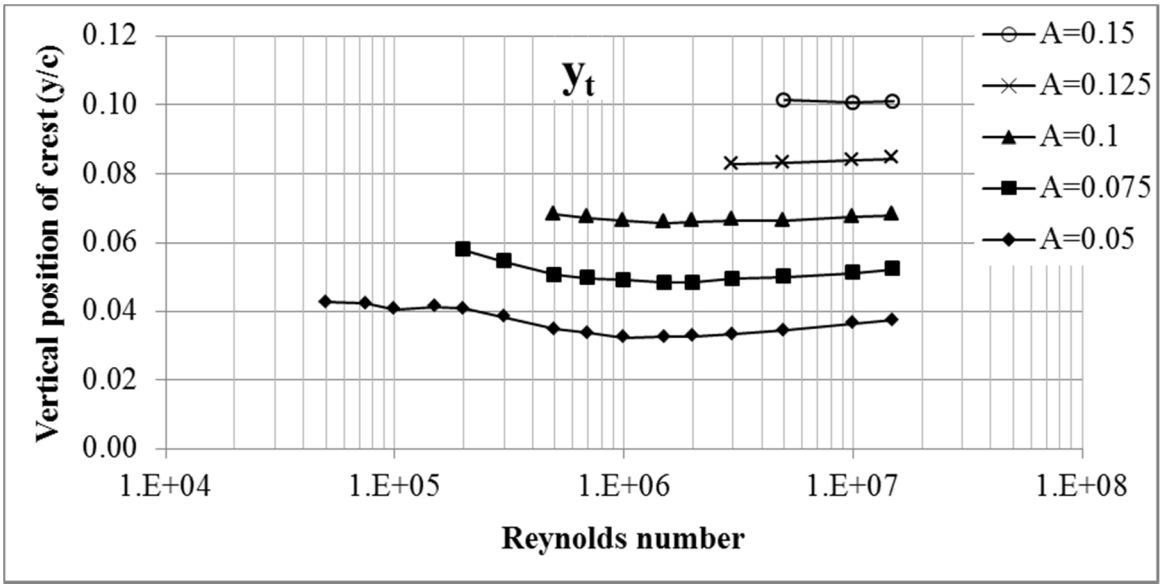


Figure 6.7 – Area constrained crest max thickness position trends

In order to explain the transition from the low variation of the crest at low Reynolds numbers to the higher linear variation at higher Reynolds numbers it is necessary to compare plots of the crest location versus the transition point of the flow from laminar to turbulent.

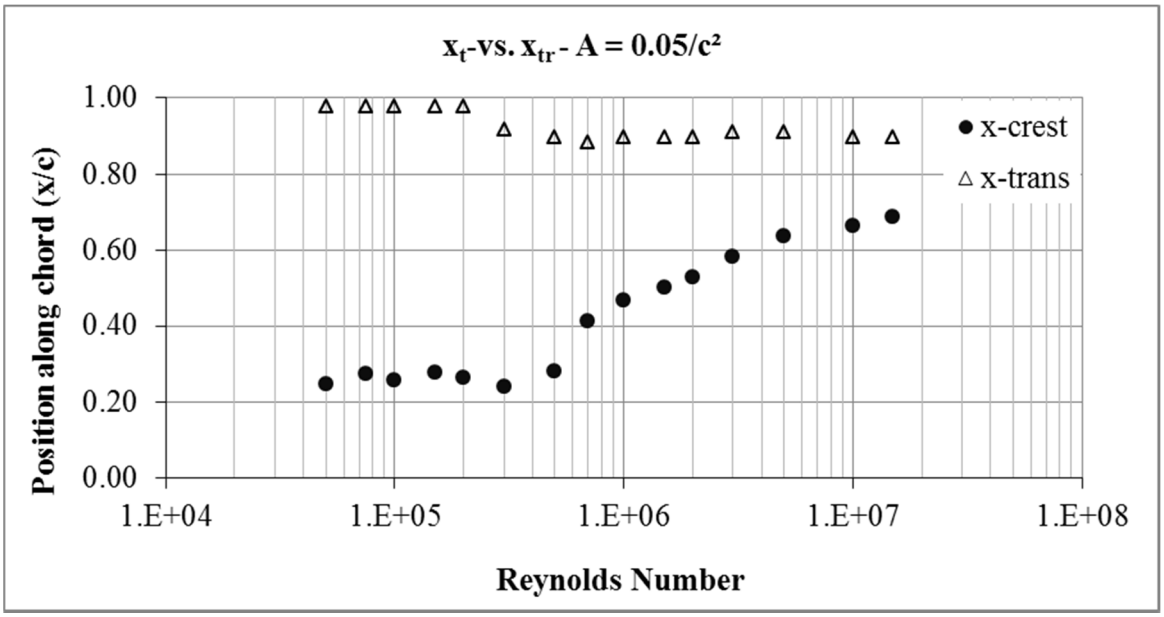


Figure 6.8 – Crest location versus flow behaviour change, A = 0.05/c²

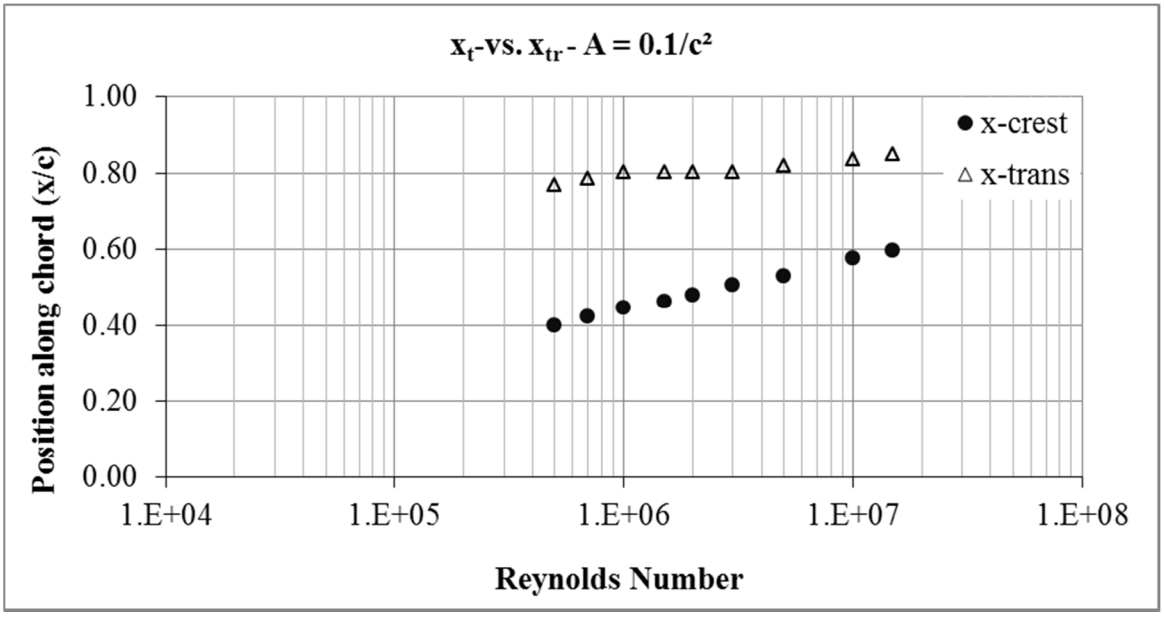


Figure 6.9 – Crest location versus flow behaviour change, A = 0.1/c²

The previous figures illustrate the transition behaviour. The upper transition points show that most of the flow is laminar and that turbulence is only experienced near the trailing edge. The large laminar portion is something to be expected since a drawback of turbulent flow conditions is a large increase in drag. Hence it can be seen that the crest location influences the point of transition and that as turbulence threatens to appear it is countered by a shift in the crest location toward the trailing edge.

The following tables provide the parameters for the optimized airfoils with area constraints. Note that amplification factors have not been applied to the applicable parameters. Some rough trends do exist within the data other than the already described location of the thickness crest. The radii of the leading edge as well as crest curvature decrease as the crest in moved towards the middle of the chord and then decrease as the chord approaches the trailing edge. A more pronounced trend for the wedge angle exists as well but cannot be easily interpreted from the data tables.

Re	r_{le}	k_t	dz_{te}	β_{te}	x_t	y_t	λ_{amp}
50000	-0.002519	-6.868816	0.000000	0.492429	0.246717	0.158438	0.268801
75000	-0.002876	-6.379561	0.000000	0.513266	0.274564	0.183056	0.230030
100000	-0.020582	-5.910243	0.000000	0.578309	0.258680	0.186871	0.217076
150000	-0.014102	-6.972657	0.000000	0.512162	0.276238	0.171218	0.240415
200000	-0.022304	-7.438792	0.000000	0.760594	0.262587	0.217364	0.187063
300000	-0.067977	-4.080060	0.000000	0.911123	0.240680	0.205366	0.185914
500000	-0.058338	-4.205771	0.000000	0.591142	0.280748	0.114943	0.301380
700000	-0.067025	-2.909892	0.000000	0.753031	0.411105	0.141302	0.237271
1000000	-0.043578	-0.585990	0.000000	0.773480	0.467545	0.192045	0.168220
1500000	-0.036563	-0.656844	0.000000	0.864011	0.502136	0.207573	0.156358
2000000	-0.037521	-0.766887	0.000000	0.940701	0.529959	0.224622	0.145776
3000000	-0.020658	-0.667966	0.000000	0.809142	0.582642	0.168696	0.197165
5000000	-0.025818	-1.156174	0.000000	0.962910	0.637599	0.209139	0.163751
10000000	-0.003604	-1.447693	0.000000	0.769877	0.663925	0.189427	0.191830
15000000	-0.002754	-2.069237	0.000000	0.776694	0.686985	0.223863	0.166647

Table 6-1 – Area = $0.05/c^2$ airfoil parameters

Re	r_{le}	k_t	dz_{te}	β_{te}	x_t	y_t	λ_{amp}
200000	-0.052732	-4.294185	0.000000	0.421937	0.224382	0.135979	0.425980
300000	-0.063409	-2.294787	0.000000	0.436156	0.261091	0.123238	0.442622
500000	-0.057893	-0.996493	0.000000	0.722971	0.396383	0.236019	0.215280
700000	-0.043888	-0.631110	0.000000	0.635661	0.434334	0.177566	0.280030
1000000	-0.038306	-0.538613	0.000000	0.575764	0.457845	0.166764	0.294900
1500000	-0.051151	-0.436778	0.000000	0.614307	0.478804	0.150186	0.323413
2000000	-0.036067	-0.410556	0.000000	0.565738	0.488091	0.141067	0.344168
3000000	-0.050488	-0.741281	0.000000	0.793285	0.524377	0.209869	0.236196
5000000	-0.041133	-0.794287	0.000000	0.779777	0.570870	0.195409	0.256720
10000000	-0.017172	-0.781629	0.000000	0.656565	0.620508	0.150205	0.341772
15000000	-0.021720	-1.259596	0.000000	0.812148	0.638449	0.205824	0.254282

Table 6-2 – Area = $0.075/c^2$ airfoil parameters

Re	r_{le}	k_t	dz_{te}	β_{te}	x_t	y_t	λ_{amp}
500000	-0.029072	-0.825989	0.000000	0.453728	0.399649	0.198059	0.344061
700000	-0.033988	-0.817892	0.000000	0.498120	0.422191	0.214324	0.314198
1000000	-0.065444	-0.628084	0.000000	0.572768	0.447010	0.188223	0.352194
1500000	-0.056683	-0.503651	0.000000	0.537844	0.462686	0.161323	0.407866
2000000	-0.062322	-0.615209	0.000000	0.614404	0.479854	0.187346	0.353140
3000000	-0.045780	-0.579015	0.000000	0.579806	0.505333	0.163701	0.406857
5000000	-0.055820	-0.642176	0.000000	0.644684	0.530774	0.185795	0.357551
10000000	-0.044748	-0.841228	0.000000	0.710019	0.575512	0.195579	0.345269
15000000	-0.038821	-0.948918	0.000000	0.727409	0.598597	0.203271	0.335023

Table 6-3 – Area = $0.1/c^2$ airfoil parameters

Re	r_{le}	k_t	dz_{te}	β_{te}	x_t	y_t	λ_{amp}
3000000	-0.055861	-0.435139	0.000000	0.451274	0.472891	0.145977	0.566373
5000000	-0.040019	-0.457176	0.000000	0.436819	0.499381	0.144301	0.576584
10000000	-0.070045	-0.768936	0.000000	0.666015	0.542581	0.208949	0.401323
15000000	-0.061488	-0.905359	0.000000	0.705742	0.561395	0.229704	0.367549

Table 6-4 – Area = $0.125/c^2$ airfoil parameters

Re	r_{le}	k_t	dz_{te}	β_{te}	x_t	y_t	λ_{amp}
5000000	-0.074316	-0.664655	0.000000	0.501256	0.487629	0.196657	0.515702
10000000	-0.072789	-0.653252	0.000000	0.554081	0.518905	0.200289	0.502154
15000000	-0.040933	-0.586177	0.000000	0.485582	0.539550	0.169367	0.596148

Table 6-5 – Area = $0.15/c^2$ airfoil parameters

It can be noted that for the wedge angle minute geometric inconsistencies occur such as having a very large wedge angle. These high values cause flow separation to occur at the trailing edge but contained to a very small region. This suggests that for these area constrained airfoils the optimized shapes are located on the boundary between attached and separated flow. To better see the behaviour of the wedge angle it is possible to manually measure the angle on a more macro scale. Figure 6.10 shows the resulting relationship obtained

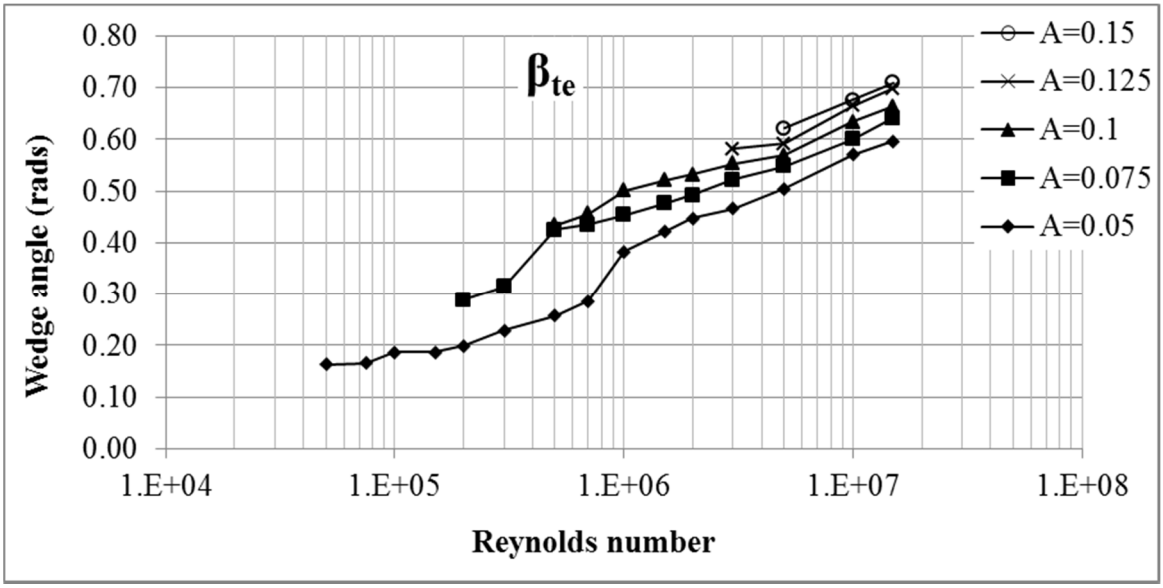


Figure 6.10 - Area constrained wedge angle trends

In addition to assessing the shapes produced it is also possible to review the behaviour of the optimization strategy. This allows for sanity checks to be completed in order to provide a high level of confidence in the resulted shapes. One method of assessing the shapes is to look at the trends of the costs that are produced. Figure 6.11 shows the plots of all of the area constrained airfoil optimizations. It can be noticed that smooth asymptotic curves are formed for each set of airfoil sizes. From an optimization point of view there are trends to assess there as well. It is expected that a period of time

(generations) is spent in an exploration phase and then a time that works towards convergence. This behaviour as shown in Figure 6.12, Figure 6.13, and Figure 6.14 confirms the predicted behaviour illustrated in Figure 4.2 - Figure 4.5.

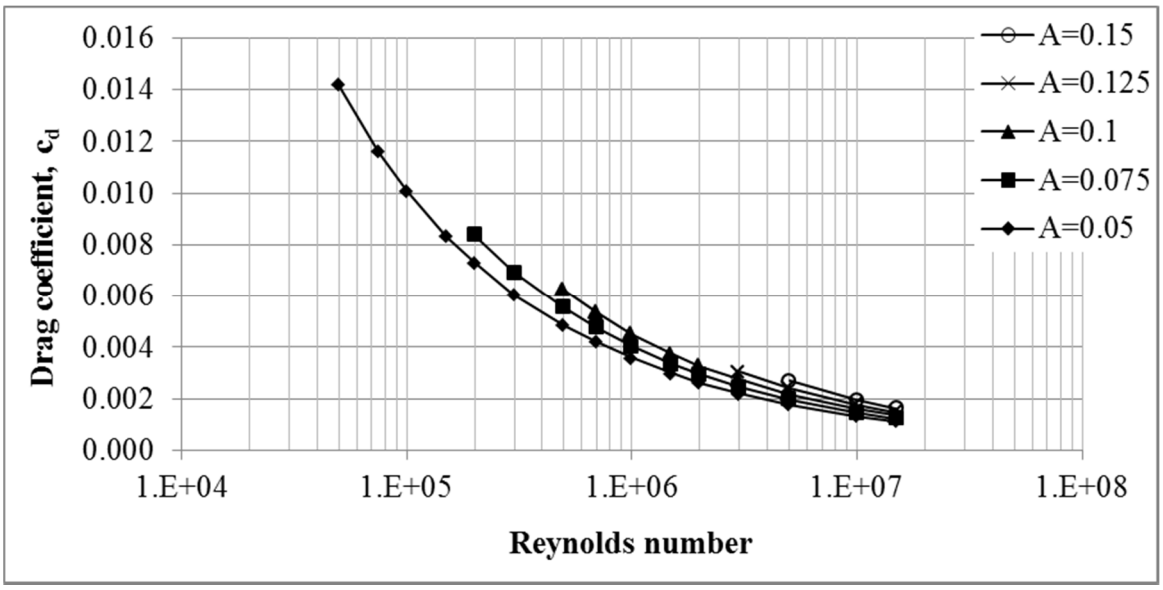


Figure 6.11 – Cost plots of optimized shapes

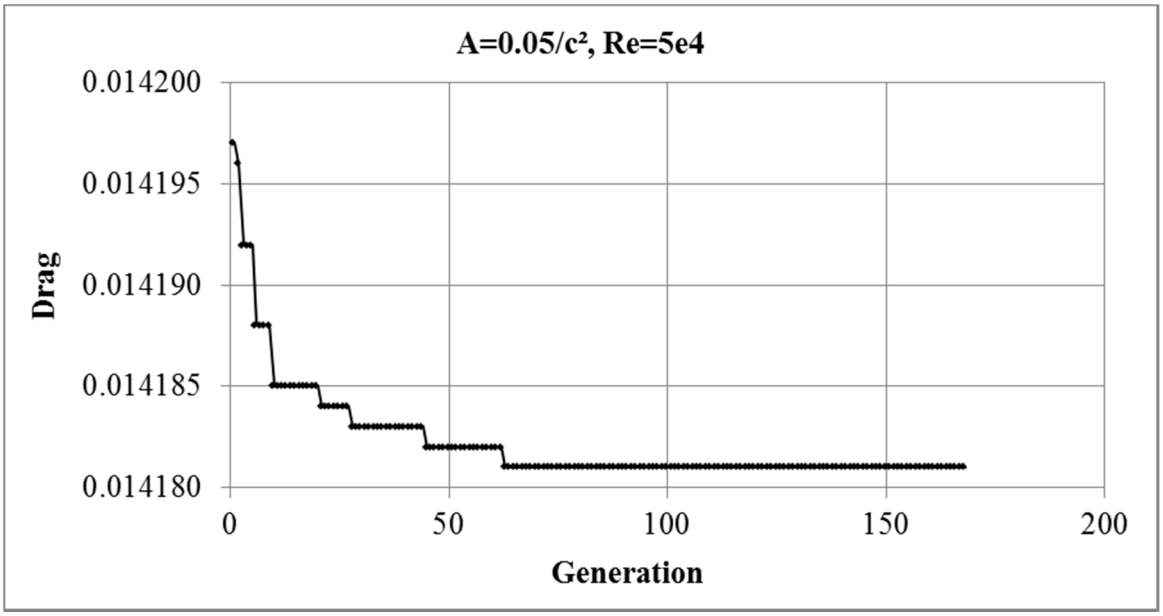


Figure 6.12 – Cost of best performing vector vs generation - A=0.05/c², Re=5e4

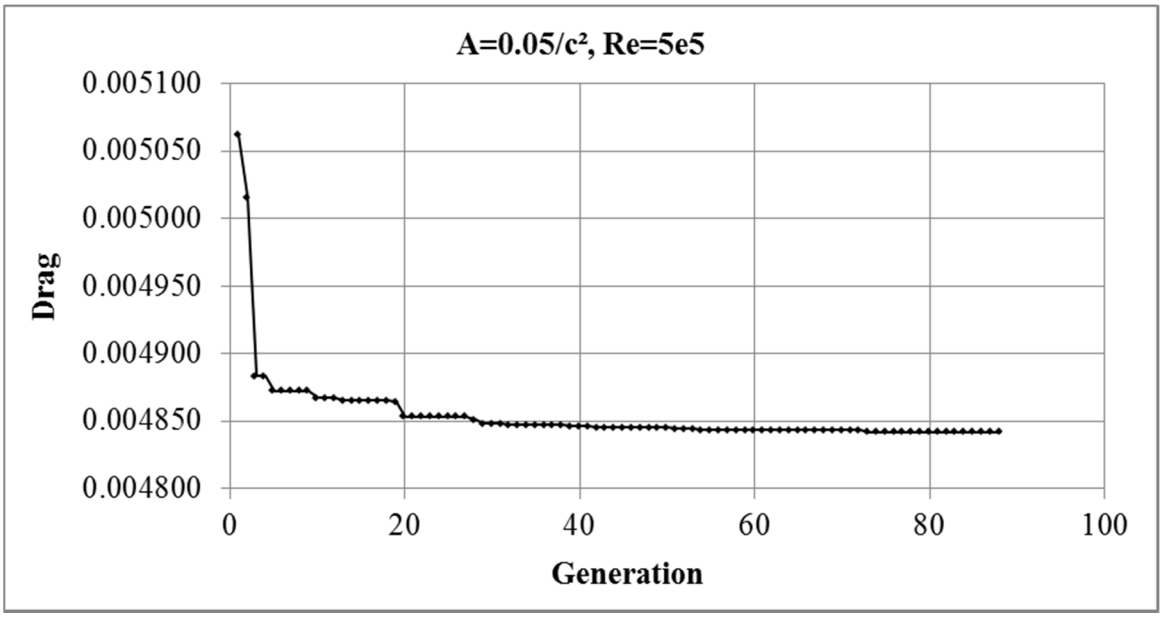


Figure 6.13 – Cost of best performing vector vs generation - A=0.05/c², Re=5e5

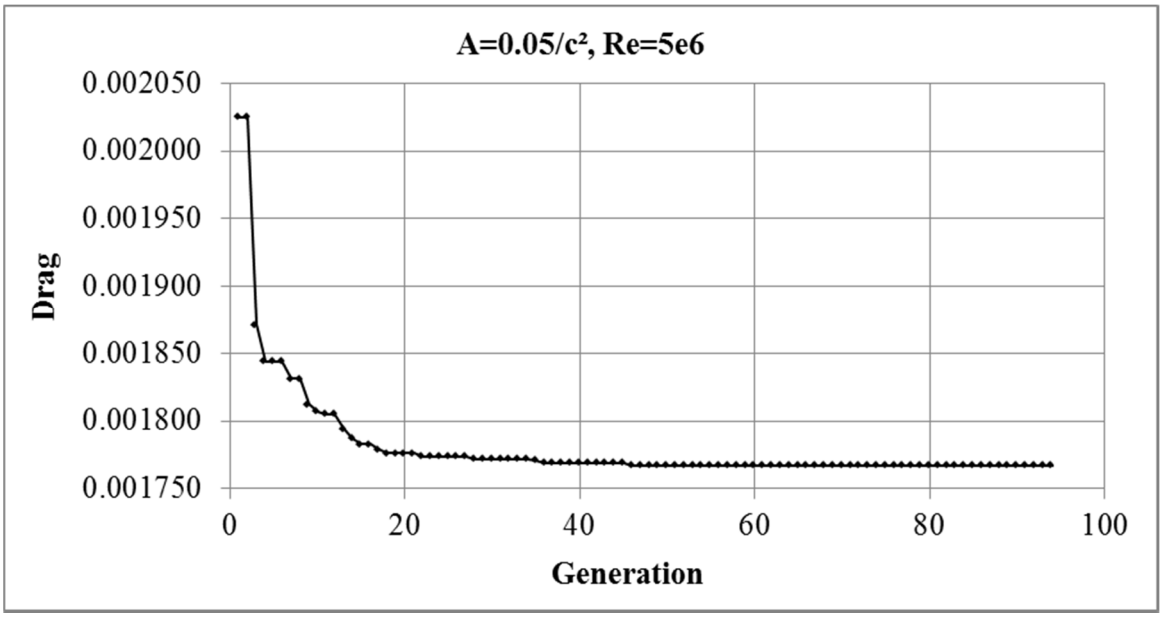


Figure 6.14 – Cost of best performing vector vs generation - A=0.05/c², Re=5e6

It can be noticed that the first 20-25% of the time is spent in the exploration phase and the remainder is spent on convergence. This does indicate that the convergence criterion is strict and inspecting a population 20 generations sooner can usually confirm this. Convergence plots that show the difference vectors trending towards zero indicate

that an optimal solution has been found. These plots resemble the previous set of plots and samples of them are provided as follows. In all cases convergence resembled that which was shown in Figure 4.5.

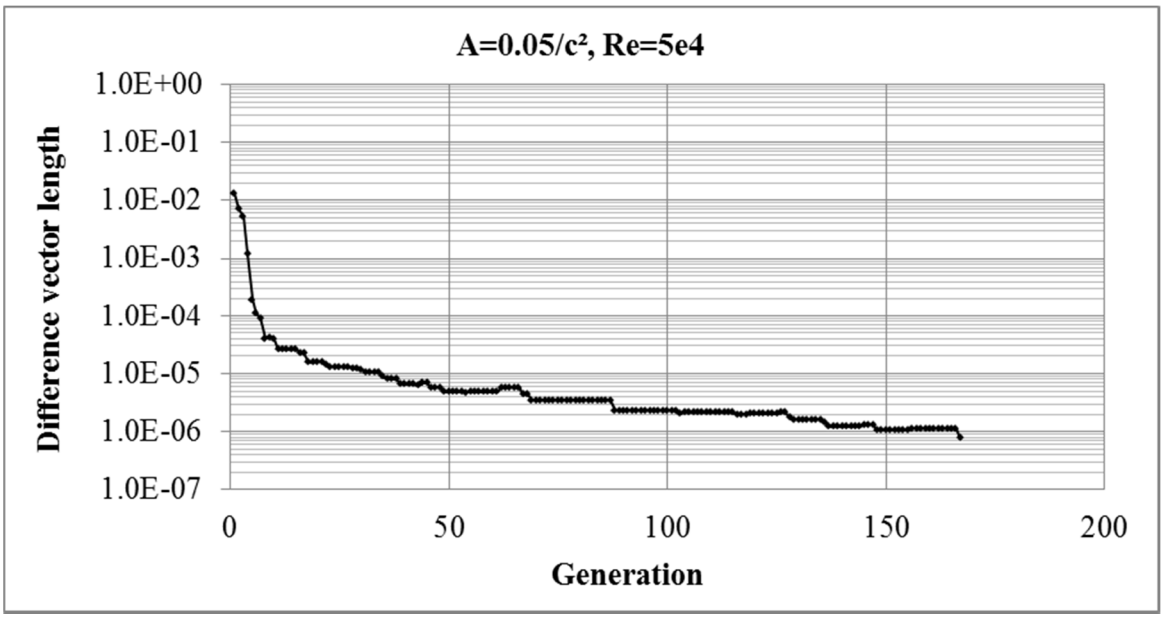


Figure 6.15 – Convergence history, A=0.05/c², Re=5e4

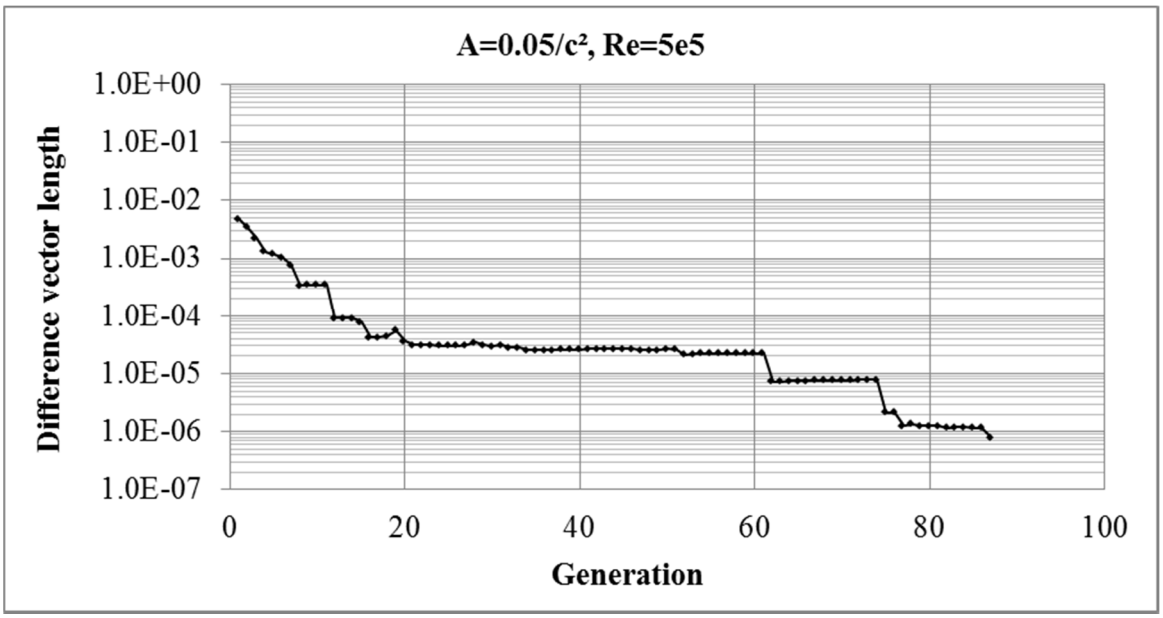


Figure 6.16 - Convergence history, A=0.05/c², Re=5e5

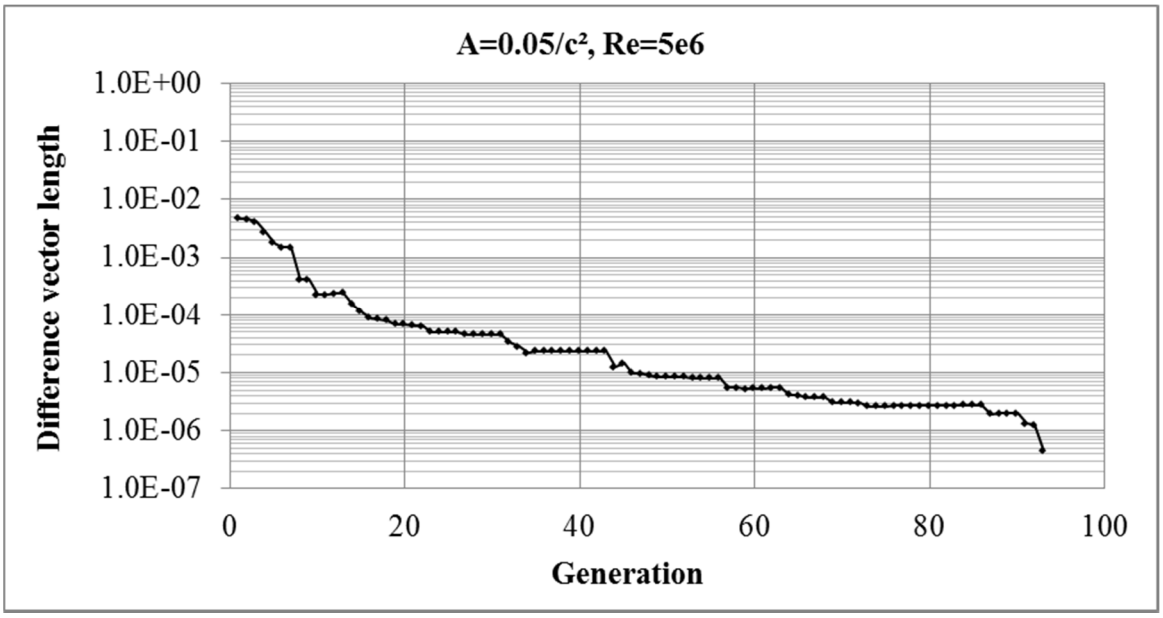


Figure 6.17 - Convergence history, A=0.05/c², Re=5e5

The convergence criteria imposed on the current optimization problem showed no trend between the number of cost evaluations (NCE's) and the corresponding Reynolds number. The following tables show the resulting NCE's for each constrained case.

