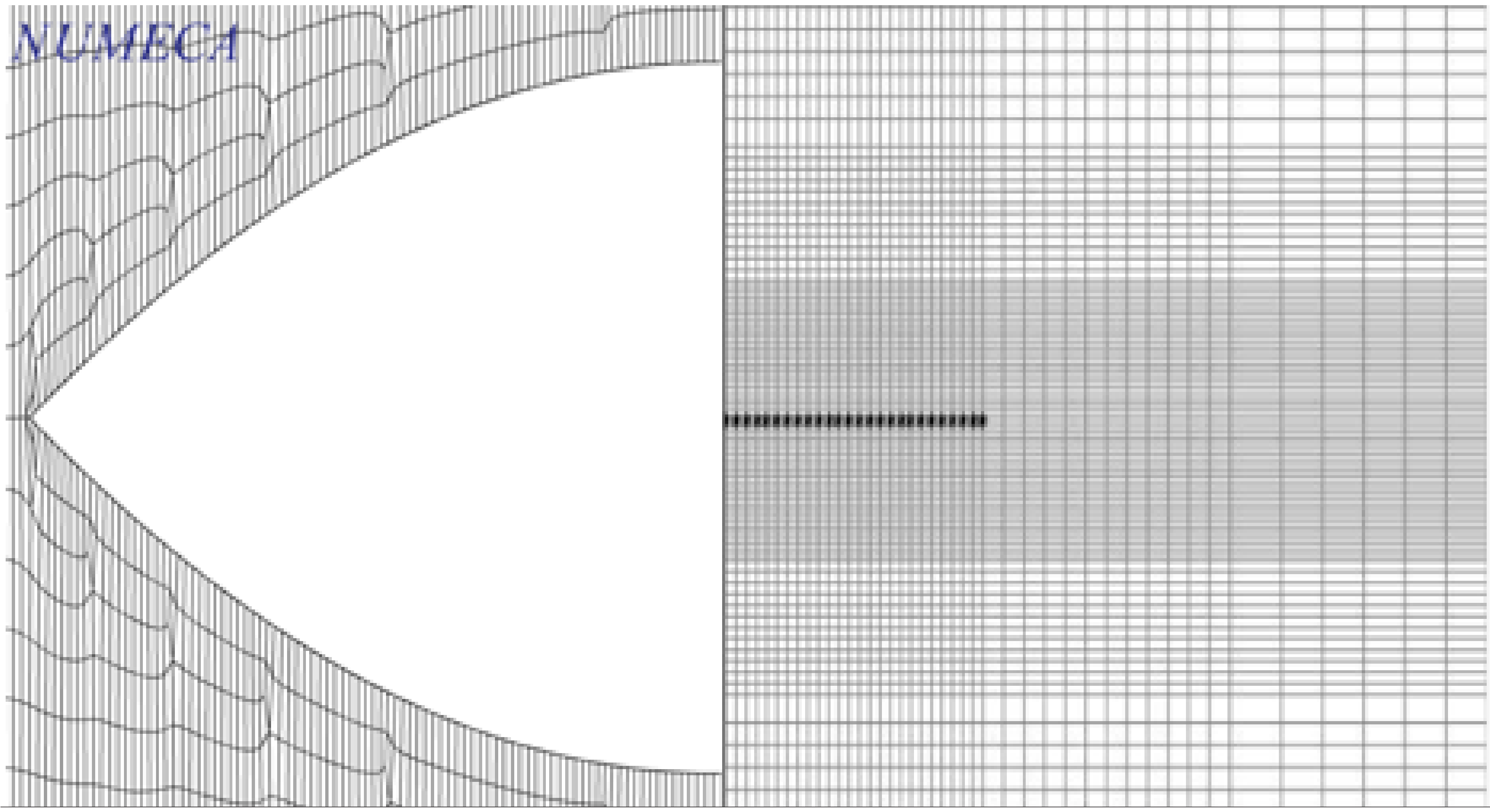
Solution to the Steady Transonic Small Disturbance Integral Equation

Emel Cankaya



Solution to the Steady Transonic Small Disturbance Integral Equation

by

Emel Cankaya

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology,

Supervisor:	Prof. dr. ir. W.L. Harris,	MIT
	Dr. ir. A.H. van Zuijlen,	TU Delft
Thesis committee:	Prof. Wesley L. Harris,	MIT
	Dr. Ir. Sander van Zuijlen,	TU Delft
	Prof. Dr. Stefan Hickel,	TU Delft
	Dr. Ir. Mark Voskuijl ,	TU Delft

Acknowledgements

It is thoroughly enjoyable to now thank the individuals who have contributed to my intellectual and personal growth during my academic years. First, a special thanks to Dr. A.H. van Zuijlen, who was my main thesis supervisor. Despite I executed my M.Sc. abroad, he took the time to guide me and answer all my questions. I appreciate his knowledge,patience, and advice. Much gratitude is also due to my second thesis supervisor, Professor Wesley L. Harris who allowed me to execute my M.Sc. thesis at the Massachusetts Institute of Technology (MIT). I value the academic advice he gave as well as his support on matters outside the academics., which made my abroad experience more enjoyable. Both advisors cared about me and showed it, which I very much appreciate.

This acknowledgement would be incomplete without mentioning my fellow students, their ability to make me laugh and joke made my time in the Delft University of Technology (TU Delft) more memorable. They supported me in tough and challenging times, and helped clear my mind by taking many coffee breaks with me. Besides that, they were always there when I had questions and needed advice. Thanks guys, for being such good friends.

A heartfelt thank you goes to Pauli Paap and Margaret Edwards, who had an enormous impact on my life. Thanks, Pauli, for always having faith in me and reminding me to keep chasing my dreams. And thanks, Margaret (Peggy), for making my day by giving compliments every single day. There are so many other people who have contributed to my success, both at academic level and in daily life. However, since I do not want to risk forgetting to mention someone, I hereby want to thank all of them.

Finally, I would like to express my extreme gratitude to my family who provided me with the best possible education. During stressful times I could be very intolerable, and keep complaining. Despite this, they have always supported me, and made me push trough. There is a lot more which I cannot describe in words, and it is in this spirit that I dedicate this thesis to them.

CONTENTS

Acronyms

BC	Boundary Condition
CFD	Computational Fluid Dynamics
GE	Governing Equation
IEM	Integral Equation Method
NS	Navier-Stokes equations
PDE	Partial Differential Equation
TSD	Transonic Small Disturbance Equation

List of Symbols

C_P	Pressure coefficient [-]
M_∞	Freestream Mach number
P_∞	Freestream static pressure [Pa]
P	Static pressure at the point of interest [Pa]
V_∞	Freestream velocity [$\frac{m}{s}$]
V	Velocity vector field [$\frac{m}{s}$]
Γ	Total circulation Γ [$\frac{m^2}{s}$]
α	Angle of attack of the airfoil [rad]
$\bar{\gamma}$	Strength of elemental vortex [$\frac{m^2}{s}$]
$\bar{t}$	Dimensional spatial coordinate t [s]
$\bar{x}$	Dimensional cartesian coordinate x [m]
$\bar{z}$	Dimensional cartesian coordinate z [m]
β	Transformation parameter [-]
ϵ_0	Initial/starting value of the thickness ratio [-]
ϵ	Thickness ratio of the airfoil, which is the ratio of maximum thickness of the airfoil to the chord at that point ($\frac{t}{b}$) [-]
γ	Specific heat ratio [-]
μ	Transformation parameter [-]
ν	Transformation parameter [-]
ϕ_0	Steady component perturbation velocity potential [-]
ϕ_1	Unsteady component perturbation velocity potential [-]
ϕ	Perturbation velocity potential [-]
ρ_∞	Freestream density [$\frac{kg}{m^3}$]
ξ	Non-dimensional x-location in the computational domain [-]
ζ	Non-dimensional z-location in the computational domain[-]
b	Airfoil semi-chord [m]
g_L	Lifting contribution to the steady velocity potential [-]
g_T	Thickness contribution to the steady velocity potential [-]
g_{NL}	Contribution of non-linear phenomena to the steady velocity potential [-]
g	Perturbation velocity potential transformed into ϵ-space [-]
$h_0^{'}$	Thickness distribution of the body (i.e. airfoil) [-]
$h_1^{'}$	Unsteady displacement of the wing mean surface about $z^{'} = 0$ [-]
$h^{'}$	Vertical coordinates of the body (i.e. airfoil) [-]
q_∞	Freestream dynamic pressure [Pa]
t	Non-dimensional spatial coordinate t [-]
u_0	Base solution perturbation velocity, i.e. initial (starting) value [$\frac{m}{s}$]
u	Perturbation velocity [$\frac{m}{s}$]
x	Non-dimensional coordinate/location along the horizontal axis of the domain/airfoil (running coordinate of the integrals) [-]
$y^{\pm}$	Vertical coordinates of the upper (+) and lower (-) side of the airfoil [-]
z	Non-dimensional coordinate/location along the vertical axis of the domain/airfoil (running coordinate of the integrals)[-]

1 Introduction

This research deals with airfoils in high-speed compressible flow. What happens when an airfoil attains high-subsonic or low-supersonic Mach numbers, close to the speed of sound? As Mach 1 is approached, the drag of an airfoil abruptly increases. This phenomenon, known as drag divergence, raises questions and concerns on aircraft design. Is it possible to deal with this drag divergence by determining its effects on the aerodynamic loading on the airfoil in a relatively simple, affordable and fast way, instead of using complex, expensive and time-consuming Computational Fluid Dynamics (CFD) tools? Wind tunnel testing as a design tool is also avoided, as the cost of the models and wind tunnel test time are too high [1]. The aforementioned question is a very important and practical question for preliminary design phases in particular, and will be addressed in this research. The need for such a model arises due to the fact that CFD models are computationally intensive, and the overall cost of the computational effort is not justified for the preliminary design phase. Also, for the industry, reducing the time (of for example aircraft design) is vitally important to have lower development costs and rapid product availability, which both contribute to the larger market share. Although CFD models result in very accurate predictions, it should be noted that these levels of accuracy are not necessary for preliminary design stage. Sufficient levels of accuracy can be reached with simpler models. The aforementioned factors of computer capacity, processing time, computation costs, and level of accuracy make CFD insuitable to solve transonic flows in the preliminary design phase of a project. A simpler, faster, and cheaper model would be more favorable. The resulting algorithm could be used for stand-alone analysis as well as for multidisciplinary design. An example of multidisciplinary design processes is for example analyzing the effect of drag divergence on flutter characteristics.

Both military and civilian operators have expressed the importance of aircraft flying in the transonic regime either in the maneuvering or cruise mode,respectively. This makes aerodynamic prediction in this range extremely important. Therefore, the transonic flight regime should be well understood. The ability to determine and predict the steady and unsteady aerodynamic loads acting on the aircraft in this flight regime is needed in order to design a vehicle that can operate safely at the desired flight conditions. Unfortunately, predicting the aerodynamic loads at transonic speeds is one of the major challenges the aircraft designer is facing, due to the fact that a typical transonic flow has a subsonic free stream from which the flow over the wing accelerates to supersonic speeds. One way the required deceleration from the supersonic speeds to the free stream velocity takes place is through shock waves, resulting in the presence of various types of flow regions in the flow field. This means that the algorithm that will be developed for this study to solve transonic flows should be able to predict the subsonic and supersonic flows simultaneously. Hence, mathematical difficulties arise, since subsonic flows are described by elliptic governing equations while supersonic flows are described by hyperbolic equations. Thus, in order for the algorithm to describe transonic flows a nonlinear governing equation of mixed elliptic/hyperbolic type must be solved. The difference in governing equations should be analyzed to understand the main difficulties in predicting aerodynamic loading in the transonic flight regime. The governing equations are given in the form of partial differential (PDE) equations, which can be divided in three categories based on the coefficients of the higher-order terms of the governing equation. These categories are: elliptic, parabolic, and hyperbolic. A more detailed description about the classification of partial differential equations can be found in Appendix A. The difficulty in solving the mixed elliptic/hyperbolic problems arise due to the fact that the behavior of subsonic and supersonic flow is different, leading to difficulties in finding a solution at the boundary of the hyperbolic and elliptic regions. Linear theory is sufficient in predicting the aerodynamic loads in the subsonic and supersonic flow regimes, whereas in the transonic flow regime even the most fundamental representation of the aerodynamics must be described by a nonlinear set of equations. Thus, the prediction method used in developing the algorithm need to be able to model all these types of flow and is derived from the Navier-Stokes (NS) equations. However, since the NS equations are difficult to solve analytically, certain simplifications are made. The first simplification is that the flow is always considered to be attached. If this is the case, the viscous terms can be neglected and the dominant nonlinearity will be caused by the shock waves. Other simplifications are possible if the geometry of the shock-wave geometry is considered to be simple across a wing, if the flow is considered to be irrotational, and if the disturbances are assumed to be small. Further simplifications are not possible without removing the dominant transonic phenomena of shock waves, resulting in the simplest physical problem that must be modeled to be that of two-dimensional, irrotational, attached flow with shock waves and small disturbances. These assumptions lead to the derivation of the Transonic Small Disturbance (TSD) equation from the NS equations.

This study is therefore concerned with the development of an accurate and efficient technique for solving the steady Transonic Small Disturbance (TSD) equation, in the form of PDEs. Since simplicity, speed, and cost are the key requirements in developing an effective algorithm, the main focus is on the integral equation method (IEM). This implies that the governing equations (GE) and boundary conditions (BC), which are expressed by PDEs, will be re-formulated into integral forms where the unknown function occurs under the integral sign [2]. The IEM has proven to be a powerful technique for solving a variety of practical problems. This study investigates whether applying this technique to the TSD will be suitable for predicting the aerodynamic loading on the airfoils in transonic flows. Further, the integral equation method offers a deeper insight into the problem that is posed by studying the nature of the PDEs they originate from, as the analytical steps involved are of physical significance. This includes in precisely identifying the different segments of the integrals, and the contour each integral applies to (such as infinity, the wake, airfoil body, shock-wave surface). A more detailed discussion regarding the steps taken and the physical significance of each step can be found in Appendix B. Moreover, for design applications repetitive and non-repetitive portions of the computations are readily separable, and the required sensitivities of aerodynamic parameter to variations in airfoil geometry can be readily calculated. Since the emphasis is on the speed of the algorithm, potential theory will be used. The numerical simulations of potential equations are computationally efficient, as the solution consists of only solving a simple scalar equation. A more complete composition of CFD, using the NS and Euler equations, consist of five coupled equations. Moreover, numerical iteration schemes for solving the potential equations characteristically converge in fewer iterations than the NS and Euler equations. Consequently, potential solvers are typically an order of magnitude (or more) faster than Euler equation solvers on comparable grids [3]. This difference in speed gets more recognizable as the problem posed gets more complex. Also, all potential formulations are isentropic and irrotational. The aforementioned assumptions are commonly consistent with subsonic, transonic, and supersonic flows, at or near the cruise conditions. In addition to this, the shock-waves are required to be weak, which holds when the upstream, normal-shock Mach number component is more than or equal to approximately 1.3. To simplify the problem even further, this special issue is devoted to a symmetric parabolic arc airfoil. If the algorithm produces reliable results, it can be expanded further to a wider range of airfoil configurations.

The objective of this study to develop an accurate and efficient technique for solving the steady TSD will be achieved by the following approach. First, a physical model will be defined to develop the analytical model for the treatment of two-dimensional airfoils in transonic flight regimes. This means that the characteristics of the flow and the shape of the airfoil will be defined as reported in Chapter 2. Subsequently, the equations of motion and the boundary conditions used in developing the analytical model will be outlined in Chapter 3, upon which the implementation of those set of equations into the technical programming language $MATLAB$ will be discussed in Chapter 4. The results obtained with this algorithm will be treated next in Chapter 5. Chapter 6 then will focus on the quality assessment of the algorithm, meaning that it will be checked whether the developed product meets the objective of this study. Finally, based on these results, conclusions will be drawn and some recommendations will be given in Chapter 7.

2 Physical Model

This chapter discusses the physical model defined to develop an analytical model for the treatment of steady, two-dimensional flow around an airfoil in high-subsonic and transonic speeds. The ultimate goal of the project is to determine the aerodynamic characteristics of the airfoil in the transonic flow regime. The flow over the airfoil and the resulting aerodynamic forces are modeled as defined in this chapter in subsequent sections.

2.1. Flow Characteristics

The characteristics of the flow over the airfoil will be treated in this section. It is important to have a well defined flow and keep track of the assumptions that are made. The underlying assumptions impose certain limitations on the final results, and should be cheered to avoid a situation in which the equation is not valid. The fluid type and flow characteristics (i.e., the assumptions that are made regarding the flow) can be seen below.

Fluid type	=	perfect gas, air
Flow characteristics	=	continuum
		steady
		inviscid
		no body forces
		irrotational
		compressible

Here, continuum is defined as a region of space filled with continuous matter with continuous properties [4], thereby defining continuous as various properties averaged on a length and time scale of interest varying smoothly within the region, except possibly for a small number of discontinuities. Steady flow is characterized as a flow of which the fluid properties at every point are time-independent, making the flow-field variables a function of spatial location only. Inviscid flow means that viscosity of the flow can be neglected. Physically, it means that viscous shear and normal stresses due to viscosity are negligible, resulting in the only stresses acting on the body surface to be the normal stresses due to the pressure. Hence, the only surface forces exerted on the fluid are the pressure forces. Due to this assumption the boundary layer on the surface of the body will be eliminated, resulting in the boundary layer to be very thin compared to the dimensions of the body. Subsequently, this absence of the boundary layer has a negligible effect pertaining to alternations of the body geometry as seen by the flow. This changes when the flow separates and the boundary layer leaves the body, which results in a big alternation of the effective geometry of the body. Also, it is assumed that no body forces (e.g., electromagnetic forces, gravity, or any other forces which "act at a distance" on the fluid) are experienced by the flow.

Irrotational flow is the flow type in which fluid particles do not rotate about their own axes and retain their original orientations. This implies that the fluid elements have no angular velocity, and their motion through space is a pure translation. This can be seen in Figure 2.1. Here, fluid elements moving along two different streamlines are shown in various modes of translation. The upper streamline shows a fluid element where the angular velocities θ of its sides are zero, while the lower streamline shows a fluid element where the angular velocities of two intersecting sides are finite but equal and opposite to each other. Thus, the sum of the angular velocities in the latter case is also equal to zero. In both cases, the angular velocity of the fluid elements is zero meaning that their motion through space is pure translation.

Before discussing compressibility, it is worth reviewing the Mach number regimes since it determines whether a flow has to be treated as compressible or incompressible. Using Mach numbers as the criterion, the following speed regimes can be identified when considering the whole flow field:

- **Subsonic flow**: Mach number is less than one everywhere in the flow. A rule of thumb for subsonic flows is that $M_\infty < 0.8$ for subsonic flow over slender bodies. This value must even be lower for subsonic flow over blunt bodies to ensure complete subsonic flow.
- **Supersonic flow**: Mach number is greater than one everywhere in the flow. The rule of thumb for supersonic flows is $M_\infty > 1.2$. Another feature is that usually shock waves arise, which causes the flow properties and streamlines to alter discontinuously.
- **Transonic flow**: A flow with mixed regions of $M < 1$ and $M > 1$. Here M represents the local Mach number. If M_∞ is subsonic but near unity, the flow can locally become supersonic ($M > 1$). A shock wave can cease this by reducing the flow behind the shock to a subsonic regime. Accordingly, this type of flow is characterized by mixed subsonic-supersonic flows. A rule of thumb for transonic flows is $0.8 < M_\infty < 1.2$ for flows over slender bodies.
- **Hypersonic flow**: This type of flow occurs when M_∞ becomes large enough such that viscous interaction and/or chemically reacting effects begin to dominate the flow. The rule of thumb for hypersonic flows is $M_\infty = 5$.

Since the current research covers high-speed flow (near Mach 1 and above), the flow has to be modeled as being compressible. This means that the density ρ can not be treated as a constant, and can vary over a wide range. Only the flow of gases at low Mach numbers ($M < 0.3$) could be treated as incompressible, meaning that the density is assumed constant in the flow field.