Re	NCE	c_d
50000	16749	0.01418
75000	23676	0.01160
100000	14466	0.01010
150000	13982	0.00833
200000	18247	0.00731
300000	10475	0.00604
500000	30578	0.00484
700000	13566	0.00420
1000000	22678	0.00358
1500000	13090	0.00299
2000000	9080	0.00262
3000000	15683	0.00220
5000000	9335	0.00177
10000000	7847	0.00133
15000000	8635	0.00113

Table 6-6 – NCE's for A=0.05/c²

Re	NCE	c_d
200000	12452	0.00838
300000	10074	0.00691
500000	6908	0.00556
700000	34719	0.00475
1000000	11526	0.00403
1500000	13870	0.00334
2000000	15291	0.00293
3000000	12407	0.00245
5000000	11006	0.00196
10000000	14057	0.00145
15000000	9141	0.00122

Table 6-7 – NCE's for $A=0.075/c^2$

Re	NCE	c_d
500000	17852	0.00625
700000	11198	0.00534
1000000	10287	0.00451
1500000	12151	0.00373
2000000	12559	0.00327
3000000	12865	0.00273
5000000	18287	0.00217
10000000	12894	0.00160
15000000	10939	0.00134

Table 6-8 – NCE's for $A=0.1/c^2$

Re	NCE	c_d
3000000	16316	0.00303
5000000	17855	0.00241
10000000	12162	0.00177
15000000	11623	0.00148

Table 6-9 – NCE's for $A=0.125/c^2$

Re	NCE	c_d
5000000	16396	0.002683
10000000	13944	0.001939
15000000	14616	0.001624

Table 6-10 – NCE's for A=0.15/c²

While it was not expected that there be a correlation between the NCE's and the Reynolds number it is expected that a direct relationship between the total solve time and the Reynolds number exists. This relationship is due to the time it takes the IBL solver to evaluate a single airfoil. Recall that boundary layer thicknesses are much greater for airfoils at low Reynolds numbers. This effect translated into the total solve times based on the CPU description presented in §4.1 being four times greater for like airfoils at Re = 50,000 than airfoils at Re = 15,000,000.

6.2 Struts loaded in bending

Optimized airfoil shapes for struts loaded in bending were produced according to the test matrix shown in Table 5-2. Many of the same observations and trends were noted for the struts loaded in bending as they were for the axially loaded struts. The following figures show some of the shapes that were obtained. A complete list of optimized shapes can be found in Appendix B of this thesis.

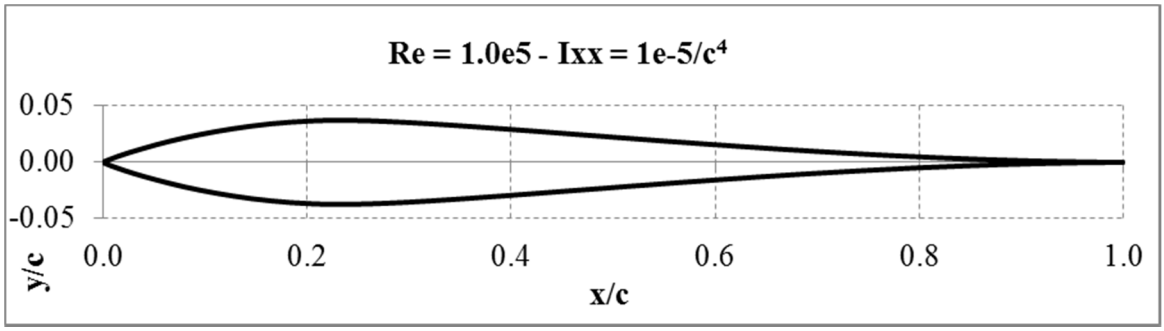


Figure 6.18 – Ixx = 1.0e-5/c^4 constrained airfoil at Re = 100,000

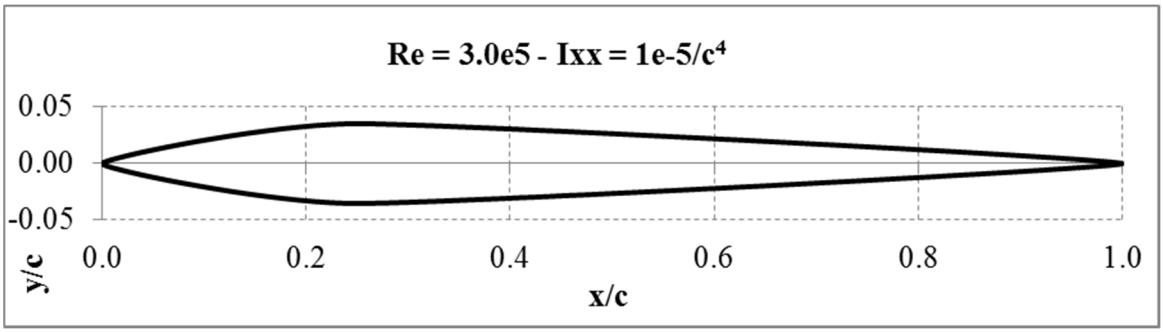


Figure 6.19 – Ixx = 1.0e-5/c^4 constrained airfoil at Re = 300,000

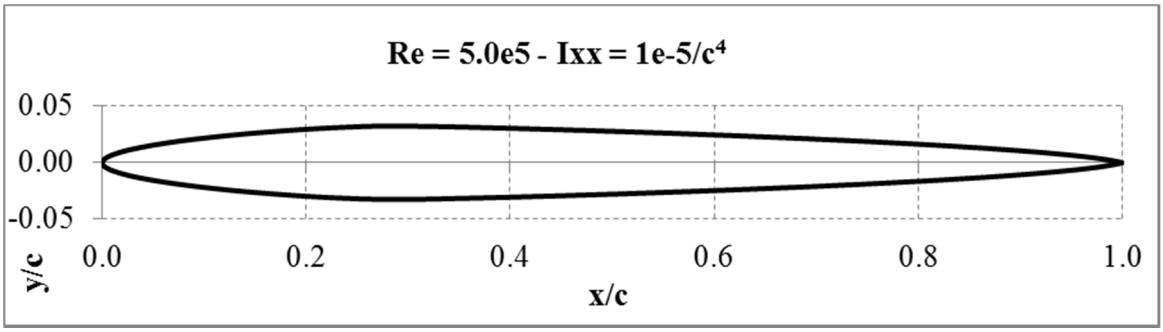


Figure 6.20 – Ixx = 1.0e-5/c^4 constrained airfoil at Re = 500,000

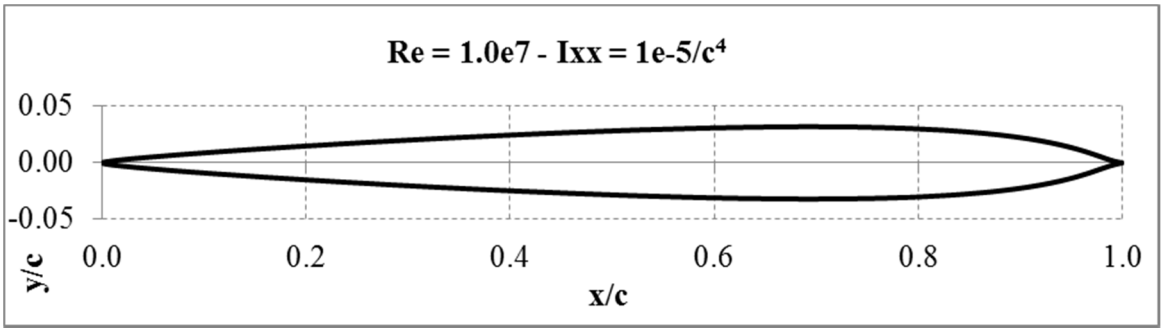


Figure 6.21 – Ixx = 1.0e-5/c^4 constrained airfoil at Re = 10,000,000

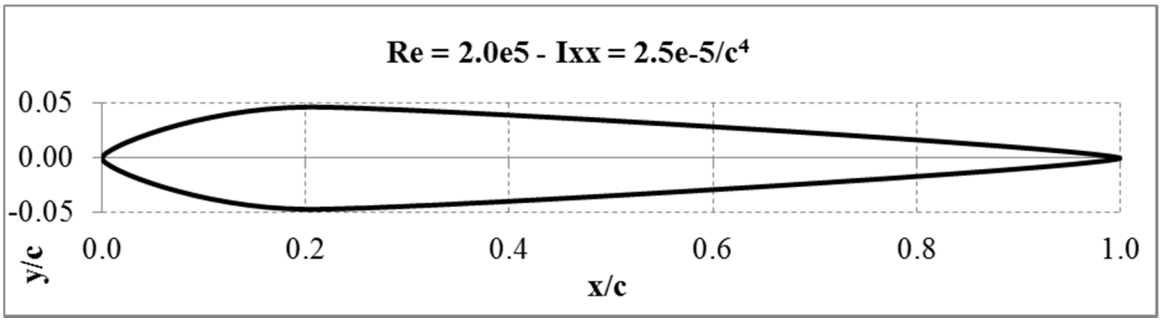


Figure 6.22 – Ixx = 2.5e-5/c^4 constrained airfoil at Re = 200,000

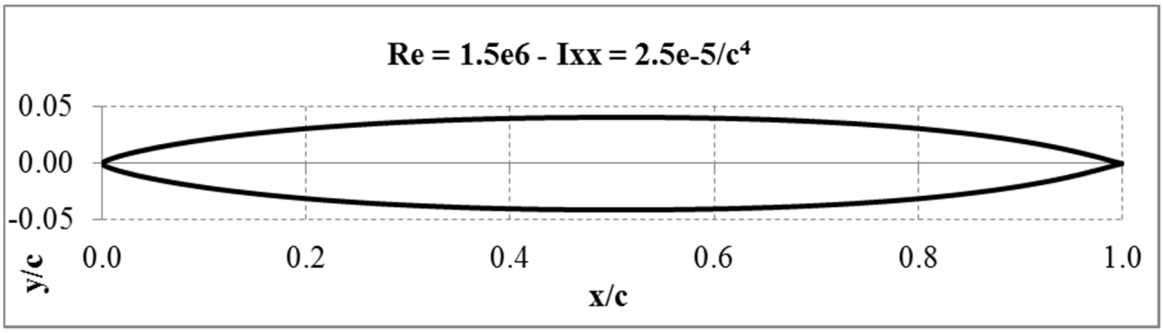


Figure 6.23 – Ixx = 2.5e-5/c^4 constrained airfoil at Re = 1,500,000

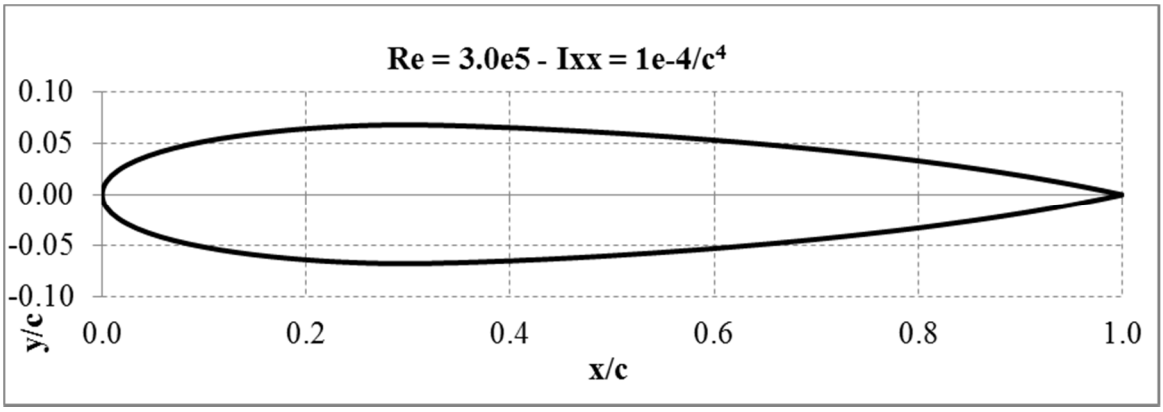


Figure 6.24 – Ixx = 1.0e-4/c^4 constrained airfoil at Re = 300,000

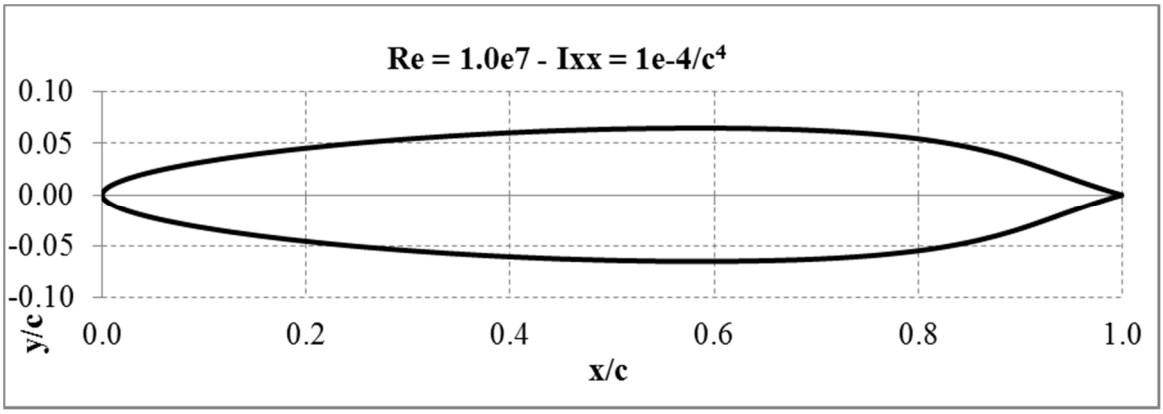


Figure 6.25 – Ixx = 1.0e-4/c^4 constrained airfoil at Re = 10,000,000

Again the crux of the results is the behaviour of the airfoil's crest, predominantly its lateral position along the chord length. For low Reynolds numbers this appears close to the leading edge and as the Reynolds number is increased this gets pushed back towards

the trailing edge. This behaviour has the similar direct effect on both the radius of the leading edge as well as the trailing edge wedge angle, that is, the position of the crest determines the size of these geometric properties. It can be seen in Figure 6.27 that the crest behaviour acts in a regular manner in the thickness direction with only a slight variation present. Lateral positioning of the thickness crest on the other hand as shown in Figure 6.26 increases in a linear manner for Reynolds numbers above 500,000 noting a sudden jump in the vicinity of Reynolds numbers 300,000-500,000. This trend was also observed for the area constrained airfoils. For low Reynolds numbers (<300,000) for the limited data points, the crest location shows minimal variation. One further observation from Figure 6.26 is that the lateral crest location reduces with a proportional increase in the airfoil's moment of inertia. As can be recalled, this is due to the thicker airfoils producing too bluff of a trailing edge resulting in flow separation and to counter this effect, the crest position is moved forward.

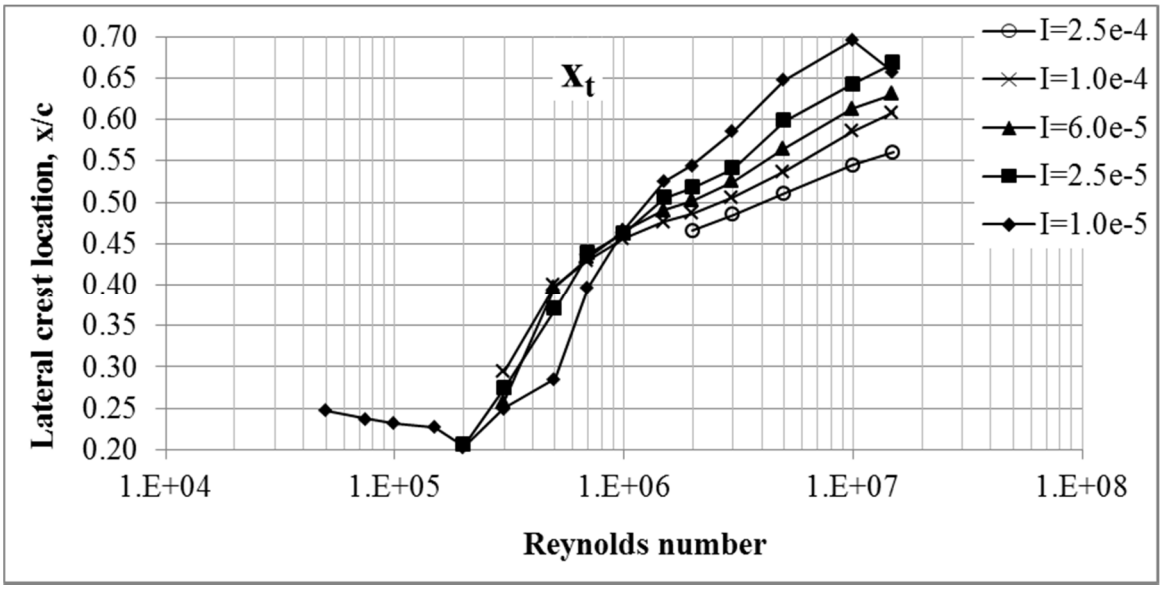


Figure 6.26 – Moment of inertia constrained crest lateral position trends

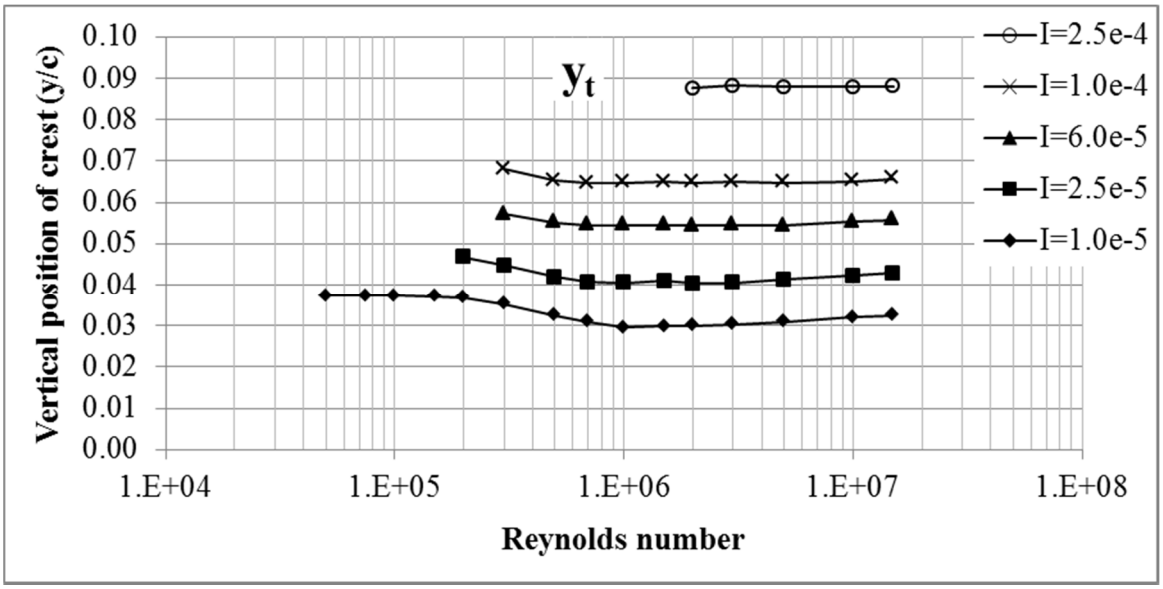


Figure 6.27 – Moment of inertia constrained crest max thickness position trends

In order to explain the transition from the low variation of the crest at low Reynolds numbers to the higher linear variation at higher Reynolds numbers it is again necessary to compare plots of the crest location versus the transition point of the flow from laminar to turbulent.

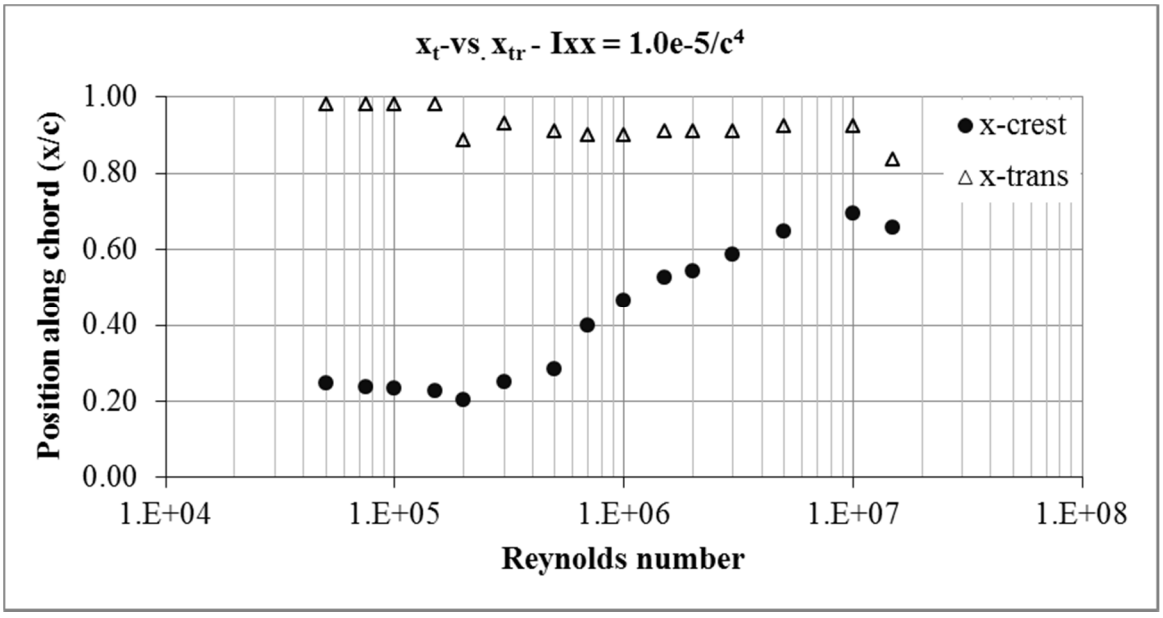


Figure 6.28 – Crest location versus flow behaviour change, Ixx = 1.0e-5/c^4

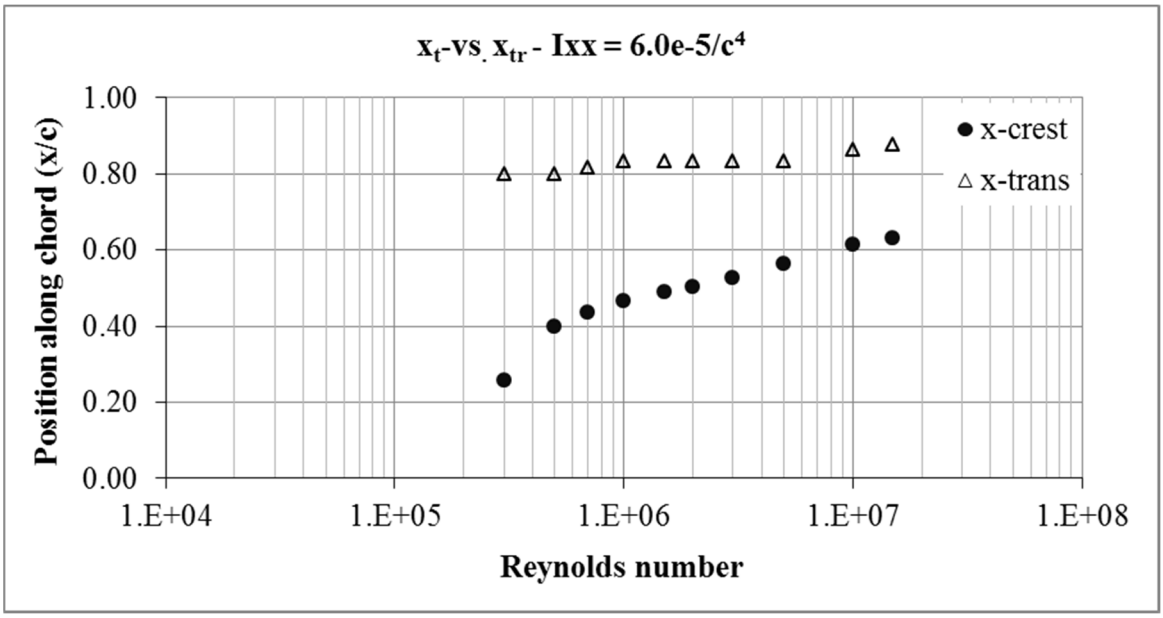


Figure 6.29 – Crest location versus flow behaviour change, Ixx = 6.0e-5/c^4

The previous figures illustrate the transition behaviour. The upper transition points show that most of the flow is laminar and that turbulence is only experienced near the trailing edge. The large laminar portion is something to be expected since a drawback of turbulent flow conditions is a large increase in drag. Hence it can be seen that the crest location influences the point of transition and that as turbulence threatens to appear it is countered by a shift in the crest location toward the trailing edge. It can also now be observed that for optimal shapes the presence of turbulence is minimal and thus it can be wondered why so much care and attention is given to the turbulence models presented in §2.2. This is because although turbulence is not widely present in the solution it was present in the early evolutionary stages particularly the exploration stage. Thus turbulence was necessary to behave in a manner similar to a penalty method.

The following tables provide the parameters for the optimized airfoils with bending moment constraints. Note that amplification factors have not been applied to the applicable parameters. Some rough trends do exist within the data other than the already

described location of the thickness crest. The radii of the leading edge as well as crest curvature decrease as the crest in moved towards the middle of the chord and then decrease as the chord approaches the trailing edge. A more pronounced trend for the wedge angle exists as well but cannot be easily interpreted from the data tables.

Re	r_{le}	k_t	dz_{te}	β_{te}	x_t	y_t	λ_{amp}
50000	-0.001032	-8.143474	0.000000	0.026835	0.247456	0.226615	0.164549
75000	-0.001045	-9.926761	0.000000	0.030977	0.237386	0.226530	0.164869
100000	-0.001010	-9.477077	0.000000	0.026405	0.232203	0.248852	0.149905
150000	-0.001048	-8.454458	0.000000	0.026532	0.227290	0.232695	0.159572
200000	-0.005933	-8.054213	0.000000	0.057745	0.203547	0.226581	0.162319
300000	-0.001068	-6.054071	0.000000	0.986970	0.250254	0.082234	0.429326
500000	-0.026005	-4.817268	0.000000	0.659537	0.285450	0.106820	0.304281
700000	-0.035672	-4.505325	0.000000	0.674235	0.397745	0.110416	0.279760
1000000	-0.028685	-0.798320	0.000000	0.942396	0.466824	0.249518	0.118260
1500000	-0.027738	-0.738687	0.000000	0.928775	0.525493	0.205703	0.144796
2000000	-0.014528	-0.881682	0.000000	0.902911	0.544637	0.230927	0.129779
3000000	-0.009789	-0.808810	0.000000	0.884698	0.585785	0.177968	0.170463
5000000	-0.005659	-0.801590	0.000000	0.714122	0.648182	0.135174	0.228509
10000000	-0.004261	-2.043471	0.000000	0.809349	0.695941	0.229999	0.138973
15000000	-0.001006	-2.182971	0.000000	0.165415	0.657251	0.248917	0.130384

Table 6-11 – Ixx = 1.0e-5/c^4 airfoil parameters

Re	r_{le}	k_t	dz_{te}	β_{te}	x_t	y_t	λ_{amp}
200000	-0.021042	-8.252378	0.000000	0.918946	0.205412	0.182018	0.256837
300000	-0.064704	-4.565886	0.000000	0.881755	0.273843	0.223630	0.199884
500000	-0.046297	-3.091776	0.000000	0.525701	0.370167	0.115451	0.363545
700000	-0.012756	-0.925828	0.000000	0.810099	0.438368	0.244027	0.166683
1000000	-0.030149	-0.853755	0.000000	0.882462	0.462410	0.235333	0.172144
1500000	-0.013704	-1.003079	0.000000	0.849134	0.505605	0.242791	0.168749
2000000	-0.037762	-0.681289	0.000000	0.811090	0.517834	0.197220	0.204589
3000000	-0.024656	-0.775384	0.000000	0.811789	0.541207	0.208027	0.195191
5000000	-0.025035	-1.068056	0.000000	0.880052	0.598627	0.219670	0.188221
10000000	-0.009840	-1.289306	0.000000	0.805759	0.642961	0.195517	0.216408
15000000	-0.005589	-1.438511	0.000000	0.754117	0.668559	0.177977	0.241274

Table 6-12 – Ixx = 2.5e-5/c^4 airfoil parameters

Re	r_{le}	k_t	dz_{te}	β_{te}	x_t	y_t	λ_{amp}
300000	-0.05458	-2.11704	0.000000	0.350191	0.257709	0.100844	0.566837
500000	-0.003829	-0.676156	0.000000	0.533549	0.397660	0.152225	0.362759
700000	-0.012065	-0.753576	0.000000	0.469026	0.433992	0.198870	0.274268
1000000	-0.013959	-0.686844	0.000000	0.495026	0.465451	0.184104	0.296099
1500000	-0.031113	-0.772017	0.000000	0.654178	0.490680	0.205186	0.265788
2000000	-0.047242	-0.804851	0.000000	0.727457	0.501882	0.221841	0.245071
3000000	-0.061422	-0.923852	0.000000	0.824489	0.525435	0.239574	0.227661
5000000	-0.045094	-0.676133	0.000000	0.721001	0.564970	0.175639	0.309625
10000000	-0.044334	-1.117587	0.000000	0.836834	0.612744	0.217193	0.254890
15000000	-0.023518	-1.098593	0.000000	0.748010	0.631281	0.197714	0.282453

Table 6-13 – Ixx = 6.0e-5/c^4 airfoil parameters

Re	r_{le}	k_t	dz_{te}	β_{te}	x_t	y_t	λ_{amp}
300000	-0.06288	-1.84294	0.000000	0.362158	0.29391	0.117648	0.578085
500000	-0.04218	-0.87634	0.000000	0.54263	0.400075	0.21051	0.309878
700000	-0.05479	-0.81394	0.000000	0.586126	0.429697	0.218583	0.295894
1000000	-0.04581	-0.88148	0.000000	0.661593	0.456641	0.228287	0.284018
1500000	-0.02203	-0.77407	0.000000	0.54406	0.476547	0.199729	0.324897
2000000	-0.03384	-0.78327	0.000000	0.606167	0.486839	0.208822	0.309929
3000000	-0.041534	-0.873211	0.000000	0.680988	0.505901	0.224081	0.289546
5000000	-0.053617	-0.806218	0.000000	0.707112	0.536852	0.206428	0.313839
10000000	-0.028056	-0.675099	0.000000	0.602785	0.585648	0.160758	0.404574
15000000	-0.047304	-1.124997	0.000000	0.805394	0.608822	0.220352	0.298042

Table 6-14 – Ixx = 1.0e-4/c^4 airfoil parameters

Re	r_{le}	k_t	dz_{te}	β_{te}	x_t	y_t	λ_{amp}
2000000	-0.05232	-0.51849	0.000000	0.452986	0.465376	0.151677	0.577526
3000000	-0.04593	-0.64836	0.000000	0.493039	0.485017	0.168096	0.525141
5000000	-0.054245	-0.589103	0.000000	0.508988	0.510384	0.164844	0.532807
10000000	-0.061019	-0.714375	0.000000	0.605949	0.544840	0.194998	0.450474
15000000	-0.051945	-0.764521	0.000000	0.623207	0.560106	0.201404	0.436956

Table 6-15 – Ixx = 2.5e-4/c^4 airfoil parameters

It can be noted that for the wedge angle minute geometric inconsistencies occur such as having a very large wedge angle. These high values cause flow separation to

occur at the trailing edge but contained to a very small region. This suggests that for these bending moment constrained airfoils that the optimized shapes are located on the boundary between attached and separated flow. To better see the behaviour of the wedge angle it is possible to manually measure the angle on a more macro scale. Figure 6.30 shows the resulting relationship obtained. An anomaly is also noted for the point I=1.0e-5, Re=1.5e7. This is likely a numerical precision error since the predicted drag values in this region are very low.

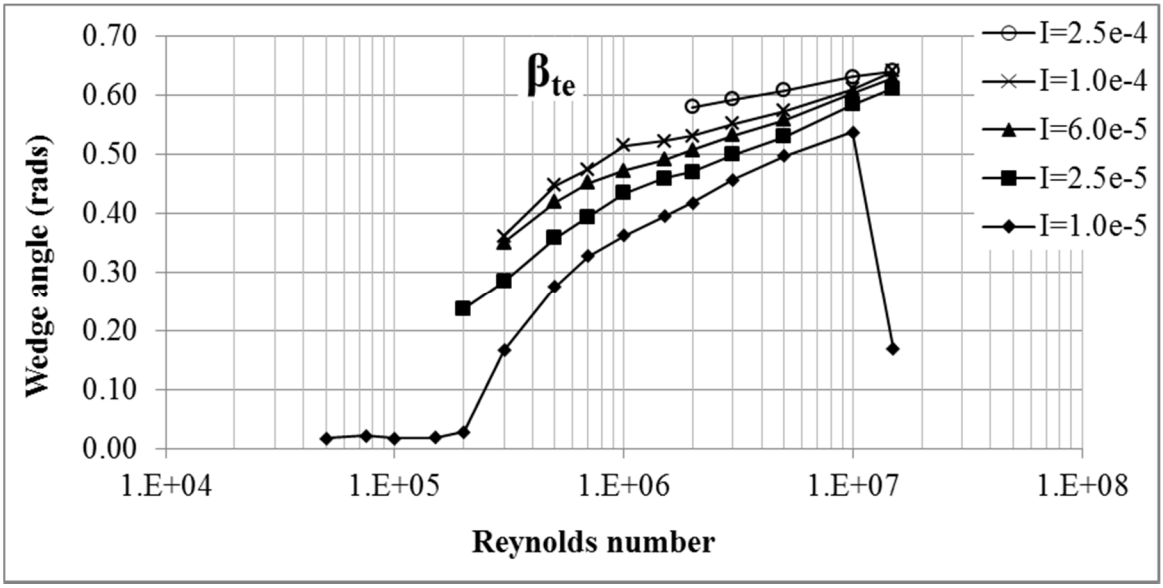


Figure 6.30 – Moment of inertia constrained wedge angle trends

In addition to assessing the shapes produced it is again also possible to review the behaviour of the optimization strategy. This allows for sanity checks to be completed in order to provide a high level of confidence in the resulted shapes. One method of assessing the shapes is to look at the trends of the costs that are produced. Figure 6.31 shows the plots of all of the bending moment constrained airfoil optimizations. This time it can be noticed that more choppy curves are formed for each set of airfoil sizes. This is also a common behaviour in optimization. From an optimization point of view there are

trends to assess there as well. It is expected that a period of time (generations) is spent in an exploration phase and then a time that works towards convergence. This behaviour as shown in Figure 6.32, Figure 6.33, and Figure 6.34 confirms the predicted behaviour illustrated in Figure 4.2 - Figure 4.5.

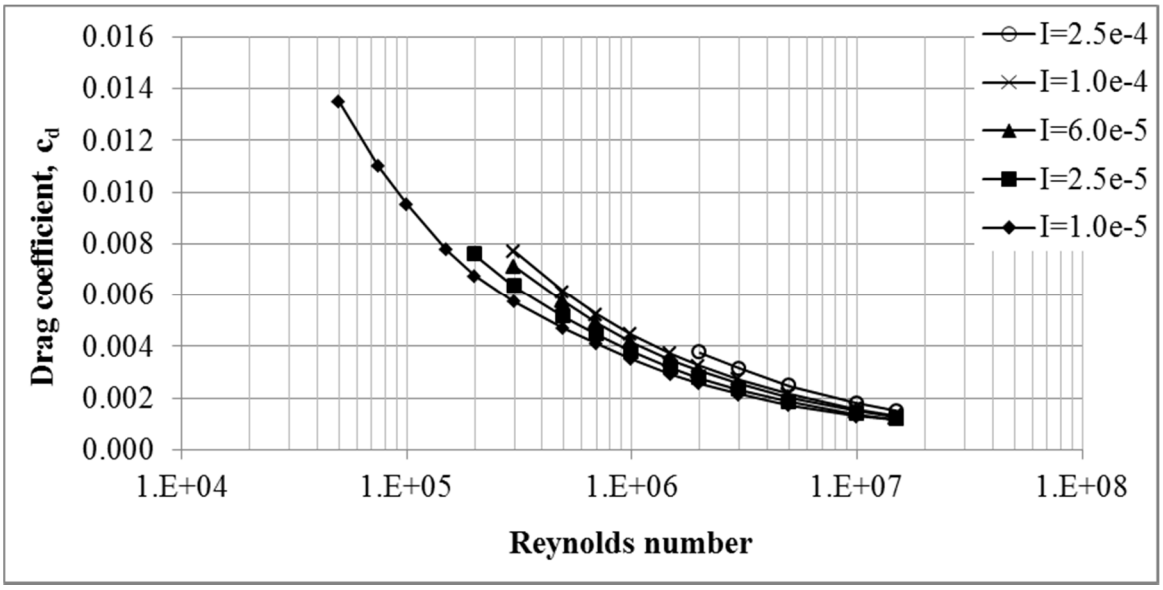


Figure 6.31 – Cost plots of moment of inertia optimized airfoils

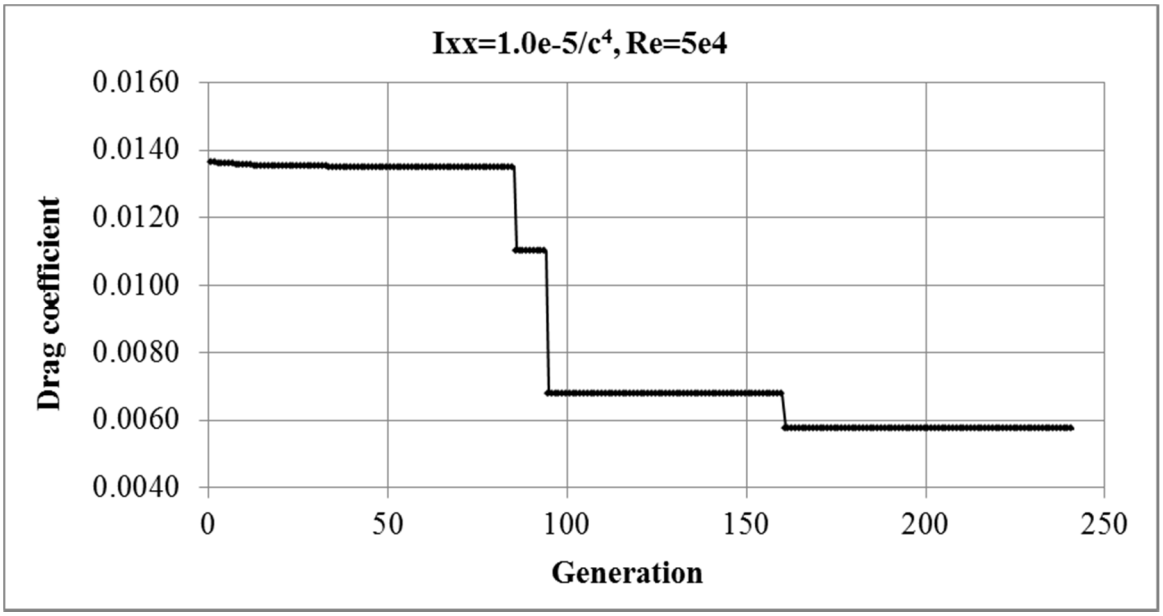


Figure 6.32 – Cost of best performing vector vs generation - Ixx=1.0e-5/c^4, Re=5e4

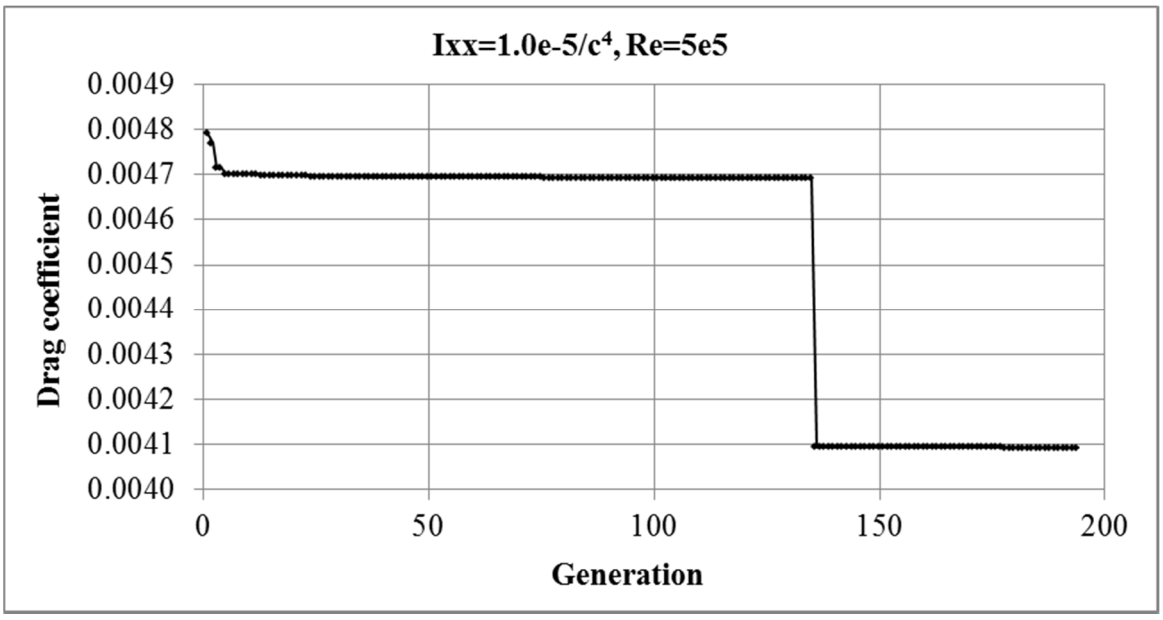


Figure 6.33 – Cost of best performing vector vs generation - Ixx=1.0e-5/c^4, Re=5e5

Figure 6.34 – Cost of best performing vector vs generation - Ixx=1.0e-5/c^4, Re=5e5

Again it can be noticed that the first 20-25% of the time is spent in the exploration phase and the remainder is spent on convergence. Convergence plots that show the difference vectors trending towards zero indicate that an optimal solution has been found.

As in the area constrained cases, convergence resembled that which was shown in Figure 4.5.

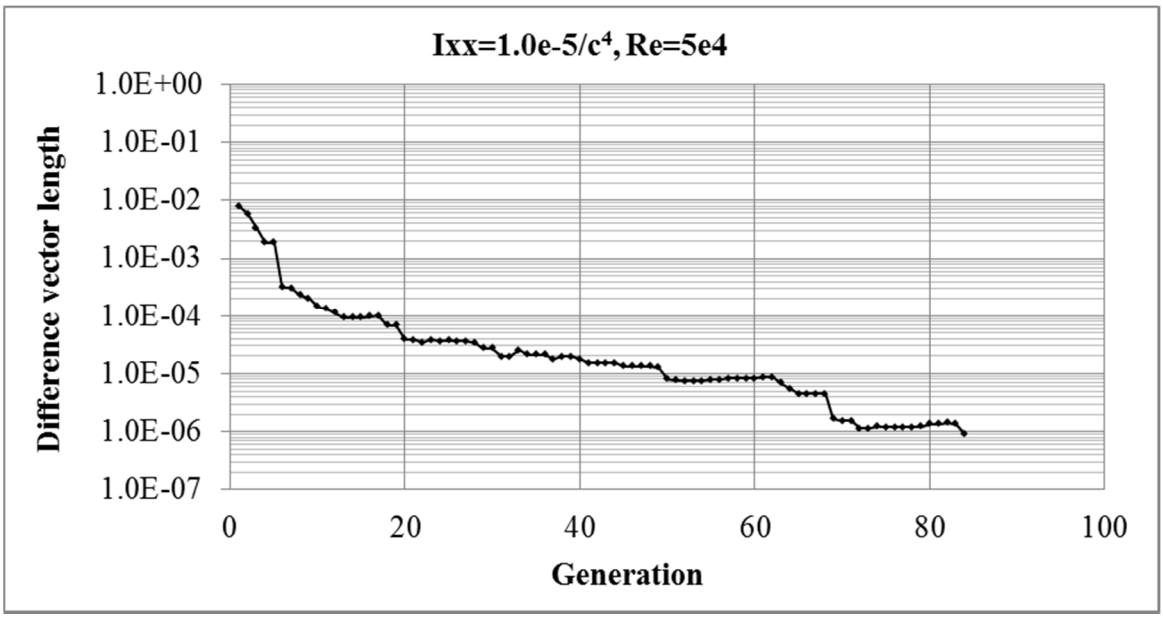


Figure 6.35 – Convergence history, Ixx=1.0e-5/c^4, Re=5e4

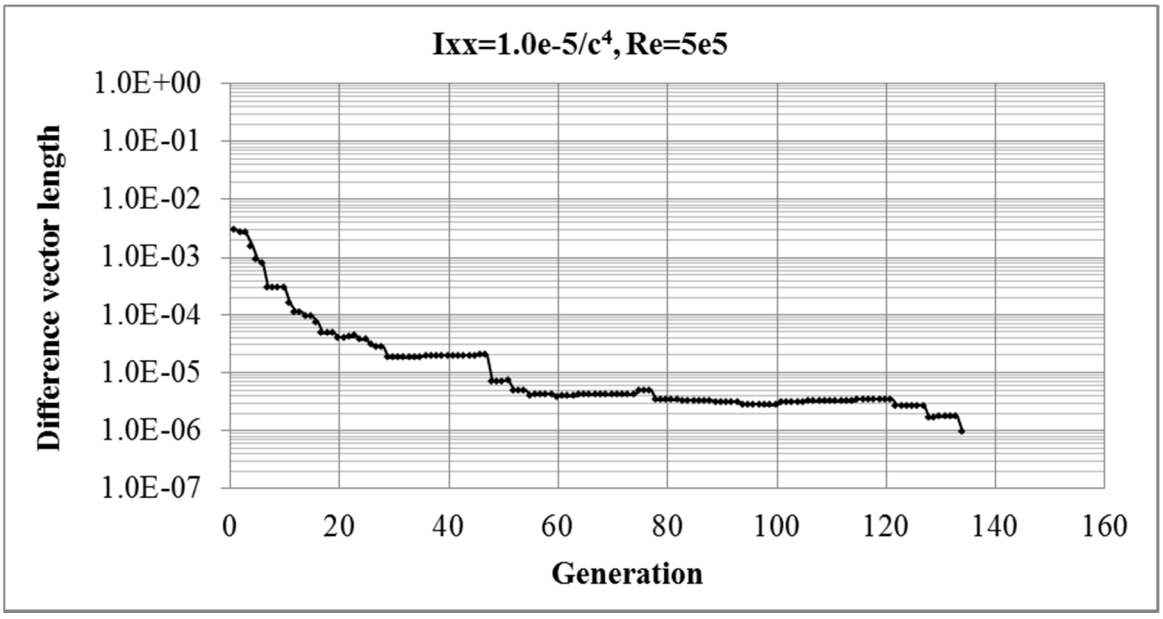


Figure 6.36 – Convergence history, Ixx=1.0e-5/c^4, Re=5e5

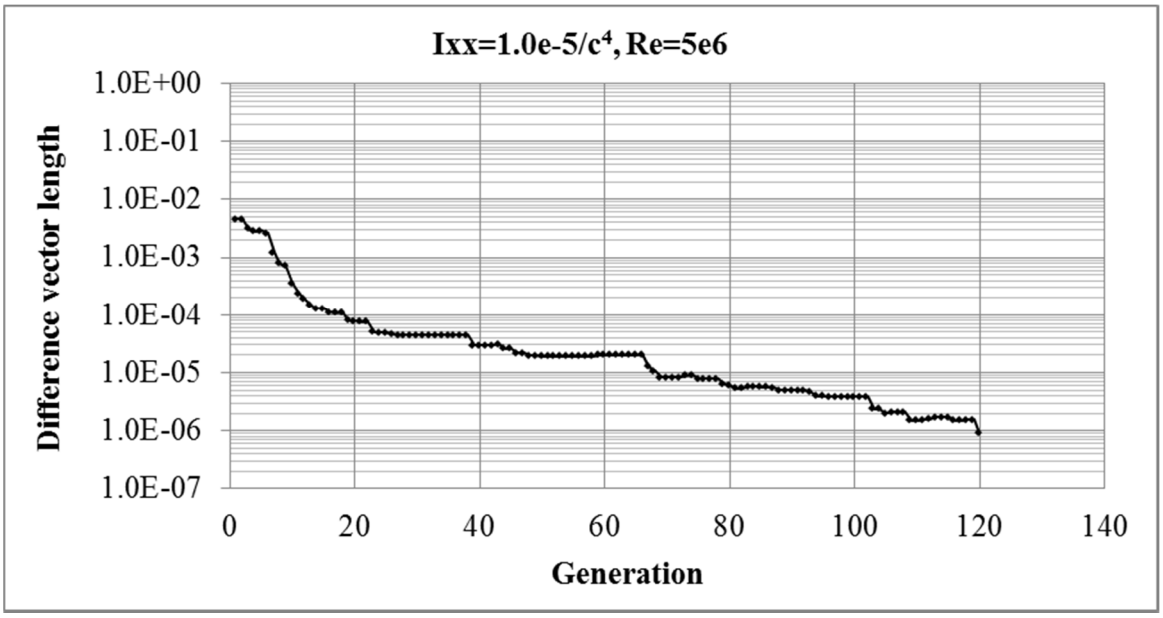


Figure 6.37 – Convergence history, Ixx=1.0e-5/c^4, Re=5e6

The convergence criteria imposed on the current optimization problem showed no trend between the number of cost evaluations (NCE's) and the corresponding Reynolds number. The following tables show the resulting NCE's for each constrained case.

Re	NCE	c_d
50000	8446	0.01351
75000	9354	0.01101
100000	8720	0.00953
150000	6832	0.00779
200000	15945	0.00677
300000	24034	0.00575
500000	13295	0.00469
700000	16791	0.00409
1000000	9990	0.00351
1500000	13996	0.00292
2000000	9686	0.00257
3000000	11776	0.00215
5000000	11987	0.00172
10000000	15767	0.00129
15000000	9257	0.00118

Table 6-16 – NCE's for Ixx=1.0e-5/c^4

Re	NCE	c_d
200000	19883	0.00758
300000	10647	0.00634
500000	7442	0.00514
700000	8584	0.00444
1000000	11563	0.00379
1500000	10163	0.00315
2000000	12713	0.00276
3000000	16198	0.00231
5000000	14932	0.00185
10000000	8430	0.00138
15000000	8750	0.00117

Table 6-17 – NCE's for Ixx=2.5e-5/c^4

Re	NCE	c_d
300000	19925	0.007112
500000	11172	0.00573
700000	19319	0.00490
1000000	23795	0.00415
1500000	17607	0.00345
2000000	21505	0.00303
3000000	9554	0.00253
5000000	49865	0.00202
10000000	8950	0.00149
15000000	11246	0.00125

Table 6-18 – NCE's for Ixx=6.0e-5/c^4

Re	NCE	c_d
300000	18450	0.007711
500000	23481	0.00612
700000	16818	0.005234
1000000	14459	0.004451
1500000	14630	0.003702
2000000	13634	0.003241
3000000	12985	0.00271
5000000	16885	0.00215
10000000	49895	0.00158
15000000	9933	0.00133

Table 6-19 – NCE's for Ixx=1.0e-4/c^4

Re	NCE	c_d
2000000	12959	0.003758
3000000	23985	0.003127
5000000	35377	0.002468
10000000	12923	0.001803
15000000	14135	0.001507

Table 6-20 – NCE's for Ixx=2.5e-4/c^4

While it was again not expected that there be a correlation between the NCE's and the Reynolds number it is expected that a direct relationship between the total solve time and the Reynolds number exists. For the same reasons as described in the area constrained case this effect translated into the total solve times based on the CPU description presented in §4.1 being four times greater for like airfoils at Re = 50,000 than airfoils at Re = 15,000,000.

7 Conclusions and future work

The objective for the work of this thesis set out to find insight and a means for obtaining optimum designs for non-lifting aerodynamic struts. This was done for axially loaded struts as well as struts loaded in bending thus the optimization was performed on struts of a fixed cross-sectional area or for a fixed bending moment. Assigning these physical constraints to the optimization problem allowed a set of airfoil parameters to be determined that resulted in an airfoil producing minimum drag within a specified flow. The flow was specified by assigning a Reynolds number to the flow which in turn altered the viscous behaviour of the flow, in both its laminar and turbulent flow regions.

By combining a laminar and turbulent flow field solver with an airfoil parameterization technique and a genetic algorithm, optimized results for many flow scenarios and shape sizes could be obtained. The numerical approach of the flow field solver and the simplicity of the genetic algorithm allowed for these results to be obtained in an acceptable timely manner. With the vast number of data points collected from all of the scenarios charts such as Figure 6.8, Figure 6.10, Figure 6.26 and Figure 6.30 could be produced. These charts allow an engineer to obtain a near-optimum shape for a given problem without having to perform the optimization. By having a large number of data points, trends were able to be deduced thus providing confidence in the obtained results.

While confidence was instilled by these results, limitations and the need for future work were also exposed. The limitations occurred in this thesis were due to the limitations of the boundary layer flow field solver. This solver did not allow airfoils of significant thickness to be evaluated thus restricting the solution space to thin airfoils. To incorporate thick airfoils a better suited and proven boundary layer solver with these

capabilities would have to be inserted in place of the existing solver. The boundary layer solver also is limited to the incompressible flow region. Again, if the need arose for compressible effects to be included the existing solver could be replaced with one containing these capabilities.

Perhaps the biggest need for future work resulting from this thesis is the need for validation of the results. There are many way that this can be achieved; from repeating this work using other optimization, parameterization, and evaluation techniques to actual wind tunnel testing. Validation would bring the high level of confidence already obtained to something even higher.

8 Bibliography

Anderson, J. D. (2011). *Fundamentals of Aerodynamics 5th ed.* New York: McGraw-Hill.

Cebeci, T. (2004). *Turbulence Models and Their Applications.* Long Beach, California: Horizons Publishing Inc.

Chang, I., Torres, G. J., & Tung, C. (1995). Geometric Analysis of Wing Sections. *NASA TM 110346.*

Coiro, D. P., & Nicolosi, F. (1995). Design and Optimization of Glider Components. *Technical Soaring*, Vol. 19, No 2.

Derksen, R. W., & Rogalsky, T. (2009). Optimum Aerofoil Parameterization for Aerodynamic Design. In S. Hernandez, & C. A. Brebbia, *Computer Aided Optimum Design in Engineering XI* (pp. 197-206). Southampton: WIT Press.

Huyse, L., & Lewis, M. R. (2001). *Airfoils, Aerodynamic Shape Optimization of Two-dimensional.* Institute for Computer Applications in Science and Engineering (ICASE).

Price, K. V. (1999). An Introduction to Differential Evolution. In D. Corne, M. Dorigo, & F. Glover, *New Ideas in Optimization* (pp. 79-108). New-York: McGraw-Hill.

Price, K., & Storn, R. (1997, April). Differential Evolution. *Dr. Dobb's Journal*, pp. 18-24.

Rogalsky, T. (2004). Acceleration of Differential Evolution for Aerodynamic Design. *Ph.D. Thesis*. University of Manitoba: Canada.

Rogalsky, T., Derksen, R. W., & Kocabiyik, S. (1999). An Aerodynamic Design Technique for Optimizing Fan Blade Spacing. *Proceedings of the 7th Annual*

Conference of the Computational Fluid Dynamics Society of Canada, (pp. 2.29-2.34).

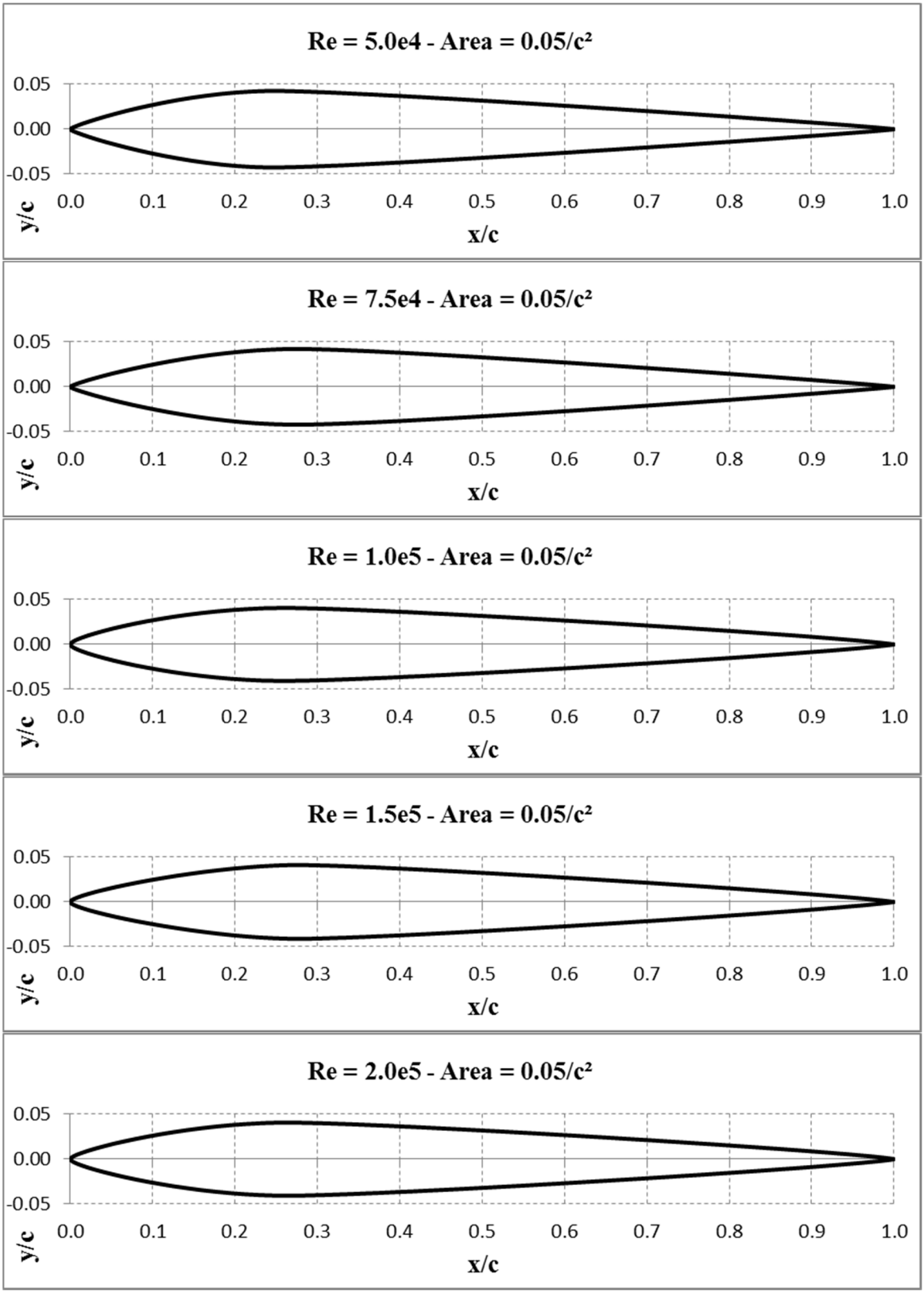
Rogalsky, T., Derksen, R. W., & Kocabiyik, S. (2000). Differential Evolution in Aerodynamic Optimization. *Canadian Aeronautics and Space Journal, Vol. 46, No. 4*, 183-190.