The research presented in this report treats the flow at freestream Mach numbers $M_\infty = 0.806$ and $M_\infty = 0.86$. The first Mach number was chosen because one of the verification sources [5] has computed the pressure distribution of a parabolic arc airfoil at this Mach number as well, since the entire flow happens to be subsonic at this Mach number. However, since the regime of interest of this research is the transonic flow regime, mixed regions of subsonic and supersonic regions have to be present. Shock waves start to arise on the surface of the parabolic arc airfoil at a Mach number of $M_\infty = 0.86$. This freestream Mach number, at which the local Mach number on the airfoil reaches $M = 1$ for the first time, is called the critical Mach number M_{cr}. This value has been found by trial and error with a CFD tool called *Numeca*. More about this tool will be explained in Chapter 6.

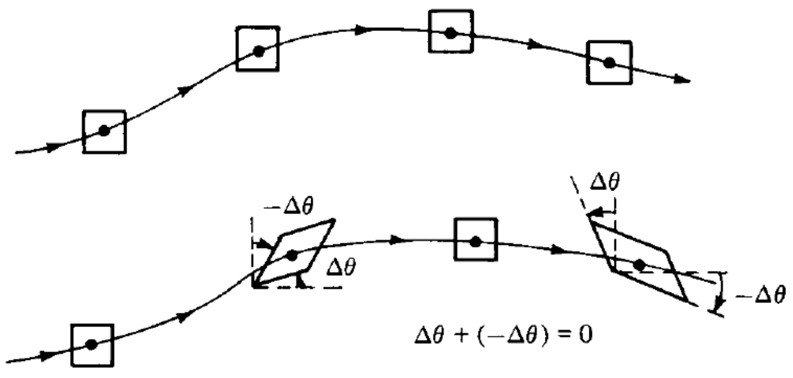


Figure 2.1: Fluid elements in an irrotational flow [1]

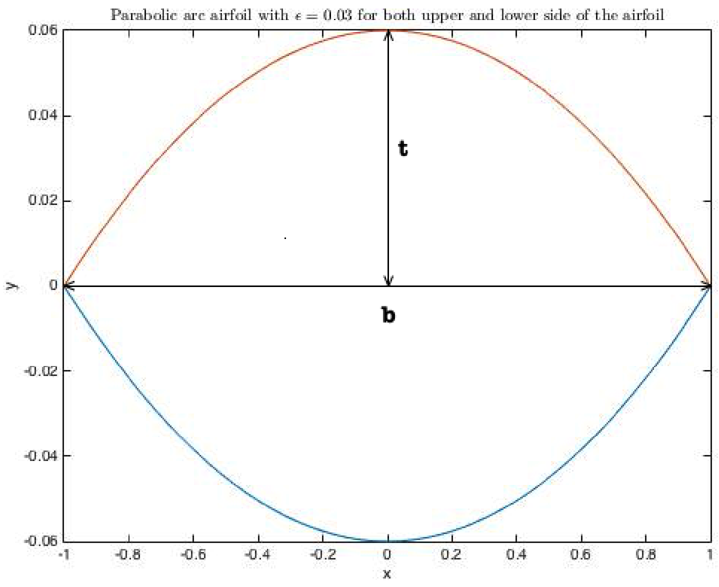


Figure 2.2: Parabolic arc airfoil for $\epsilon = 0.03$ and $\alpha = 0°$

2.2. AIRFOIL

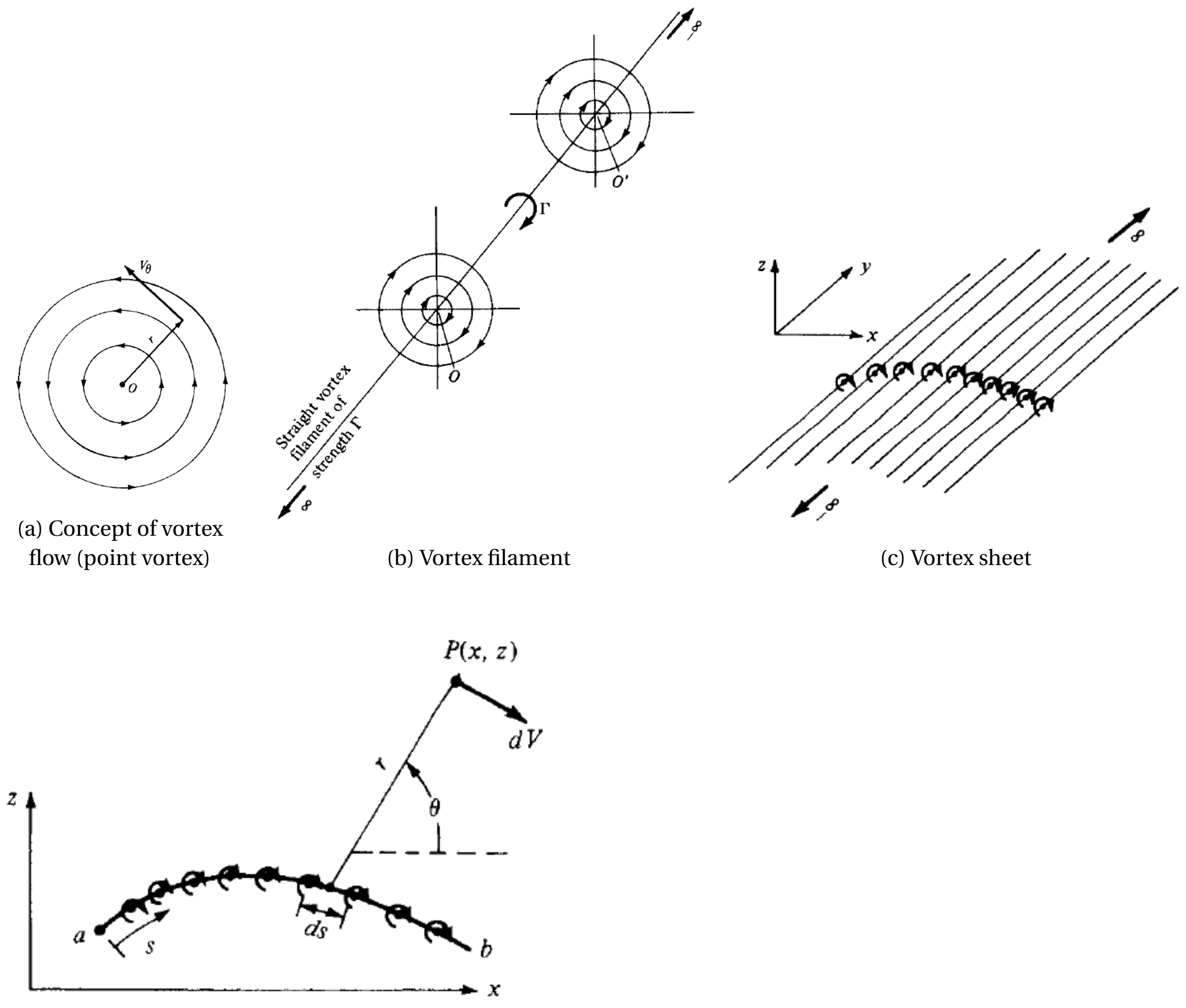
Now the characteristics of the flow have been defined, the airfoil on which the aerodynamic forces have been computed will be treated in this section. The coordinates of the upper and lower sides of the airfoil are determined using Equation 2.1.

$$y^{\pm} = \pm 2\epsilon(1 - x^2) - \alpha x \tag{2.1}$$

where ϵ represents the thickness ratio of the airfoil, which is the ratio of maximum thickness of the airfoil to the chord at that point ($\frac{t}{b}$). The angle of attack is denoted by α $[rad]$. A graphical representation of this airfoil for $\epsilon = 0.03$ for both the upper and lower side of the airfoil can be seen in Figure 2.2. This value of the thickness ratio is chosen for reasons which will be explained later in this report.

The computations will be done on a mesh created around the airfoil. Since placing the mesh points on the surface of a rounded object with equal distances will be too complex, the airfoil will be projected on the $x-$ axis, making the airfoil a flat plate. Also, in order to simplify the theoretical prediction of the characteristics of the airfoil, the thin airfoil theory is applied. In order to get familiarized with this theory, it is important to understand the concepts of vortex flow, vortex filament and vortex sheet. Vortex flow type of elementary flow is used to obtain lifting flow over several body shapes. A vortex flow is a type of flow where all the streamlines are concentric circles about a given point. The velocity along each streamline is constant but varies from one streamline to another inversely with distance from the center of the circle. The circulation Γ around this point vortex is equal to the strength of the vortex. A sketch of this point vortex can be seen in Figure 2.3a, where V_r and V_θ represent the radial and tangential velocity components respectively. Figure 2.3b displays a vortex filament, which is a straight line perpendicular to the page, going through point O, and extending to infinity both out of and into the page. By placing an infinite number of those straight vortex filaments side by side, a vortex sheet can be formed. A three dimensional view of this vortex sheet can be seen in Figure 2.3c, where the elemental vortex γds indicates the strength of an infinitesimal portion ds of the sheet and is variable along the sheet.

According to the principle of thin airfoil theory, a thin airfoil can be simulated by placing a vortex sheet on the surface of the airfoil as shown in Figure 2.3d. An important feature of thin airfoil theory is that for symmetric airfoils, this vortex sheet is placed along the chord of the airfoil, while for cambered airfoils the vortex sheet is placed along the camber line of the airfoil. Each section of this vortex sheet induces an infinitesimally small velocity in a direction perpendicular to r at point P. The total velocity induced at P is the summation of the induced velocities by the individual vortex sheet segments, which can be obtained by integrating the infinitesimally small induced velocities from the leading edge a to the trailing edge b. However, it should be noted that the individual small induced velocities are perpendicular to r, meaning that it changes direction



(a) Concept of vortex flow (point vortex)

(b) Vortex filament

(c) Vortex sheet

(d) Vortex sheet placed on airfoil surface

Figure 2.3: Application of vortex flow on airfoil [1]

at point P when summing up the individual induced velocities from a to b. Thus, the incremental velocities induced at P by different sections of the vortex sheet must be added vectorially. Therefore, it is more convenient to deal with the velocity potential. The objective now is to generate this velocity potential. Referring to Figure 2.3d the total velocity potential at P induced by the entire vortex sheet from a to b is:

$$\phi(x,z) = -\frac{1}{2\pi}\int_a^b \theta\bar{\gamma}ds \tag{2.2}$$

The objective is to find that particular $\gamma(S)$ such that the airfoil surface, chord, or camber line become a streamline of the flow and and such that the Kutta condition of finite velocities is satisfied at the trailing edge $(\gamma(s)_{TE} = 0)$. Once this particular variation of $\gamma(s)$ is found, the total circulation Γ around the vortex sheet along the airfoil can be found by the sum of the the strengths of the elemental vortices:

$$\Gamma = \int_a^b \bar{\gamma}ds \tag{2.3}$$

The determined value of circulation can than be used to calculate the resulting lift.

2.3. POTENTIAL FLOW

Potential flow studies the properties of forces, which follow the law of gravitation. A field of gravitational forces is called a potential field, and the corresponding function is the potential. This theory arose after it was realized that the fundamental forces of nature could be modeled using potentials that satisfy Laplace's equation ($\nabla^2\phi = 0$). Potential theory can also be applied to aeronautical problems, where the flow surrounding various shapes of bodies could be modeled in order to determine the flow properties. The potential flow theory states that flow properties (such as the flow velocity V) can be written as the gradient of a scalar function ϕ, which is called the velocity potential. In other words, it treats properties of vector fields (i.e. $V = (u, v, w)$) generated by gradients of a potential function ϕ (i.e. $V = \nabla\phi$). The gradient of a function is a vector field, where the vector at each point, points in the direction in which the function increases most rapidly (steepest ascent). The potential flow theory works by first introducing a potential function, known as velocity potential ϕ, from which the individual velocity components (i.e. u, v, w) can be determined. Subsequently, the equations of motion for irrotational flow reduce to a single partial differential equation for velocity potential. This equation of motion is known as the Laplace equation, and is used in solving the unsteady transonic small disturbance equation, as explained in Chapter 3. A base solution, as a starting value of the perturbation velocity will be constructed from Laplace equation satisfying the boundary conditions of the flow.
Potential flows by definition are irrotational. The physical meaning of irrotational flows was given in Section 2.1. This section will dive in the mathematical definition of it. In order to do so, a function f will be introduced. Integrating the gradient of a function f along a curve, results in the difference between f at the end of the curve and f at the beginning of the curve $\left(\int_c \nabla f \mathrm{d}c = f_{end} - f_{start}\right)$. However, integrating the curl of a function f along a surface, gives the integral of f along the boundary of the surface . This means looping from some point f on the boundary of the surface back to itself along the boundary $\left(\int_S \nabla \times f \mathrm{d}s = \int_c f \mathrm{d}c = f(x) - f(x)\right)$. Thus, integrating the curl of the gradient of a function f across any surface results in the integral of the gradient of f along the boundary of that surface, meaning again looping from some point f on the boundary of the surface back to itself along the boundary. It's the difference between f at that ending point and f at that same starting point, which is zero $\left(\int_S \nabla \times \nabla f \mathrm{d}s = \int_c \nabla f \mathrm{d}c = f(x) - f(x) = 0\right)$. The curl of the gradient of f always integrates to zero, leading to a feature of potential flow; potential flows are irrotational. The local rotation of a vector field (i.e. the gradient of a function, $\nabla\phi$) is proportional to its curl, so that potential flows do not rotate as they deform. This can be written in mathematical symbols as follows:

$$V = \nabla\phi \tag{2.4a}$$

$$\nabla \times \nabla\phi = 0 \tag{2.4b}$$

$$\nabla \times V = 0 \tag{2.4c}$$

It should be noted that potential flow is restricted to situations where vorticity is not present, and will not be able to forecast the flow properly when vorticity becomes important. The latter can occur for boundary layers and wakes. But since the boundary layer will be neglected due to the assumption of inviscid flow, the assumption of irrotationality is valid too. Therefore, potential flow can be applied to the current problem. The governing equations and boundary conditions will be derived in Chapter 3, based on this theory.

The reason for applying potential flow theory to the current problem is due to the way the Transonic Small Disturbance (TSD) equations are derived. The derivation starts from the Navier-Stokes (NS) equations, and simplifies to the TSD by assuming inviscid, irrotational flow with small perturbation velocities. The assumption of irrotational flow is where the non-physical scalar potential ϕ is being defined, since irrotationality comes with the curl and gradient of the velocity potential as mentioned above.

3 Mathematical Model

This chapter outlines the equations of motion (EOM) and the corresponding boundary conditions (BC) used in determining the aerodynamic loading on two-dimensional airfoils in transonic flight regimes. These equations are based on the potential flow theory, as explained in Chapter 2.3. As mentioned in Chapter 1, the numerical analyses should be computationally less intensive than the state-of-the art CFD tools and thus cheaper. Therefore, the integral forms of EOM and BC will be solved, rather than the PDEs.

First the equation used to obtain the governing equations will be discussed in Section 3.1. Then, the EOM and BC in differential form (arising out of the aforementioned equation type) will be outlined in Section 3.2, after which they will be transformed into integral form in Section 3.3 by applying Green's theorem.

3.1. Transonic Small Disturbance Equation

This research is based on implementing the unsteady TSD equation for high-subsonic and transonic speeds. Before deriving the governing equations and boundary conditions for steady transonic flow, solving the TSD equation will be justified.
Figure 3.1, along with Table 3.1, gives an overview of the commonly solved equations along with their corresponding assumptions. The full Navier-Stokes (NS) equations describe how the velocity, pressure, temperature, and density of a moving fluid are related [6]. The equations are a set of coupled, highly nonlinear partial differential equations (PDEs), consisting of one continuity equation (i.e. conservation of mass), three conservation of momentum equations, and one conservation of energy equation. Those equations are all time-dependent. The numerical solution of these equations is a challenge, as well as its analytical solution. Depending on the complexity of the problem, high computing power and time is required to solve the equations using different numerical techniques as finite difference - , finite volume - , and finite element methods. This branch of fluid mechanics is called Computational Fluid Dynamics (CFD). Solutions of the full NS equations show the onset of turbulence, the interaction of shear layers, and almost all of the interesting aerodynamic phenomena.
Since the NS equations are too difficult to solve both numerically and analytically, due to the highly nonlinear nature of the five coupled set of PDEs, and the purpose of this study is to develop a semi-analytical method, further assumptions and simplifications are made to derive an analytically solvable equation. These assumptions and simplifications about the fluid flow will simplify the governing equations and their numerical solution, which results in the reduction of computational costs. The hierarchy of the governing flow equations based on their assumptions about the fluid flow is shown in Figure 3.1.

This figure can be divided into two categories; the approaches displayed on the left side of the figure neglects the effect of viscosity, while the approaches on the right side include them into the analysis. However, since the current study focuses on an analytically solvable approach, the right side of this figure will not be elaborated further. Only a very brief introduction of the approaches on the right side will be given, whereafter the left side of the figure will be elaborated on.

Above a certain critical value of the Reynolds number (Re) the entire flow becomes turbulent, at which point

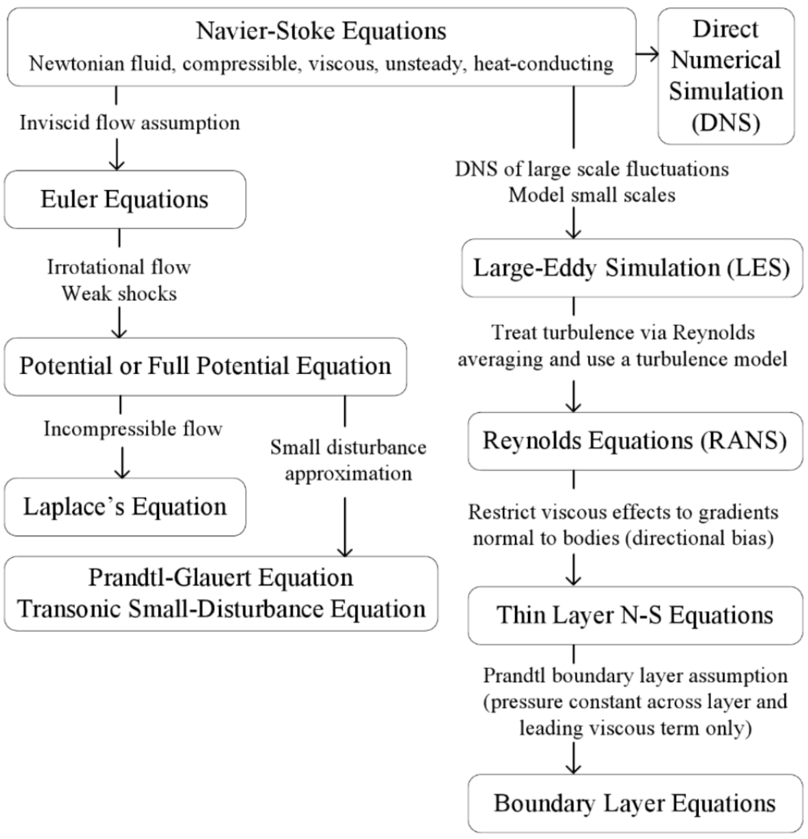
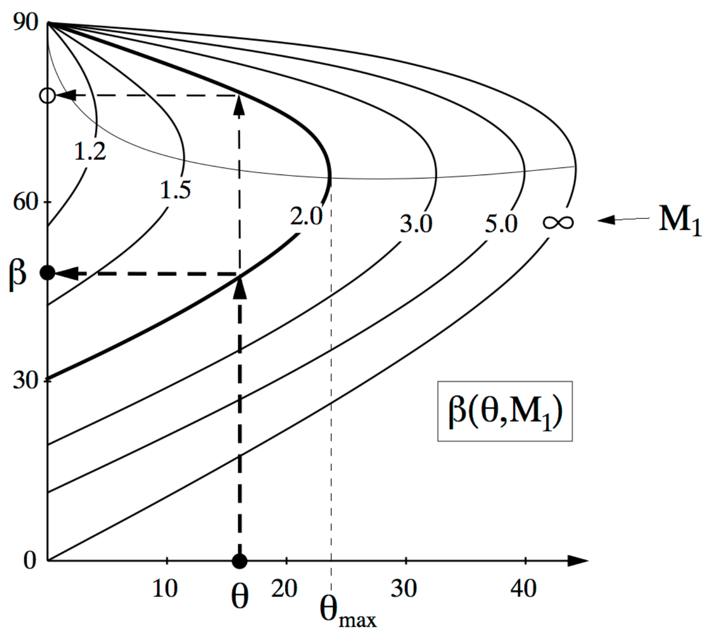

Navier-Stoke Equations
Newtonian fluid, compressible, viscous, unsteady, heat-conducting
Direct Numerical Simulation (DNS)
Inviscid flow assumption
Euler Equations
Irrotational flow
Weak shocks
Potential or Full Potential Equation
Incompressible flow
Laplace's Equation
Small disturbance approximation
Prandtl-Glauert Equation
Transonic Small-Disturbance Equation
DNS of large scale fluctuations
Model small scales
Large-Eddy Simulation (LES)
Treat turbulence via Reynolds averaging and use a turbulence model
Reynolds Equations (RANS)
Restrict viscous effects to gradients normal to bodies (directional bias)
Thin Layer N-S Equations
Prandtl boundary layer assumption (pressure constant across layer and leading viscous term only)
Boundary Layer Equations

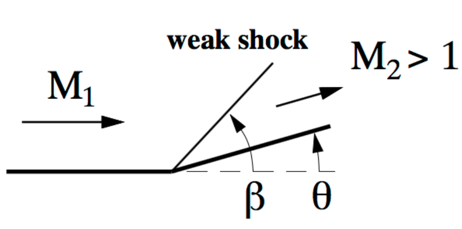
Figure 3.1: The governing flow equations based on the fluid flow assumptions [7]

the flow is characterized by the presence of statistical fluctuations of all the fluid flow variables (such as the density, pressure, velocities, etc.) around mean values as a consequence of the nonlinear convection terms. The Direct Numerical Simulation (DNS) simulates the whole range of these turbulent statistical fluctuations at all relevant physical scales in space and time. The Large-Eddy Simulation (LES) computes, just like the DNS, directly the turbulent fluctuations, but only above a certain scale. Below this scale, the turbulence is modeled by semi-empirical laws. This means that that the model makes some estimations by considering experimental data for its constituent part in concert with some analytical models. As many calculations as possible are being replaced with pre-calculated data that are an integral part of the program, which speeds up the calculations.The Reynolds-Averaged Navier-Stokes equations (RANS) approach reduces the computational effort even further, since it treats turbulence through turbulence models. Therefore, it is not required to resolve all the turbulent scales (which is the case for DNS and partially for LES). To acquire the RANS approach a turbulent quantity is averaged over time into a mean and fluctuating component. By doing this, the viscous effects in the flow are retained.

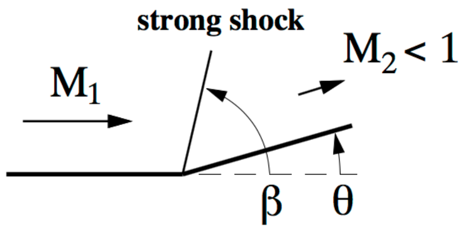
Proceeding to the left side of the figure, the Euler equations can be encountered, which are obtained by assuming inviscid flow. They are valid for any arbitrary body shape at any angle of attack. However, they will not produce reliable results if separation or other strong viscous effects are present. In transonic flow, the supersonic region along the airfoil is being terminated by a shock wave, in order to reduce the flow behind the shock wave to subsonic speeds. Besides reducing the flow speed, and thus the Mach number, discontinuously across this shock wave, the pressure, temperature, density, and entropy are being increased. A distinction can be made between weak and strong shock waves, which can be explained with the oblique shock chart in Figure 3.2a. With the given upstream Mach number M_1 and deflection angle θ the wave angle β can be determined by using this oblique shock chart. Two oblique shock waves with two different wave angles β are possible if $\theta \leq \theta_{max}$, where the smaller β corresponds to a weak shock while the larger β corresponds to a strong shock. However, in supersonic flow weak shock waves are most likely to occur (up to freestream Mach numbers of approximately $M_\infty = 1.3$). Both shock waves can be seen in figures 3.2b and 3.2c. Here, M_2 represents the Mach number of the flow after the shock waves. This is also an indication of how strong the shock wave is. The lower the Mach number behind the shock wave, the stronger the shock. In addition, the flow of weak shock waves can assumed to be isentropic, because of the small entropy change across the shocks. This is due to the fact that the change in entropy is related to the shock strength by a third order relation. The assumption of isentropic flow then allows for the assumption of irrotational flow, which cascades the Euler equations to the full potential equation. This equation is a single, nonlinear PDE. Since irrotational flow has resulted from the assumption of inviscid flow, it should be noted that viscous effects can be neglected for flows with hight Reynolds numbers (i.e Reynolds numbers of typical aircraft range from 10^5 to 10^7 [7]) dominated dominated by convective transport of momentum. Consequently, external laminar remaining flows over solid surfaces or objects at high Reynolds numbers can be analyzed with potential flow, since this flow is assumed to be irrotational [8]. Furthermore, for flow with high speeds over surfaces, and thus high Reynolds numbers through the relation $Re \sim V$, the viscous boundary layer growing next to the solid body will be thin. In this case, the existence of the boundary layer can be neglected when analyzing the potential flow region, since assuming potential flow would be improper for viscous flows. Consequently, neglecting boundary layers would mean the potential flow to follow the contours of the solid surface. In addition to the assumption of irrotationality, a small-disturbance assumption can be made if the airfoil is assumed to be a slender body at a small angle of attack. This assumption transforms the full potential equation to the TSD equation. It should be noted that all of the three sets of equations described in this paragraph represent models for the analysis of inviscid, transonic flow past an airfoil and can be applied effectively for high-speed airfoil designs.
The TSD can not be simplified further, since this would result in linear theories. These theories will fail in describing high-subsonic/low-supersonic and transonic flows, since the transonic flow regime must be described by a non-linear equation or set of equations due to the mixed elliptic/hyperbolic and non-linear regions involved. These sets of equations, along with their assumptions and simplifications, have been summarized in Table 3.1. The Laplace equation has been included in this table as well, since it will be used in determining the initial perturbation velocity only. It can not be used for further computations, since this equation is only valid for incompressible flows.
Thus, it is concluded that the most simplified equation that should be solved for treatment of steady, two-dimensional flows around airfoils in high-subsonic, transonic, and low-supersonic speeds is the TSD equation. The Laplace equation will be used to determine the initial perturbation velocity.



(a) Oblique shock chart



(b) Weak shock



(c) Strong shock

Figure 3.2: Weak and strong shock waves [1]

Equation	Inviscid	Irrotational	Small Perturbations	Incompressible	Notes
Navier - Stokes					Homogeneous
Euler	X				
Full Potential	X	X			
Transonic Small Disturbance	X	X	X		
Prandtl - Glauert	X	X	X		Linearized
Acoustic	X	X	X		Linearized
Laplace	X	X		X	

Table 3.1: Commonly solved equations with corresponding assumptions [9]

3.2. DIFFERENTIAL EQUATIONS

The non-linear two-dimensional unsteady TSD equation, which is the governing equation, for inviscid, irrotational, compressible flow past a thin airfoil at small angles of attack, can be written as:

$$\left[1-M_\infty{}^2-M_\infty{}^2(\gamma+1)\frac{\delta\phi}{\delta\bar{x}}\right]\frac{\delta^2\phi}{\delta\bar{x}^2}+\frac{\delta^2\phi}{\delta\bar{z}^2}-2M_\infty{}^2\frac{\delta^2\phi}{\delta\bar{x}\delta\bar{t}}-M_\infty{}^2\frac{\delta^2\phi}{\delta\bar{t}^2}=0 \tag{3.1}$$

where

ϕ	=	perturbation velocity potential	$[-]$
M_∞	=	undisturbed free-stream Mach number	$[-]$
$\bar{x},\bar{z}$	=	cartesian coordinates	$[m]$
$\bar{t}$	=	time	$[s]$

For the derivation of this equation the reader is referred to Appendix B. In order to make Equation 3.1 entirely non-dimensional, the cartesian coordinates and time are non-dimensionalized with respect to the airfoil semi-chord b and the free-stream velocity U_∞. The non-dimensionalized cartesian coordinates and time are given by:

$$x=\frac{\bar{x}}{b} \tag{3.2a}$$

$$z=\frac{\bar{z}}{b} \tag{3.2b}$$

$$t=\frac{\bar{t}V_\infty}{b} \tag{3.2c}$$

The BC [1, 5] that will be imposed on the flow boundaries are as follows:

- Flow tangency condition: the resultant velocity vector is always tangent to the local airfoil surface.
- The pressure is continuous off the airfoil, particularly across the wake. This means that there is a zero pressure difference across the plane $z=0$ at all regions outside of the wing.
- The Kutta condition of finite velocities is satisfied at the trailing edge of the airfoil.
- At distances far upstream of the airfoil, the perturbation velocity potential and its derivatives fade out.

Since the governing equation given by Equation 3.1 is of non-linear form, it is assumed that ϕ consists of the sum of a steady component ϕ_0 and a time-dependent component ϕ_1:

$$\phi(x,z,t)=\phi_0(x,z)+\phi_1(x,z,t) \tag{3.3}$$

This decomposition is performed to solve Equation 3.1. On the basis of small perturbations, it is assumed that $\phi_1 \ll \phi_0$. Substituting Equation 3.3 into Equation 3.1, and equating equal orders of magnitude, results in the following set of governing equations:

$$(1-M_\infty{}^2)\frac{\delta^2\phi_0}{\delta x^2}+\frac{\delta^2\phi_0}{\delta z^2}=M_\infty{}^2(\gamma+1)\frac{\delta}{\delta x}\left(\frac{1}{2}\left(\frac{\delta^2\phi_0}{\delta x^2}\right)^2\right) \tag{3.4a}$$

$$(1-M_\infty{}^2)\frac{\delta^2\phi_1}{\delta x^2}+\frac{\delta^2\phi_1}{\delta z^2}-2M_\infty{}^2\frac{\delta^2\phi_1}{\delta x\delta t}-M_\infty{}^2\frac{\delta^2\phi_1}{\delta t^2}=M_\infty{}^2(\gamma+1)\frac{\delta}{\delta x}\left(\frac{\delta\phi_0}{\delta x}\frac{\delta\phi_1}{\delta x}\right) \tag{3.4b}$$

The obtained set of equations (especially Equation 3.4b) show that in order to solve the unsteady component ϕ_1, prior knowledge of the steady velocities $\frac{\delta\phi_0}{\delta x}$ in the flowfield is required. Although the current project only focuses on determining the steady perturbation potential ϕ_0 and the steady velocities $\frac{\delta\phi_0}{\delta x}$, the governing equations and boundary conditions for both situations are outlined.
The governing equations 3.4 are transformed into a more familiar form, by introducing the following parameters and variables:

Parameters:	Variables:
$\beta = \sqrt{1 - M_\infty{}^2}$	$x^{'} = \bar{x}$
$\mu = \frac{(\gamma+1)M_\infty{}^2}{\beta^3}$	$z^{'} = \beta\bar{z}$
$\nu = \beta\mu$	$t^{'} = \frac{\beta}{M_\infty}\bar{t}$
	$\phi^{'} = \nu\bar{\phi}$

Applying those parameters and variables to Equation 3.4 results in the following steady and unsteady governing equations:

$$\nabla^2 \phi_0^{'} = \frac{\delta}{\delta x}\left(\frac{1}{2}\left(\frac{\delta \phi_0^{'}}{\delta x^{'}}\right)^2\right) \tag{3.5a}$$

$$\left(\frac{D\phi_1^{'}}{Dt^{'}}\right)^2 + \frac{2M_\infty}{\beta}\frac{\delta^2\phi_1^{'}}{\delta x^{'}\delta t^{'}} = -\frac{\delta}{\delta x}\left(\frac{\delta \phi_0^{'}}{\delta x^{'}}\frac{\delta \phi_1^{'}}{\delta x^{'}}\right) \tag{3.5b}$$

Here, ∇^2 and $\left(\frac{D}{Dt}\right)^2$, are:

$$\nabla^2 \equiv \frac{\delta^2}{\delta x^{'2}} + \frac{\delta^2}{\delta z^{'2}} \quad \text{and} \quad \left(\frac{D}{Dt^{'}}\right)^2 \equiv \frac{\delta^2}{\delta t^{'2}} - \nabla^2$$

The governing equations are obtained, and the boundary conditions mentioned earlier are written in equation form using the tangency condition as follows:

$$\left[\frac{\delta\phi^{'}}{\delta z^{'}}\right]_{z^{'}=h^{'}} = \mu\left(\frac{\delta h^{'}}{\delta x^{'}} + \frac{\beta}{M_\infty}\frac{\delta h^{'}}{\delta t^{'}}\right) \qquad , -1 \leq x^{'} \leq 1 \tag{3.6}$$

The variable $h^{'}$ in this equation is defined as the vertical coordinate of the body surface. However, it should be noted that this boundary condition is valid only along the surface of the airfoil and thus enforced on the actual surface. This is indicated by $z^{'} = h^{'}$ in equation. The airfoil is located between $x' = -1$ and $x' = 1$, meaning that the chord of the airfoil is equal to $b = 2$. Therefore, the boundary condition given by Equation 3.6 is valid for $|x'| \leq 1$. In order to simplify the problem further, a thin airfoil is considered, and thus can be prescribed by placing a vortex sheet on the chord of the airfoil b. This theory is known as the thin airfoil theory, and is discussed in Section 2.2. Due to this assumption, the tangency boundary condition given by Equation 3.6 is satisfied on the airfoil chord $z^{'} = 0$, that is:

$$\left[\frac{\delta\phi^{'}}{\delta z^{'}}\right]_{z^{'}=0} = \mu\left(\frac{\delta h^{'}}{\delta x^{'}} + \frac{\beta}{M_\infty}\frac{\delta h^{'}}{\delta t^{'}}\right) \tag{3.7}$$

The next step is to decompose the vertical coordinates of the body $h^{'}$ into a steady component and unsteady component:

$$h^{'}(x^{'}, t^{'}) = h_0^{'}(x^{'}) + h_1^{'}(x^{'}, t^{'}) \tag{3.8}$$

where

$h_0^{'}(x^{'})$	=	thickness distribution
$h_1^{'}(x^{'}, t^{'})$	=	unsteady displacement of the wing mean surface about $z^{'} = 0$

Now, inserting equations 3.3 and 3.8 into Equation 3.7 results in the tangency boundary condition for the steady and unsteady case respectively:

$$\left[\frac{\delta\phi_0'}{\delta z'}\right]_{z'=0} = \mu\frac{\delta h_0'}{\delta x'} \qquad ,-1\leq x'\leq 1 \tag{3.9}$$

$$\left[\frac{\delta\phi_1'}{\delta z'}\right]_{z'=0} = \mu\left(\frac{\delta h_1'}{\delta x'} + \frac{\beta}{M_\infty}\frac{\delta h_1'}{\delta t'}\right) \qquad ,-1\leq x'\leq 1 \tag{3.10}$$

Since this study is concerned with solving the steady TSD equations, the derivation of the steady tangency condition will be shown in Appendix B.

Next, the wake boundary condition is defined, with continuous pressure across the wake. The wake condition is valid for $x' \geq 1$, since the wake arises at the trailing edge of the airfoil when the flow starts separating from the airfoil as can be seen in Figure 3.3. During flight, the speed of the aircraft and angle of attack of the wings (and thus the airfoils) change continuously. This in turn varies the strength of the trailing edge vortices, which is accounted for by a continuous sheet of weak vorticity known as the wake. These wake related boundary conditions are formulated as follows:

$$\Delta\left(\frac{\delta\phi_0'}{\delta x'}\right) = 0 \tag{3.11a}$$

$$\Delta\left(\frac{\delta\phi_1'}{\delta x'}\right) + \frac{\beta}{M_\infty}\Delta\left(\frac{\delta\phi_1'}{\delta t'}\right) = 0 \tag{3.11b}$$

where $\Delta((\phi_0')_x)$ represents the jump in $((\phi_0')_x)$ across the wake.

The goal of this research was to determine the aerodynamic loading on airfoils in transonic regimes, which can be done by computing the pressure distribution along the airfoil. However, the potential flow equations discussed so far are posed in terms of state variables and do not include the pressures. The differential equations and boundary conditions allow the computation of the local velocities, but not the pressure distributions. Therefore, after obtaining the velocities, the pressure coefficient $C_P \equiv \frac{P-P_\infty}{q_\infty}$ will be used to find the local pressures as can be seen in Appendix B. This leads to an expression for the pressure coefficient C_P:

$$C_P(x',z',t') = -\frac{2}{\nu b}\left(\frac{\delta\phi'}{\delta x'} + \frac{\beta}{M_\infty}\frac{\delta\phi'}{\delta t'}\right) \tag{3.12}$$

Again, substituting Equation 3.3 into Equation 3.12 results in:

$$C_P(x',z',t') = -\frac{2}{\nu b}\frac{\delta\phi_0'}{\delta x'} - \frac{2}{\nu b}\left(\frac{\delta\phi_1'}{\delta x'} + \frac{\beta}{M_\infty}\frac{\phi_1'}{\delta t'}\right) \tag{3.13}$$

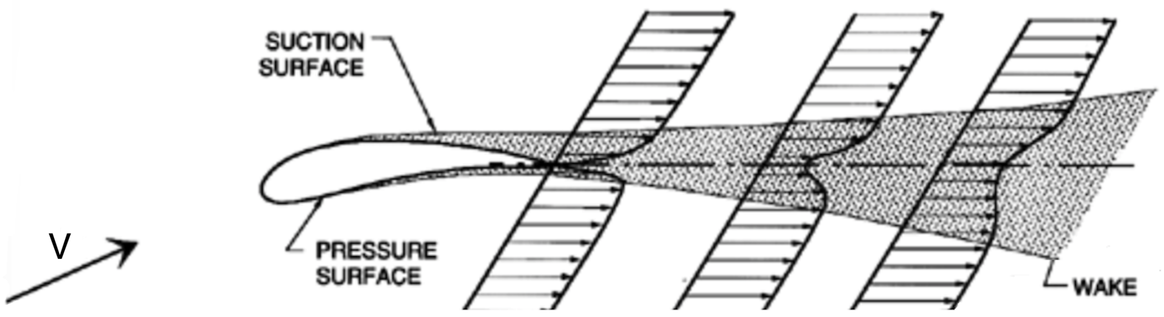


Figure 3.3: Airfoil wake [1]

Thus, the unsteady transonic small perturbation flow problem about two-dimensional lifting airfoils is formulated by the following governing equations and boundary conditions in differential form:

	Steady Component	Unsteady Component	
Governing Equation	$\nabla^2\phi_0' = \frac{\delta}{\delta x}\left(\frac{1}{2}\left(\frac{\delta\phi_0'}{\delta x'}\right)^2\right)$	$\left(\frac{D\phi_1'}{Dt'}\right)^2 + \frac{2M_\infty}{\beta}\frac{\delta^2\phi_1'}{\delta x'\delta t'} = -\frac{\delta}{\delta x}\left(\frac{\delta\phi_0'}{\delta x'}\frac{\delta\phi_1'}{\delta x'}\right)$	
Tangency Condition	$\left[\frac{\delta\phi_0'}{\delta z'}\right]_{z'=0} = \mu\frac{\delta h_0'}{\delta x'}$	$\left[\frac{\delta\phi_1'}{\delta z'}\right]_{z'=0} = \mu\left(\frac{\delta h_1'}{\delta x'} + \frac{\beta}{M_\infty}\frac{\delta h_1'}{\delta t'}\right)$	$\lvert x'\rvert \leq 1$
Wake Condition	$\Delta\left(\frac{\delta\phi_0'}{\delta x'}\right) = 0$	$\Delta\left(\frac{\delta\phi_1'}{\delta x'}\right) + \frac{\beta}{M_\infty}\Delta\left(\frac{\delta\phi_1'}{\delta t'}\right) = 0$	$x' \geq 1$