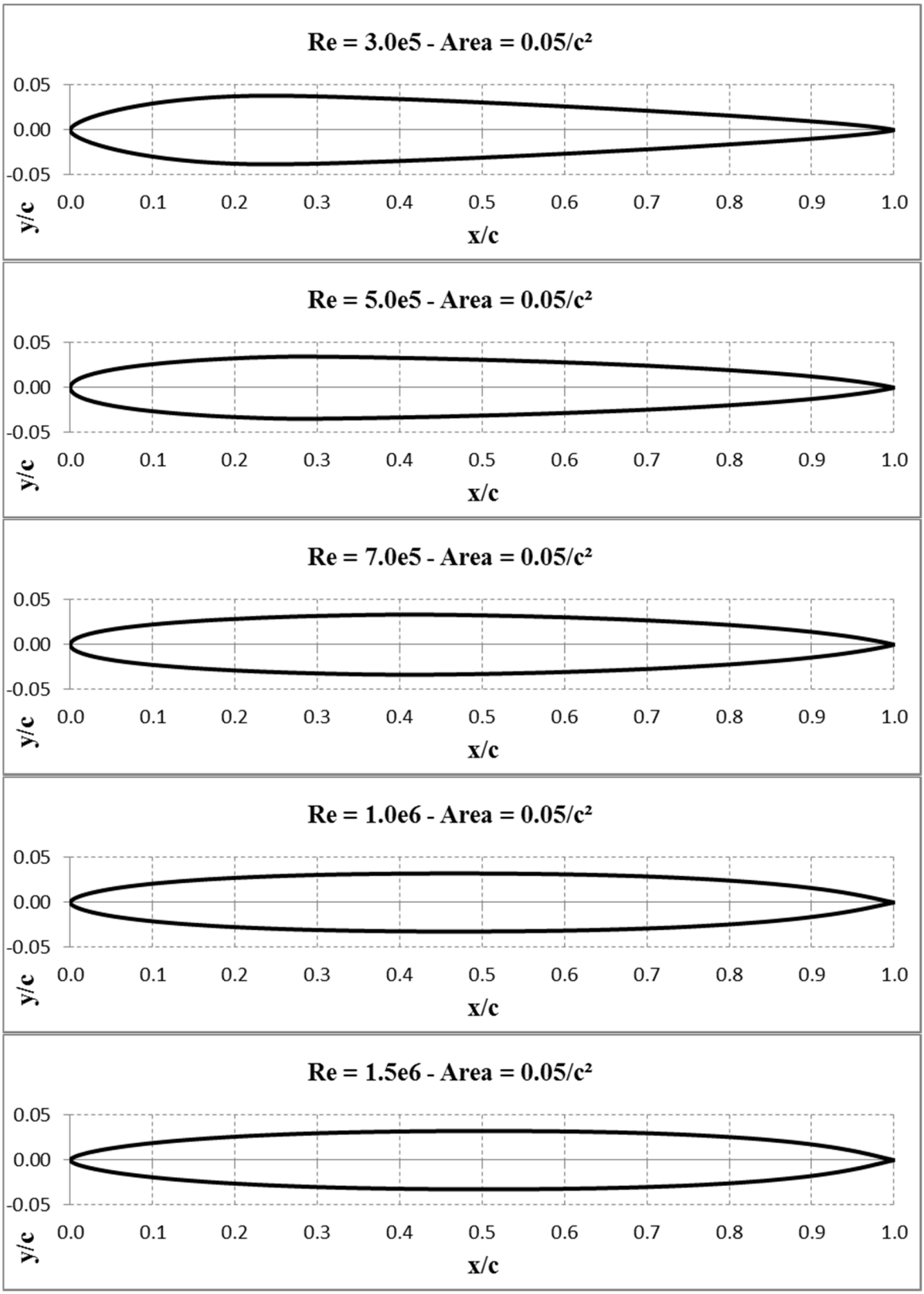
Sobieczky, H. (1999). Parametric Airfoils and Wings. In K. Fujii, & G. S. Dulikravich, *Recent Development of Aerodynamic Design Methodologies* (pp. 71-87). Braunschweig/Wiesbaden: Friedr. Vieweg & Sohn Verlagsgesellschaft mbH.

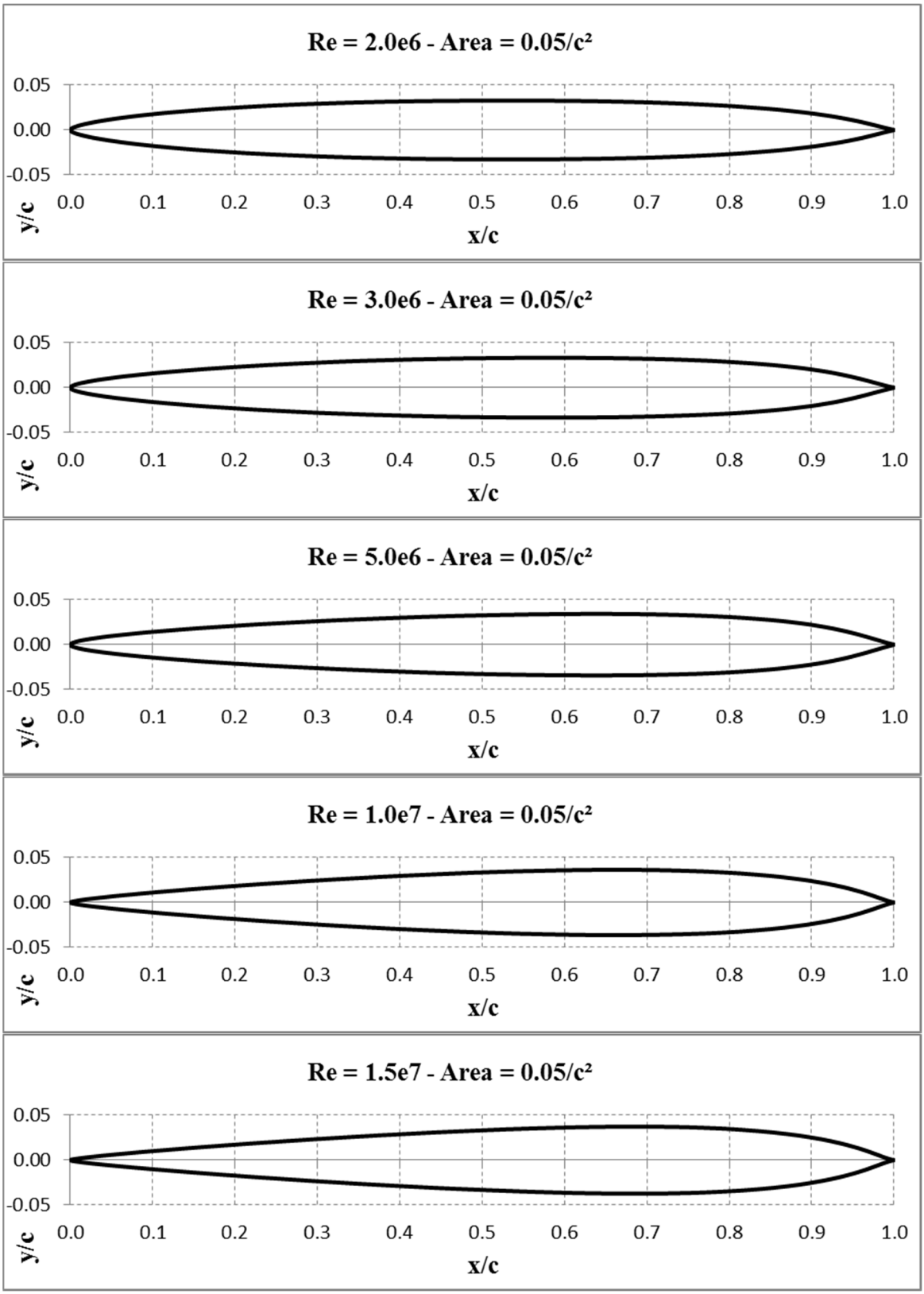
Venkataraman, P. (1995). A new procedure for airfoil definition. *13th Applied Aerodynamics Conference, AIAA Paper 95-1875-CP.* San Diego.

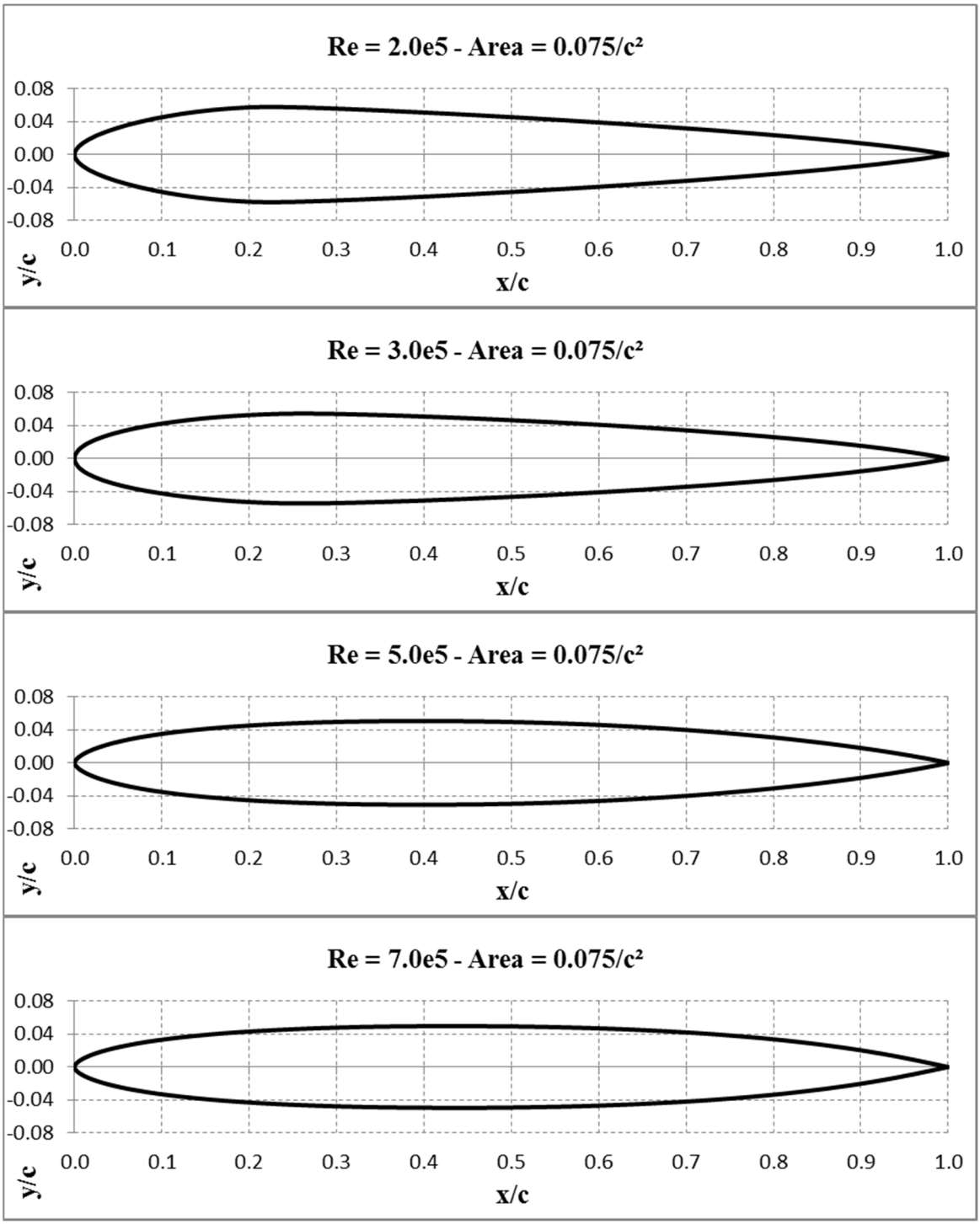
Zielinski, K., Weitkemper, P., Laur, R., & Kammeyer, K. D. (2006, May 18-19). Examination of Stopping Criteria for Differential. *10th Int. Conf. Optimization Electrical Electronic Equipment* (pp. 149-156). Brasov, Romania: May 18-19.

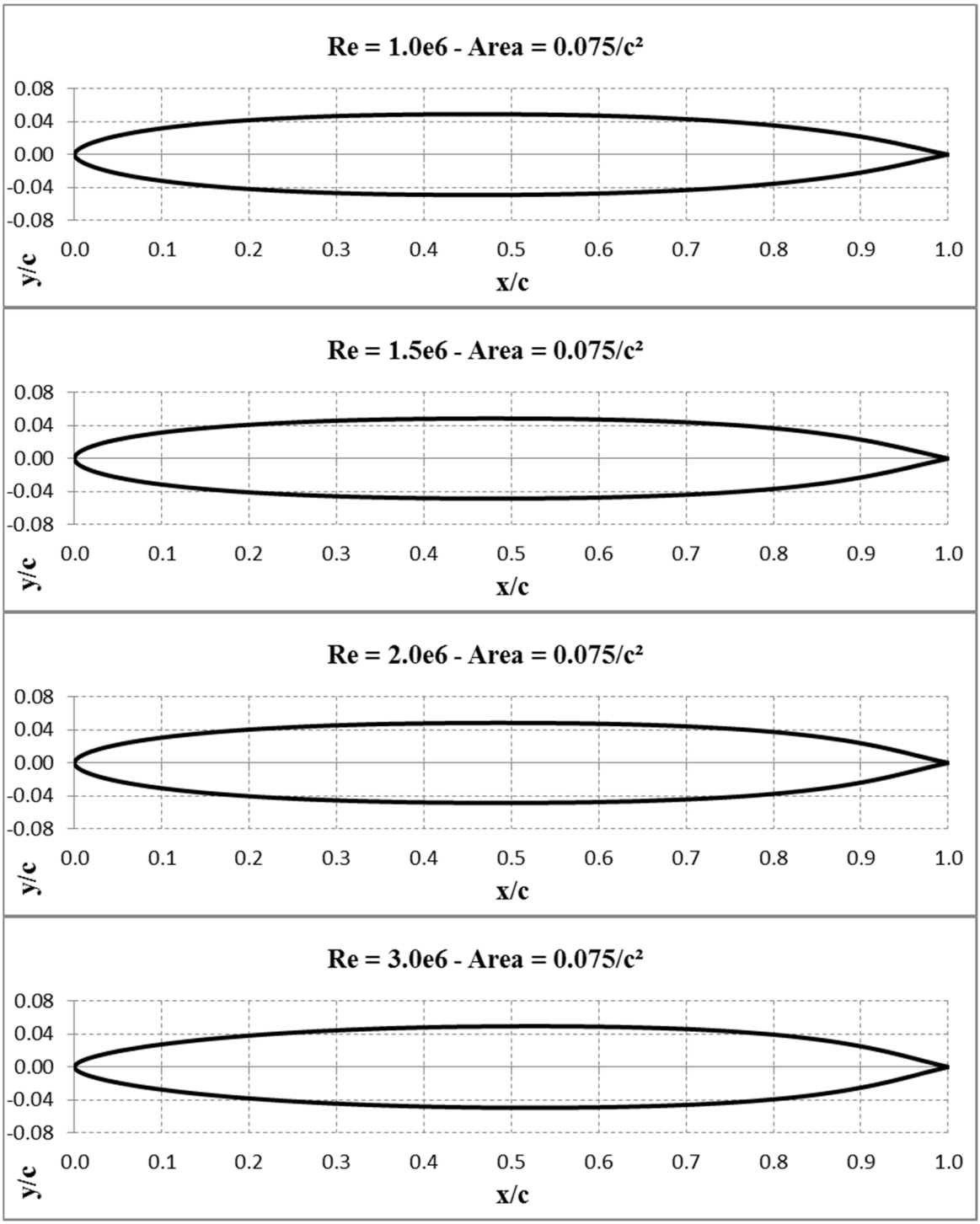
Appendix A – Optimized area constrained airfoil shapes

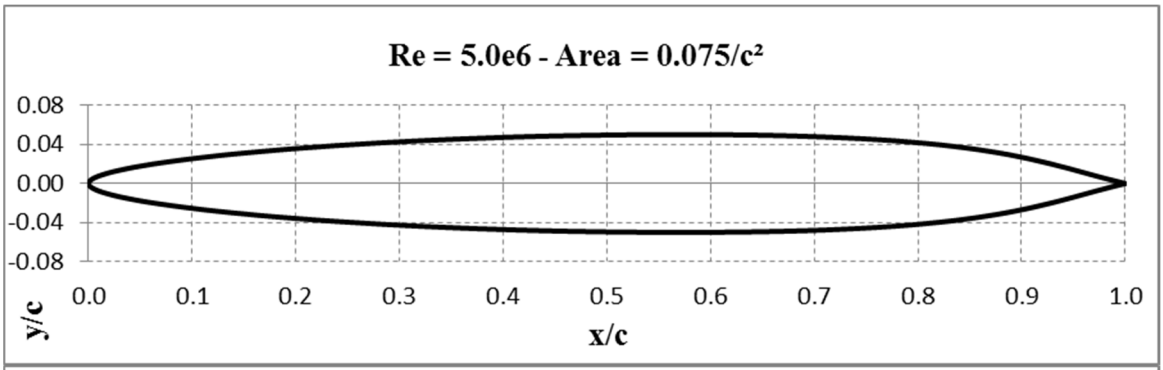


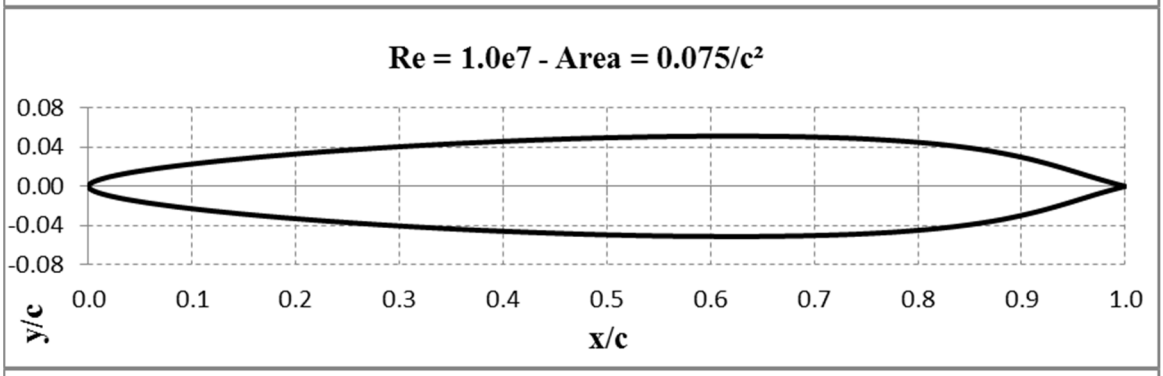

Re = 3.0e5 - Area = 0.05/c²
Re = 5.0e5 - Area = 0.05/c²
Re = 7.0e5 - Area = 0.05/c²
Re = 1.0e6 - Area = 0.05/c²
Re = 1.5e6 - Area = 0.05/c²
0.05
0.00
-0.05
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

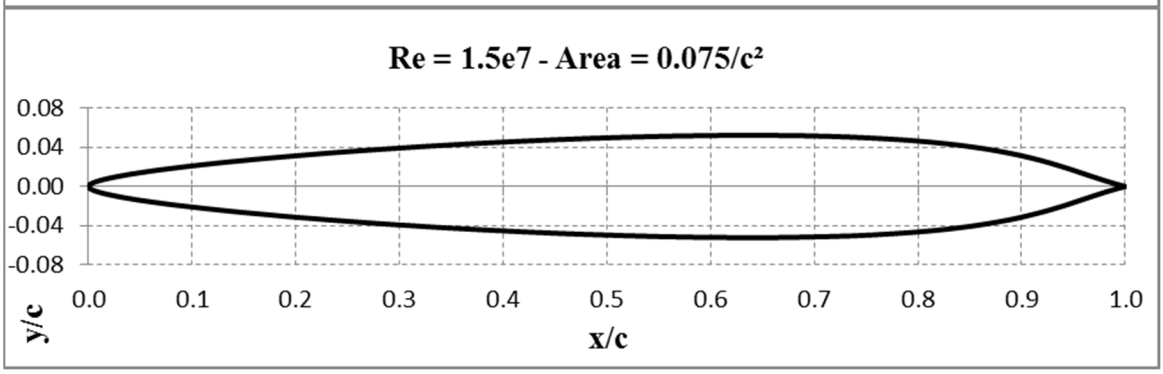

Re = 2.0e6 - Area = 0.05/c²
Re = 3.0e6 - Area = 0.05/c²
Re = 5.0e6 - Area = 0.05/c²
Re = 1.0e7 - Area = 0.05/c²
Re = 1.5e7 - Area = 0.05/c²
0.05
0.00
-0.05
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

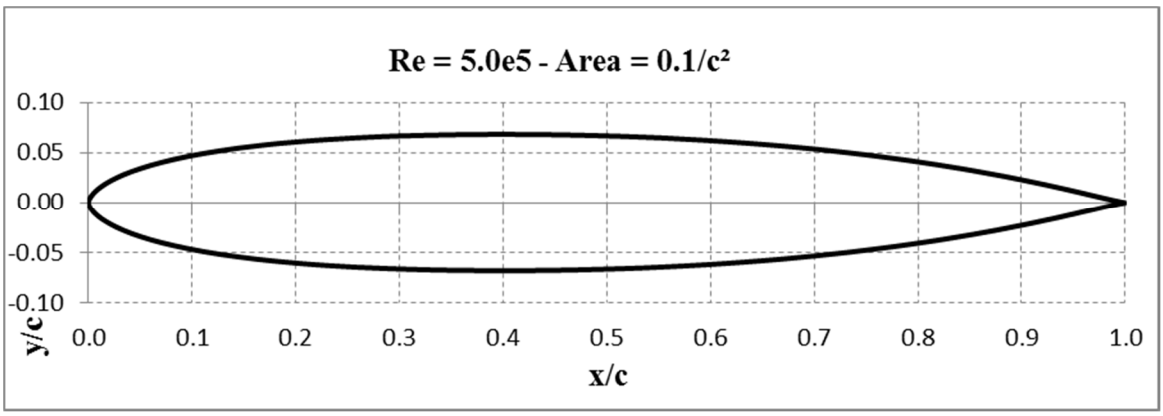

Re = 2.0e5 - Area = 0.075/c²
Re = 3.0e5 - Area = 0.075/c²
Re = 5.0e5 - Area = 0.075/c²
Re = 7.0e5 - Area = 0.075/c²
0.08
0.04
0.00
-0.04
-0.08
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
y/c
x/c

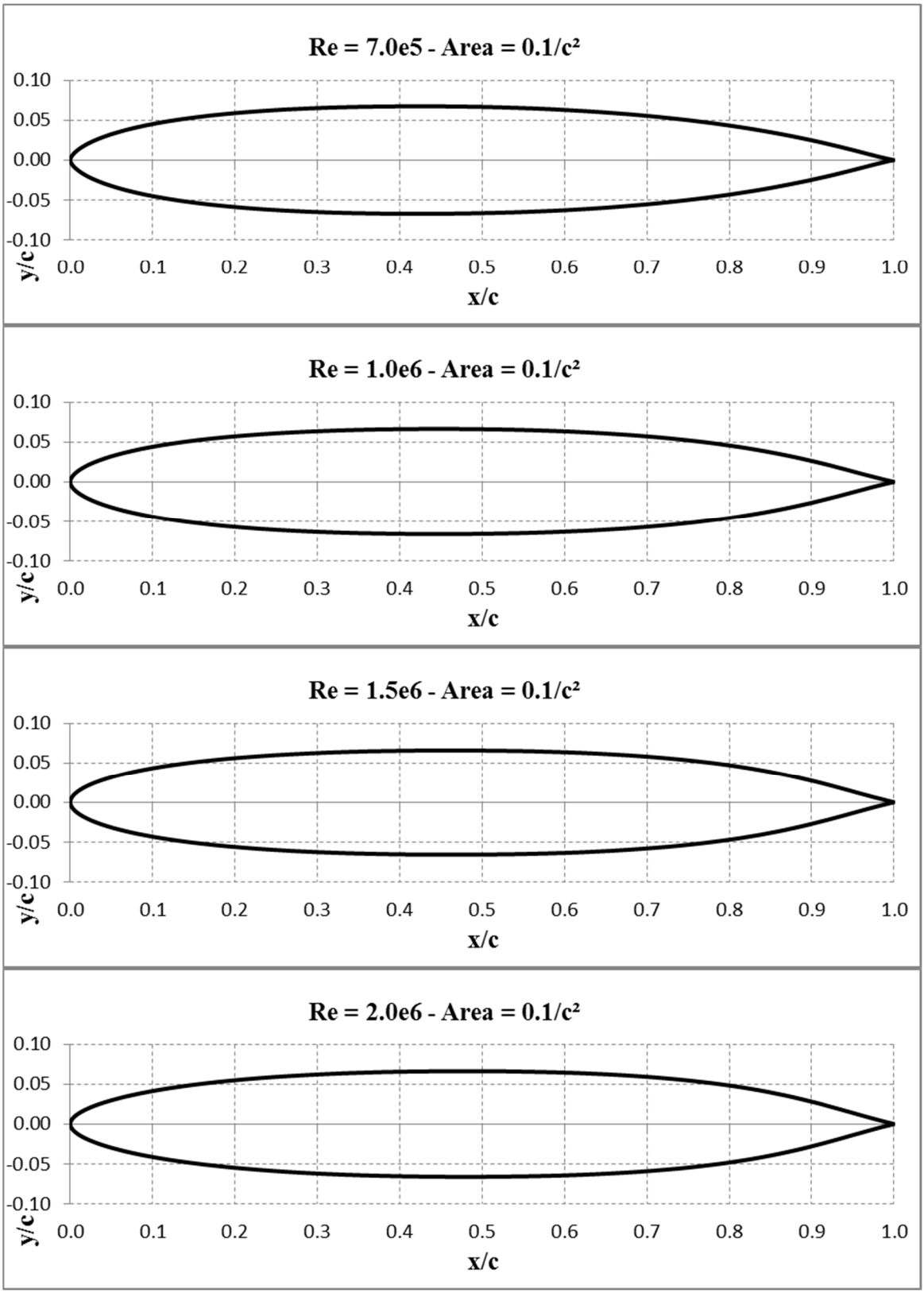

Re = 1.0e6 - Area = 0.075/c²
0.08
0.04
0.00
-0.04
-0.08
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
y/c
x/c
Re = 1.5e6 - Area = 0.075/c²
0.08
0.04
0.00
-0.04
-0.08
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
y/c
x/c
Re = 2.0e6 - Area = 0.075/c²
0.08
0.04
0.00
-0.04
-0.08
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
y/c
x/c
Re = 3.0e6 - Area = 0.075/c²
0.08
0.04
0.00
-0.04
-0.08
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
y/c
x/c

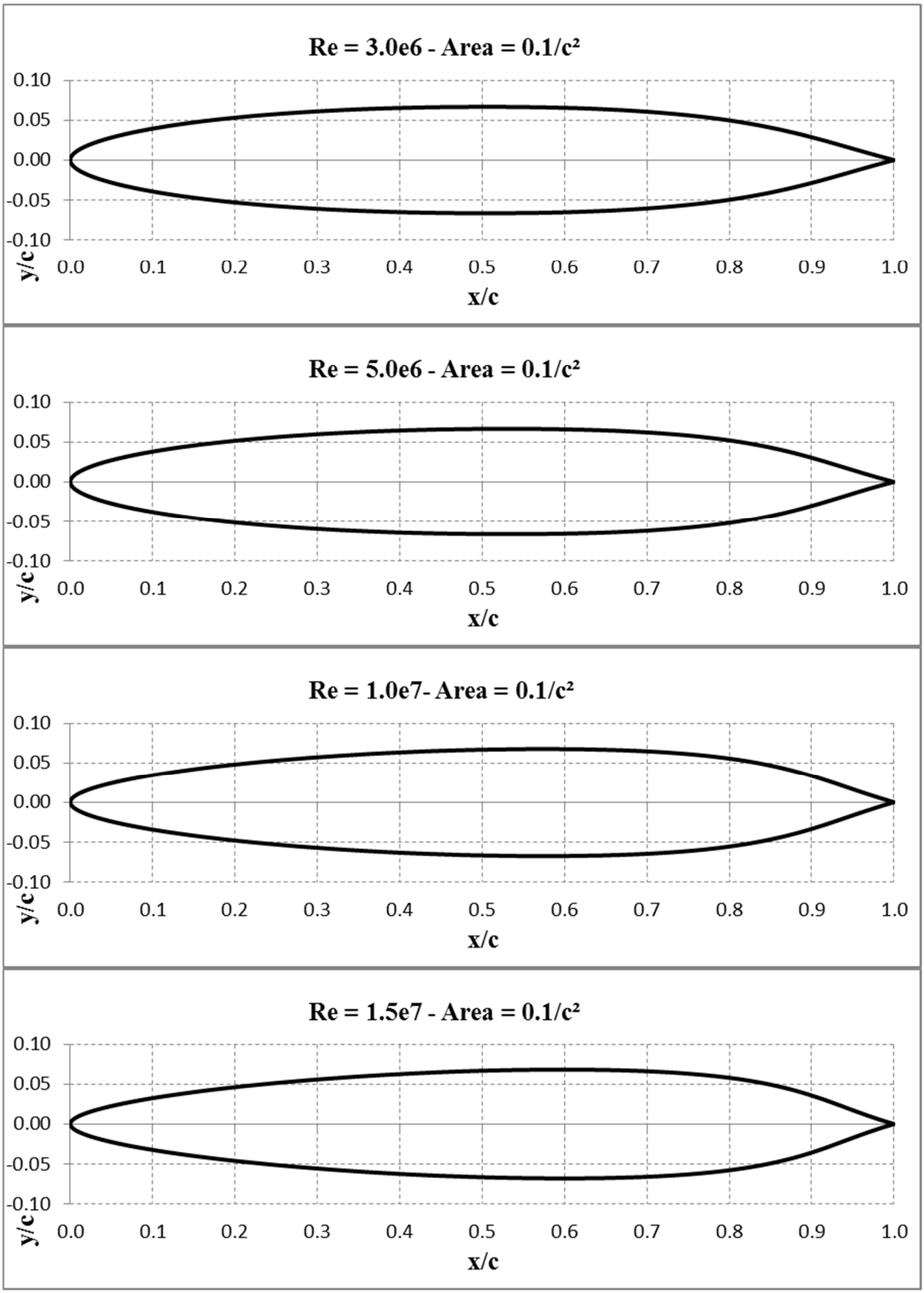

Re = 5.0e6 - Area = 0.075/c²
0.08
0.04
0.00
-0.04
-0.08
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