3.3. INTEGRAL EQUATIONS

In this section integral representation corresponding to the steady transonic differential equations will be derived. This will be done by linearizing equations 3.5a, 3.9 and 3.11a with the method of parametric differentiation, and applying Green's theorem [10]. Since this research focuses on applying the method of parametric differentiation to the transonic differential equations, the method itself will not be elaborated on. However, a parameter of interest is needed in using this method, which is taken to be the thickness ratio ϵ for this study. Equations 3.5a, 3.9 and 3.11a will be transformed to ϵ-space with $g = \frac{\delta\phi_0'}{\delta\epsilon}$, and result into the following linear governing equation and boundary conditions:

$$\nabla^2 g = \frac{\delta}{\delta x}\left(u\frac{\delta g}{\delta x}\right) \tag{3.14a}$$

$$\left[\frac{\delta g}{\delta z}\right]_{z=0} = \mu\frac{\delta}{\delta\epsilon}\left(\frac{\delta h_0}{\delta x}\right), |x| \leq 1 \tag{3.14b}$$

$$\Delta\frac{\delta g}{\delta x} = 0, x \geq 1 \tag{3.14c}$$

Note that the accents have been removed for simplicity, and u in Equation 3.14a denotes the steady component of longitudinal velocity $\frac{\delta\phi_o}{\delta x}$. This transformation process to ϵ-space can be reviewed in Appendix B. Now, let (x, z) be the running coordinates in the subsequent integrations. Then, for $M_\infty < 1$, application of Green's theorem converts Equation 3.14a into the following integral relation [11]:

$$g(\xi,\zeta) = g_T(\xi,\zeta) + g_L(\xi,\zeta) + g_{NL}(\xi,\zeta) \tag{3.15}$$

This means that the steady velocity potential consists of three contributions, namely:

1. thickness effects
2. lifting
3. non-linear phenomenon

Before discussing those contributions, the region of integration will be shown which can be used to obtain the integral equations of the different contributions. This region can be seen in Figure 3.4. The flow embodied by the circle in this figure (i.e., reasonably large finite region of the flow) defines a control volume $\mathscr{V}$ to which the fundamental physical principles are applied, and a control surface $\mathscr{S}$ is defined as the control surface which bounds the control volume. Instead of looking at the whole flow field at once, the attention has been limited to just the fluid in the finite region of the volume itself. The control volume shown in Figure 3.4 is bounded by the circle C_∞ around the airfoil (AB), and a cut that surrounds and wraps the surface of the body $(BCDA)$. Stations 1 and 2 are inflow and outflow stations, respectively. Assume that the contour C_∞ is far enough from the body such that the pressure is same everywhere on C_∞ and equal to the freestream pressure $P = P_\infty$, meaning that the perturbation velocity decays away from the airfoil. This assumption ensures that the real life situation of undisturbed flow far away from the airfoil is imitated as good as possible, and that the only disturbances are due to the airfoil. Consequently, the perturbation velocity fades away as the air moves farther from the airfoil.

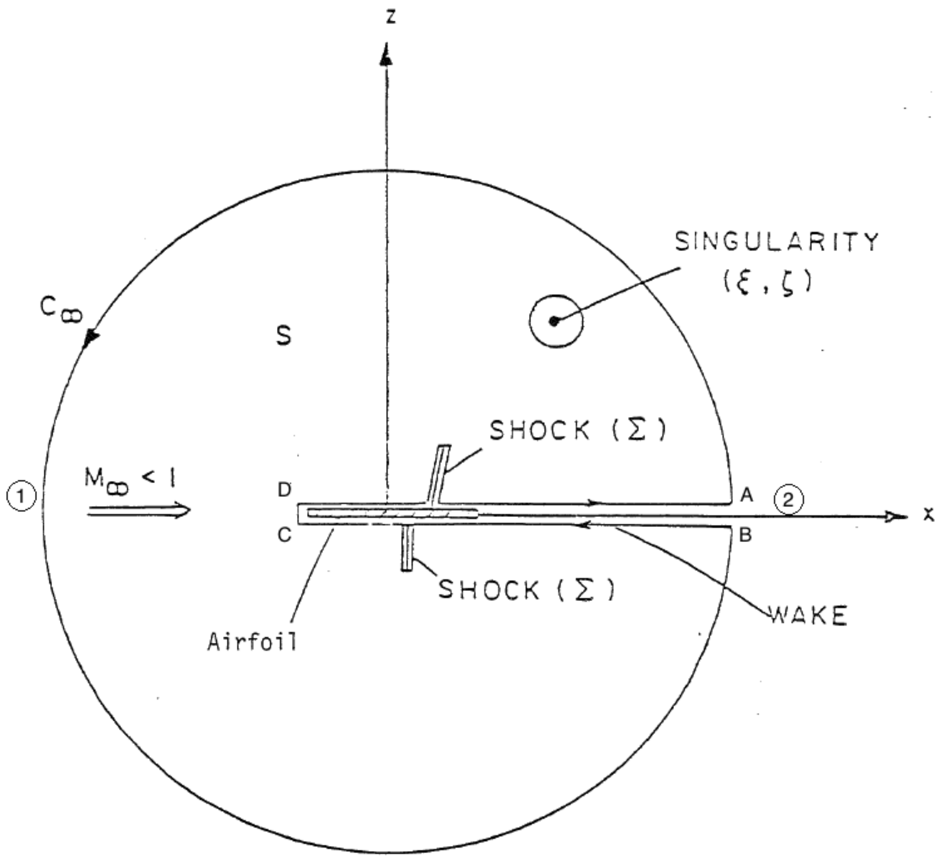


Figure 3.4: Integration region used in obtaining velocity potential contributions due to thickness, lift, shock-wave and non-linear phenoma [5]

3.3.1. THICKNESS EFFECT CONTRIBUTION

The contribution to the velocity potential due to thickness effects is [11]:

$$g_T(\xi,\zeta) = \frac{1}{2\pi}\int_{-1}^{1}\left[\Delta\frac{\delta g}{\delta z}\right]_{z=0} ln\left(\sqrt{(x-\xi)^2+\zeta^2}\right)dx \tag{3.16}$$

The term $\sqrt{(x-\xi)^2+\zeta^2}$ in this equation represents the distance from a vortex sheet segment to a point in the flow, as indicated in Figure 2.3d. However, point P for the thickness contribution case is placed between $x=-1$ and $x=1$ on the chord line, because of the thin airfoil theory mentioned in Chapter 2.2. Therefore the distance r varies only due to the variation of the location on the $x-$ axis.

3.3.2. LIFTING CONTRIBUTION

The lifting contribution to the velocity potential is represented by [11]:

$$g_L(\xi,\zeta) = \frac{1}{2\pi}\int_{-1}^{\infty}[\Delta g]_{z=0}\frac{\zeta}{(x-\xi)^2+\zeta^2}dx \tag{3.17}$$

The distance term appears in this equation as well, and is the same as for the thickness contribution case. The only difference here is that the integration boundaries in Equation 3.17 are defined by $x=-1$ and $x=\infty$. This is due to the fact that both the thin airfoil and wake behind the airfoil (which expands to infinity) can be simulated by placing a vortex sheet on them, resulting in both the airfoil and wake contributing to lift. The reader is referred to Chapter 2.2 for a more detailed explanation of this principle. However, this difference does not influence the $z-axis$; point P in Figure 2.3d is still located on the chord line. Therefore the distance term in equations 3.16 and 3.17 is the same. In order to maintain consistency in determining the several contributions to the velocity potential, Equation 3.17 will be transposed to an integral with boundaries from

$x = -1$ to $x = 1$. This is done using integration by parts once and knowing that $\left[\Delta\frac{\delta g}{\delta x}\right]_{x=-1} = 0$ and $[\Delta g]_{x=\infty} = [\Delta g]_{x=1}$, and results in:

$$g_L(\xi,\zeta) = \frac{1}{2\pi}\int_{-1}^{1}\left[\Delta\frac{\delta g}{\delta x}\right]_{z=0}\left[\frac{\pi}{2}sgn(\zeta) + tan^{-1}\left(\frac{\xi - x}{\zeta}\right)\right]dx \tag{3.18}$$

3.3.3. NON-LINEAR CONTRIBUTION

The last contribution to the velocity potential is due to the non-linear contributions, and is given by [11]:

$$g_{NL}(\xi,\zeta) = -\frac{1}{2\pi b}\int\int\frac{(x-\xi)}{(x-\xi)^2+(z-\zeta)^2}\left(u\frac{\delta g}{\delta x}\right)dxdz \tag{3.19}$$

where u denotes the x-component of the perturbation velocity. The distance from an element of the vortex sheet to a point in the flow field is indicated by $r = (x-\xi)^2 + (z-\zeta)^2$, and can be seen in the denominator of Equation 3.19. This distance differs from the distance term in equations 3.16 and 3.17, since Equation 3.19 is a surface integral. The region of integration is the whole flow field domain, instead of only the chord line.
An example of a non-linear contribution is due to the shock wave, which most likely will arise at transonic regimes.

3.3.4. TOTAL SET OF EQUATIONS

Now that the separate contributions are obtained, they can be put together to obtain the total set of equations, as given below:

$$\begin{aligned}
g(\xi,\zeta) = &\frac{1}{2\pi}\int_{-1}^{1}\left[\Delta\frac{\delta g}{\delta z}\right]_{z=0} ln\left(\sqrt{(x-\xi)^2+\zeta^2}\right)dx\\
&+\frac{1}{2\pi}\int_{-1}^{1}\left[\Delta\frac{\delta g}{\delta x}\right]_{z=0}\left[\frac{\pi}{2}sgn(\zeta) + tan^{-1}\left(\frac{\xi - x}{\zeta}\right)\right]dx\\
&-\frac{1}{2\pi b}\int\int\frac{(x-\xi)}{(x-\xi)^2+(z-\zeta)^2}\left(u\frac{\delta g}{\delta x}\right)dxdzvmberthis
\end{aligned}$$

The calculation of this equation proceeds in two steps:

1. First, the perturbation of the stream-wise derivative of the perturbation potential g_ξ (i.e. due to the nonlinear character of the flow) is determined. The derivation of g_ξ can be found in Appendix C.

2. Subsequently, the surface distribution of the vortex strength is determined. Here, it is important that the surface tangency boundary condition is satisfied. The derivation of this vortex strength is also outlined in Appendix C.

Those two steps of calculation are strongly coupled and must be solved numerically by an iterative procedure, until g_ξ converges to a certain value and does not change anymore. Then this value will be used to determine the perturbation velocity distribution $u(\xi,\zeta)$, which in turn can be used to determine the pressure coefficient distribution $C_P(\xi,\zeta)$ along the airfoil. A diagrammatic representation of this solution model is displayed in Figure 3.5, and will be explained in Chapter 4.

This section is concluded with a summary of the governing equation and boundary conditions of the steady transonic differential equation:

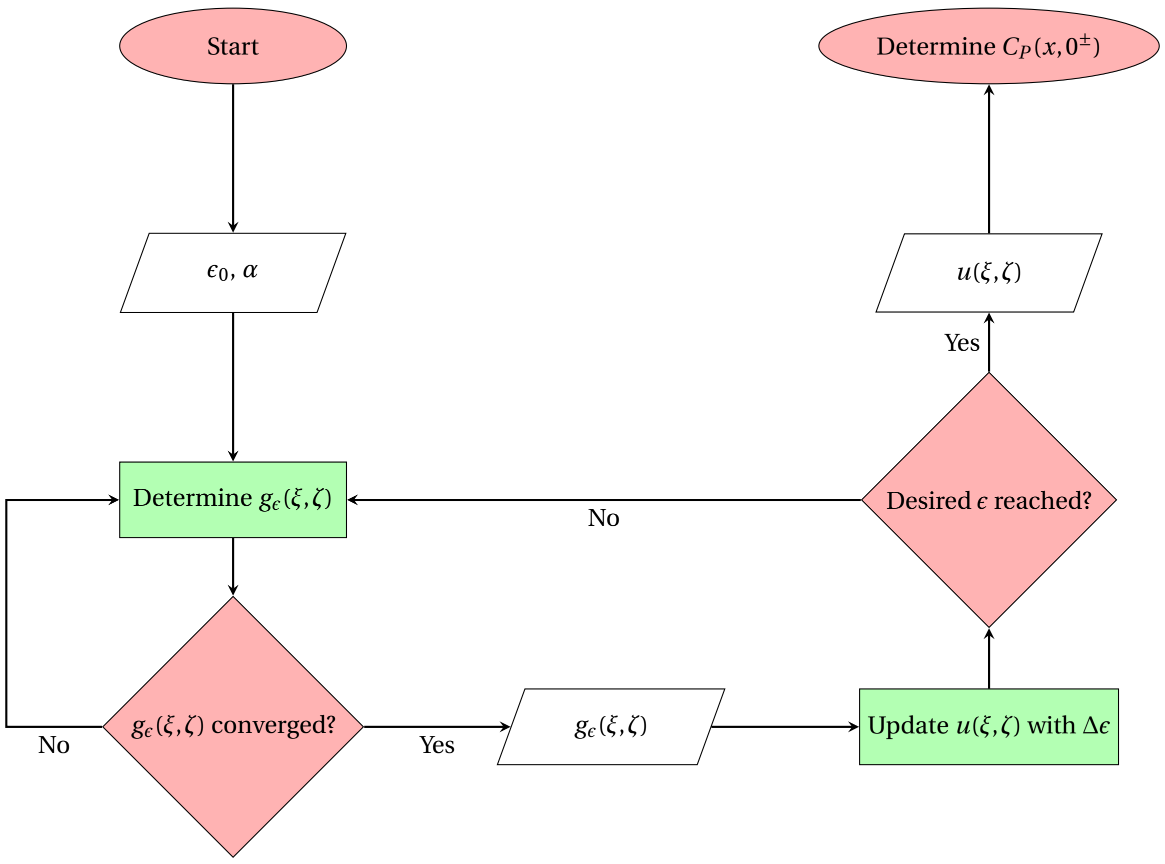

Start
ϵ_0, α
Determine $g_\epsilon(\xi,\zeta)$
$g_\epsilon(\xi,\zeta)$ converged?
No
Yes
$g_\epsilon(\xi,\zeta)$
Update $u(\xi,\zeta)$ with $\Delta\epsilon$
Desired ϵ reached?
No
Yes
$u(\xi,\zeta)$
Determine $C_P(x,0^\pm)$

Figure 3.5: Flowchart of solution model

	Steady Component	
Governing Equation	$g(\xi,\zeta) = \frac{1}{2\pi}\int_{-1}^{1}\left[\Delta\frac{\delta g}{\delta z}\right]_{z=0} ln\left(\sqrt{(x-\xi)^2+\zeta^2}\right)dx + \frac{1}{2\pi}\int_{-1}^{1}\left[\frac{\Delta g}{\delta x}\right]_{z=0}\left[\frac{\pi}{2}sgn(\zeta)+tan^{-1}\left(\frac{\xi-x}{\zeta}\right)\right]dx - \frac{1}{2\pi b}\int\int\frac{(x-\xi)}{(x-\xi)^2+(z-\zeta)^2}\left(u\frac{\delta g}{\delta x}\right)dxdz$	
Tangency Condition	$\left[\frac{\delta g}{\delta z}\right]_{z=0} = \mu\frac{\delta}{\delta\epsilon}\left(\frac{\delta h_0}{\delta x}\right)$	$\|x\| \le 1$
Wake Condition	$\Delta\frac{\delta g}{\delta x} = 0$	$x \ge 1$

4
Numerical Implementation

Once the integral representations of the governing equation and corresponding boundary conditions of to the steady transonic small disturbance equation are obtained, the equations are implemented numerically using a technical programming language. *MATLAB* is the program that is used for this project, and this chapter treats the implementation of the obtained set of equations into this program. First, the mesh structure is discussed in Section 4.1. This covers the type and number of cells on which the flow equations are solved, and highly affects the rate of convergence and the solution accuracy. As the grid cells become smaller, and the number of cells in the flow domain increase (i.e. fine grid rather than a coarse grid), the rate of convergence will most likely increase too. At the same time, the solutions will become more accurate. As a result, a trade off for the algorithm has to be made between rate of convergence (computation time) and solution accuracy. Section 4.2 then covers the implementation of the equations on such mesh, and is performed as explained earlier in the diagrammatic representation of the solution model given by Figure 3.5.

4.1. Mesh Structure

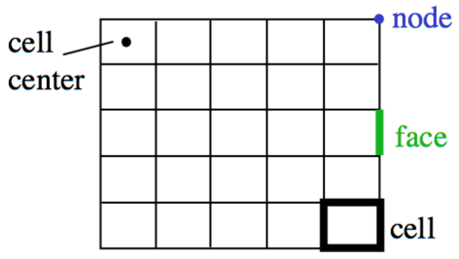
This section covers the mesh structure for the flow simulations performed under this study. The meshing starts by creating a grid, which qualifies the cells/elements the flow will be solved on, and represents the geometry of the problem. A grid has a huge significance, as it impacts the rate of convergence, the solution accuracy, and the required processing time. The quality of the mesh depends highly on the grid type. A sketch of a two-dimensional and three-dimensional computational grid is as shown in Figure 4.1 [12]. The terms used in this figure are as explained below:

Cell	=	control volume into which the domain is broken up
Cell center	=	center of a cell
node	=	grid point
edge	=	boundary of face
face	=	boundary of a cell
Zone	=	grouping of nodes, faces, and cells (wall boundary zone, fluid cell zone)
Domain	=	group of nodes, faces, and cell zones

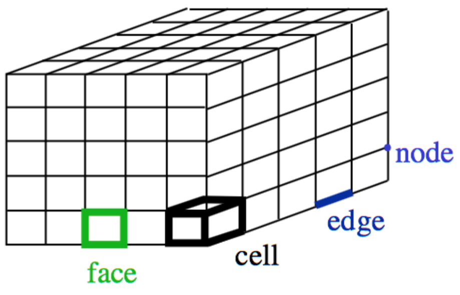
From the figure, it is clear that certain decisions regarding the grid type and shape are required for performing accurate flow simulations, which are further discussed in subsequent sections.

4.1.1. Cell type

The choice of the cell type depends on the problem and the solver capabilities. Figures 4.2 and 4.3 show the different cell types for two-dimensional and three-dimensional computations, respectively. One can see from these figures that a tetrahedron is the three-dimensional form of a triangular cell, and a hexahedron is a three-dimensional form of a quadrilateral cell. Since the objective of this research is to solve two-dimensional flow, the choice of cell types reduces to two cases as can be seen from Figure 4.2. Because of the ease of creation the quadrilateral cell is chosen to be the cell type that will be used in creating the computational domain.



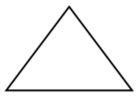
(a) 2D Computational Grid



(b) 3D Computational Grid

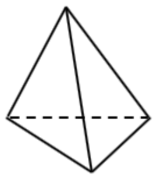
Figure 4.1: 2D and 3D computational grids [12]

(a) Quadrilateral

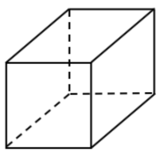


(b) Triangle

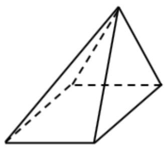
Figure 4.2: 2D cell types [12]



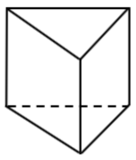
(a) Tetrahedron



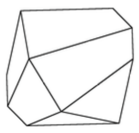
(b) Hexahedron



(c) Pyramid



(d) Wedge



(e) Polyhedron

Figure 4.3: 3D cell types [12]

4.1.2. GRID TYPE

This section outlines the most common grid types, after which the choice of grid type for the current project is explained. The four most common used grid types and their definitions are as follows:

- **Structured grid**: This mesh consists out of different types of hexahedral grids, and the mesh is represented in a single block. The connectivity information is such that the mesh follows a structured i, j, k convention.
- **Unstructured grid**: The cells of this type of grid are arranged in an arbitrary fashion, without i, j, k indices. Thus, the mesh has no logical representation. Also, the processing and computation time with these kind of grids are higher as compared to the structured grids.
- **Multiblock**: This type of mesh consists of multiple blocks. Each block can be meshed using a structured or unstructured meshing approach.
- **Hybrid**: This type of grid is created by using the most appropriate cell type in any combination; triangle and quadrilaterals in two-dimensional computations, and tetrahedra, prisms, and pyramids in three-dimensional computations. Also, the grid lines do not need to match at block boundaries.

Since the geometry of the airfoil used in this study to determine the aerodynamic loading represents a simple geometrical profile, and as the focus is on fast processing time, the structured grid will be used for the numerical computations. Furthermore, the coordinate system defined is a non-uniform rectangular grid centered at the mid-chord of the airfoil and symmetric with respect to the mid-chord. Also, the airfoil surface is projected on the plane $z = 0$ in the interval $-1 \leq x \leq 1$. This is depicted in Figure 4.4.
As shown in Figure 4.4, the final mesh generated is non-uniform over the whole domain owing to the reduction in processing time. The quality of the generated mesh is further discussed in this section. One measure

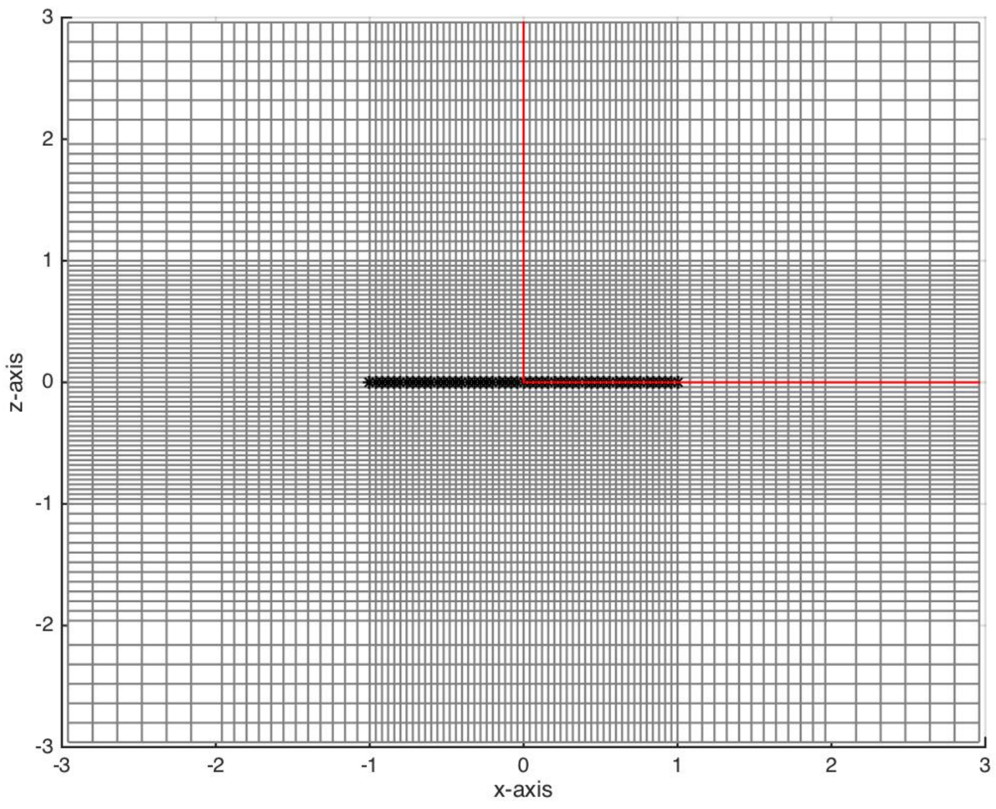


Figure 4.4: Mesh structure

of the mesh quality is the mesh density, which is required to be high enough to capture all relevant flow features, and can be stretched where the flow is fully-developed. Another important feature is the total number of cells. More cells can give a higher accuracy, with a downside of an increase memory and processing time. Since the objective of the current research is to develop a tool that is accurate enough, but with a low processing time, the cell count are kept low. This can be done by using a non-uniform grid to cluster cells only where they are needed. The symmetry of the mesh is another factor that affects the mesh quality. In order to get reliable results, a symmetrical mesh is desired. As such, the mesh spacing is decided to be non-uniform, with a finer mesh in the vicinity of the airfoil which gets coarser further away from the airfoil. This is due to the fact that the velocity gradients close to the airfoil surface are higher, and vanish when moving further away from the airfoil. Therefore, the grid points have been clustered in this region to capture these rapid velocity changes. Due to this type of non-uniform grid, the total number of mesh points are kept to a minimum while as much area of interest as possible will be covered. In addition, the mesh should be suitable for the integration and differentiation operations performed in the required numerical calculations. This will be explained in the next section. Therefore, taking all requirements into account, a symmetric and non-uniform structured grid consisting of hexahedral cells is used for the computations. Figure 4.4 shows that both the x - and z-directions are divided into three regions, which are indicated by the red and blue lines for the x- and z-directions respectively:

- The vicinity of the airfoil ($-1 < x < 1$ and $-1 < z < 1$)
- An intermediate region ($-2 < x < -1$, $1 < x < 2$, $-2 < z < -1$, $1 < z < 2$)
- The far field ($x < -2$, $x > 2$, $z < -2$, $z > 2$)

The mesh sizes dx and dz within each region are uniform but with different cell spacing; the grid region further away from the airfoil is less dense compared to the region around the airfoil. This is due to the boundary condition stating that he perturbation velocity potential and its derivatives fade out at large distances from the airfoil. Since the algorithm computes the velocity distribution potentials and its derivatives, which decreases in magnitude further away from the airfoil, a coarser grid is sufficient for the numerical computations. In particular, the mesh size is uniform along the airfoil surface, with 87 number of nodes in both the x and z-direction.

4.2. IMPLEMENTATION OF EQUATIONS

Once the mesh structure to perform the computations on is obtained, the way of implementing the mathematical model (which was outlined in Chapter 3) into *MATLAB* is discussed. This is done by using the flow chart displayed in Figure 3.5 as a guideline. Before analyzing the flowchart, an overview of the equations that will be implemented into the code will be given. Those equations are applied to a parabolic arc airfoil with a thickness ratio ϵ and under small angles of attack α. Hence, this section starts with an expression for the upper and lower boundaries of the profile:

$$h_0^{\pm} = \pm 2\epsilon(1-x^2) - \alpha x \tag{4.1}$$

The coordinates in this equation are normalized with respect to the airfoil semi-chord b. Equation 4.1 is used to determine the jump in normal velocity across the surface, by substituting it into the equation for the surface tangency condition given by Equation 3.9. This results in:

$$\begin{aligned}\left[\Delta\frac{\delta\phi_0}{\delta z}\right]_{z=0} &= \mu\left(\frac{\delta h_0^+}{\delta x} - \frac{\delta h_0^-}{\delta x}\right)\\ &= -8\mu\epsilon x\end{aligned} \tag{4.2}$$

Equation 3.14b is be used to transform Equation 4.2 into the ϵ parameter-space, by differentiating it with respect to ϵ. This yields in:

$$\left[\Delta\frac{\delta g}{\delta z}\right]_{z=0} = -8\mu x \tag{4.3}$$

These equations are implemented into equations C.1 and C.4 to obtain the complete set of integral equations formulating steady transonic flow about a parabolic arc airfoil. It is given as follows:

$$\begin{aligned}\frac{\delta g}{\delta\xi}(\xi,\zeta) &= \frac{4\mu}{\pi}\int_{-1}^{1}\frac{x(x-\xi)}{(x-\xi)^2+\zeta^2}dx\\ &\quad + \frac{1}{2\pi}\int_{-1}^{1}\left[\Delta\frac{\delta g}{\delta x}\right]_{z=0}\frac{\zeta}{(x-\xi)^2+\zeta^2}dx\\ &\quad - \frac{1}{2\pi b}\frac{\delta}{\delta\xi}\left[\iint_S\frac{(x-\xi)}{(x-\xi)^2+(z-\zeta)^2}\left(u\frac{\delta g}{\delta x}\right)dxdz\right]\\ \Delta\frac{\delta g}{\delta\xi}(\xi,0^{\pm}) &= \frac{2}{\pi}\sqrt{\frac{1-\xi}{1+\xi}}\int_{-1}^{1}\frac{I(x)}{\xi-x}\sqrt{\frac{1+x}{1-x}}dx\\ I(\xi) &= \frac{1}{\pi b}\iint_S\frac{2z(x-\xi)}{[(x-\xi)^2+z^2]^2}\left(u\frac{\delta g}{\delta x}\right)dxdz\end{aligned} \tag{4.4}$$

In order to start the computations, an initial solution is required. This will be done by choosing a base solution corresponding to some initial value of the thickness parameter ϵ_0. A convenient choice is $\epsilon \ll 1$, since the base solution can then be constructed from Laplace's equation introduced in Chapter 2.3 (i.e. $\nabla^2\phi = 0$). Thus, equating the steady component of the governing equation to zero, and satisfying the boundary conditions of the flow results in the base solution. Since the perturbation velocity u is needed in the equations given by Equation 4.4, the base solution is written in terms of the perturbation velocity as well. This can be done, since:

$$\frac{\delta u}{\delta\epsilon} = \frac{\delta g}{\delta x} \tag{4.5}$$

That is:

$$u_0(\xi,\zeta) = -\frac{1}{2\pi}\int_{-1}^{1}\left[\Delta\frac{\delta\phi}{\delta z}\right]_{z=0}\frac{(x-\xi)}{(x-\xi)^2+\zeta^2}dx + \frac{1}{2\pi}\int_{-1}^{1}[\Delta u]_{z=0}\frac{\zeta}{(x-\xi)^2+\zeta^2}dx \tag{4.6}$$

Here, $\left[\Delta\frac{\delta\phi}{\delta z}\right]_{z=0}$ is related to the airfoil geometry through Equation 4.2 and $[\Delta u]_{z=0}$ denotes the vortex singularity distribution along the chord. This quantity is given by the flat plate relation in the linearized theory, that is:

$$[\Delta u]_{z=0} = 2\mu\alpha\sqrt{\frac{(1-x)}{(1+x)}} \tag{4.7}$$

This relation satisfies the Kutta condition at the trailing edge ($x = 1$) and features a square root singularity at the leading edge ($x = -1$). Thus, combining equations 4.2, 4.6 and 4.7 results in the final expression of the base solution:

$$u_0(\xi,\zeta) = \frac{4\mu\epsilon_0}{\pi}\int_{-1}^{1}\frac{x(x-\xi)}{(x-\xi)^2+\zeta^2}dx + \frac{\alpha\mu}{\pi}\int_{-1}^{1}\sqrt{\frac{(1-x)}{(1+x)}}\frac{\zeta}{(x-\xi)^2+\zeta^2}dx \tag{4.8}$$

Implementing the coupled system of equations 4.4 and 4.8 in *MATLAB* is a complicated process, since iterative procedures are involved; updating for $\frac{\delta g}{\delta\xi}$, u and ϵ. These iterative procedures are displayed in Figure 3.5 by different colors. The computation process starts with determining a base perturbation velocity solution u_0 with the help of Equation 4.8, by choosing an initial thickness parameter ϵ_0. Here, the subscript 0 denotes that the determined values are the initial values to start the computations (i.e. base solutions). The next step is solving for $\frac{\delta g}{\delta\xi}$ by using Equation 4.4. This is done at a fixed ϵ-level (ϵ_0 in this case), and iterating the value of $\frac{\delta g}{\delta\xi}$ while keeping $u(x,z)$ ($u_0(x,z)$ in this case) constant. Once the value of $\frac{\delta g}{\delta\xi}$ is converged, the perturbation velocity $u(\xi,\zeta)$ is updated by the following formula:

$$\begin{aligned} u(\epsilon+\Delta\epsilon) &= u(\epsilon) + \int_{\epsilon}^{\epsilon+\Delta\epsilon}\frac{\delta u}{\delta\epsilon}d\epsilon \\ &= u(\epsilon) + \int_{\epsilon}^{\epsilon+\Delta\epsilon}\frac{\delta g}{\delta x}d\epsilon \end{aligned} \tag{4.9}$$

As shown by Equation 4.4 that the base value of $\frac{\delta g}{\delta\xi}$ consist of only the thickness effect contribution to the velocity potential g_T, since the lifting contribution g_L (second term) and non-linear contribution g_D(third term) disappear because of the non-existence of $\frac{\delta g}{\delta x}$ for the very first, thus initial, step.Thus, it is remarkable that the thickness effect contribution g_T is independent of $\frac{\delta g}{\delta x}$ and $u(\xi,\zeta)$ and can be placed outside the iterative loops in *MATLAB*.

After having calculated the first updated value of the perturbation velocity $u_1(\xi,\zeta)$ with $u_1(\xi,\zeta) = u_0(\xi,\zeta) + \left(\frac{\delta g}{\delta\xi}\right)_0\Delta\epsilon$, the aforementioned process of obtaining a converged value of $\frac{\delta g}{\delta\xi}$, thus $\left(\frac{\delta g}{\delta\xi}\right)_1$ in this case,with the new perturbation velocity $u_1(\xi,\zeta)$ is carried out. Then, the velocity potential is updated again with $u_2(\xi,\zeta) = u_1(\xi,\zeta) + \left(\frac{\delta g}{\delta\xi}\right)_1\Delta\epsilon$, after which the converged value of $\left(\frac{\delta g}{\delta\xi}\right)_2$ will be determined and so on.

The green symbols in the flowchart in Figure 3.5 indicate the iteration of $\frac{\delta g}{\delta\xi}$ until a converged value is found. This iteration is incorporated into the *MATLAB* code with an inner-loop, and indexed with n. The process of updating $u(\xi,\zeta)$, indicated by the red symbols in the flowchart, is incorporated with an outer-loop and indexed with m. It is important to note that $\Delta\epsilon$ is the same for all iterations, thus there is a uniform increment in the thickness ratio ϵ.

The iterative procedure is run until the desired thickness ratio of ϵ is reached, the pressure coefficient $C_P(x,0^{\pm})$ on the $z = 0$ can be computed with:

$$C_P(x,0^{\pm}) = -\frac{2}{bv}u(x,0^{\pm}) \tag{4.10}$$

This algorithm can also be found in equation form in Appendix D. Note that the perturbation velocity is being calculated in the whole flow domain, but only the value on the plane $z = 0$ should be used to determine the pressure coefficient C_P since the airfoil was projected onto this plane. The *MATLAB* file with the entire code can be found in Appendix E. One can see that the code is build up as follows:

- Defining parameters
- Mesh generation for vicinity of the airfoil, intermediate region, far field , and combining the separate meshes into a single mesh for the whole flow domain
- Computing the base solution;
  - Base solution in terms of the perturbation velocity $u(\xi,\zeta)$.
  - Base solution in terms of $\frac{\delta g}{\delta \xi}(\xi,\zeta)$, which consists of only the thickness effect contribution g_T as explained before.

  Both quantities are independent of $u(x,0^{\pm})$ and $\frac{\delta g}{\delta x}(\xi,\zeta)$, and are therefore placed outside the inner and outer loops.
- Inner loop, thus updating for $\frac{\delta g}{\delta \xi}(\xi,\zeta)$ until it converges;
  - Computing $I(\xi)$
  - Computing $\Delta\frac{\delta \xi}{\delta \xi}(\xi,0^{\pm})$
  - Computing the lift contribution g_L to $\frac{\delta g}{\delta x}(\xi,\zeta)$
  - Computing the non-linear contribution g_{NL} to $\frac{\delta g}{\delta x}(\xi,\zeta)$

  An interesting parameter tho note is the under-relaxation factor d at the end of the inner-loop. This factor is introduced, because without this factor the solutions of $\frac{\delta g}{\delta \xi}(\xi,\zeta)$ would be unstable. That is, when plotting $\frac{\delta g}{\delta \xi}(\xi,\zeta)$ versus iterations, a non-smooth curve with lots of rapidly changing ups and downs were noticed. Therefore, the under-relaxation factor was introduced since this is a significant parameter affecting the convergence of the numerical scheme, and represents the fraction of the solution being carried forward from one iteration (j) to the next $(j+1)$ for the computation of $\frac{\delta g}{\delta \xi}(\xi,\zeta)$ as follows:

$$\left(\frac{\delta \tilde{g}}{\delta \xi}\right)_{j+1} = \left(\frac{\delta g}{\delta \xi}\right)_j + d\left[\left(\frac{\delta g}{\delta \xi}\right)_{j+1} - \left(\frac{\delta g}{\delta \xi}\right)_j\right]$$

  For a more detailed overview of this process the reader is referred to Appendix D. An optimized value for this constant is therefore very important. Computations without an under-relaxation factor correspond to $d = 0$. By trial-and error, thus by starting with $d = 0$ and increasing this value, an under-relaxation factor of $d = 0.1$ is found to be optimal.
- Outer loop, thus updating for $u(x,0^{\pm})$ by increasing ϵ every iteration by $\Delta\epsilon$, as follows:

$$u_{i+1} = u_i + \left(\frac{\delta g}{\delta \xi}\right)_n \Delta\epsilon$$

  Appendix D shows in more detail how this update is achieved.

The numerical techniques used to evaluate the integrals on the mesh, as well as the numerical treatment of the differentiation showing up in the non-linear contribution to $\frac{\delta g}{\delta \xi}(\xi,\zeta)$ as shown in Equation 4.4, are outlined in Appendix C.

4.3. SUMMARY

The computations in this study are performed on a symmetric and non-uniform structured grid made up of hexahedral cells. Two loops have been incorporated, viz. inner-loop and outer-loop. The goal of the inner-loop is to find the converged value of $\frac{\delta g}{\delta \xi}$ in Equation 4.4, which then will be used in the outer-loop to update the perturbation velocity u. The number of iterations of the outer- loop is determined by the thickness ratio ϵ. In the current study the starting value of this thickness ratio ϵ and the increment in ϵ are set to be $\epsilon_0 = 0.00125$. The prediction of the loads is aimed to be performed for a 6% thick airfoil, i.e. resulting in a thickness ratio of $\epsilon = 0.06$. Both the upper and lower side of the airfoil contribute to this 6% thickness. Since the parabolic arc airfoil under consideration is a symmetric airfoil, the contribution of both sides is equal. Therefore, the thickness of both sides of the airfoil (as seen from the middle of the airfoil) is 3%. This is illustrated in Figure 4.5. Starting from $\epsilon_0 = 0.00125$, 24 incremental steps are required to reach this 3% thickness in incremental steps of $\Delta\epsilon = 0.00125$ (i.e. $\epsilon_0 = 0.00125$ is counted as the first incremental step). Therefore, 24 outer-loop iterations will be needed for the computations (indicated by $m = 24$ in the algorithm). After having computed the perturbation velocity with the desired thickness ratio, the pressure coefficient on the $z = 0$ are computed.