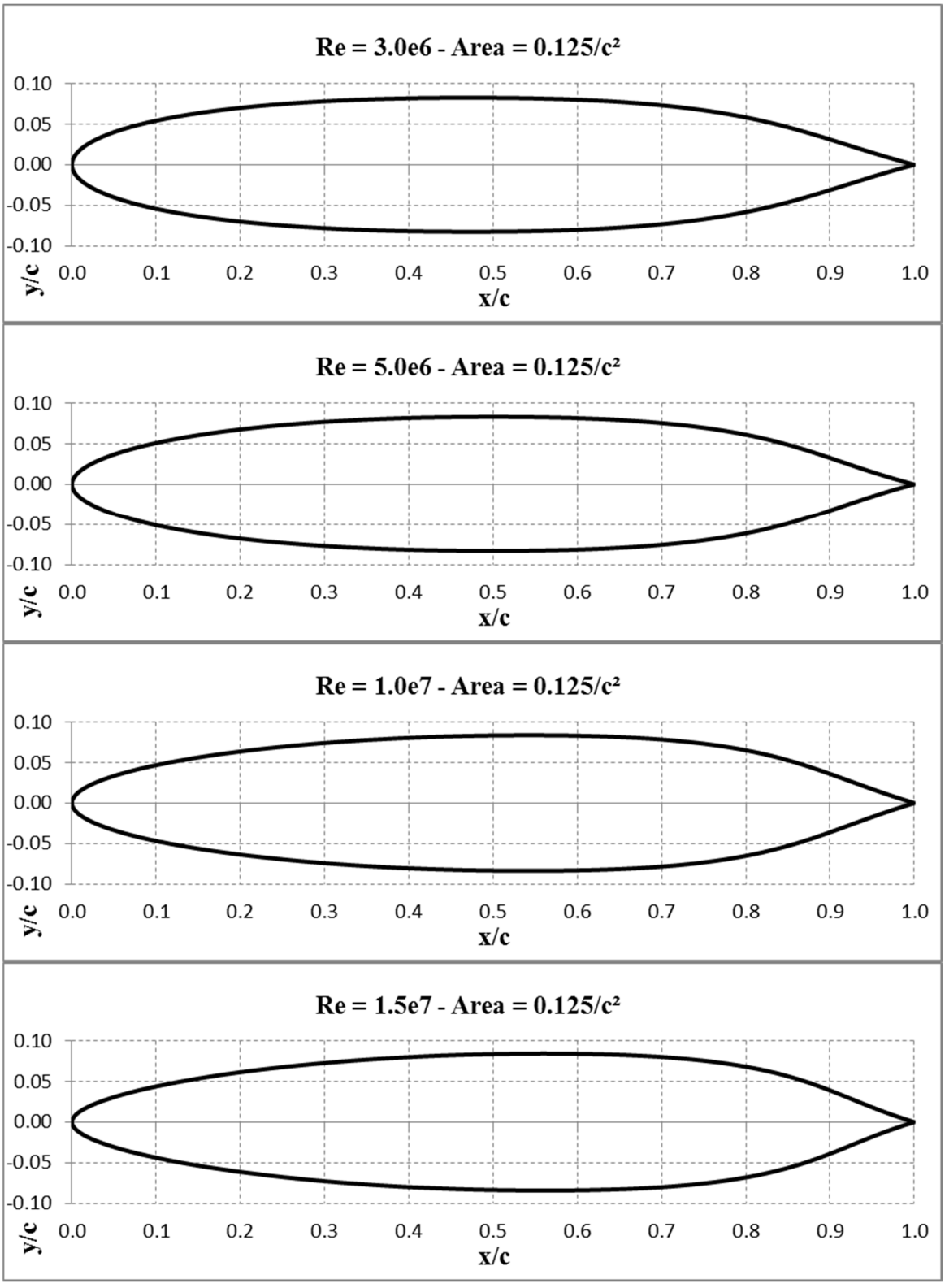

Re = 1.0e7 - Area = 0.075/c²
0.08
0.04
0.00
-0.04
-0.08
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

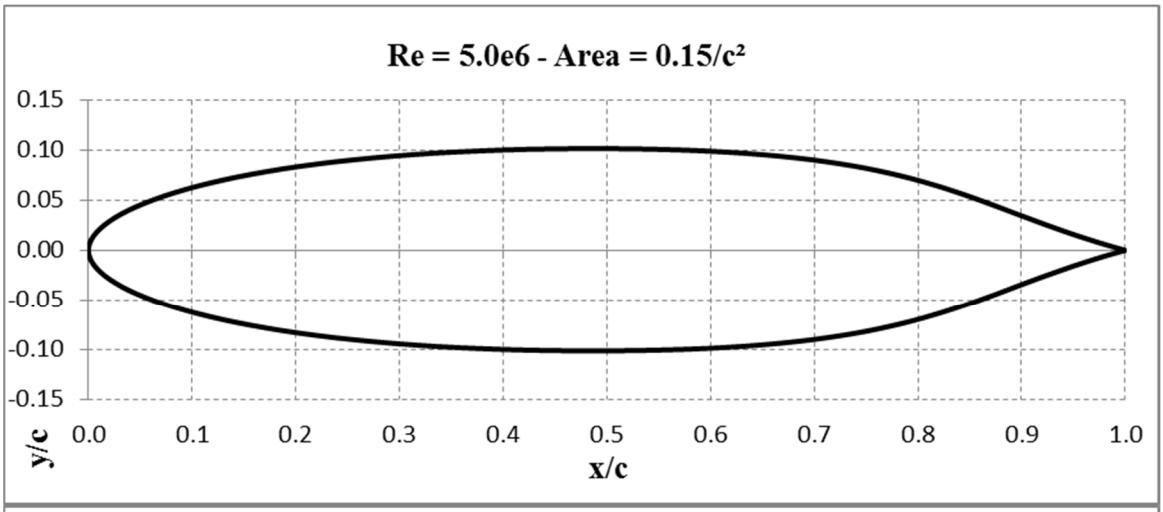

Re = 1.5e7 - Area = 0.075/c²
0.08
0.04
0.00
-0.04
-0.08
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

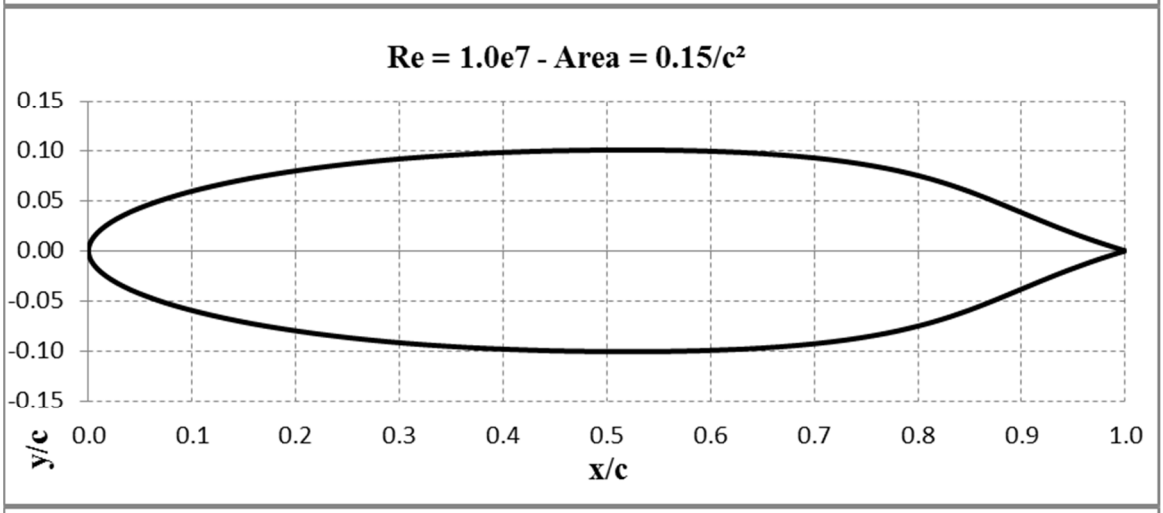

Re = 5.0e5 - Area = 0.1/c²
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

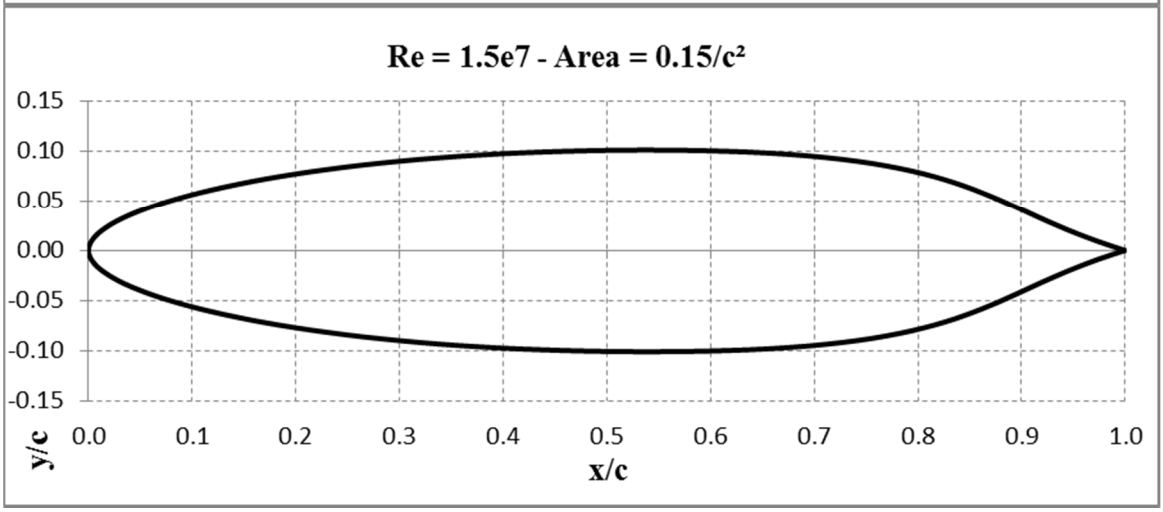

Re = 7.0e5 - Area = 0.1/c²
Re = 1.0e6 - Area = 0.1/c²
Re = 1.5e6 - Area = 0.1/c²
Re = 2.0e6 - Area = 0.1/c²
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c


Re = 3.0e6 - Area = 0.1/c²
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c
Re = 5.0e6 - Area = 0.1/c²
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c
Re = 1.0e7- Area = 0.1/c²
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c
Re = 1.5e7 - Area = 0.1/c²
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

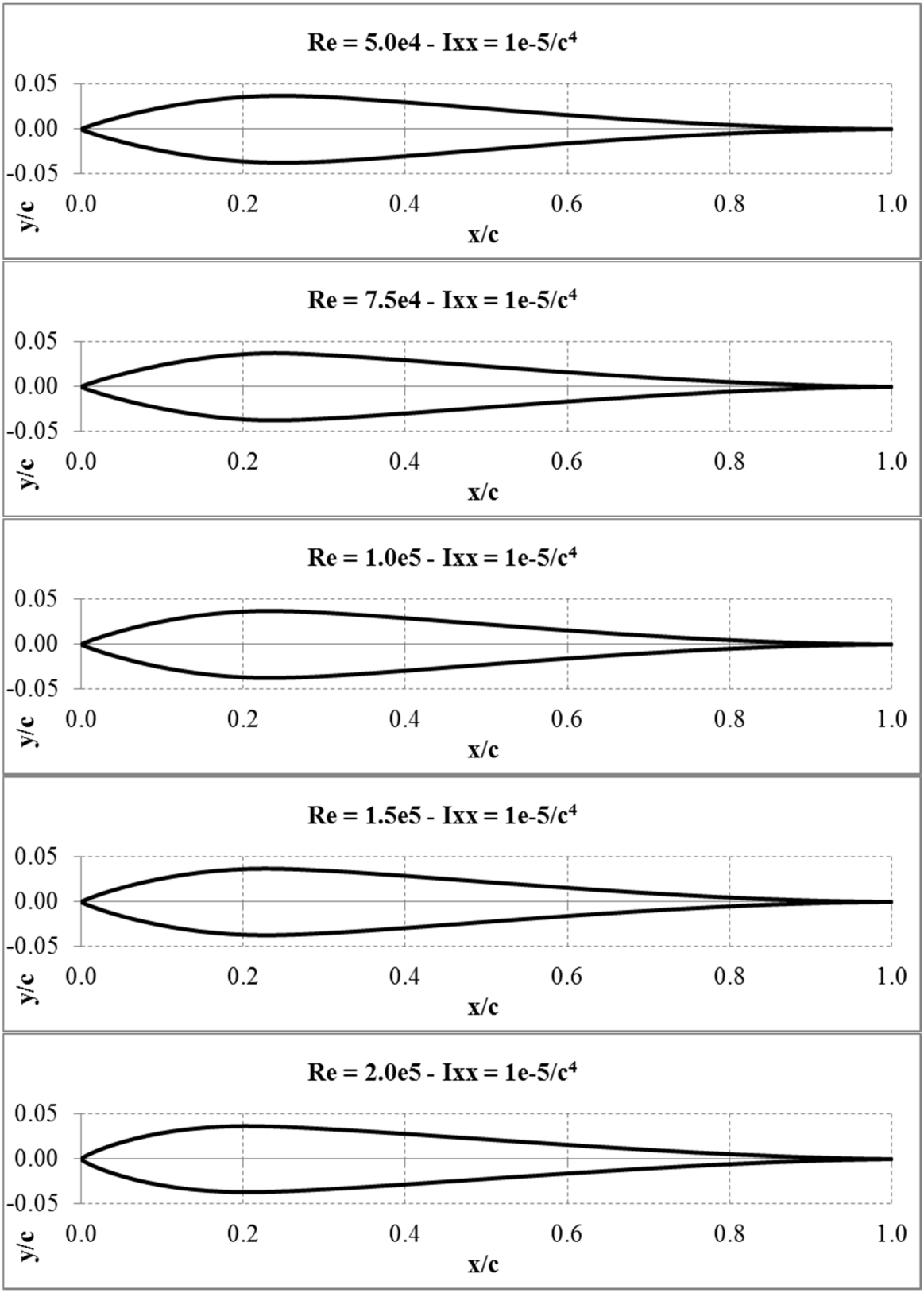

Re = 3.0e6 - Area = 0.125/c²
Re = 5.0e6 - Area = 0.125/c²
Re = 1.0e7 - Area = 0.125/c²
Re = 1.5e7 - Area = 0.125/c²
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

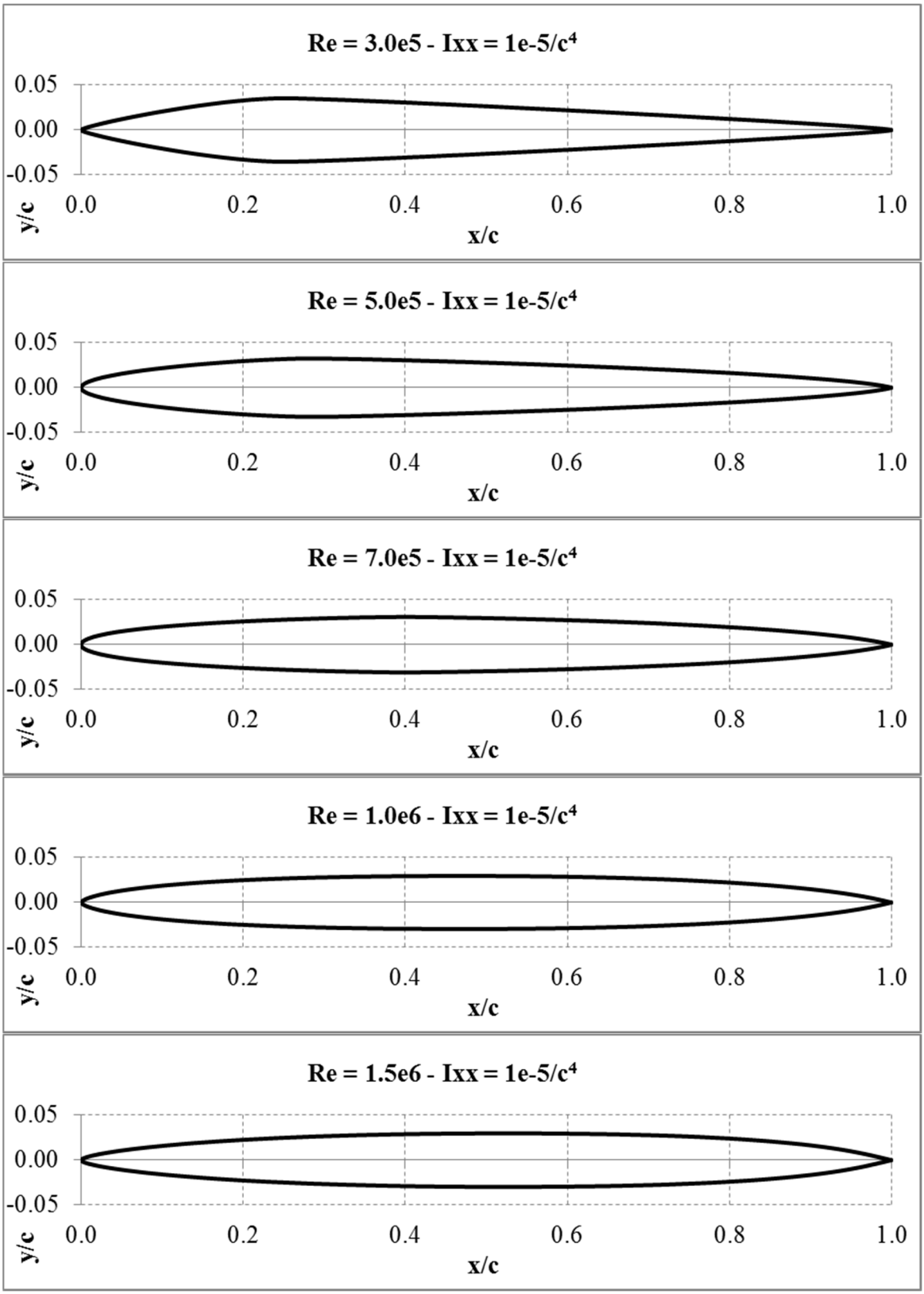

Re = 5.0e6 - Area = 0.15/c²
0.15
0.10
0.05
0.00
-0.05
-0.10
-0.15
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

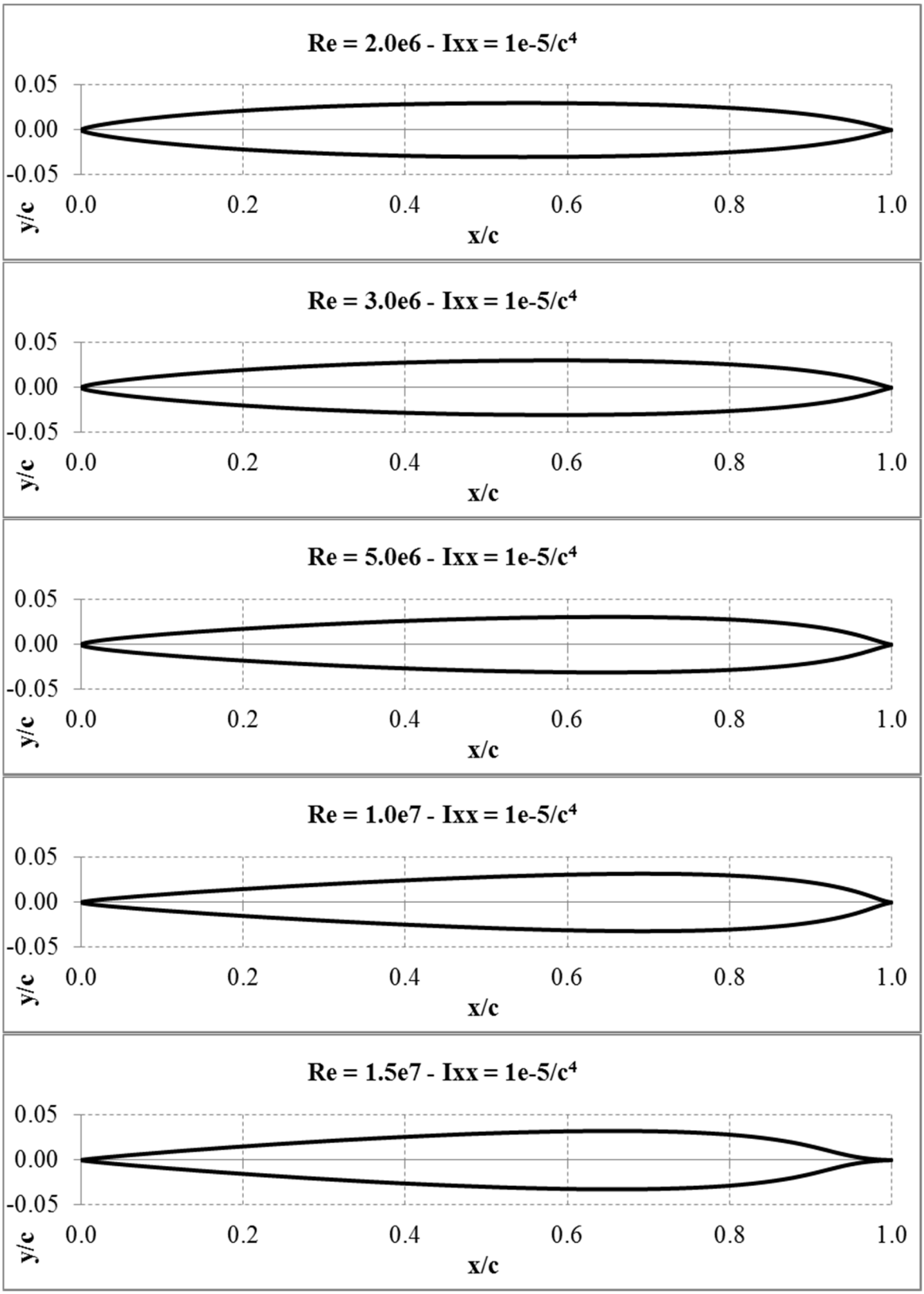

Re = 1.0e7 - Area = 0.15/c²
0.15
0.10
0.05
0.00
-0.05
-0.10
-0.15
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

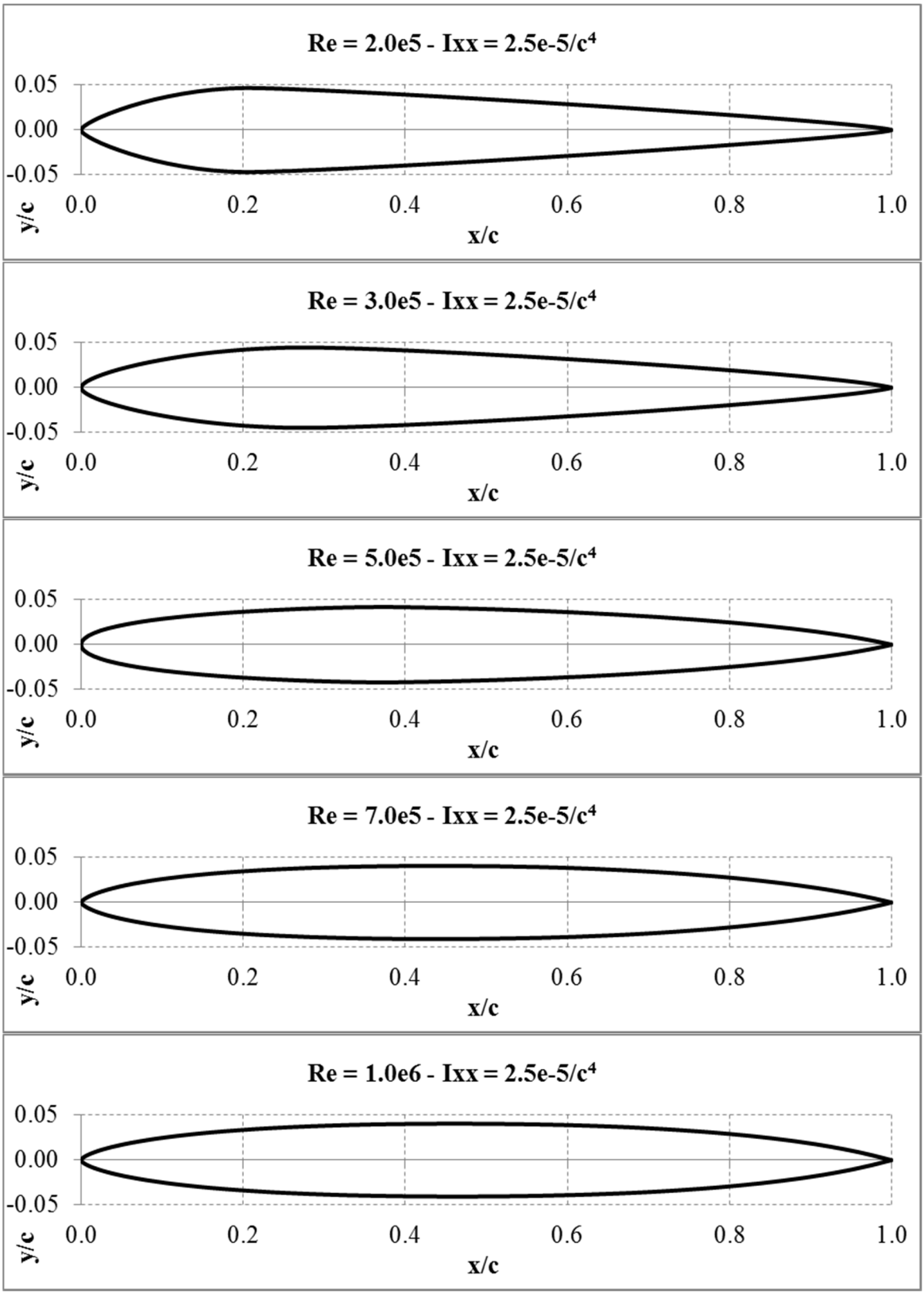

Re = 1.5e7 - Area = 0.15/c²
0.15
0.10
0.05
0.00
-0.05
-0.10
-0.15
y/c
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
x/c

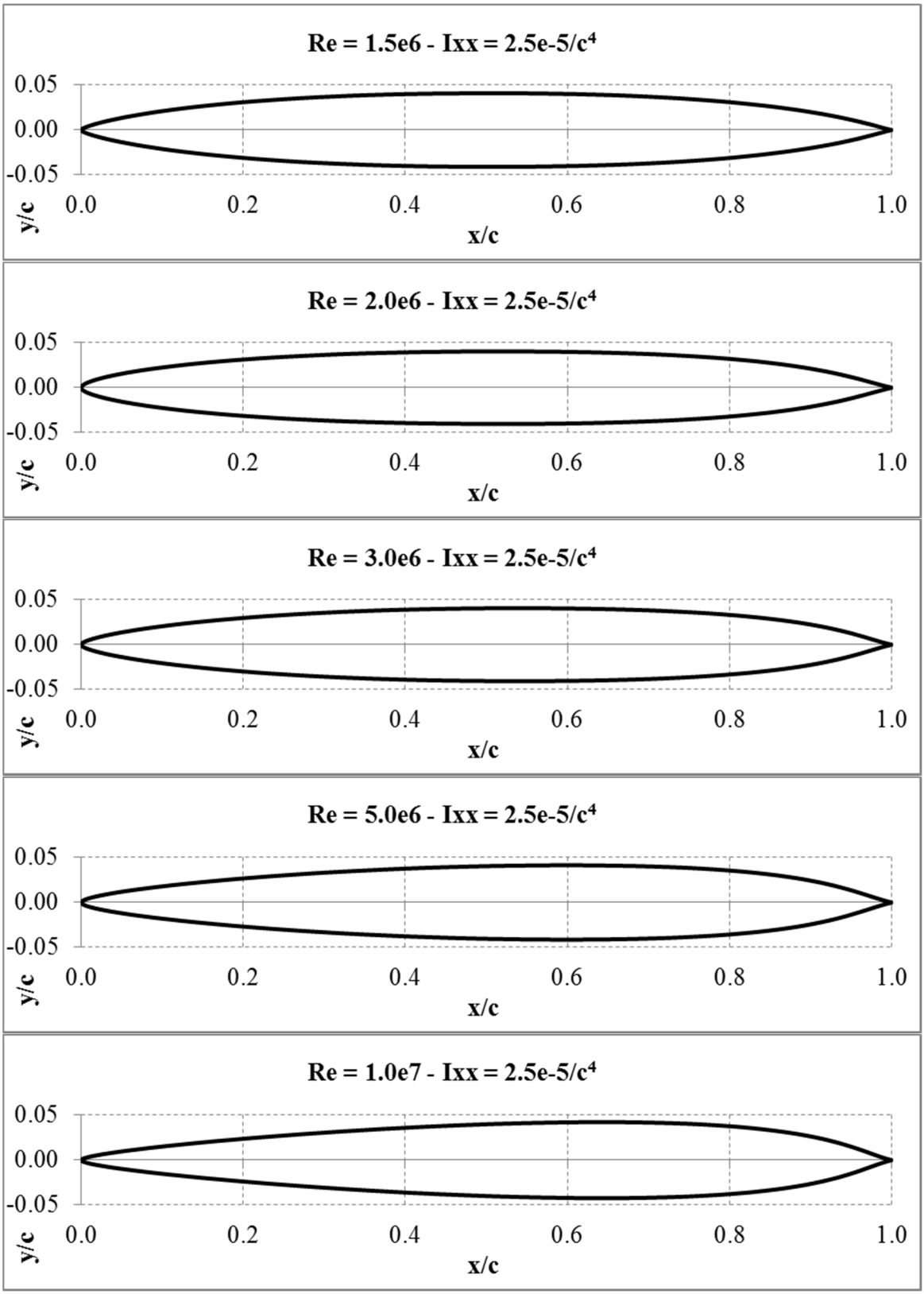
Appendix B – Optimized moment of inertia constrained airfoil shapes

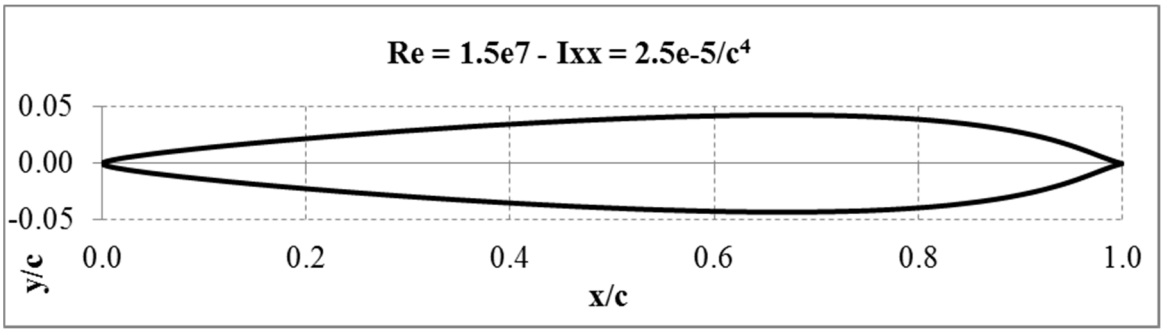


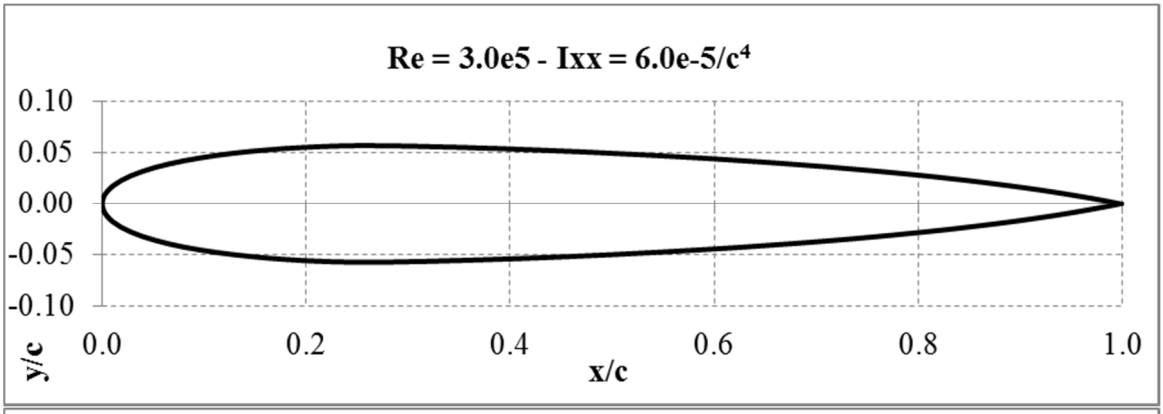

Re = 3.0e5 - Ixx = 1e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 5.0e5 - Ixx = 1e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 7.0e5 - Ixx = 1e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 1.0e6 - Ixx = 1e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 1.5e6 - Ixx = 1e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