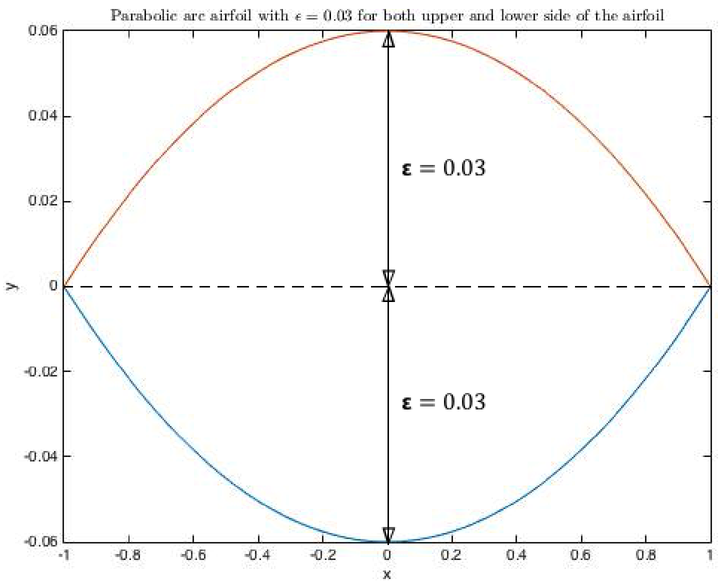


Figure 4.5: Thickness ratio of $\epsilon = 0.03$ for the upper- and lower-side of the parabolic arc airfoil

5 Results

This chapter discusses the results of the tool developed to solve transonic flow around the symmetrical parabolic arc airfoil. The pressure coefficient distributions have been determined for the freestream Mach numbers $M_\infty = 0.806$ and $M_\infty = 0.86$. At each Mach number, the angles of attack of $\alpha = 0°$, $\alpha = 0.5°$, and $\alpha = 1°$ have been treated. The number of mesh cells used for these calculations are 87 in both the x and z direction. Coarser or finer meshes have not been researched, since the resulting pressure coefficients were in good agreement with $NumecaFine^{TM}/Open$ (as will be explained in Chapter 6). Also, due to the inefficient implementation of mesh elements in the generated code, the number of cells were restricted to a particular constant values and no further changes were made.
Furthermore, the initial thickness ratio is taken to be $\epsilon_0 = 0.00125$, which in 24 iterative steps of $\delta\epsilon$ (outer-loop) is increased to $\epsilon = 0.03$ on both sides of the airfoil. This results in a total thickness of 6%. This airfoil thickness was investigated for comparison purposes with Lamah's results [5] in Chapter 6.The number of iterations for the x-derivative of the perturbation velocity potential g_x (inner-loop) is chosen to be $n = 10$. The number of iterations of the inner- and outer-loop ($n = 10$ and $m = 24$, respectively) have been obtained by starting with $m = 1$, and varying the number of the iterations of the inner-loop, until no changes in the final pressure coefficient distributions could be identified up to two decimal places. Then, the number of iterations of the outer-loop were varied. This was accompanied by changing the initial thickness ratio ϵ_0. This, again, has been done until a converged solution was reached. For the computations of g_x at the end of the inner-loop, an under-relaxation factor of $d = 0.1$ has been used for stability purposes. This value determines the faction of the difference between $\frac{\delta g}{\delta \xi}$ at the start of the iteration and $\frac{\delta g}{\delta \xi}$ computed at the end of the inner-loop. This difference is being added to the value of $\frac{\delta g}{\delta \xi}$ at the start of every iteration. The resulting $\left(\frac{\delta g}{\delta \xi}\right)_{j+1}$ is then used as the starting value for the next iteration. A value of $d = 0.1$ was found by starting with no under-relaxation factor, and increasing it with intervals of $\Delta d = 0.1$ to determine the effect of this factor on the final results. No under-relaxation factor thus means that the value of $\frac{\delta g}{\delta \xi}$ taken for the next iteration is equal to the value of $\frac{\delta g}{\delta \xi}$ determined at the end of the previous iteration. The reader is referred to Appendix D for a complete overview of this process. After having run several simulations with increasing d, it was observed that $d > 0.1$ results in no significant changes in the pressure coefficient distribution. However, the computation time was marginally increased (maximum time difference of about one minute). Therefore, since no significant difference in results was noticed, $d = 0.1$ was selected to minimize the resulting computational time.

First, the pressure coefficient distribution for the airfoil for zero angle of attack is treated in Section 5.1. The results for $\alpha = 0.5°$ and $\alpha = 1°$ are then reported in sections 5.2 and 5.3 respectively.

5.1. $\alpha = 0°$

This section presents the pressure coefficient distribution of the parabolic arc airfoil under zero angle of attack, immersed in free stream Mach numbers of $M_\infty = 0.806$ and $M_\infty = 0.86$. The C_P distributions are as shown in Figure 5.1. Before discussing these pressure distributions, the mathematical definition of the pressure coefficient C_P will be given by Equation 5.1a.

$$C_P = \frac{P - P_\infty}{q_\infty} \tag{5.1a}$$

$$q_\infty = \frac{1}{2}\rho_\infty V_\infty^2 \tag{5.1b}$$

where

C_P	=	Pressure coefficient	[-]
P	=	Static pressure at the point of interest	[Pa]
P_∞	=	Free stream static pressure	[Pa]
V_∞	=	Free stream velocity	$\left[\frac{m}{s}\right]$
ρ_∞	=	Free stream density	$\left[\frac{kg}{m^3}\right]$

When there is no perturbation in the flow field, the static pressure at the point of interest P is equal to the freestream static pressure P_∞. Therefore, the pressure coefficient C_P will be zero. However, when the freestream flow is disturbed by an object immersed in the flow, the local pressure changes. Due to this inequality between the pressure at the point of interest and the freestream static pressure, the pressure coefficient C_P at the point of interest will not be equal to zero anymore, as can be derived from Equation 5.1a. Remarkable features of the pressure coefficient distributions in figures 5.1a and 5.1b are itemized below.

- The pressure coefficient distributions for the upper and lower side of the airfoil lie on top of each other, meaning that those values are the same. This is as expected, since those pressure distributions are computed for a symmetric airfoil under zero angle of attack. Also, due to the symmetry of the airfoil in both the x- and z-directions, the pressure coefficients are mirrored in the z-axis at $x = 0$. This means that the pressure coefficients at $-1 \le x < 0$ are identical to the pressure coefficients at $0 > x \ge 1$.

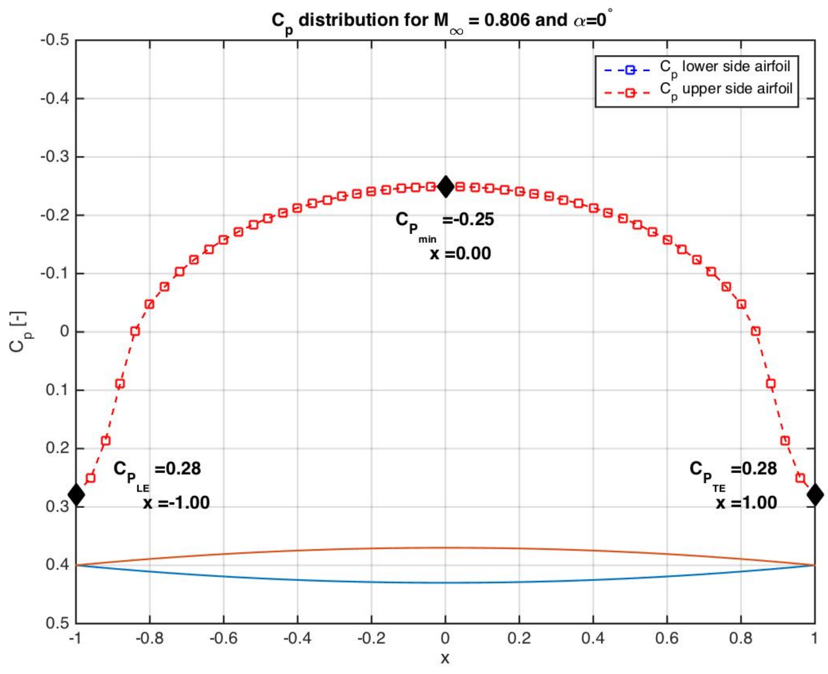
- For $M_\infty = 0.806$ the C_P distribution in Figure 5.1a starts out at the value of $C_{P_{LE}} = 0.28$ at the nose. The value of C_P drops as the flow expands around the nose, yielding a minimum value of $C_{P_{min}} = -0.25$ downstream of the nose. This occurs at the thickest point of the airfoil, which is at $x = 0$ for the symmetrical parabolic arc airfoil under consideration. Further downstream the pressure tries to recover and approaches a value of $C_{P_{TE}} \approx 0.28$ at the trailing edge. Such a region of increasing pressure in the direction of the flow is called an adverse pressure gradient. Too severe adverse pressure gradient leads to boundary layer transition, and possibly separation. The pressure distribution in Figure 5.1a does not indicate separation, since otherwise this value would be far below zero. In contrary, there is a positive pressure coefficient at the trailing edge, which is exactly equal to the pressure coefficient at the leading edge due to the symmetry of the airfoil.
  This trend in the C_P distribution is reasonable, since a typical pressure coefficient for symmetric airfoils starts with a positive value at the leading edge (where the local velocity at the nose has come to rest), whereafter it starts to decrease further downstream of the airfoil until it reaches its minimum value. After this point the pressure coefficients increases again and reaches a value similar to the value at the leading edge.

- For $M_\infty = 0.86$ the C_P distribution in Figure 5.1b starts out at the value of $C_{P_{LE}} = 0.29$ at the nose. The value of C_P drops as the flow expands around the nose, thereby decreasing the local pressure until it yields a minimum value of $C_{P_{min}} = -0.28$ downstream of the nose. This occurs at the thickest point of the airfoil, which is at $x = 0$ for the symmetrical parabolic arc airfoil under consideration. Further downstream the pressure recovers and approaches a value of $C_{P_{TE}} \approx 0.29$ at the trailing edge. Again, the pressure distribution in Figure 5.1b does not indicate separation and the positive pressure coefficient at the trailing edge is exactly equal to the pressure coefficient at the leading edge due to the symmetry of the airfoil. It should be mentioned here that no separation was expected after the location of minimum pressure coefficient, since this was already confirmed with the results of $NumecaFine^{TM}/Open$ (which will be shown in Chapter 6). $M_\infty = 0.86$ was determined to be the critical Mach number.
  The C_P distribution looks reasonable again for the same reason as for $M_\infty = 0.806$.

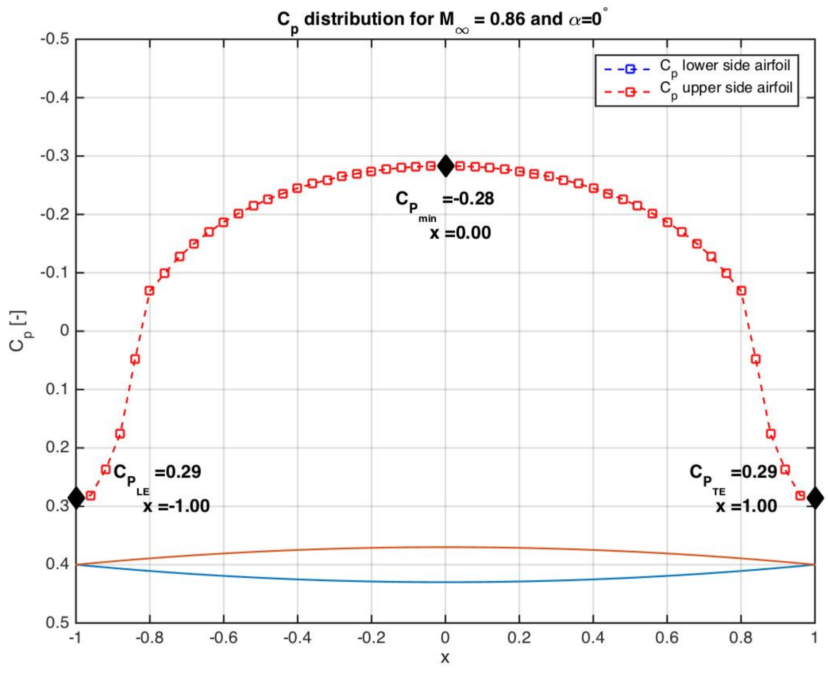
- When comparing the C_P distribution of the airfoil immersed in a flow with freestream Mach numbers $M_\infty = 0.806$ and $M_\infty = 0.86$, it can be observed that there is not a huge difference between the values of

C_P at the leading and trailing edges and the middle of the airfoil for the two Mach numbers. The values of C_P for $M_\infty = 0.86$ are slightly higher than for $M_\infty = 0.806$, which makes sense since there is more perturbation in case of a higher Mach number.

- The runtimes for $M_\infty = 0.806$ and $M_\infty = 0.86$ are roughly 5 minutes.



(a) $M_\infty = 0.806$

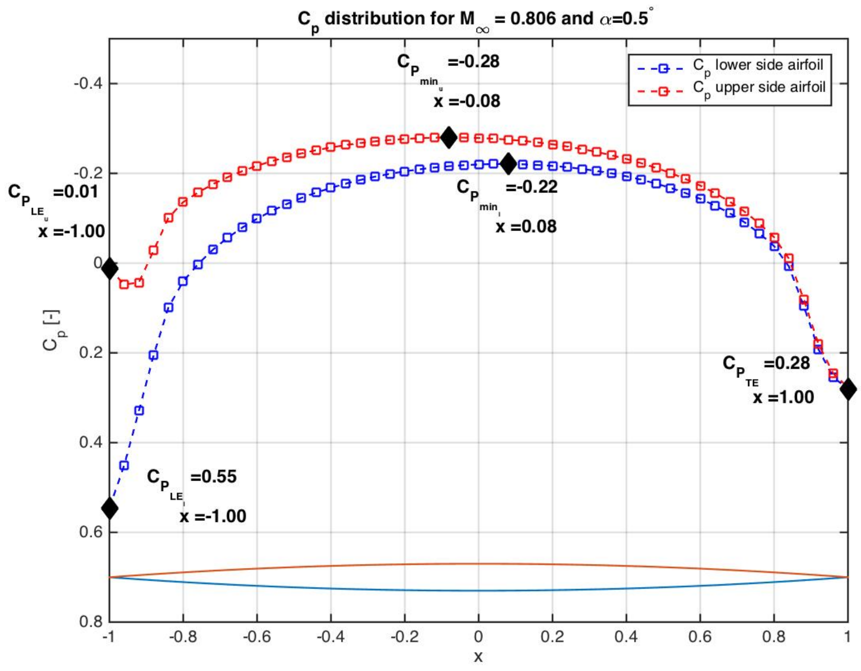


(b) $M_\infty = 0.86$

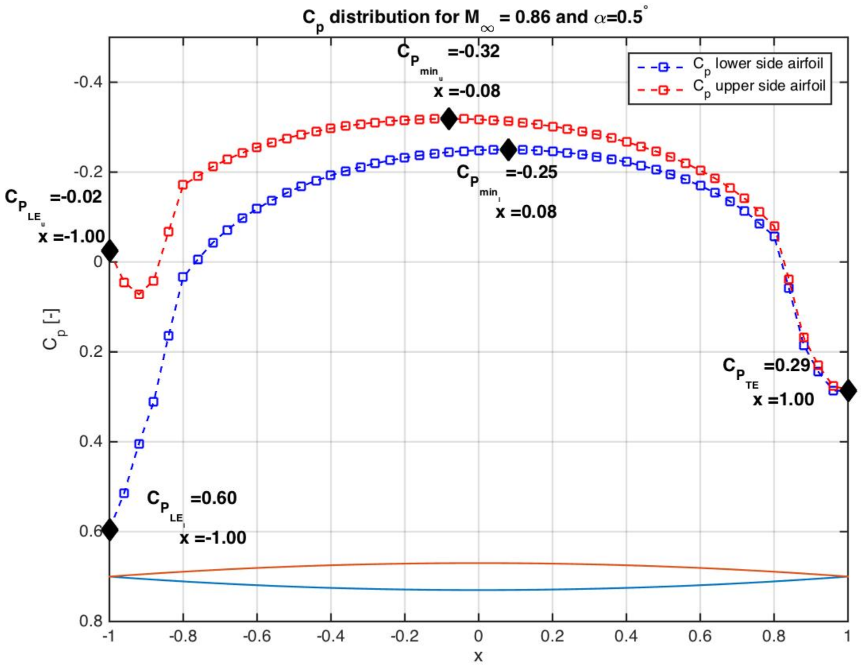
Figure 5.1: C_P distribution for $\alpha = 0°$

5.2. $\alpha = 0.5°$

The C_P distributions for the airfoil under an angle of attack of $\alpha = 0.5°$ and immersed in a flow with freestream Mach numbers $M_\infty = 0.806$ and $M_\infty = 0.86$ is treated in this section, as presented in Figure 5.2.



(a) $M_\infty = 0.806$



(b) $M_\infty = 0.86$

Figure 5.2: C_P distribution for $\alpha = 0.5°$

- When the angle of attack increases from $\alpha = 0°$ to $\alpha = 0.5°$, the C_P distributions along the upper and lower side of the airfoil start to separate from each other at the leading edge, rather than lying on top of each other, as can be seen from Figure 5.2. They meet again at the trailing edge, and reach a $C_{P_{TE}}$ of 0.29 and 0.30 for $M_\infty = 0.806$ and $M_\infty = 0.86$, respectively.

- For $M_\infty = 0.806$, C_P along the upper side of the airfoil starts at $C_{P_{LE}} = 0.01$, after which it decreases to $C_{P_{min}} = -0.28$ at the location $x = -0.08$. Further downstream C_P gradually increases and reaches $C_{P_{TE}} = 0.28$ at the trailing edge at the location $x = 1.00$.
  The C_P distribution along the lower side of the airfoil appears similar as on the upper side, but the level is higher. It starts with $C_{P_{LE}} = 0.55$ at the leading edge (i.e.$x = -1.00$), decreases to $C_{P_{min}} = -0.22$ at the location $x = 0.08$, and increases further downstream to $C_{P_{TE}} = 0.28$ at the trailing edge (i.e. $x = 1.00$).
- For $M_\infty = 0.86$, C_P along the upper side of the airfoil starts at $C_{P_{LE}} = -0.02$, after which it decreases to $C_{P_{min}} = -0.32$ at the location $x = -0.08$. Further downstream C_P gradually increases and reaches $C_{P_{TE}} = 0.29$ at the trailing edge at the location $x = 1.00$.
  The C_P distribution along the lower side of the airfoil again appears similar as the upper side, but its level is higher. It starts with $C_{P_{LE}} = 0.60$ at the leading edge (i.e.$x = -1.00$), decreases to $C_{P_{min}} = -0.25$ at the location $x = 0.08$, and increases further downstream to $C_{P_{TE}} = 0.29$ at the trailing edge (i.e. $x = 1.00$).
- For both Mach numbers, a remarkable behavior occurs on the upper side of the airfoil close to the leading edge, where a local maximum pressure coefficient occurs. The values of these maximum pressure coefficients are $C_P = 0.05$ and $C_P = 0.07$ respectively for $M_\infty = 0.806$ and $M_\infty = 0.86$. They occur at $x = -0.96$ and $x = -0.92$, respectively.
- The runtimes for $M_\infty = 0.806$ and $M_\infty = 0.86$ are roughly 5 minutes.

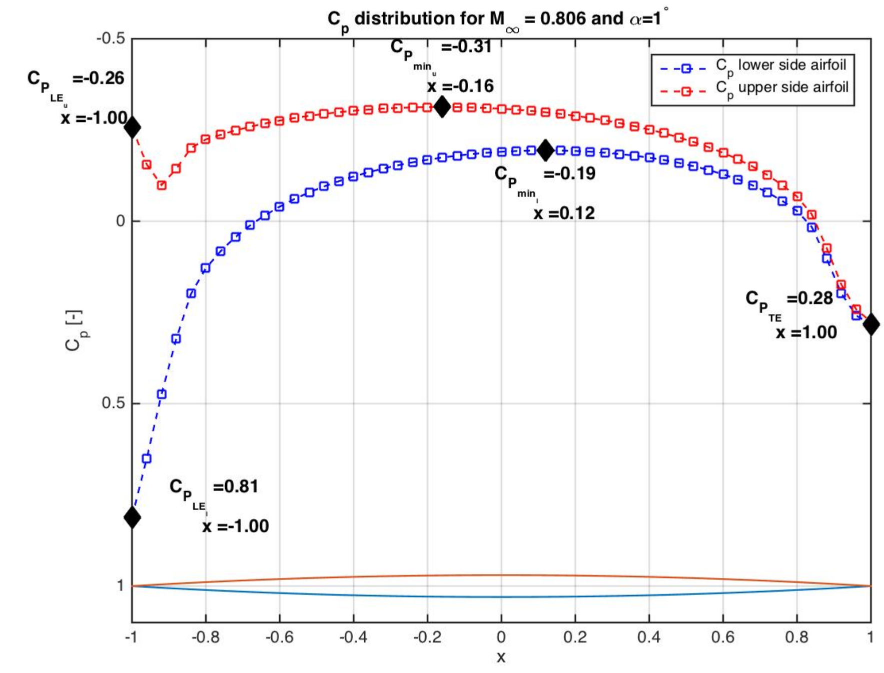
5.3. $\alpha = 1°$

The pressure coefficient distributions for an airfoil under an angle of attack of $\alpha = 1°$ immersed in a flow with freestream Mach numbers $M_\infty = 0.806$ and $M_\infty = 0.86$ are treated in this section, and are represented in figures 5.3a and 5.3b respectively. The explanatios of these distributions are as follows:

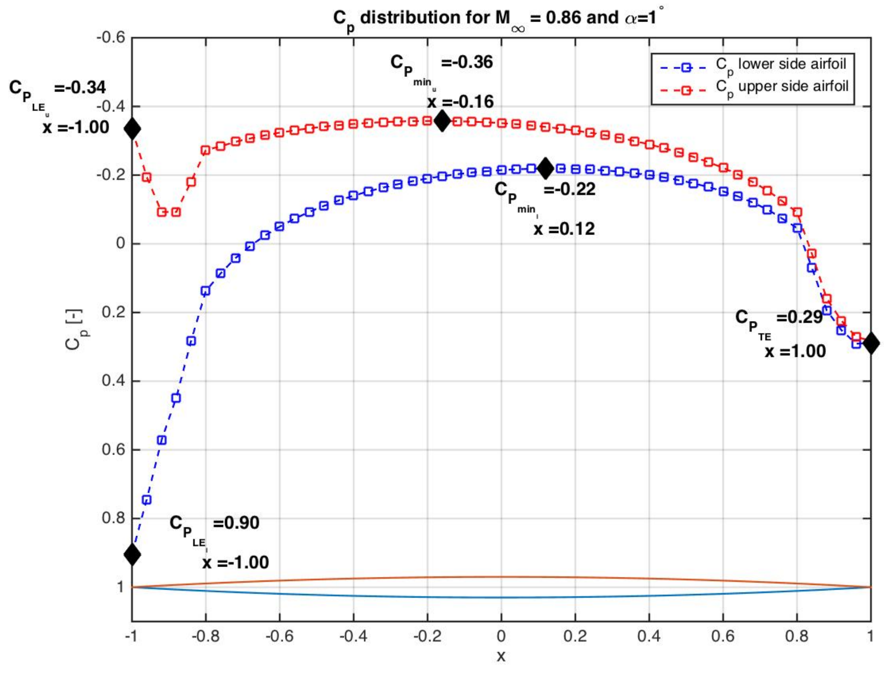
- The C_P distributions of the upper and lower side of the airfoil separate further from each other when the angle of attack is increased to $\alpha = 1°$, after which they merge at the trailing edge and reach a value of $C_{P_{TE}} = 0.28$ and $C_{P_{TE}} = 0.29$ for $M_\infty = 0.806$ and $M_\infty = 0.86$, respectively.
- For $M_\infty = 0.806$, the C_P distribution along the upper side of the airfoil starts at $C_{P_{LE}} = -0.26$, after which it decreases to a value of $C_{P_{min}} = -0.31$ at the location $x = -0.16$, and then increases again towards a value of $C_{P_{TE}} = 0.28$ at the leading edge (i.e. $x = 1.00$).
  The C_P distribution along the lower side of the airfoil starts at $C_{P_{LE}} = 0.81$ at the leading edge (i.e. $x = -1.00$), reaches a minimum value of $C_{P_{min}} = -0.19$ at the location $x = 0.12$, and increases again until it reaches a value of $C_{P_{TE}} = 0.28$ at the trailing edge (i.e. $x = 1.00$).
- For $M_\infty = 0.86$, the C_P distribution along the upper side of the airfoil starts at $C_{P_{LE}} = -0.34$, after which it decreases and reaches a value of $C_{P_{min}} = -0.36$ at the location $x = -0.16$, and then increases again towards a value of $C_{P_{TE}} = 0.29$ at the leading edge (i.e. $x = 1.00$).
  The C_P distribution along the lower side of the airfoil starts at $C_{P_{LE}} = 0.90$ at the leading edge (i.e. $x = -1.00$), reaches a minimum value of $C_{P_{min}} = -0.22$ at the location $x = 0.12$, and increases again until it reaches a value of $C_{P_{TE}} = 0.29$ at the trailing edge (i.e. $x = 1.00$).
- When the angle of attack increases from $\alpha = 0.5°$ to $\alpha = 1°$ the local pressure increase on the upper side of the airfoil close to the leading edge becomes more significant. The values of these local maximum pressure coefficients, together with their locations, for $M_\infty = 0.806$ and $M_\infty = 0.86$ have been determined to be $C_P = -0.10$ ($x = -0.92$) and $C_P = -0.09$ ($x = -0.92$).
- The runtimes for $M_\infty = 0.806$ and $M_\infty = 0.86$ are roughly 5 minutes.

5.4. Conclusions

In this section conclusions are drawn based on the pressure coefficient distributions of figures 5.1, 5.2, and 5.3. The present calculations require approximately 5 minutes of CPU time for all the cases. First, the effect of



(a) $M_\infty = 0.806$



(b) $M_\infty = 0.86$

Figure 5.3: C_P distribution for $\alpha = 1°$

increasing angle of attack α are discussed, followed by the effect of increasing freestream Mach number M_∞

Considering the upper side of the airfoil, the increasing angle of attack α causes the pressure coefficients at the leading edge to decrease (i.e. become more negative). The minimum pressure coefficients along the upper side of the airfoil also decrease with an increasing angle of attack, while the locations where they occur

move upstream towards the leading edge. The pressure coefficients at the trailing edge were observed to be uneffected by the increasing angle of attack.
Along the lower side of the airfoil, the increase in α affects the pressure coefficients at the leading edge and the minimum pressure coefficients in the opposite way as for the upper side of the airfoil. The pressure coefficients at the leading edge increase (i.e. become more positive) with increasing angle of attack. The minimum pressure coefficients also increase with increasing angle of attack, but their locations on the contrary move further downstream towards the trailing edge.
The values of the local increase in pressure coefficient along the upper side of the airfoil do change with increasing α too, in that their values increase (i.e. become more positive). The locations where they occur, however, almost remain the same. Only for $M_\infty = 0.806$ this location moves further downstream (towards the trailing edge of the airfoil) along the the upper side of the airfoil.

Examining the effects of the increasing freestream Mach number M_∞ on the pressure coefficients, it can be concluded that for all angles of attack α an increase in M_∞ leads to increasing pressure coefficients at both the upper and lower sides of the leading edge (i.e. they become more positive). The minimum pressure coefficients on the contrary, become more negative (i.e. they decrease) with increasing M_∞. This is as expected, since more suction (i.e. more negative pressure coefficient) is expected when the aircraft goes faster, resulting in more lift. The locations where the minimum pressure coefficients occur remain the same, except for $\alpha = 1°$, where the locations along the lower side of the airfoil move further upstream (i.e. towards the leading edge). Finally, the pressure coefficients at the trailing edges increase with increasing M_∞ (i.e. become more positive). Subsequently the values of the local pressure increase along the upper side of the airfoil increase with increasing freestream Mach numbers for $\alpha = 0.5°$, while their locations move further downstream (i.e. towards the trailing edge). For $\alpha = 1°$, this value decreases with $\Delta C_P \approx 0.01$, while the locations where they occur remain the same.

5.5. LIMITATIONS

The algorithm developed for this study has some limitations, and thus generates reasonable results within certain limits. These limits are set by some factors, which are discussed here.

The first parameter is the number of inner and outer iterations, which determines the convergence rate and thus the speed of the algorithm. The initial parameters of the algorithm were $n = 120$ and $m = 6$ (with an initial thickness parameter of $\epsilon_0 = 0.005$, where n and m represent the number of inner and outer iterations, respectively. The runtime of the algorithm with these initial runtime parameters was approximately 17 minutes for all Mach numbers and angle of attacks. However, since run time speed was a key requirement for the algorithm, this code is optimized and made faster by finding the most efficient combination of inner and outer loop iterations. This combination turned out to be $n = 10$ and $m = 24$ ($\epsilon_0 = 0.00125$), which reduced the run time of the algorithm to approximately 5 minutes.

Finding the optimal number of iterations was accompanied by finding the optimal under-relaxation factor, which is a restraint on the change of a dependent variable (the derivative of the disturbance velocity potential g_x in this study) from one iteration to the next and is needed for stability purposes. Stability of the coupled, (non-)linear system of equations is essential, since this factor could result in faster convergence or in preventing divergence. Including an under-relaxation factor affects the number of iterations needed to reach convergence. However, as long as the simulation is fully converged, the same result will be obtained, irrespective of under-relaxation values. Thus, under-relaxation factors do not affect the solution value if converged, but they only effect the time (number of iterations) taken to solution convergence.
This optimal under-relaxation factor has been found by trial and error, as explained before. First, the computations have been run without an under-relaxation factor, which resulted in unstable solutions of g_x (i.e. non-smooth curves with rapidly changing ups and downs). Therefore, an under-relaxation factor has been introduced, starting with $d = 0.1$. Various computations have been run by increasing this value in steps of $\Delta d = 0.1$. The outcome of this process was that for $d > 0.1$ the final solution was not changing anymore, but there was a small increase in computation time. Since the objective of this study was to develop a fast algorithm, the smallest possible under-relaxation factor has been chosen. While optimizing the algorithm, it is observed that an under-relaxation factor becomes essential when the number of iterations increases. An increase in the number of iterations of the inner-loop in particular resulted in introducing an under-relaxation factor. This is due to the fact that with increasing iterations, the value of the dependent variable

(i.e. g_x), becomes larger. However, since the optimal number of inner-iterations was found to be $n = 10$, no under-relaxation factor was required while computing the pressure coefficient distribution for $M_\infty = 0.806$. Performing the same computations for $M_\infty = 0.86$ raised concerns, since the pressure coefficient values especially at the leading and trailing edges of the airfoil became extremely high. Therefore, an under-relaxation factor of $d = 0.1$ was introduced.
Thus, it was observed that the number of iterations of the inner and outer loop, freestream Mach number, and under-relaxation factor affect each other. The most optimal value of the number of iterations and the under-relaxation factor have been chosen by taking this dependency into account.

The steady TSD formulation given in Chapter 3 is valid for isentropic, irrotational flows with negligible viscous effects (flow separation in particular). It turned out that the current algorithm does not provide reliable results when M_∞ exceeds 0.92. At this Mach number, the pressure coefficient values at the leading and trailing edges in particular, fluctuate and become very high. The algorithm also does not generate results for $M_\infty = 0$, since all terms of the governing equations and boundary conditions either become zero or infinite at this Mach number. Thus, this algorithm produces results for Mach numbers ranging from $M_\infty = 0.01$ to $M_\infty = 0.92$. However, the reliability of these results for all Mach numbers in this range is doubtful for reasons that are discussed in Chapter 6. The algorithm generated for this study produces reliable results for freestream Mach numbers up to $M_\infty = 0.85$, since shock-waves start to arise at $M_\infty = 0.86$ for the case treated in this study. Thus, a shock-wave should be observed in the pressure coefficient distribution for $M_\infty = 0.86$, which is not the case as can be seen from the results displayed in this chapter. Hence, this algorithm produces reliable results for $0.01 \le M_\infty \le 0.85$.

6 Verification & Validation

This chapter will analyses the quality of the algorithm developed as part of this study. The purpose of this study is to develop a tool to predict the pressure coefficient distribution on a symmetric parabolic arc airfoil flying at transonic speeds (i.e. high-subsonic, low-supersonic), with varying angle of attacks in a flow with varying freestream Mach numbers. This airfoil in particular has been selected in order to be able to compare the outcomes of the resulting algorithm to Lamah's results [5]. By executing verification and validation studies, it is checked whether the developed product meets the underlaying purpose.

First, the verification of the tool is performed. Verification deals with whether the product is built right, thus whether the equations are solved right, how well the product approximates the model, and whether the obtained results are reasonable. This is done by comparing the results of this product with the result of other reliable sources. For the verification of the product that resulted from this study, two sources have been chosen. The first source is the master's thesis of Charles A. Lamah [5], who was supervised by one of the two supervisors (i.e. Professor Wesley L. Harris, MIT) of the study presented in this report. The second source is a computational fluid dynamics (CFD) tool called $NumecaFine^{TM}/Open$. This tool is a fully integrated platform of CFD based on unstructured grid systems. A more detailed explanation of these sources, as well as the verification process, can be found in Section 6.1.

The second step in the quality assessment process (i.e. validation) is carried out in Section 6.2. Validation answers the question whether the right product has been built, thus whether the right equations have been solved, whether the model includes all the physics to approximate reality, whether the product produces the results aimed for and does what it is built for.

6.1. Verification

As mentioned in the introduction of this chapter, verification answers the question whether the product is built right. This applies to the results of the product, and assess whether the obtained results are reliable. In order to decide on the quality of the results, a comparison study is performed using data from reliable sources. These sources, and their explanations can be found in Section 6.1.1. The results arising from these sources, which have been obtained for the three angles of attack $\alpha = 0°$, $\alpha = 0.5°$, and $\alpha = 1°$, can be reviewed in Section 6.1.2.

Thus, in this section the question regarding whether the algorithm, is built right is answered by comparing the pressure coefficient distributions resulting from the algorithm and Lamah's results [5] along with the pressure coefficients obtained using $Numeca$. It should also be noticed that Lamah only generated pressure coefficients for the freestream Mach number $M_\infty = 0.806$, and angles of attack $\alpha = 0.5°$ and $\alpha = 1°$. Thus, for the remaining freestream Mach number and angles of attack this source of comparison is lacking.

6.1.1. Tools

Lamah

Charles A. Lamah has presented a numerical method for the solution of the transonic small disturbance equation in his master's thesis [5] at the Massachusetts Institute of Technology (MIT). The analysis combines the method of parametric differentiation and the integral equation technique to predict aerodynamic loadings on thin lifting airfoils in steady, subcritical, two-dimensional flow. The developed product has been applied to a 6% thick symmetric parabolic arc airfoil, immersed in a freestream Mach number of $M_\infty = 0.806$ under angle of attacks $\alpha = 0.5°$ and $\alpha = 1°$.
Lamah's work has been appointed as a verification tool, because it models the same equations and has been verified with other sources.

$NumecaFine^{TM}/Open$

$NumecaFine^{TM}/Open$ is a fully integrated platform of CFD based on unstructured grid systems. It is designed to solve complex internal and external flows and is dedicated to any flow, from incompressible to compressible and low speed to high speed flows. It combines completely unstructured hexahedral grids with an efficient preconditioned compressible solver with fast agglomerated multigrid acceleration and adaptation techniques. The simulation setup and settings can be found in Appendix F.
By using this tool, the pressure coefficient distribution along the symmetric parabolic arc airfoil in consideration is computed by solving the Euler equations from Figure 3.1. This is done for three different angles of attack α; $\alpha = 0°$, $\alpha = 0.5°$, and $\alpha = 1°$. For each of these angles of attack, there are different test-setups, which are discussed in Appendix F. The resulting pressure coefficient distributions were outlined and discussed in Appendix G. The setups will be recalled here again.

- $\alpha = 0°$:
  - The airfoil is created in $Catia$ by inserting the airfoil coordinates with Equation 4.1. Then, the borders of the domain are created within $Catia$ as well.
  - The airfoil is created in $Catia$ by inserting the airfoil coordinates with Equation 4.1, but the borders of the domain are created using $Numeca$.
- $\alpha = 0.5°$:
  - The airfoil under an angle of attack $\alpha = 0°$ is created in $Catia$ by inserting the airfoil coordinates with Equation 4.1. Then, the borders of the domain are created within $Catia$ as well. The airfoil is put under an an angle of attack of $\alpha = 0.5°$ within $Numeca$.
  - The airfoil under an angle of attack $\alpha = 0°$ is created in $Catia$ by inserting the airfoil coordinates with Equation 4.1. Then, the borders of the domain are created within $Numeca$, and the airfoil is put under an angle of attack of $\alpha = 0.5°$ within $Numeca$.
  - The airfoil under an angle of attack of $\alpha = 0.5°$ is created in $Catia$ with Equation 4.1, as well as the borders of the domain itself. Then, the computations are performed by:
    - ◇ not decomposing the lift and drag computations of the external flow in a $x-$ and z-direction. This means that lift is fully oriented in the positive z-direction, while drag is fully oriented in the positive x-direction.
    - ◇ decomposing the lift and drag computations of the external flow in a $x-$ and z-direction, since the airfoil is placed under an angle of attack. This decomposition is done as explained in Appendix F.
  - The airfoil under an angle of attack of $\alpha = 0.5°$ is created in $Catia$ with Equation 4.1, but the borders of the domain are created within $Numeca$. Then, the computations are performed by:
    - ◇ not decomposing the lift and drag computations of the external flow in a $x-$ and z-direction. This means that lift is fully oriented in the positive z-direction, while drag is fully oriented in the positive x-direction.
    - ◇ decomposing the lift and drag computations of the external flow in a $x-$ and z-direction, since the airfoil is placed under an angle of attack. This decomposition is done as explained in Appendix F.
- $\alpha = 1°$:

- The airfoil under an angle of attack $\alpha = 0°$ is created in $Catia$ by inserting the airfoil coordinates with Equation 4.1. Then, the borders of the domain are created within $Catia$ as well. The airfoil is put under an angle of attack of $\alpha = 1°$ within $Numeca$.
- The airfoil under an angle of attack $\alpha = 0°$ is created in $Catia$ by inserting the airfoil coordinates with Equation 4.1. Then, the borders of the domain are created within $Numeca$, and the airfoil is put under an angle of attack of $\alpha = 1°$ within $Numeca$.
- The airfoil under an angle of attack of $\alpha = 1°$ is created in $Catia$ with Equation 4.1, as well as the borders of the domain itself. Then, the computations are performed by:
  - not decomposing the lift and drag computations of the external flow in a $x-$ and z-direction. This means that lift is fully oriented in the positive z-direction, while drag is fully oriented in the positive x-direction.
  - decomposing the lift and drag computations of the external flow in a $x-$ and z-direction, since the airfoil is placed under an angle of attack. This decomposition is done as explained in Appendix F.
- The airfoil under an angle of attack of $\alpha = 1°$ is created in $Catia$ with Equation 4.1, but the borders of the domain are created within $Numeca$. Then, the computations are performed by:
  - not decomposing the lift and drag computations of the external flow in a $x-$ and z-direction. This means that lift is fully oriented in the positive z-direction, while drag is fully oriented in the positive x-direction.
  - decomposing the lift and drag computations of the external flow in a $x-$ and z-direction, since the airfoil is placed under an angle of attack. This decomposition is done as explained in Appendix F.

The objective of performing different tests for the borders of the domains created in $Catia$ and $Numeca$, is to examine whether one of the two programs is more efficient. Another important point to mention is the way the airfoil is created within $Catia$. This is done by first computing the airfoil coordinates with Equation 4.1 with $MicrosoftExcel$, and then importing those coordinates into $Catia$ and interpolating in between the coordinates with the $GSD_PointSplineLoftFromExcel$ feature of $Catia$. Finally, the reason for decomposing and not decomposing the lift and drag computations of the external flow in $x-$ and $z-$direction is to examine whether this decomposition is needed when the airfoil is already put under an angle of attack within $Catia$.
Each one of these computations is carried out for different number of mesh cells in the $x-$ and z-directions, i.e. 20, 40, 60, 80, 100, 250, 500, 750, and 1000 (same amount in both directions since the mesh is symmetric; i.e. 20 mesh cells in both $x-$ and $z-$directions). The simulations have been conducted for an increasing number of mesh cells to obtain converged solutions, i.e. as a convergence check.

6.1.2. RESULTS

The way this section is structured is by treating the results generated for $M_\infty = 0.806$ and $M_\infty = 0.86$ in separate sections. First, the pressure coefficients obtained for $M_\infty = 0.806$ are discussed, followed by the results generated for $M_\infty = 0.86$.