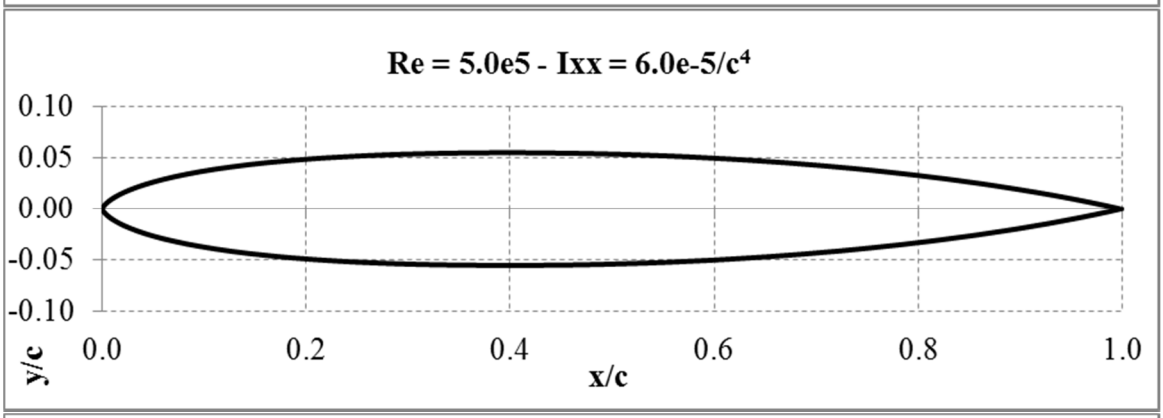

Re = 2.0e6 - Ixx = 1e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 3.0e6 - Ixx = 1e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 5.0e6 - Ixx = 1e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 1.0e7 - Ixx = 1e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 1.5e7 - Ixx = 1e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

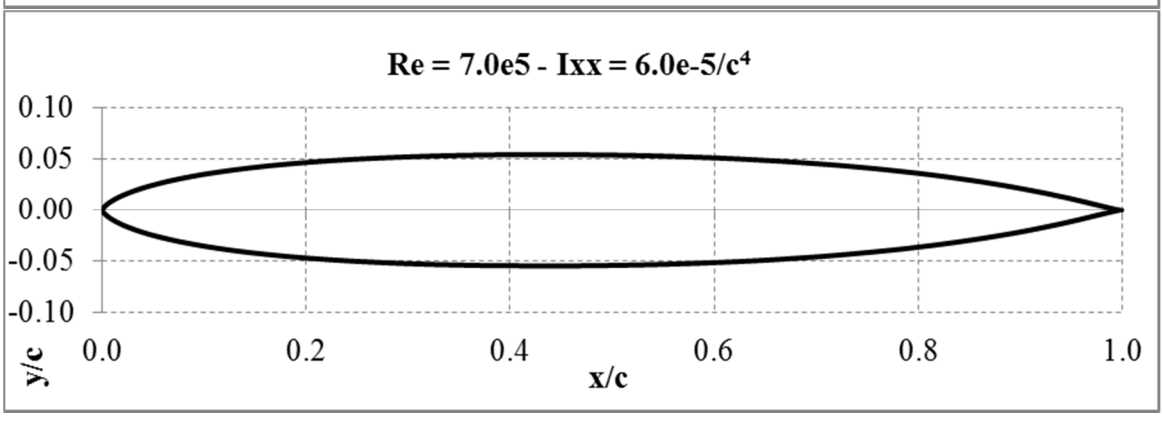

Re = 2.0e5 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 3.0e5 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 5.0e5 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 7.0e5 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 1.0e6 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

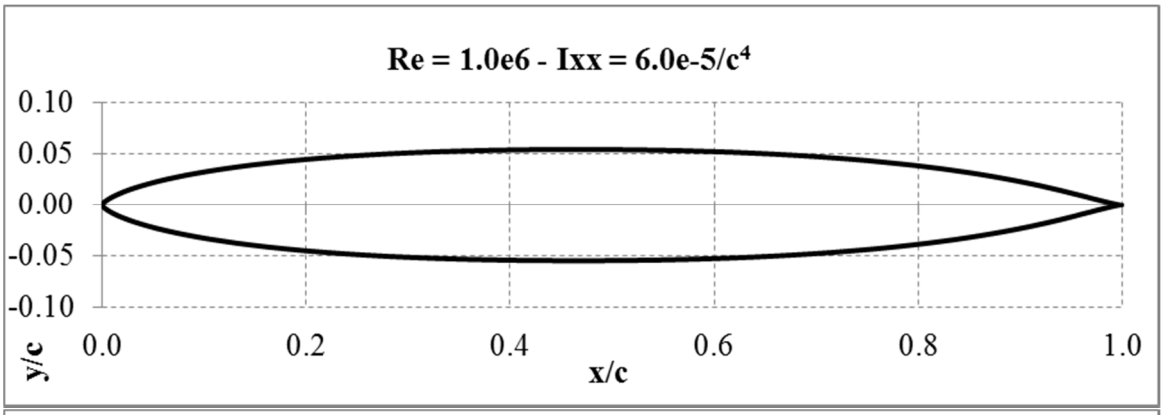

Re = 1.5e6 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 2.0e6 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 3.0e6 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 5.0e6 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 1.0e7 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

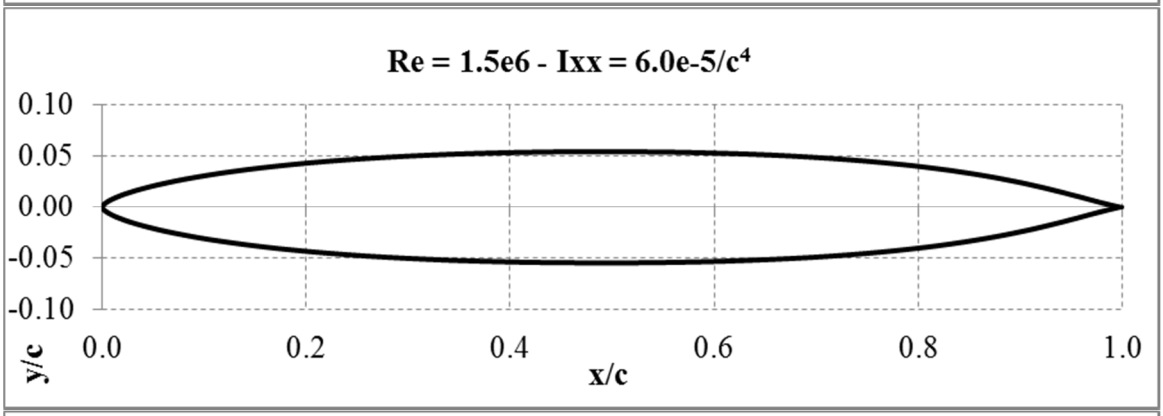

Re = 1.5e7 - Ixx = 2.5e-5/c⁴
0.05
0.00
-0.05
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

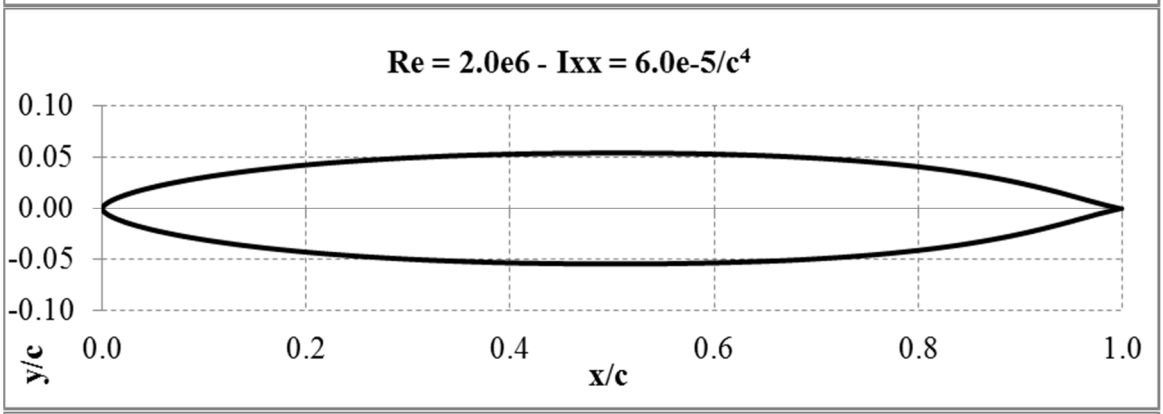

Re = 3.0e5 - Ixx = 6.0e-5/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

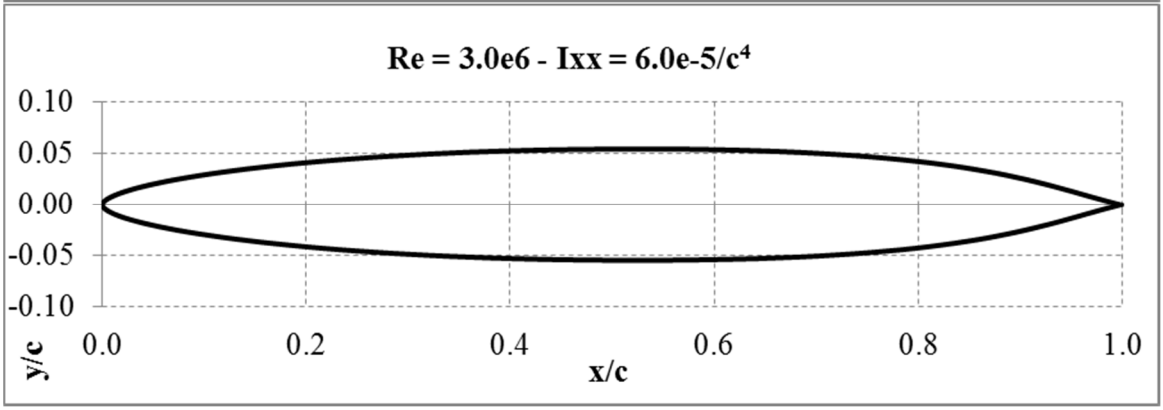

Re = 5.0e5 - Ixx = 6.0e-5/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

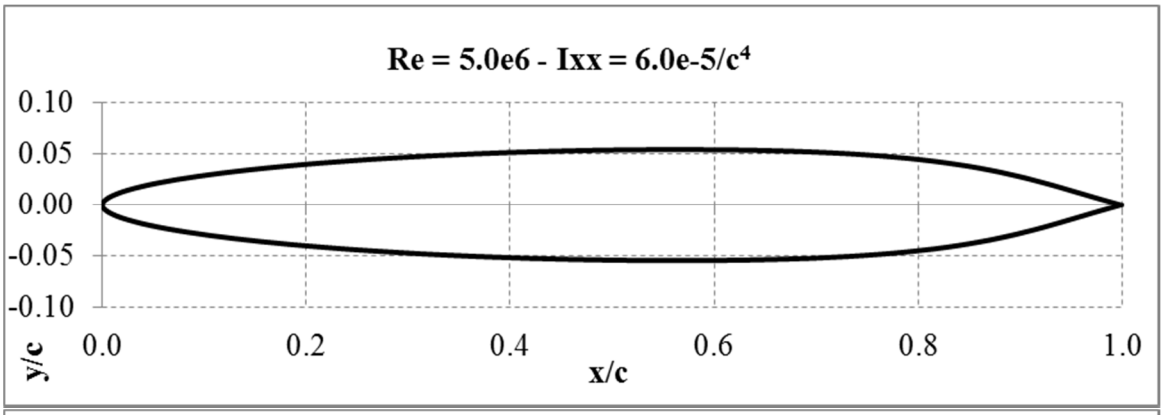

Re = 7.0e5 - Ixx = 6.0e-5/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

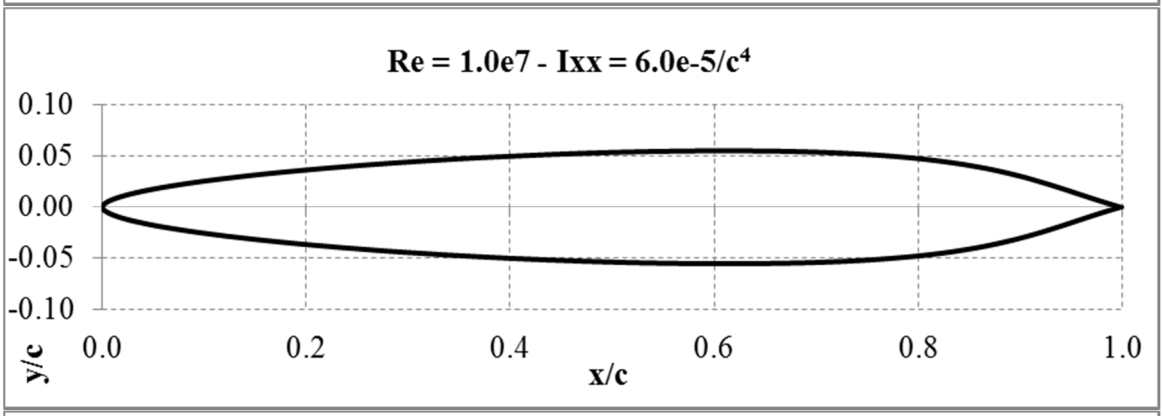

Re = 1.0e6 - Ixx = 6.0e-5/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

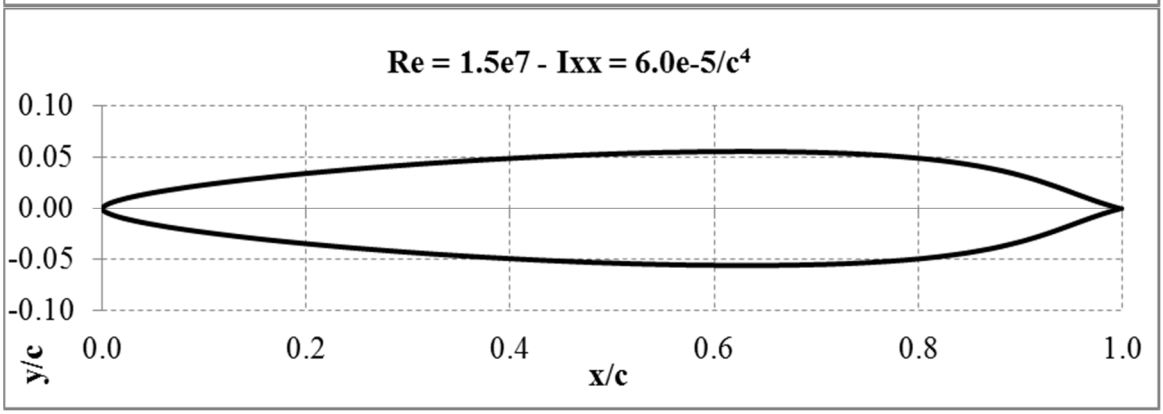

Re = 1.5e6 - Ixx = 6.0e-5/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

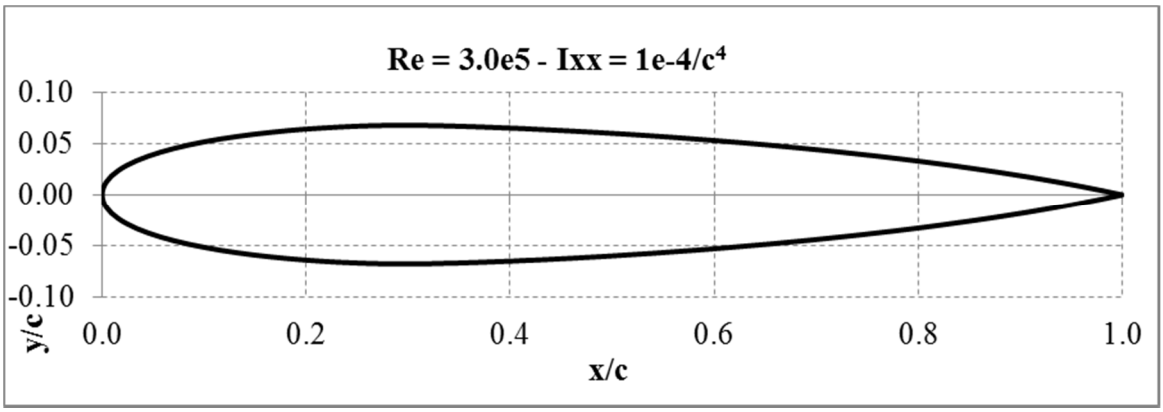

Re = 2.0e6 - Ixx = 6.0e-5/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

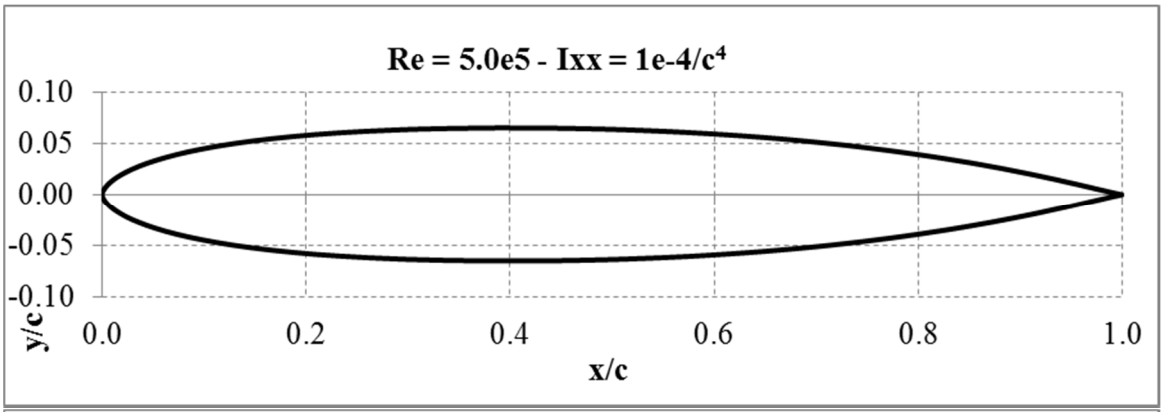

Re = 3.0e6 - Ixx = 6.0e-5/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

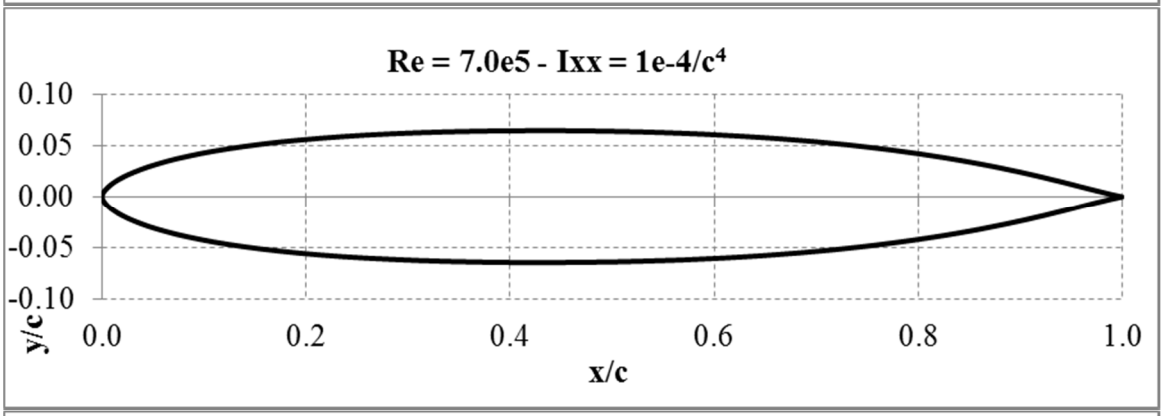

Re = 5.0e6 - Ixx = 6.0e-5/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

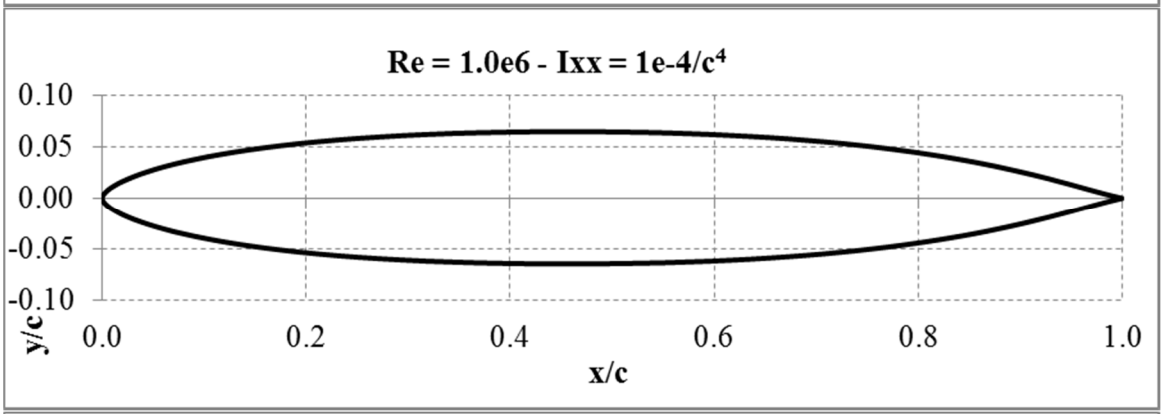

Re = 1.0e7 - Ixx = 6.0e-5/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

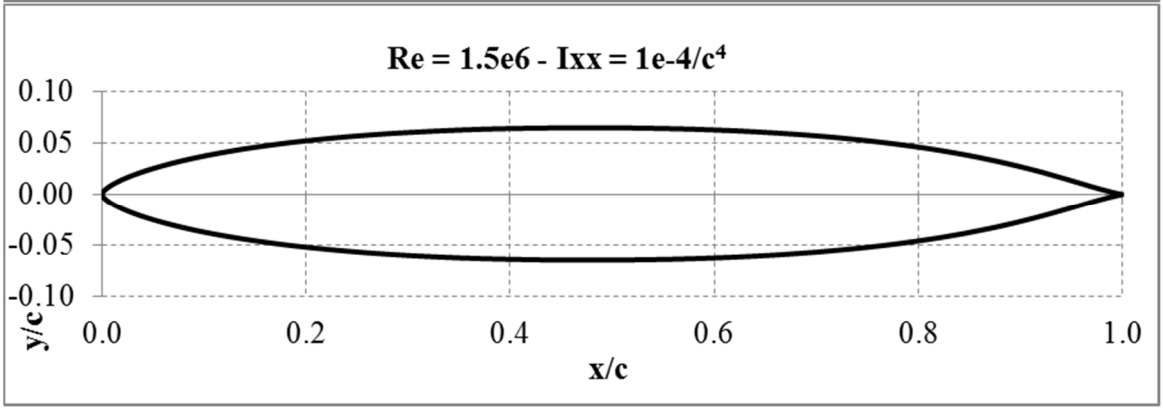

Re = 1.5e7 - Ixx = 6.0e-5/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

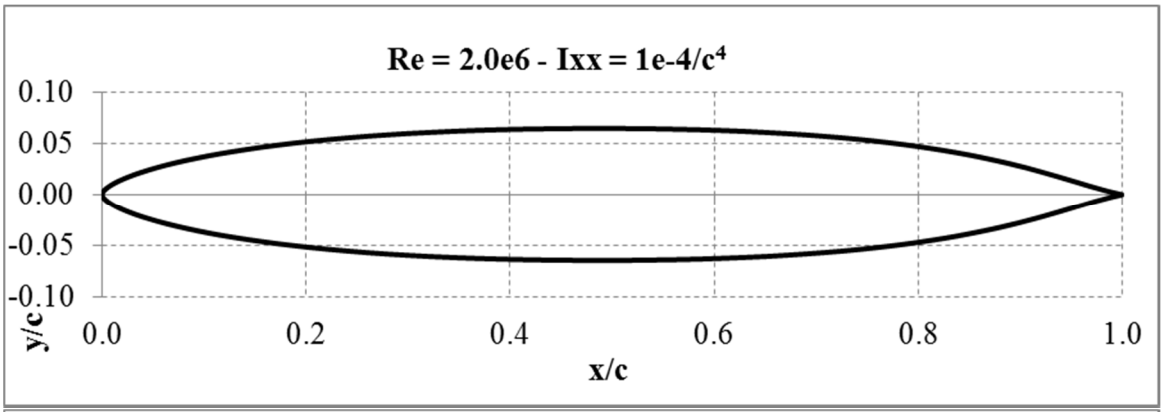

Re = 3.0e5 - Ixx = 1e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

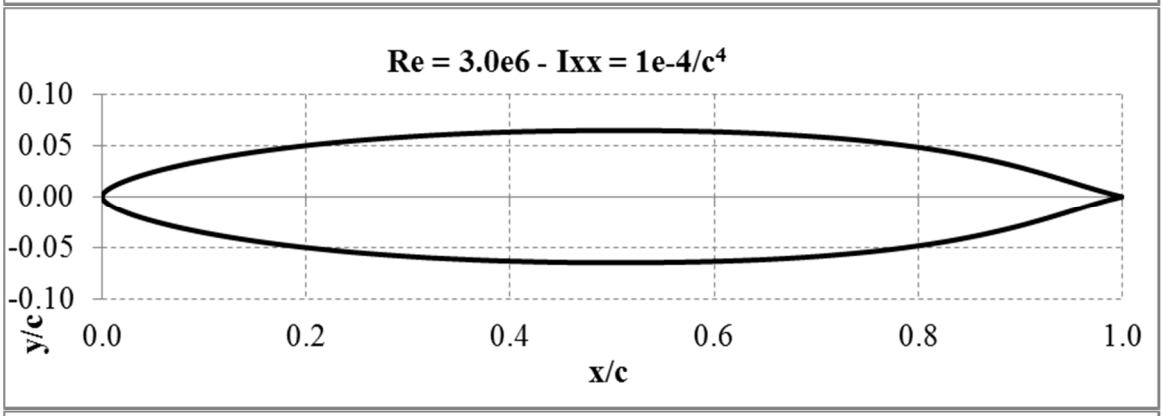

Re = 5.0e5 - Ixx = 1e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

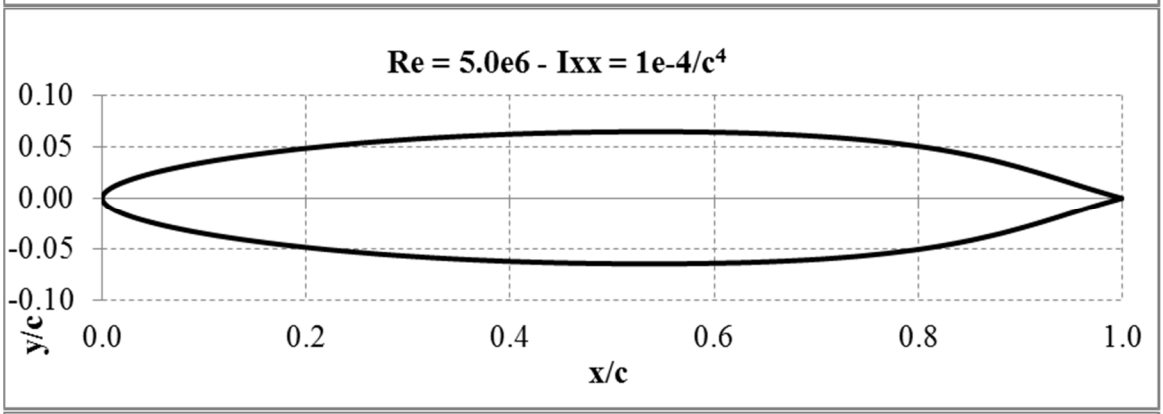

Re = 7.0e5 - Ixx = 1e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

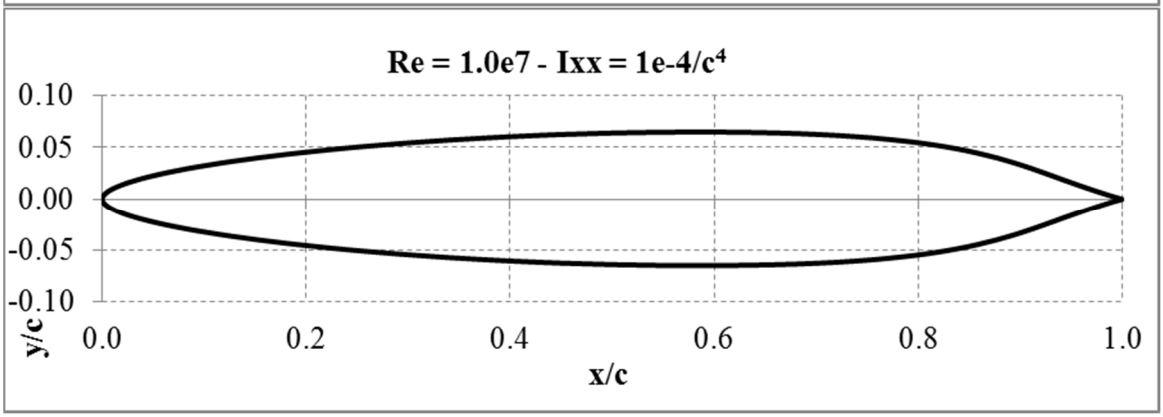

Re = 1.0e6 - Ixx = 1e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

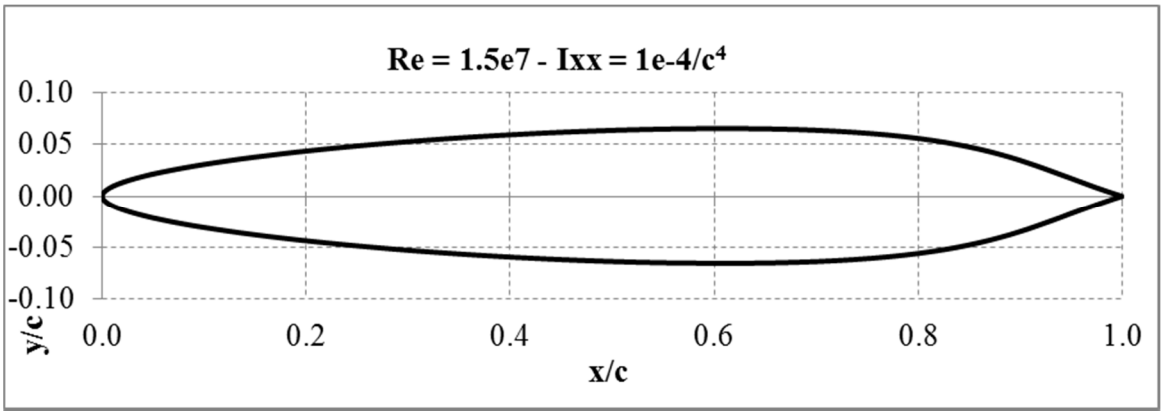

Re = 1.5e6 - Ixx = 1e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

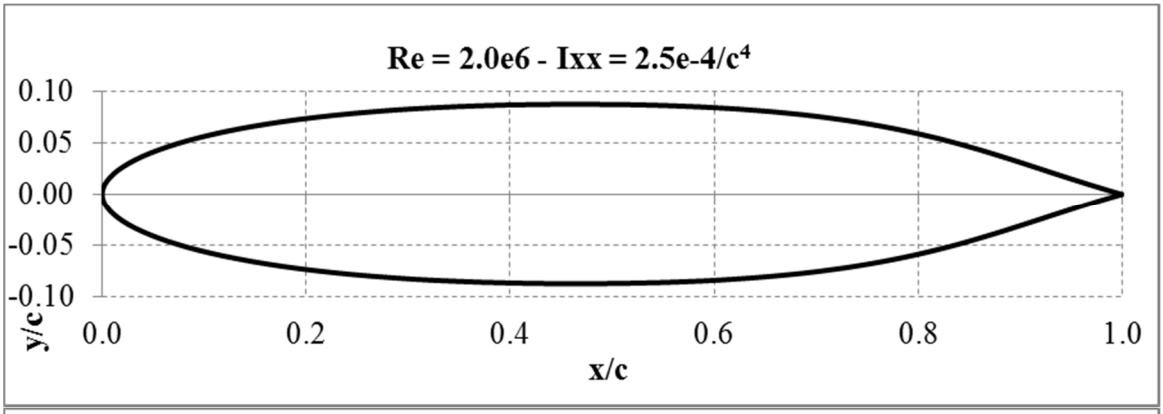

Re = 2.0e6 - Ixx = 1e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

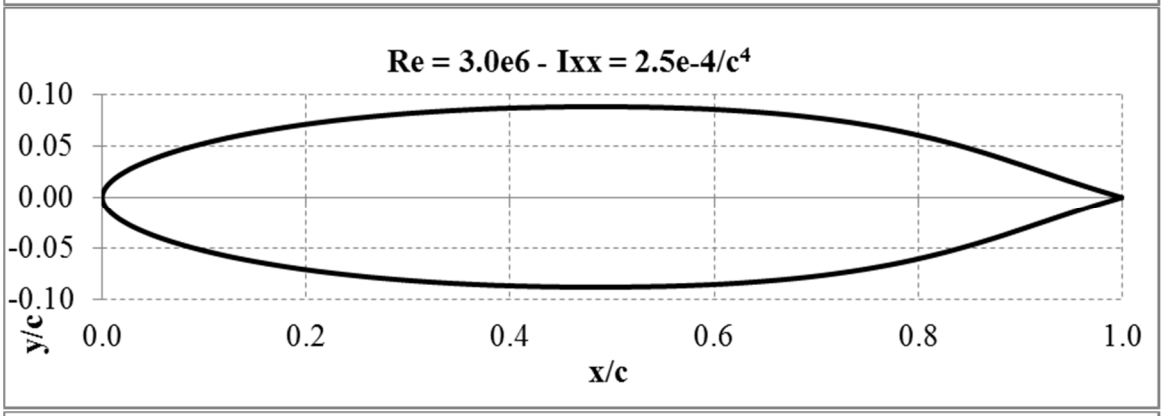

Re = 3.0e6 - Ixx = 1e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

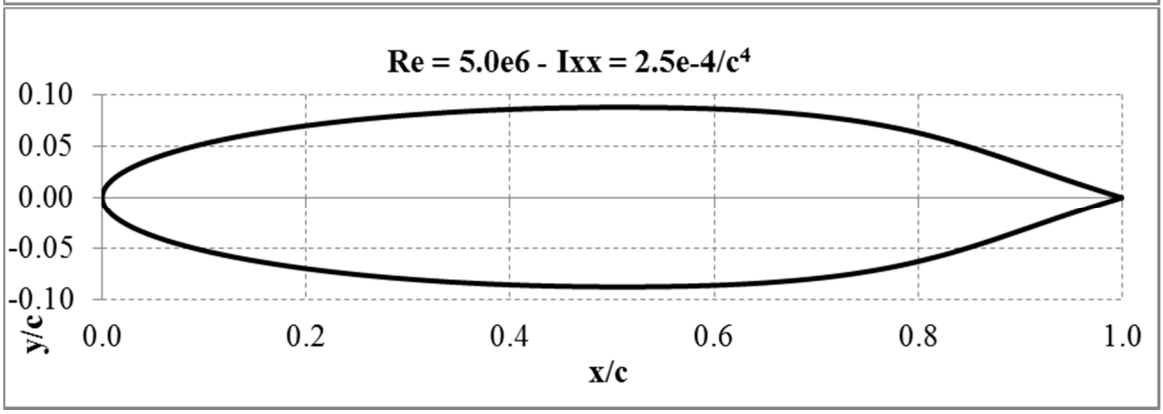

Re = 5.0e6 - Ixx = 1e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

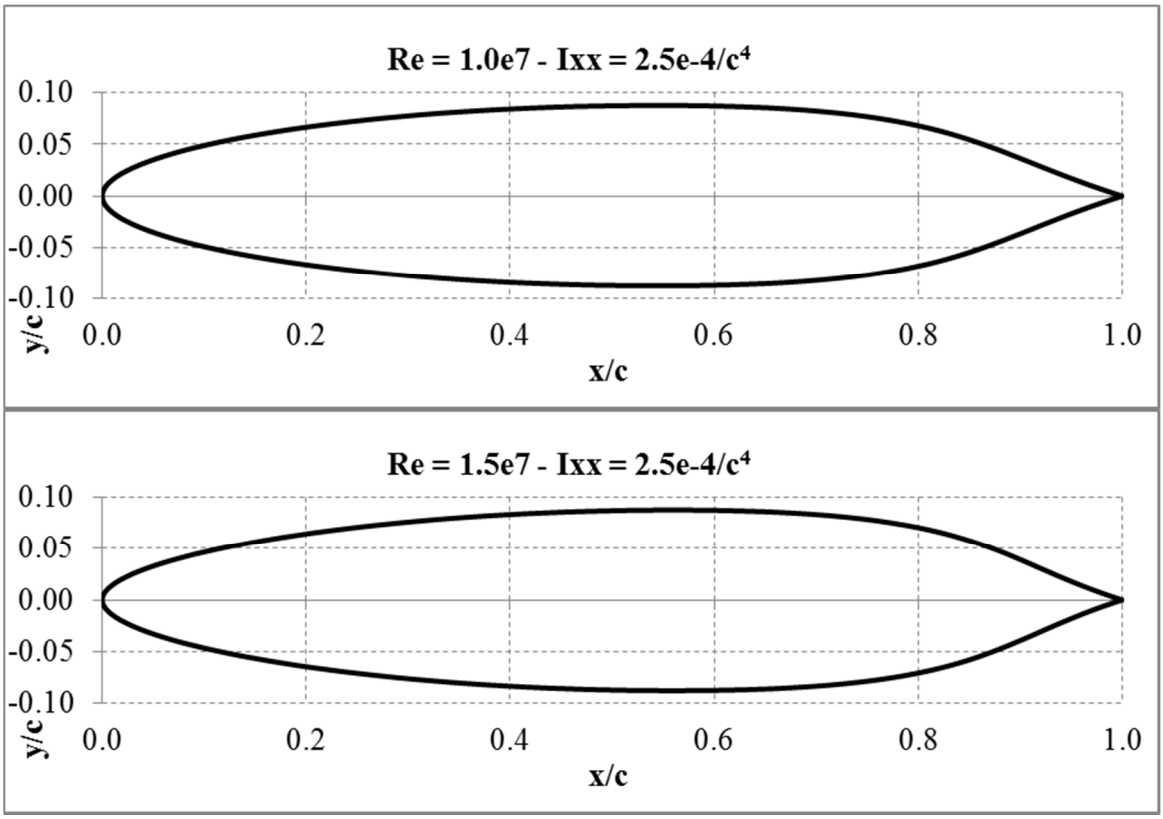

Re = 1.0e7 - Ixx = 1e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c


Re = 1.5e7 - Ixx = 1e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c


Re = 2.0e6 - Ixx = 2.5e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c


Re = 3.0e6 - Ixx = 2.5e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c


Re = 5.0e6 - Ixx = 2.5e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c


Re = 1.0e7 - Ixx = 2.5e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Re = 1.5e7 - Ixx = 2.5e-4/c⁴
0.10
0.05
0.00
-0.05
-0.10
y/c
0.0
0.2
0.4
0.6
0.8
1.0
x/c

Appendix C – Fortran90 computer code (Area constrained)

```
!---File: main.f90 ---------------------------------------------------+
!
    PROGRAM main
!
    USE interfaces
!
!---Variable Declarations
!---User Inputs
    REAL :: REYN_NUM(10)                !Inputted below
    REAL :: MACH_NUM = 0.1
    REAL :: ALPHA = 0.0
    INTEGER, PARAMETER :: D = 6         !Number of parameters (fixed)
    INTEGER, PARAMETER :: NP = 100      !Number of population members
    INTEGER, PARAMETER :: GMAX = 500    !Max num of generations
    INTEGER, PARAMETER :: DE_SET = 2    !DE Operator scheme
    REAL :: F = 0.3                     !Scale factor
    REAL :: CR = 0.8                    !Cross-over control constant
    REAL :: KC = 1.0                    !Coefficient of combination
    REAL :: DES_AREA = 0.125            !Set cross-sectional area
    REAL :: CONVERGE = 0.000001         !Convergence criteria
!
!---DE arrays
    REAL :: XP(NP,D), VP(NP,D), UP(NP,D), XLO(D), XHI(D)
    REAL :: COST(NP,GMAX)
    REAL :: AFOIL(1,D), U_AMPF(NP), X_AMPF(NP), O_AMPF
    INTEGER :: G, ICOUNT=0, IERR
!
!---Output files
    CHARACTER*32 :: FILENAME1, FILENAME2, FILENAME3
    CHARACTER*32 :: FILENAME4, FILENAME5
!====================================================================+
!Panel and Inverse Boundary-Layer Method Variables
    INTEGER, PARAMETER :: NODTOT=160, NANGLE=1
    REAL :: ALPHAS(NANGLE), X(NODTOT+1), Y(NODTOT+1)
    INTEGER :: NBT
    REAL :: CL_RES(NANGLE), CD_RES(NANGLE), CM_RES(NANGLE)
!
    NBT = NODTOT+1
    LXY = NODTOT+1
    LAA = 1
    ALPHAS(1) = ALPHA
!====================================================================+
!---Range of Reynolds Numbers
    REYN_NUM(1) = 15000000.
    REYN_NUM(2) = 10000000.
    REYN_NUM(3) = 5000000.
    REYN_NUM(4) = 3000000.
    REYN_NUM(5) = 2000000.
    REYN_NUM(6) = 1500000.
    REYN_NUM(7) = 1000000.
    REYN_NUM(8) = 700000.
    REYN_NUM(9) = 500000.
    REYN_NUM(10)= 300000.
!
    DO K = 1, 7
!
!---Set the parameter constraints
        CALL PARAMS(XLO, XHI, D)
        ICOUNT = 0
        I = 1
        DO WHILE(I .LE. NP)
!
!           Generate initial population member
            CALL INIPOP(XP, XLO, XHI, NP, D, I)
!
```

```
!           Generate the points for the panel method
            CALL GEOM(X, Y, XP, DES_AREA, X_AMPF(I), NODTOT, I, NP, D, IERR)
!
!           If the member is valid then evaluate it
            IF(IERR .EQ. 0) THEN
!               Evaluate the cost of each of the initial members
                ICOUNT = ICOUNT + 1
                CALL PANEL_IBL(NODTOT, 1, X, Y, LXY,                    &
                &              ALPHAS, LAA, MACH_NUM, REYN_NUM(K),      &
                &              CL_RES, CD_RES, CM_RES, 1)
!
                COST(I,1) = CD_RES(1)
                IF( COST(I,1) .LT. 0.0001 ) COST(I,1) = 0.1
                I = I + 1
!           If the member is invalid then recreate it (i is not indexed)
            ELSE
                I = I
            END IF
        END DO
!
        DO I = 1, NP
            OPTCOST = 10.0  !Keep track of best population member
!           Check if current population member is the best
            IF(COST(I,1) .LT. OPTCOST) THEN
!               Store the best set of genes
                DO J = 1, D
                    AFOIL(1,J) = XP(I,J)
                END DO
                OPTCOST = COST(I,1)
            END IF
        END DO
        G = 1
        RESIDUAL = 1.0
!
        WRITE(FILENAME1,310) K
310     FORMAT('converge_',I0,'.out')
        OPEN(21,FILE=FILENAME1)
        OPEN(26,FILE="STATUS.TXT")
!
        DO WHILE(G .LT. GMAX .AND. CONVERGE .LT. RESIDUAL)
!
            G = G + 1
            WRITE(26,*) G
!
            DO I = 1, NP
                R1 = I
                R2 = I
                R3 = I
                R4 = I
                R5 = I
                DO WHILE(R1 .EQ. I)
                    CALL RANDOM_NUMBER(RAND)
                    R1 = INT(RAND * NP) + 1
                END DO
                DO WHILE(R2 .EQ. R1 .OR. R2 .EQ. I)
                    CALL RANDOM_NUMBER(RAND)
                    R2 = INT(RAND * NP) + 1
                END DO
                DO WHILE(R3 .EQ. R2 .OR. R3 .EQ. R1 .OR.                &
                &        R3 .EQ. I)
                    CALL RANDOM_NUMBER(RAND)
                    R3 = INT(RAND * NP) + 1
                END DO
                DO WHILE(R4 .EQ. R3 .OR. R4 .EQ. R2 .OR.                &
                &        R4 .EQ. R1 .OR. R4 .EQ. I)
```

```
                    CALL RANDOM_NUMBER(RAND)
                    R4 = INT(RAND * NP) + 1
                END DO
                DO WHILE(R5 .EQ. R4 .OR. R5 .EQ. R3 .OR.                &
                &        R5 .EQ. R2 .OR. R5 .EQ. R1 .OR.                &
                &        R5 .EQ. I)
                    CALL RANDOM_NUMBER(RAND)
                    R5 = INT(RAND * NP) + 1
                END DO
!
!               Mutation and Crossover
                DO J = 1, D
                    IF(DE_SET .EQ. 1) THEN
!                   "DE/rand/1"
                    VP(I,J) = XP(R1,J) + F*(XP(R2,J) - XP(R3,J))
                    ELSE IF(DE_SET .EQ. 2) THEN
!                   "DE/rand-to-best/2"
                    VP(I,J) = XP(R1,J) + F*(AFOIL(1,J) - XP(R1,J)) +    &
                    &         F*(XP(R2,J) - XP(R3,J)) +                 &
                    &         F*(XP(R4,J) - XP(R5,J))
                    END IF
!
                    CALL RANDOM_NUMBER(RAND1)
                    IF(RAND1 .LE. CR) THEN
                        UP(I,J) = VP(I,J)
                    ELSE
                        UP(I,J) = XP(I,J)
                    END IF
                END DO
!
!               Check to ensure new values are within constraints
                DO J = 1, D
                    IF(UP(I,J) .LT. XLO(J)) THEN
                        UP(I,J) = (XP(I,J) + XLO(J)) / 2.0
                    ELSE IF(UP(I,J) .GT. XHI(J)) THEN
                        UP(I,J) = (XP(I,J) + XHI(J)) / 2.0
                    END IF
                END DO
            END DO
!
            DO I = 1, NP
!               Generate the points for the panel method
                CALL GEOM(X, Y, UP, DES_AREA, U_AMPF(I), NODTOT,        &
                &         I, NP, D, IERR)
!
                IF(IERR .EQ. 0) THEN
!                   Evaluate the cost of each of the initial members
                    ICOUNT = ICOUNT + 1
!
                    CALL PANEL_IBL(NODTOT, 1, X, Y, LXY,                &
                    &              ALPHAS, LAA, MACH_NUM, REYN_NUM(K),  &
                    &              CL_RES, CD_RES, CM_RES, 1)
!
                    SCORE = CD_RES(1)
                    IF( SCORE .LT. 0.0001 ) SCORE = 0.1
                ELSE
                    SCORE = 10.0
                END IF
!
                IF(SCORE .LT. COST(I,G-1)) THEN
                    DO J = 1, D
                        XP(I,J) = UP(I,J)
                        X_AMPF(I) = U_AMPF(I)
                    END DO
                    COST(I,G) = SCORE
```

```
                ELSE
                    COST(I,G) = COST(I,G-1)
                END IF
!
!               Check if current population member is the best
                IF(COST(I,G) .LT. OPTCOST) THEN
!                   Store the best set of genes
                    DO J = 1, D
                        AFOIL(1,J) = XP(I,J)
                    END DO
                    O_AMPF = X_AMPF(I)
                    OPTCOST = COST(I,G)
                END IF
            END DO
!
            CALL RESID(RESIDUAL, COST, G, NP, GMAX)
            WRITE(21,*) RESIDUAL
!
        END DO
!
        CLOSE(21)
        CLOSE(26)
!
!       Write the cost array to file
        WRITE(FILENAME2,320) K
320     FORMAT('cost_',I0,'.out')
        OPEN(22,FILE=FILENAME2, STATUS="REPLACE", RECL=9*NP)
        WRITE(22,100) COST
100     FORMAT(100(2X, F7.6))
        CLOSE(22)
!
!       Write the DNA of the last generation to file
        WRITE(FILENAME3,330) K
330     FORMAT('DNA_',I0,'.out')
        OPEN(23,FILE=FILENAME3, STATUS="REPLACE")
        DO I = 1, NP
            WRITE(23,200) XP(I,1), XP(I,2), XP(I,3), XP(I,4),          &
            &             XP(I,5), XP(I,6), X_AMPF(I)
200         FORMAT(7(2X, F11.8))
        END DO
        CLOSE(23)
!
!   Generate the points of the best panel and write to file
!
        CALL GEOM(X, Y, AFOIL, DES_AREA, O_AMPF, NODTOT, 1, 1, D, IERR)
        WRITE(FILENAME4,340) K
340     FORMAT('afoil_',I0,'.out')
        OPEN(24,FILE=FILENAME4, STATUS="REPLACE")
        DO I = 1, NODTOT+1
            WRITE(24,*) X(I), Y(I)
        END DO
        CLOSE(24)
!
!       Write the summary file
        WRITE(FILENAME5,350) K
350     FORMAT('summary_',I0,'.out')
        OPEN(25,FILE=FILENAME5, STATUS="REPLACE")
        CALL POSTPRO(REYN_NUM(K), MACH_NUM, ALPHA, CONVERGE,           &
        &            NP, D, F, CR, KC, G, ICOUNT, OPTCOST,             &
        &            AFOIL, O_AMPF, DES_AREA)
        CLOSE(25)
    END DO
!
    STOP
    END PROGRAM main
```

```
!---File: interfaces.f90 ------------------------------------------+
!
MODULE interfaces

!====================================================================+
  INTERFACE
!--------------------------------------------------------------------+
    SUBROUTINE PANEL_IBL(numPanel, jjangle, xgeom, ygeom, Lxy,        &
    &                    attackangles, LAA, S_mach, Reynolds,         &
    &                    Cl_Res, Cd_Res, Cm_Res,iterative)
!--------------------------------------------------------------------+
    INTEGER :: LAA, Lxy, jjangle, numPanel
    REAL :: xgeom(Lxy), ygeom(Lxy), attackangles(LAA)
    REAL :: Cl_Res(jjangle), Cd_Res(jjangle), Cm_Res(jjangle)
    REAL :: S_mach, Reynolds
!--------------------------------------------------------------------+
    END SUBROUTINE PANEL_IBL
!--------------------------------------------------------------------+
  END INTERFACE

!====================================================================+
  INTERFACE
!--------------------------------------------------------------------+
    SUBROUTINE INIPOP(PROP, XLO, XHI, NUMP, D, ID)
!--------------------------------------------------------------------+
    INTEGER :: NUMP, D, ID
    REAL :: PROP(NUMP,D)
    REAL :: XLO(D), XHI(D)
!--------------------------------------------------------------------+
    END SUBROUTINE INIPOP
!--------------------------------------------------------------------+
  END INTERFACE

!====================================================================+
  INTERFACE
!--------------------------------------------------------------------+
    SUBROUTINE ROOTS(B9, RLE, KT, DZTE, BTE, XT, YT)
!--------------------------------------------------------------------+
    REAL :: B9
    REAL :: RLE, KT, DZTE, BTE, XT, YT
!--------------------------------------------------------------------+
    END SUBROUTINE ROOTS
!--------------------------------------------------------------------+
  END INTERFACE

!====================================================================+
  INTERFACE
!--------------------------------------------------------------------+
    SUBROUTINE GEOM(XCO, YCO, PROP, DES_AREA, AMPF, NODTOT, N, NUMP, D, IERR)
!--------------------------------------------------------------------+
    INTEGER :: NODTOT, N, NUMP, D
    REAL :: XCO(NODTOT+1), YCO(NODTOT+1), PROP(NUMP,D)
    REAL :: DES_AREA, AMPF
    INTEGER :: IERR
!--------------------------------------------------------------------+
    END SUBROUTINE GEOM
!--------------------------------------------------------------------+
  END INTERFACE

!====================================================================+
  INTERFACE
!--------------------------------------------------------------------+
    FUNCTION X_AREA(XCO, YCO, NODTOT)
!--------------------------------------------------------------------+
    INTEGER :: NODTOT
```

```
    REAL :: AREA
    REAL :: XCO(NODTOT+1), YCO(NODTOT+1)
!-----------------------------------------------------------------------+
    END FUNCTION X_AREA
!-----------------------------------------------------------------------+
  END INTERFACE

!=======================================================================+
  INTERFACE
!-----------------------------------------------------------------------+
    SUBROUTINE PARAMS(XLO, XHI, D)
!-----------------------------------------------------------------------+
    INTEGER :: D
    REAL :: XLO(D), XHI(D)
!-----------------------------------------------------------------------+
    END SUBROUTINE PARAMS
!-----------------------------------------------------------------------+
  END INTERFACE

!=======================================================================+
  INTERFACE
!-----------------------------------------------------------------------+
    SUBROUTINE RESID(RESIDUAL, COST, GENERATION, NUMP, GMAX)
!-----------------------------------------------------------------------+
    INTEGER :: NUMP, GMAX
    REAL :: COST(NUMP,GMAX)
    REAL :: RESIDUAL
    INTEGER :: GENERATION
!-----------------------------------------------------------------------+
    END SUBROUTINE RESID
!-----------------------------------------------------------------------+
  END INTERFACE

!=======================================================================+
  INTERFACE
!-----------------------------------------------------------------------+
    SUBROUTINE POSTPRO(REYN_NUM, MACH_NUM, ALPHA, CONVERGE,           &
    &                  NUMP, D, F, CR, K, GENERATION, ICOUNT, OPTCOST,  &
    &                  AFOIL, AMPF, DES_AREA)
!-----------------------------------------------------------------------+
    REAL :: REYN_NUM, MACH_NUM, ALPHA, CONVERGE
    INTEGER :: NUMP, D, GENERATION, ICOUNT
    REAL :: F, CR, K, OPTCOST, DES_AREA
    REAL :: AFOIL(1,D), AMPF
!-----------------------------------------------------------------------+
    END SUBROUTINE POSTPRO
!-----------------------------------------------------------------------+
  END INTERFACE

!=======================================================================+


END MODULE interfaces
```

```
!---File: params.f90 ------------------------------------------------+
    SUBROUTINE PARAMS(XLO, XHI, D)
!-------------------------------------------------------------------+
!
!   Calling arguments
    INTEGER :: D
    REAL :: XLO(D), XHI(D)

!   Local variables
!---Geometry constraints
    REAL :: RLE_MIN = -0.075    !Leading edge radius min (-0.04)
    REAL :: RLE_MAX = -0.001    !Leading edge radius max (-0.001)
    REAL :: KT_MIN  = -10.0     !Juncture curvature min (-25.0)
    REAL :: KT_MAX  = -0.01     !Juncture curvature max (-0.01)
    REAL :: DZTE_MIN= 0.0       !Trailing edge thick min (0.0)
    REAL :: DZTE_MAX= 0.0       !Trailing edge thick max (0.001)
    REAL :: BTE_MIN = 0.001     !Trailing edge wedge angle min (0.001)
    REAL :: BTE_MAX = 1.00      !Trailing edge wedge angle max (1.57)
    REAL :: XT_MIN  = 0.15      !Juncture (crest) x location min (0.15)
    REAL :: XT_MAX  = 0.70      !Juncture (crest) x location max (0.70)
    REAL :: YT_MIN  = 0.05      !Juncture (crest) y location min (0.05)
    REAL :: YT_MAX  = 0.25      !Juncture (crest) y location max (0.25)
!
!   Store the max and min values in gene vectors
    XLO(1) = RLE_MIN
    XHI(1) = RLE_MAX
    XLO(2) = KT_MIN
    XHI(2) = KT_MAX
    XLO(3) = DZTE_MIN
    XHI(3) = DZTE_MAX
    XLO(4) = BTE_MIN
    XHI(4) = BTE_MAX
    XLO(5) = XT_MIN
    XHI(5) = XT_MAX
    XLO(6) = YT_MIN
    XHI(6) = YT_MAX
!
    RETURN
    END SUBROUTINE PARAMS
```

```
!---File: inipop.f90 -------------------------------------------------+
    SUBROUTINE INIPOP(PROP, XLO, XHI, NUMP, D, ID)
!--------------------------------------------------------------------+
!
!---Calling arguments
    INTEGER :: NUMP, D, ID
    REAL :: PROP(NUMP,D)
    REAL :: XLO(D), XHI(D)
!
!---Local variables
    REAL :: RAND
!
    DO J = 1, D
        CALL RANDOM_NUMBER(RAND)
        PROP(ID,J) = XLO(J) + RAND * (XHI(J) - XLO(J))
    END DO
!
    RETURN
    END SUBROUTINE INIPOP
```

```
!---File: geom.f90 ---------------------------------------------------+
    SUBROUTINE GEOM(XCO, YCO, PROP, DES_AREA, AMPF, NODTOT, N, NUMP, D, IERR)
!--------------------------------------------------------------------+
!
    USE interfaces
!
!   Calling arguments
    INTEGER :: NODTOT, N, NUMP, D
    REAL :: XCO(NODTOT+1), YCO(NODTOT+1), PROP(NUMP,D)
    REAL :: DES_AREA, AMPF
    INTEGER :: IERR
!
!   Local variables
    REAL :: XTEMP(NODTOT+1), YTEMP(NODTOT+1)
    REAL :: RLE, KT, DZTE, BTE, XT, YT
    REAL :: XLT0, XLT1, XLT2, XLT3, YLT0, YLT1, YLT2, YLT3
    REAL :: XTT0, XTT1, XTT2, XTT3, YTT0, YTT1, YTT2, YTT3
    REAL :: B9
    REAL :: U, UINT, AREA
    INTEGER :: I, K
!
    IERR = 0
    XCO = 0.0
    YCO = 0.0
    XTEMP = 0.0
    YTEMP = 0.0
!
    RLE  = PROP(N,1)
    KT   = PROP(N,2)
    DZTE = PROP(N,3)
    BTE  = PROP(N,4)
    XT   = PROP(N,5)
    YT   = PROP(N,6)
!
    CALL ROOTS(B9, RLE, KT, DZTE, BTE, XT, YT)
!
    IF(B9 .EQ. 9999.0) THEN
        IERR = 1
        RETURN
    END IF
!
    XLT0 = 0.0
    XLT1 = 0.0
    XLT2 = B9
    XLT3 = XT
    YLT0 = 0.0
    YLT1 = (3.0/2.0) * KT * (XT - B9)**2 + YT
    YLT2 = YT
    YLT3 = YT
    XTT0 = XT
    XTT1 = 2 * XT - B9
    XTT2 = 1.0 + (DZTE - ((3.0/2.0) * KT * (XT - B9)**2 + YT))        &
    &      / TAN(BTE)
    XTT3 = 1.0
    YTT0 = YT
    YTT1 = YT
    YTT2 = (3.0/2.0) * KT * (XT - B9)**2 + YT
    YTT3 = DZTE
!
    UINT = 1.0 / FLOAT(NODTOT/2-1)
!
    IF(ABS(XLT2-XLT3) .LT. 0.05) THEN
        IERR = 1
        RETURN
    END IF
```

```
    IF(ABS(XLT0-XTT1) .LT. 0.05) THEN
        IERR = 1
        RETURN
    END IF
    IF(ABS(XLT1-XTT2) .LT. 0.05) THEN
        IERR = 1
        RETURN
    END IF
!
    I = 1
    U = 0.0
    DO WHILE (U .LT. 1.0)
        XTEMP(I) = XLT0*(1.0-U)**3 + 3*XLT1*U*(1.0-U)**2 +            &
        &      3*XLT2*U**2*(1.0-U) + XLT3*U**3
        YTEMP(I) = YLT0*(1.0-U)**3 + 3*YLT1*U*(1.0-U)**2 +            &
        &      3*YLT2*U**2*(1.0-U) + YLT3*U**3
        U = U + UINT / XT
        I = I + 1
    END DO
!
    U = 0.0
    DO WHILE (U .LT. 1.0)
        XTEMP(I) = XTT0*(1.0-U)**3 + 3*XTT1*U*(1.0-U)**2 +            &
        &      3*XTT2*U**2*(1.0-U) + XTT3*U**3
        YTEMP(I) = YTT0*(1.0-U)**3 + 3*YTT1*U*(1.0-U)**2 +            &
        &      3*YTT2*U**2*(1.0-U) + YTT3*U**3
        U = U + UINT / (1.0 - XT)
        I = I + 1
    END DO
    XCO(1) = XTT3
    YCO(1) = YTT3
!
!   Reorder coordinates
    K = NODTOT/2
    DO I = 2, NODTOT/2 + 1
        XCO(I) = XTEMP(K)
        YCO(I) = -YTEMP(K)
        K = K - 1
    END DO
!
!   Check to insure that the coordinates increase monotonically
    DO I = 1, NODTOT/2
        IF( XCO(I) .LE. XCO(I+1)  ) THEN
            IERR = 1
            RETURN
        END IF
    END DO
!
!   Mirror the coordinates to complete the top half
    K = NODTOT/2
    DO I = NODTOT/2 + 2, NODTOT+1
        XCO(I) = XCO(K)
        YCO(I) = -YCO(K)
        K = K - 1
    END DO
!
!   Scale the geometry (y-only) to obtain the desired area
    AREA = X_AREA(XCO, YCO, NODTOT)
    AMPF = DES_AREA / AREA
    DO I = 1, NODTOT+1
        YCO(I) = AMPF * YCO(I)
    END DO
!
    RETURN
    END SUBROUTINE GEOM
```

```
!---File: roots.f90 ------------------------------------------------+
    SUBROUTINE ROOTS(B9, RLE, KT, DZTE, BTE, XT, YT)
!-------------------------------------------------------------------+
!
!   Calling arguments
    REAL :: B9
    REAL :: RLE, KT, DZTE, BTE, XT, YT
!
!   Local variables
    REAL :: A, B, C, D, DEN
    REAL :: DEL0, DEL1
    COMPLEX :: U, V, W, Q
    COMPLEX :: R1, R2, R3, R4
    REAL :: RR1, RR2, RR3, RR4
    REAL :: UB
    REAL :: TOL
!
    DEN = (27.0/4.0) * KT**2
    A = (-27 * KT**2 * XT) / DEN
    B = (9.0 * KT * YT + (81.0/2.0) * KT**2 * XT**2) /               &
    &   DEN
    C = (2.0 * RLE - 18.0 * KT * XT * YT - 27.0 * KT**2 * XT**3) /    &
    &   DEN
    D = (3.0 * YT**2 + 9.0*KT*XT**2*YT + (27.0/4.0) * KT**2 * XT**4) /&
    &   DEN
!
    DEL0 = B**2 - 3.0*A*C + 12.0*D
    DEL1 = 2.0*B**3 - 9.0*A*B*C + 27.0*A**2*D + 27*C**2 - 72.0*B*D
!
    Q = ((DEL1 + SQRT(CMPLX(DEL1**2 - 4.0*DEL0**3))) / 2.0)**(1.0/3.0)
    U = (3*A**2 - 8*B) / 12.0
    V = (Q**2 + DEL0) / (3.0*Q)
    W = (A**3 - 4.0*A*B + 8.0*C) / (4 * SQRT(CMPLX(U + V)))
!
    R1 = -(A/4.0) + SQRT(CMPLX(U+V))/2.0 + SQRT(CMPLX(2.0*U - V - W)) / 2.0
    R2 = -(A/4.0) + SQRT(CMPLX(U+V))/2.0 - SQRT(CMPLX(2.0*U - V - W)) / 2.0
    R3 = -(A/4.0) - SQRT(CMPLX(U+V))/2.0 + SQRT(CMPLX(2.0*U - V + W)) / 2.0
    R4 = -(A/4.0) - SQRT(CMPLX(U+V))/2.0 - SQRT(CMPLX(2.0*U - V + W)) / 2.0
!
!write(6,*) R1
!write(6,*) R2
!write(6,*) R3
!write(6,*) R4
!
    IF(ABS(AIMAG(R1)) .LT. 0.0001) THEN
        RR1 = REAL(R1)
    ELSE
        RR1 = 9999.0
    END IF
!
    IF(ABS(AIMAG(R2)) .LT. 0.0001) THEN
        RR2 = REAL(R2)
    ELSE
        RR2 = 9999.0
    END IF
!
    IF(ABS(AIMAG(R3)) .LT. 0.0001) THEN
        RR3 = REAL(R3)
    ELSE
        RR3 = 9999.0
    END IF
!
    IF(ABS(AIMAG(R4)) .LT. 0.0001) THEN
        RR4 = REAL(R4)
    ELSE
```

```
        RR4 = 9999.0
    END IF
!
    UB = MAX(0.0,XT - SQRT((-2.0*YT)/(3.0*KT)))
    IF(RR1 .LT. UB .OR. RR1 .GT. XT) RR1 = 9999.0
    IF(RR2 .LT. UB .OR. RR2 .GT. XT) RR2 = 9999.0
    IF(RR3 .LT. UB .OR. RR3 .GT. XT) RR3 = 9999.0
    IF(RR4 .LT. UB .OR. RR4 .GT. XT) RR4 = 9999.0
!
    B9 = MIN(RR1, RR2, RR3, RR4)
!
    RETURN
    END SUBROUTINE ROOTS
```

```
!---File: x_area.f90 -------------------------------------------------+
    FUNCTION X_AREA(XCO, YCO, NODTOT)
!--------------------------------------------------------------------+
!
!   Calling arguments
    INTEGER :: NODTOT
    REAL :: AREA
    REAL :: XCO(NODTOT+1), YCO(NODTOT+1)
!
!   Local variables
    INTEGER :: I, J
!
    X_AREA = 0.0
    J = NODTOT
!
    DO I = 1, NODTOT
        X_AREA = X_AREA + (XCO(J) + XCO(I)) * (YCO(J) - YCO(I))
        J = I
    END DO
    X_AREA = X_AREA / 2.0
!
    RETURN
    END FUNCTION X_AREA
```

```
!---File: resid.f90 -------------------------------------------------+
    SUBROUTINE RESID(RESIDUAL, COST, GENERATION, NUMP, GMAX)
!-------------------------------------------------------------------+
!
!---Calling arguments
    INTEGER :: NUMP, GMAX
    REAL :: COST(NUMP,GMAX)
    REAL :: RESIDUAL
    INTEGER :: GENERATION
!
!---Local variables
    INTEGER :: I, J
    REAL :: MAXCOST, MINCOST
!
    MAXCOST = COST(1,GENERATION)
    MINCOST = COST(1,GENERATION)
!
    DO I = 1, NUMP
        IF( COST(I,GENERATION) .GT. MAXCOST ) THEN
            MAXCOST = COST(I,GENERATION)
        ELSE IF( COST(I,GENERATION) .LT. MINCOST ) THEN
            MINCOST = COST(I,GENERATION)
        END IF
    END DO
!
    RESIDUAL = (MAXCOST - MINCOST) / 1.0
!
    RETURN
    END SUBROUTINE RESID
```

```
!---File: postpro.f90 -------------------------------------------------+
    SUBROUTINE POSTPRO(REYN_NUM, MACH_NUM, ALPHA, CONVERGE,           &
    &                  NUMP, D, F, CR, K, GENERATION, ICOUNT,         &
    &                  OPTCOST, AFOIL, AMPF, DES_AREA)
!---------------------------------------------------------------------+
!
!---Calling arguments
    REAL :: REYN_NUM, MACH_NUM, ALPHA, CONVERGE
    INTEGER :: NUMP, D, GENERATION, ICOUNT
    REAL :: F, CR, K, OPTCOST, DES_AREA
    REAL :: AFOIL(1,D), AMPF
!
!---Local variables
    REAL :: RLE, KT, DZTE, BTE, XT, YT
!
    RLE  = AFOIL(1,1)
    KT   = AFOIL(1,2)
    DZTE = AFOIL(1,3)
    BTE  = AFOIL(1,4)
    XT   = AFOIL(1,5)
    YT   = AFOIL(1,6)
!
    WRITE(25,100)
    WRITE(25,110) REYN_NUM, MACH_NUM, ALPHA, DES_AREA, CONVERGE
    WRITE(25,120)
    WRITE(25,130) NUMP, F, CR, K
    WRITE(25,140) GENERATION, ICOUNT
    WRITE(25,150) OPTCOST
    WRITE(25,160)
    WRITE(25,170) RLE, KT, DZTE, BTE, XT, YT, AMPF
    WRITE(25,180)


100 FORMAT(5X,70('-'),                                                &
    &     /29X, 'OPTIMUM AIRFOIL SUMMARY',                            &
    &     /5X,70('-'),/)
110 FORMAT(7X, 'INPUT REYNOLDS NUMBER     :',6X,F11.3,/               &
    &      7X, 'INPUT MACH NUMBER         :',6X,F11.3,/               &
    &      7X, 'INPUT ANGLE OF ATTACK     :',6X,F11.3,/               &
    &      7X, 'X-AREA OF AIRFOIL         :',6X,F11.3,//              &
    &      7X, 'CONVERGENCE CRITERIA      :',6X,E11.3,///)
120 FORMAT(5X, 'DE OPTIMIZATION PARAMETERS',/)
130 FORMAT(7X, 'POPULATION SIZE           :',6X,I11,/                 &
    &      7X, 'SCALING FACTOR            :',6X,F11.2,/               &
    &      7X, 'CROSSOVER PROBABILITY     :',6X,F11.2,/               &
    &      7X, 'COEFFICIENT OF COMBINATION:',6X,F11.2,//)
140 FORMAT(7X, 'GENERATIONS TO SOLUTION   :',6X,I11,/                 &
    &      7X, 'COST EVALUATIONS TO SOL.  :',6X,I11,//)
150 FORMAT(7X, 'COST OF OPTIMUM DESIGN    :',6X,F11.7,///)
160 FORMAT(5X, 'DNA OF OPTIMUM DESIGN',/)
170 FORMAT(7X, 'LEADING EDGE RADIUS       :',6X,F11.7,/               &
    &      7X, 'THICKNESS CREST CURVATURE :',6X,F11.7,/               &
    &      7X, 'TRAILING EDGE THICKNESS   :',6X,F11.7,/               &
    &      7X, 'WEDGE ANGLE [RADS]        :',6X,F11.7,/               &
    &      7X, 'X-LOCATION OF THICK CREST :',6X,F11.7,/               &
    &      7X, 'Y-LOCATION OF THICK CREST :',6X,F11.7,/               &
    &      7X, 'Y AMPLIFICATION FACTOR    :',6X,F11.7,//)
180 FORMAT(5X,70('-'))
!
    RETURN
    END SUBROUTINE POSTPRO
```

Appendix D – Fortran90 computer code (Ixx constrained)

```
!---File: main.f90 ---------------------------------------------------+
!
    PROGRAM main
!
    USE interfaces
!
!---Variable Declarations
!---User Inputs
    REAL :: REYN_NUM(10)                !Inputted below
    REAL :: MACH_NUM = 0.1
    REAL :: ALPHA = 0.0
    INTEGER, PARAMETER :: D = 6         !Number of parameters (fixed)
    INTEGER, PARAMETER :: NP = 100      !Number of population members
    INTEGER, PARAMETER :: GMAX = 500    !Max num of generations
    INTEGER, PARAMETER :: DE_SET = 2    !DE Operator scheme
    REAL :: F = 0.3                     !Scale factor
    REAL :: CR = 0.8                    !Cross-over control constant
    REAL :: KC = 1.0                    !Coefficient of combination
    REAL :: DES_IXX = 0.00006           !Set second moment of inertia
    REAL :: CONVERGE = 0.000001         !Convergence criteria
!
!---DE arrays
    REAL :: XP(NP,D), VP(NP,D), UP(NP,D), XLO(D), XHI(D)
    REAL :: COST(NP,GMAX)
    REAL :: AFOIL(1,D), U_AMPF(NP), X_AMPF(NP), O_AMPF
    INTEGER :: G, ICOUNT=0, IERR
!
!---Output files
    CHARACTER*32 :: FILENAME1, FILENAME2, FILENAME3
    CHARACTER*32 :: FILENAME4, FILENAME5
!=====================================================================+
!Panel and Inverse Boundary-Layer Method Variables
    INTEGER, PARAMETER :: NODTOT=160, NANGLE=1
    REAL :: ALPHAS(NANGLE), X(NODTOT+1), Y(NODTOT+1)
    INTEGER :: NBT
    REAL :: CL_RES(NANGLE), CD_RES(NANGLE), CM_RES(NANGLE)
!
    NBT = NODTOT+1
    LXY = NODTOT+1
    LAA = 1
    ALPHAS(1) = ALPHA
!=====================================================================+
!---Range of Reynolds Numbers
    REYN_NUM(1) = 15000000.
    REYN_NUM(2) = 10000000.
    REYN_NUM(3) = 5000000.
    REYN_NUM(4) = 3000000.
    REYN_NUM(5) = 2000000.
    REYN_NUM(6) = 1500000.
    REYN_NUM(7) = 1000000.
    REYN_NUM(8) = 700000.
    REYN_NUM(9) = 500000.
    REYN_NUM(10)= 300000.
!
    DO K = 1, 10
!
!---Set the parameter constraints
        CALL PARAMS(XLO, XHI, D)
        ICOUNT = 0
        I = 1
        DO WHILE(I .LE. NP)
!
!           Generate initial population member
            CALL INIPOP(XP, XLO, XHI, NP, D, I)
!
```

```
!           Generate the points for the panel method
            CALL GEOM(X, Y, XP, DES_IXX, X_AMPF(I), NODTOT, I, NP, D, IERR)
!
!           If the member is valid then evaluate it
            IF(IERR .EQ. 0) THEN
!               Evaluate the cost of each of the initial members
                ICOUNT = ICOUNT + 1
                CALL PANEL_IBL(NODTOT, 1, X, Y, LXY,                    &
                &              ALPHAS, LAA, MACH_NUM, REYN_NUM(K),      &
                &              CL_RES, CD_RES, CM_RES, 1)
!
                COST(I,1) = CD_RES(1)
                IF( COST(I,1) .LT. 0.0001 ) COST(I,1) = 0.1
                I = I + 1
!           If the member is invalid then recreate it (i is not indexed)
            ELSE
                I = I
            END IF
        END DO
!
        DO I = 1, NP
            OPTCOST = 10.0  !Keep track of best population member
!           Check if current population member is the best
            IF(COST(I,1) .LT. OPTCOST) THEN
!               Store the best set of genes
                DO J = 1, D
                    AFOIL(1,J) = XP(I,J)
                END DO
                OPTCOST = COST(I,1)
            END IF
        END DO
        G = 1
        RESIDUAL = 1.0
!
        WRITE(FILENAME1,310) K
310     FORMAT('converge_',I0,'.out')
        OPEN(21,FILE=FILENAME1)
        OPEN(26,FILE="STATUS.TXT")
!
        DO WHILE(G .LT. GMAX .AND. CONVERGE .LT. RESIDUAL)
!
            G = G + 1
            WRITE(26,*) G
!
            DO I = 1, NP
                R1 = I
                R2 = I
                R3 = I
                R4 = I
                R5 = I
                DO WHILE(R1 .EQ. I)
                    CALL RANDOM_NUMBER(RAND)
                    R1 = INT(RAND * NP) + 1
                END DO
                DO WHILE(R2 .EQ. R1 .OR. R2 .EQ. I)
                    CALL RANDOM_NUMBER(RAND)
                    R2 = INT(RAND * NP) + 1
                END DO
                DO WHILE(R3 .EQ. R2 .OR. R3 .EQ. R1 .OR.                &
                &        R3 .EQ. I)
                    CALL RANDOM_NUMBER(RAND)
                    R3 = INT(RAND * NP) + 1
                END DO
                DO WHILE(R4 .EQ. R3 .OR. R4 .EQ. R2 .OR.                &
                &        R4 .EQ. R1 .OR. R4 .EQ. I)
```

```
                    CALL RANDOM_NUMBER(RAND)
                    R4 = INT(RAND * NP) + 1
                END DO
                DO WHILE(R5 .EQ. R4 .OR. R5 .EQ. R3 .OR.                &
                &        R5 .EQ. R2 .OR. R5 .EQ. R1 .OR.                &
                &        R5 .EQ. I)
                    CALL RANDOM_NUMBER(RAND)
                    R5 = INT(RAND * NP) + 1
                END DO
!
!               Mutation and Crossover
                DO J = 1, D
                    IF(DE_SET .EQ. 1) THEN
!                   "DE/rand/1"
                    VP(I,J) = XP(R1,J) + F*(XP(R2,J) - XP(R3,J))
                    ELSE IF(DE_SET .EQ. 2) THEN
!                   "DE/rand-to-best/2"
                    VP(I,J) = XP(R1,J) + F*(AFOIL(1,J) - XP(R1,J)) +    &
                    &         F*(XP(R2,J) - XP(R3,J)) +                 &
                    &         F*(XP(R4,J) - XP(R5,J))
                    END IF
!
                    CALL RANDOM_NUMBER(RAND1)
                    IF(RAND1 .LE. CR) THEN
                        UP(I,J) = VP(I,J)
                    ELSE
                        UP(I,J) = XP(I,J)
                    END IF
                END DO
!
!               Check to ensure new values are within constraints
                DO J = 1, D
                    IF(UP(I,J) .LT. XLO(J)) THEN
                        UP(I,J) = (XP(I,J) + XLO(J)) / 2.0
                    ELSE IF(UP(I,J) .GT. XHI(J)) THEN
                        UP(I,J) = (XP(I,J) + XHI(J)) / 2.0
                    END IF
                END DO
            END DO
!
            DO I = 1, NP
!               Generate the points for the panel method
                CALL GEOM(X, Y, UP, DES_IXX, U_AMPF(I), NODTOT,         &
                &         I, NP, D, IERR)
!
                IF(IERR .EQ. 0) THEN
!                   Evaluate the cost of each of the initial members
                    ICOUNT = ICOUNT + 1
!
                    CALL PANEL_IBL(NODTOT, 1, X, Y, LXY,                &
                    &              ALPHAS, LAA, MACH_NUM, REYN_NUM(K),  &
                    &              CL_RES, CD_RES, CM_RES, 1)
!
                    SCORE = CD_RES(1)
                    IF( SCORE .LT. 0.0001 ) SCORE = 0.1
                ELSE
                    SCORE = 10.0
                END IF
!
                IF(SCORE .LT. COST(I,G-1)) THEN
                    DO J = 1, D
                        XP(I,J) = UP(I,J)
                        X_AMPF(I) = U_AMPF(I)
                    END DO
                    COST(I,G) = SCORE
```

```
                ELSE
                    COST(I,G) = COST(I,G-1)
                END IF
!
!               Check if current population member is the best
                IF(COST(I,G) .LT. OPTCOST) THEN
!                   Store the best set of genes
                    DO J = 1, D
                        AFOIL(1,J) = XP(I,J)
                    END DO
                    O_AMPF = X_AMPF(I)
                    OPTCOST = COST(I,G)
                END IF
            END DO
!
            CALL RESID(RESIDUAL, COST, G, NP, GMAX)
            WRITE(21,*) RESIDUAL
!
        END DO
!
        CLOSE(21)
        CLOSE(26)
!
!       Write the cost array to file
        WRITE(FILENAME2,320) K
320     FORMAT('cost_',I0,'.out')
        OPEN(22,FILE=FILENAME2, STATUS="REPLACE", RECL=9*NP)
        WRITE(22,100) COST
100     FORMAT(100(2X, F7.6))
        CLOSE(22)
!
!       Write the DNA of the last generation to file
        WRITE(FILENAME3,330) K
330     FORMAT('DNA_',I0,'.out')
        OPEN(23,FILE=FILENAME3, STATUS="REPLACE")
        DO I = 1, NP
            WRITE(23,200) XP(I,1), XP(I,2), XP(I,3), XP(I,4),          &
            &             XP(I,5), XP(I,6), X_AMPF(I)
200         FORMAT(7(2X, F11.8))
        END DO
        CLOSE(23)
!
!   Generate the points of the best panel and write to file
!
        CALL GEOM(X, Y, AFOIL, DES_IXX, O_AMPF, NODTOT, 1, 1, D, IERR)
        WRITE(FILENAME4,340) K
340     FORMAT('afoil_',I0,'.out')
        OPEN(24,FILE=FILENAME4, STATUS="REPLACE")
        DO I = 1, NODTOT+1
            WRITE(24,*) X(I), Y(I)
        END DO
        CLOSE(24)
!
!       Write the summary file
        WRITE(FILENAME5,350) K
350     FORMAT('summary_',I0,'.out')
        OPEN(25,FILE=FILENAME5, STATUS="REPLACE")
        CALL POSTPRO(REYN_NUM(K), MACH_NUM, ALPHA, CONVERGE,           &
        &            NP, D, F, CR, KC, G, ICOUNT, OPTCOST,             &
        &            AFOIL, O_AMPF, DES_IXX)
        CLOSE(25)
    END DO
!
    STOP
    END PROGRAM main
```

```
!---File: interfaces.f90 ------------------------------------------+
!
MODULE interfaces

!====================================================================+
  INTERFACE
!--------------------------------------------------------------------+
    SUBROUTINE PANEL_IBL(numPanel, jjangle, xgeom, ygeom, Lxy,        &
    &                    attackangles, LAA, S_mach, Reynolds,         &
    &                    Cl_Res, Cd_Res, Cm_Res,iterative)
!--------------------------------------------------------------------+
    INTEGER :: LAA, Lxy, jjangle, numPanel
    REAL :: xgeom(Lxy), ygeom(Lxy), attackangles(LAA)
    REAL :: Cl_Res(jjangle), Cd_Res(jjangle), Cm_Res(jjangle)
    REAL :: S_mach, Reynolds
!--------------------------------------------------------------------+
    END SUBROUTINE PANEL_IBL
!--------------------------------------------------------------------+
  END INTERFACE

!====================================================================+
  INTERFACE
!--------------------------------------------------------------------+
    SUBROUTINE INIPOP(PROP, XLO, XHI, NUMP, D, ID)
!--------------------------------------------------------------------+
    INTEGER :: NUMP, D, ID
    REAL :: PROP(NUMP,D)
    REAL :: XLO(D), XHI(D)
!--------------------------------------------------------------------+
    END SUBROUTINE INIPOP
!--------------------------------------------------------------------+
  END INTERFACE

!====================================================================+
  INTERFACE
!--------------------------------------------------------------------+
    SUBROUTINE ROOTS(B9, RLE, KT, DZTE, BTE, XT, YT)
!--------------------------------------------------------------------+
    REAL :: B9
    REAL :: RLE, KT, DZTE, BTE, XT, YT
!--------------------------------------------------------------------+
    END SUBROUTINE ROOTS
!--------------------------------------------------------------------+
  END INTERFACE

!====================================================================+
  INTERFACE
!--------------------------------------------------------------------+
    SUBROUTINE GEOM(X, Y, PROP, DES_IXX, AMPF, NODTOT, N, NP, D, IERR)
!--------------------------------------------------------------------+
    INTEGER :: NODTOT, N, NP, D
    REAL :: X(NODTOT+1), Y(NODTOT+1), PROP(NP,D)
    REAL :: DES_IXX, AMPF
    INTEGER :: IERR
!--------------------------------------------------------------------+
    END SUBROUTINE GEOM
!--------------------------------------------------------------------+
  END INTERFACE

!====================================================================+
  INTERFACE
!--------------------------------------------------------------------+
    FUNCTION X_IXX(X, Y, NODTOT)
!--------------------------------------------------------------------+
    INTEGER :: NODTOT
```

```
    REAL :: IXX
    REAL :: X(NODTOT+1), Y(NODTOT+1)
!-----------------------------------------------------------------------+
    END FUNCTION X_IXX
!-----------------------------------------------------------------------+
  END INTERFACE

!=======================================================================+
  INTERFACE
!-----------------------------------------------------------------------+
    SUBROUTINE PARAMS(XLO, XHI, D)
!-----------------------------------------------------------------------+
    INTEGER :: D
    REAL :: XLO(D), XHI(D)
!-----------------------------------------------------------------------+
    END SUBROUTINE PARAMS
!-----------------------------------------------------------------------+
  END INTERFACE

!=======================================================================+
  INTERFACE
!-----------------------------------------------------------------------+
    SUBROUTINE RESID(RESIDUAL, COST, GENERATION, NUMP, GMAX)
!-----------------------------------------------------------------------+
    INTEGER :: NUMP, GMAX
    REAL :: COST(NUMP,GMAX)
    REAL :: RESIDUAL
    INTEGER :: GENERATION
!-----------------------------------------------------------------------+
    END SUBROUTINE RESID
!-----------------------------------------------------------------------+
  END INTERFACE

!=======================================================================+
  INTERFACE
!-----------------------------------------------------------------------+
    SUBROUTINE POSTPRO(REYN_NUM, MACH_NUM, ALPHA, CONVERGE,           &
    &                  NUMP, D, F, CR, K, GENERATION, ICOUNT, OPTCOST,  &
    &                  AFOIL, AMPF, DES_IXX)
!-----------------------------------------------------------------------+
    REAL :: REYN_NUM, MACH_NUM, ALPHA, CONVERGE
    INTEGER :: NUMP, D, GENERATION, ICOUNT
    REAL :: F, CR, K, OPTCOST, DES_IXX
    REAL :: AFOIL(1,D), AMPF
!-----------------------------------------------------------------------+
    END SUBROUTINE POSTPRO
!-----------------------------------------------------------------------+
  END INTERFACE

!=======================================================================+
!
END MODULE interfaces
```

```
!---File: geom.f90 ---------------------------------------------------+
    SUBROUTINE GEOM(X, Y, PROP, DES_IXX, AMPF, NODTOT, N, NP, D, IERR)
!--------------------------------------------------------------------+
    USE interfaces
!
!   Calling arguments
    INTEGER :: NODTOT, N, NP, D
    REAL :: X(NODTOT+1), Y(NODTOT+1), PROP(NP,D)
    REAL :: DES_IXX, AMPF
    INTEGER :: IERR
!
!   Local variables
    REAL :: XTEMP(NODTOT+1), YTEMP(NODTOT+1)
    REAL :: RLE, KT, DZTE, BTE, XT, YT
    REAL :: XLT0, XLT1, XLT2, XLT3, YLT0, YLT1, YLT2, YLT3
    REAL :: XTT0, XTT1, XTT2, XTT3, YTT0, YTT1, YTT2, YTT3
    REAL :: B9
    REAL :: U, UINT, IXX
    INTEGER :: I, K
!
    IERR = 0
    X = 0.0
    Y = 0.0
    XTEMP = 0.0
    YTEMP = 0.0
!
    RLE  = PROP(N,1)
    KT   = PROP(N,2)
    DZTE = PROP(N,3)
    BTE  = PROP(N,4)
    XT   = PROP(N,5)
    YT   = PROP(N,6)
!
    CALL ROOTS(B9, RLE, KT, DZTE, BTE, XT, YT)
!
    IF(B9 .EQ. 9999.0) THEN
        IERR = 1
        RETURN
    END IF
!
    XLT0 = 0.0
    XLT1 = 0.0
    XLT2 = B9
    XLT3 = XT
    YLT0 = 0.0
    YLT1 = (3.0/2.0) * KT * (XT - B9)**2 + YT
    YLT2 = YT
    YLT3 = YT
    XTT0 = XT
    XTT1 = 2 * XT - B9
    XTT2 = 1.0 + (DZTE - ((3.0/2.0) * KT * (XT - B9)**2 + YT))        &
    &      / TAN(BTE)
    XTT3 = 1.0
    YTT0 = YT
    YTT1 = YT
    YTT2 = (3.0/2.0) * KT * (XT - B9)**2 + YT
    YTT3 = DZTE
!
    UINT = 1.0 / FLOAT(NODTOT/2-1)
!
    IF(ABS(XLT2-XLT3) .LT. 0.05) THEN
        IERR = 1
        RETURN
    END IF
    IF(ABS(XLT0-XTT1) .LT. 0.05) THEN
```

```
        IERR = 1
        RETURN
    END IF
    IF(ABS(XLT1-XTT2) .LT. 0.05) THEN
        IERR = 1
        RETURN
    END IF
!
    I = 1
    U = 0.0
    DO WHILE (U .LT. 1.0)
        XTEMP(I) = XLT0*(1.0-U)**3 + 3*XLT1*U*(1.0-U)**2 +            &
        &      3*XLT2*U**2*(1.0-U) + XLT3*U**3
        YTEMP(I) = YLT0*(1.0-U)**3 + 3*YLT1*U*(1.0-U)**2 +            &
        &      3*YLT2*U**2*(1.0-U) + YLT3*U**3
        U = U + UINT / XT
        I = I + 1
    END DO
!
    U = 0.0
    DO WHILE (U .LT. 1.0)
        XTEMP(I) = XTT0*(1.0-U)**3 + 3*XTT1*U*(1.0-U)**2 +            &
        &      3*XTT2*U**2*(1.0-U) + XTT3*U**3
        YTEMP(I) = YTT0*(1.0-U)**3 + 3*YTT1*U*(1.0-U)**2 +            &
        &      3*YTT2*U**2*(1.0-U) + YTT3*U**3
        U = U + UINT / (1.0 - XT)
        I = I + 1
    END DO
    X(1) = XTT3
    Y(1) = YTT3
!
!   Reorder coordinates
    K = NODTOT/2
    DO I = 2, NODTOT/2 + 1
        X(I) = XTEMP(K)
        Y(I) = -YTEMP(K)
        K = K - 1
    END DO
!
!   Check to insure that the coordinates increase monotonically
    DO I = 1, NODTOT/2
        IF( X(I) .LE. X(I+1)  ) THEN
            IERR = 1
            RETURN
        END IF
    END DO
!
!   Mirror the coordinates to complete the top half
    K = NODTOT/2
    DO I = NODTOT/2 + 2, NODTOT+1
        X(I) = X(K)
        Y(I) = -Y(K)
        K = K - 1
    END DO
!
!   Scale the geometry (y-only) to obtain the desired IXX
    IXX = X_IXX(X, Y, NODTOT)
    AMPF = (DES_IXX / IXX)**(1./3.)
    DO I = 1, NODTOT+1
        Y(I) = AMPF * Y(I)
    END DO
    IXX = X_IXX(X, Y, NODTOT)
!
    RETURN
    END SUBROUTINE GEOM
```

```
!---File: x_area.f90 -------------------------------------------------+
    FUNCTION X_IXX(X, Y, NODTOT)
!--------------------------------------------------------------------+
!
!   Calling arguments
    INTEGER :: NODTOT
    REAL :: AI
    REAL :: X(NODTOT+1), Y(NODTOT+1)
!
!   Local variables
    INTEGER :: I, J
!
    X_IXX = 0.0
    J = NODTOT
!
    DO I = 1, NODTOT
        AI = (X(I+1) - X(I)) * (Y(I+1) + Y(I))
        X_IXX = X_IXX + ( Y(I)**2 + Y(I+1)**2 ) * AI
    END DO
    X_IXX = X_IXX / 12.0
!
    RETURN
    END FUNCTION X_IXX
```

```
!---File: postpro.f90 -------------------------------------------------+
    SUBROUTINE POSTPRO(REYN_NUM, MACH_NUM, ALPHA, CONVERGE,           &
    &                  NUMP, D, F, CR, K, GENERATION, ICOUNT,         &
    &                  OPTCOST, AFOIL, AMPF, DES_IXX)
!---------------------------------------------------------------------+
!
!---Calling arguments
    REAL :: REYN_NUM, MACH_NUM, ALPHA, CONVERGE
    INTEGER :: NUMP, D, GENERATION, ICOUNT
    REAL :: F, CR, K, OPTCOST, DES_IXX
    REAL :: AFOIL(1,D), AMPF
!
!---Local variables
    REAL :: RLE, KT, DZTE, BTE, XT, YT
!
    RLE  = AFOIL(1,1)
    KT   = AFOIL(1,2)
    DZTE = AFOIL(1,3)
    BTE  = AFOIL(1,4)
    XT   = AFOIL(1,5)
    YT   = AFOIL(1,6)
!
    WRITE(25,100)
    WRITE(25,110) REYN_NUM, MACH_NUM, ALPHA, DES_IXX, CONVERGE
    WRITE(25,120)
    WRITE(25,130) NUMP, F, CR, K
    WRITE(25,140) GENERATION, ICOUNT
    WRITE(25,150) OPTCOST
    WRITE(25,160)
    WRITE(25,170) RLE, KT, DZTE, BTE, XT, YT, AMPF
    WRITE(25,180)


100 FORMAT(5X,70('-'),                                                &
    &     /29X, 'OPTIMUM AIRFOIL SUMMARY',                            &
    &     /5X,70('-'),/)
110 FORMAT(7X, 'INPUT REYNOLDS NUMBER     :',6X,F11.3,/               &
    &      7X, 'INPUT MACH NUMBER         :',6X,F11.3,/               &
    &      7X, 'INPUT ANGLE OF ATTACK     :',6X,F11.3,/               &
    &      7X, 'IXX OF AIRFOIL            :',6X,F11.3,//              &
    &      7X, 'CONVERGENCE CRITERIA      :',6X,E11.3,///)
120 FORMAT(5X, 'DE OPTIMIZATION PARAMETERS',/)
130 FORMAT(7X, 'POPULATION SIZE           :',6X,I11,/                 &
    &      7X, 'SCALING FACTOR            :',6X,F11.2,/               &
    &      7X, 'CROSSOVER PROBABILITY     :',6X,F11.2,/               &
    &      7X, 'COEFFICIENT OF COMBINATION:',6X,F11.2,//)
140 FORMAT(7X, 'GENERATIONS TO SOLUTION   :',6X,I11,/                 &
    &      7X, 'COST EVALUATIONS TO SOL.  :',6X,I11,//)
150 FORMAT(7X, 'COST OF OPTIMUM DESIGN    :',6X,F11.7,///)
160 FORMAT(5X, 'DNA OF OPTIMUM DESIGN',/)
170 FORMAT(7X, 'LEADING EDGE RADIUS       :',6X,F11.7,/               &
    &      7X, 'THICKNESS CREST CURVATURE :',6X,F11.7,/               &
    &      7X, 'TRAILING EDGE THICKNESS   :',6X,F11.7,/               &
    &      7X, 'WEDGE ANGLE [RADS]        :',6X,F11.7,/               &
    &      7X, 'X-LOCATION OF THICK CREST :',6X,F11.7,/               &
    &      7X, 'Y-LOCATION OF THICK CREST :',6X,F11.7,/               &
    &      7X, 'Y AMPLIFICATION FACTOR    :',6X,F11.7,//)
180 FORMAT(5X,70('-'))
!
    RETURN
    END SUBROUTINE POSTPRO
```