$M_\infty = 0.806$

The pressure coefficients of the parabolic arc airfoil immersed in a flow with freestream Mach number $M_\infty = 0.806$, and at angles of attack $\alpha = 0°$, $\alpha = 0.5°$, and $\alpha = 1°$ are displayed in figures 6.1, 6.2, and 6.3. These results also are summarized in Table 6.1 for a better overview.

Starting with Figure 6.1, one can notice that only one verification tool is provided here ($Numeca$). This is due to the fact that Lamah's work does not contain C_P distributions for an airfoil under zero angle of attack. Therefore only the results obtained with $NumecaFine^{TM}/Open$ will be used for comparative study. From this figure it can be observed that the two sets of data show good agreement in the shape of the pressure coefficient distributions. The locations of $C_{P_{LE}}$, $C_{P_{min}}$, and $C_{P_{TE}}$ are exactly the same for the results of both tools (the algorithm and $Numeca$). However, there is some difference in the magnitude of the pressure coefficients itself. The biggest discrepancies take place at the leading and trailing edges, with discrepancies of 61% and 32%, respectively. It is also striking that the data obtained with $Numeca$ are not symmetrical, meaning that the pressure coefficients at the leading and trailing edges ($C_{P_{LE}}$ and $C_{P_{TE}}$, respectively) differ. The data obtained with the algorithm on the contrary, result in the same pressure coefficient at the leading and trailing edges, which is as expected since the computations are applied to a symmetrical parabolic arc airfoil. This inequality of the pressure coefficients at the leading and trailing edges, computed with *Numeca*, and the discrepancies in pressure coefficients between the data obtained with the algorithm and *Numeca*, might be caused by three factors. The first explanation is due to the way the integral equations are solved with the algorithm. At the leading and trailing edges of the airfoil (i.e. these locations correspond to the lower and upper limits of the integrals in Chapter 3.3), singularities were occurring when solving these integrals. Slightly changing the lower limit in the algorithm from $x = -0.99$ to $x = -1$ solved this problem. This also explains the bigger discrepancy at the leading edge. Another factor causing the discrepancies in the pressure coefficients might be due to the difference in methodology. *Numeca* solves the Euler equation, which is the inviscid form of the Navier-Stokes equations. However, the present study solves the TSD equation to compute the pressure coefficients. This equation is derived by assuming irrotational flow, assuming that perturbation velocities are small, and by relating the local speed of sound to the freestream value by making use of the isentropic relations. These simplifications cause the results to become less accurate, which explains the discrepancies in the pressure coefficients at the leading and trailing edges of the airfoil computed with the algorithm and *Numeca*. This last factor also explains the discrepancy in the magnitude of the minimum pressure coefficient, which differs with 20%. The assumption of irrotationality results in a reduction of accuracy, which causes the equality in pressure coefficients at the leading and trailing edges (since no other factors have been taken into account in this case). Another factor causing the discrepancy at the leading edge might arise from thin airfoil considerations in the formulation of the boundary conditions. The surface tangency condition is applied on the profile mean chord rather than on the actual surface. The third factor could be the mesh resolution, which is different for the algorithm and *Numeca*. Although the number of mesh cells of both applications is almost the same (87 vs. 80 for the algorithm and *Numeca*, respectively), *Numeca* uses advanced cells and adapts the mesh to the geometry. The algorithm however, assumes the airfoil to be a flat plate and creates a mesh around that plate.

Before observing Figure 6.2, a typo has been identified along the $\frac{x}{c}$ axis. Here, $\frac{x}{c} = 0.8$ has been mentioned twice.The most right one should be $\frac{x}{c} = 0.9$. Also, the scale of the x-axis is not consistent with the scale of the x-axis of the results generated with the algorithm, as can be seen in Chapter 5 (i.e.$0 < x < 1$ for Lamah's results vs. $-1 < x < 1$ for the results generated with the algorithm). When mentioning the values along the x-axis in Figure 6.2b, the converted values to the scaling system of the algorithm will be given. Another remark is regarding the orientation of the C_P-axis. It seems that Lamah has flipped the pressure coefficient distributions, without flipping the axis itself (while should have been done). As a result, the C_P values resulting from Figure 6.2b should be multiplied by negative unity in order to obtain a reasonable value.
When observing this figure, it is noticed that the shapes of the pressure coefficients resulting from the three sets of data are in good agreement. However, both the magnitudes and the locations of $C_{P_{LE}}$, $C_{P_{min}}$, and $C_{P_{TE}}$ computed with the algorithm differ from Lamah's results and the results generated with *Numeca*. Comparing the pressure coefficients resulting from the algorithm to Lamah's pressure coefficient distributions, it is observed that the disagreements are considerably large. The biggest discrepancies regarding the magnitude of the pressure coefficient can be found at the leading edge along the upper and lower sides of the airfoil (300% and 56% respectively) and at the trailing edge (64%). The disagreements of the minimum pressure coefficients are less severe (23% vs. 14% along the lower and upper sides respectively). These large discrepancies might be caused by the differences in the mesh used between the algorithm and Lamah's code, in terms of the number of points on the x-axis, z-axis, and the body. Lamah's work does not provide the reader with this information. Also, Lamah uses a semi-chord of $b = 0.5$ in his calculations, while the upper and lower limits of the integrals used range from -1 to 1. The combination of these differences might cause the pressure coefficients to differ substantially.
The data of the algorithm and *Numeca* show a much better agreement. The biggest discrepancies regarding the magnitude of the pressure coefficients can again be found at the leading edge along the upper side of the airfoil ($\Delta C_{P_{u_{LE}}} \approx 500\%$) and at the trailing edge ($\Delta C_{P_{TE}} \approx 36\%$). The disagreement regarding the location of these pressure coefficients is negligibly small. The pressure coefficients at the leading edge along the lower side of the airfoil, and the minimum pressure coefficients along both sides of the airfoil, differ in a less extent and with approximately the same amount ($\Delta C_{P_{l_{LE}}} \approx 25\%$, $\Delta C_{P_{u_{min}}} \approx 18\%$, $\Delta C_{P_{l_{min}}} \approx 18\%$). The reason of these disagreements is most likely due to the way the algorithm deals with the singularities, and due to the differences in methodology.

Observing Figure 6.3 results in the conclusion that the disagreements between the data of the algorithm, Lamah and *Numeca* are less pronounced. The biggest discrepancies are between the algorithm and Lamah's data, and in particular at the leading edge along both sides of the airfoil and at the trailing edge ($\Delta C_{P_{l_{LE}}} \approx 53\%$, $\Delta C_{P_{u_{LE}}} \approx 35\%$, and $C_{P_{TE}} \approx 64\%$). The minimum pressure coefficients differ less ($C_{P_{l_{min}}} \approx 11\%$ and $C_{P_{u_{min}}} \approx 13\%$). The locations of these pressure coefficients also vary, but to a lesser extent compared to the disagreements in their magnitudes. The explanation of these discrepancies is again the same as for $\alpha = 0.5°$, and therefore will not be repeated here.
Observing the data of algorithm and *Numeca*, it is again evident that these data are in better agreement. The most notable difference occurs along the upper and lower sides at the leading edge ($C_{P_{u_{LE}}} \approx 54\%$ and $C_{P_{l_{LE}}} \approx 54\%$). The differences between the remaining pressure differences range from 16% to 32%. The factors causing these differences are again the way the algorithm deals with the singularities and the differences in methodology.

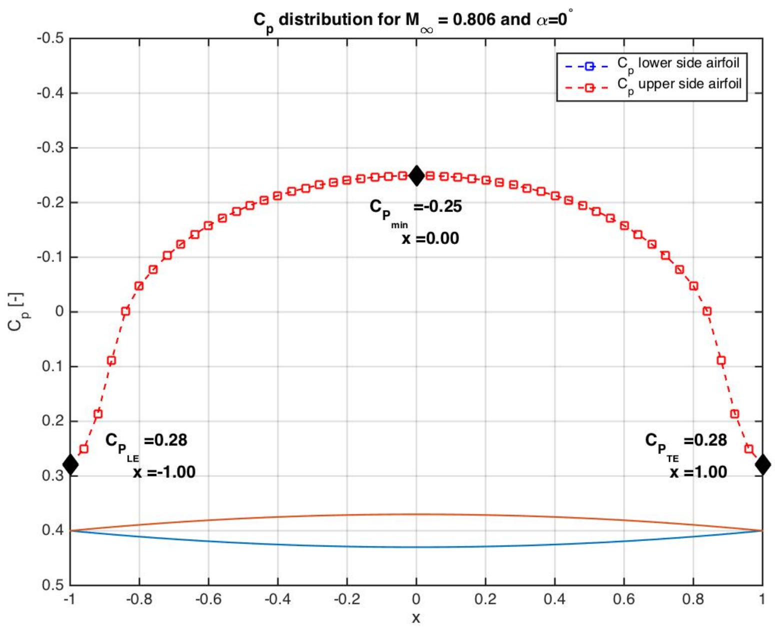
			Lower Side								Upper Side											
			$M_\infty = 0.806$				$M_\infty = 0.86$				$M_\infty = 0.806$						$M_\infty = 0.86$					
			$C_{P_{LE}}$	x	$C_{P_{min}}$	x	$C_{P_{LE}}$	x	$C_{P_{min}}$	x	$C_{P_{LE}}$	x	$C_{P_{min}}$	x	$C_{P_{TE}}$	x	$C_{P_{LE}}$	x	$C_{P_{min}}$	x	$C_{P_{TE}}$	x
α	0	*Algorithm*	-	-	-	-	-	-	-	-	0.29	-1.00	-0.25	0.00	0.29	1.00	0.30	-1.00	-0.28	0.00	0.30	1.00
		Numeca	-	-	-	-	-	-	-	-	0.45	-1.00	-0.30	0.00	0.37	1.00	0.50	-1.00	-0.42	0.23	0.40	1.00
	0.5	*Algorithm*	0.56	-1.00	-0.22	0.08	0.61	-1.00	-0.25	0.08	0.05	-0.96	-0.28	-0.08	0.29	1.00	0.08	-0.92	-0.32	-0.08	0.30	1.00
		Lamah	0.24	-0.91	-0.27	0.13	-	-	-	-	-0.03	-0.91	-0.32	-0.16	0.10	0.91	-	-	-	-	-	-
		Numeca	0.76	-0.98	-0.26	0.08	0.76	-0.98	-0.33	0.14	0.06	-0.94	-0.33	-0.06	0.38	1.00	0.10	-0.94	-0.58	0.43	0.40	1.00
	1	*Algorithm*	0.82	-1.00	-0.19	0.12	0.92	-1.00	-0.22	0.12	-0.10	-0.92	-0.31	-0.16	0.29	1.00	-0.09	-0.92	-0.36	-0.20	0.30	1.00
		Lamah	0.38	-0.91	-0.21	0.14	-	-	-	-	-0.17	-0.91	-0.35	-0.18	0.10	0.91	-	-	-	-	-	-
		Numeca	0.97	-0.98	-0.22	0.13	0.97	-0.98	-0.27	0.16	-0.14	-0.94	-0.40	-0.10	0.37	1.00	-0.09	-0.94	-0.75	0.53	0.40	1.00

Table 6.1: Pressure coefficients at the leading and trailing edges and the suction peak for the angles of attack $\alpha = 0°$, $\alpha = 0.5°$, $\alpha = 1°$ and freestream Mach numbers $M_\infty = 0.806$ and $M_\infty = 0.86$ resulting from the algorithm, Lamah, and *Numeca*.

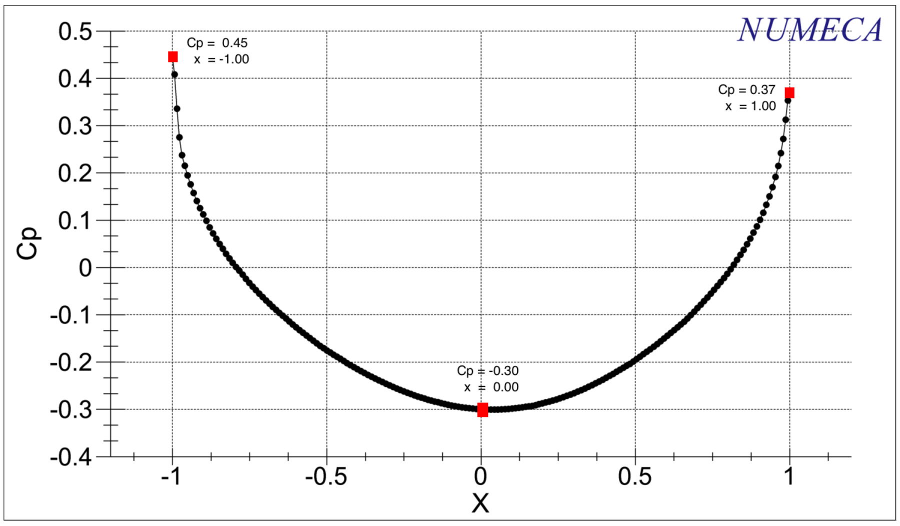
$M_\infty = 0.86$

Starting with the assessment of Figure 6.4, it is observed that the biggest discrepancies in the magnitude of the pressure coefficients between the results of the algorithm and *Numeca* occur at the leading edge of the airfoil ($\Delta C_{P_{LE}} \approx 72\%$). Moving further downstream the differences in the magnitude of the pressure differences become smaller, with a difference of $\Delta C_{P_{min}} \approx 50\%$ for the minimum pressure coefficient and $\Delta C_{P_{TE}} \approx 38\%$ for the pressure coefficient at the trailing edge. The location where $C_{P_{min}}$ occurs also differs with $\Delta x \approx 23\%$.
It is remarkable though, that the pressure coefficients at the leading and trailing edges generated with *Numeca* differ in value. The pressure coefficients at the leading and trailing edges, resulting from the algorithm, have exactly the same values. This is as expected, since the airfoil under consideration is a symmetrical parabolic arc airfoil under zero angle of attack. The inequality of the pressure coefficients at the leading and trailing edges, generated with *Numeca*, might originate from the difference in the method used for computing the pressure coefficients. *Numeca* solves the Euler equations, which are less simplified than the TSD equations used for the algorithm. The assumptions of irrotationality and small perturbations don't hold in this case, which might be the cause of the difference in pressure coefficients at the leading and trailing edges of the airfoil.
Another striking difference between the data generated with the algorithm and *Numeca*, is that a shock wave is clearly present in the data generated with *Numeca*, while no shock wave can be seen in the pressure coefficient distribution generated with the algorithm. This originates from the fact that the algorithm does not include the integral for the shock wave contribution. Adding a separate shock wave contribution integral to the perturbation velocity potential within the algorithm will solve this problem. However, in order to do so the location of the shock wave has to be predicted (without using an external tool such as *CFD* tools). Moving to Figure 6.5, one can see that along the upper side of the airfoil the biggest discrepancy in the magnitude of the pressure coefficient holds for the minimum pressure coefficient ($C_{P_{LE}} \approx 250\%$). The discrepancies of the minimum pressure coefficient and at the trailing edges are less pronounced, with difference of $\Delta C_{P_{LE}} \approx 81\%$ and $\Delta C_{P_{TE}} \approx 38\%$. The locations where $C_{P_{min}}$ occurs differ for both methods as well. Another significant difference between the two data is that the data generated with *Numeca* show a shock wave, while the data generated with the algorithm does not.
The lower side of the airfoil shows differences too, with the biggest discrepancy being the magnitudes of $C_{P_{min}}$ (32%). The difference in the magnitudes of $C_{P_{LE}}$ is less pronounced, with a difference of $\Delta C_{P_{LE}} \approx 27\%$. The difference in the location of the minimum pressure coefficient is determined to be $\Delta x \approx 75\%$.

The differences in pressure coefficients along the upper side of the airfoil in Figure 6.6 show the same trend as in Figure 6.5. The biggest discrepancy occurs between the magnitudes of the minimum pressure coefficient, being $\Delta C_{P_{min}} \approx 108\%$. The pressure coefficient at the trailing edge is less pronounced, and differs with $\Delta C_{P_{TE}} \approx 38\%$. However, the value of the pressure coefficient at the leading edge is exactly the same for both methods. The location of these pressure coefficients resulting from the algorithm and *Numeca* differs as well, with the biggest discrepancy being the location of $C_{P_{min}}$. Additionally, the appearance of a shock wave can be observed in the pressure coefficient distribution generated with *Numeca*, while the data generated with the algorithm does not show any trail of the shock wave.
However, the data along the lower side of the airfoil show better agreement. The biggest discrepancy observed is between the magnitude of the pressure coefficients at the leading edge($\Delta C_{P_{LE}} \approx 54\%$). The minimum pressure coefficients differ less, i.e. a difference of $\Delta C_{P_{min}} \approx 23\%$. The locations of the minimum pressure coefficients show bigger discrepancies, with a difference of $\Delta x \approx 33\%$.

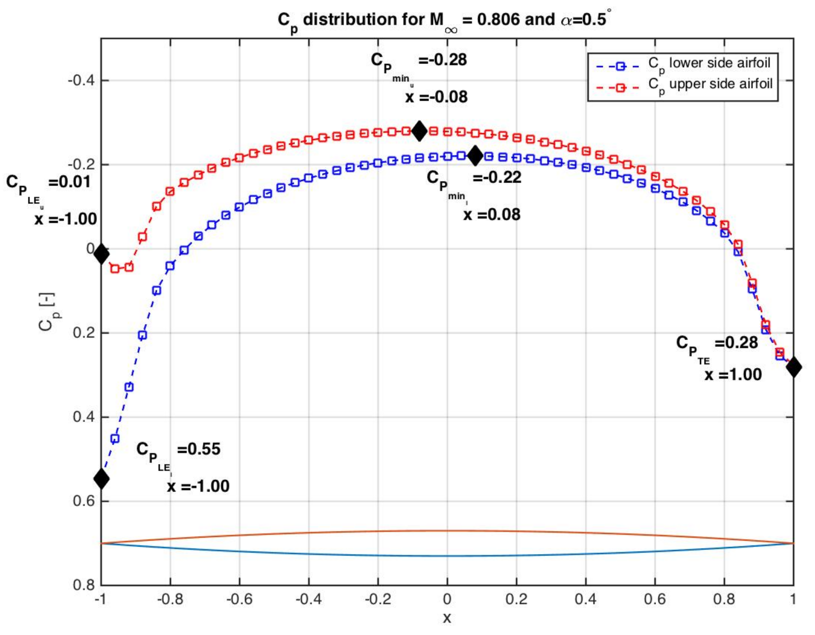


(a) *Algorithm*

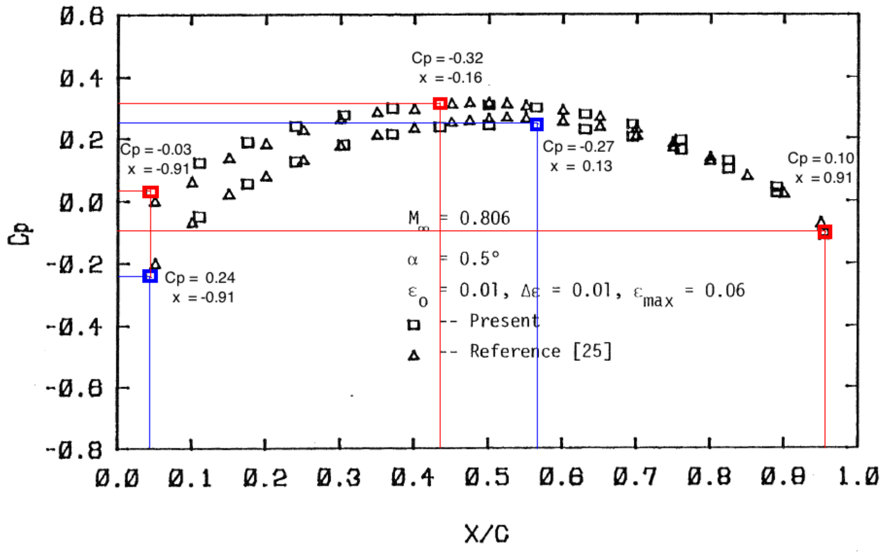


(b) *Numeca*

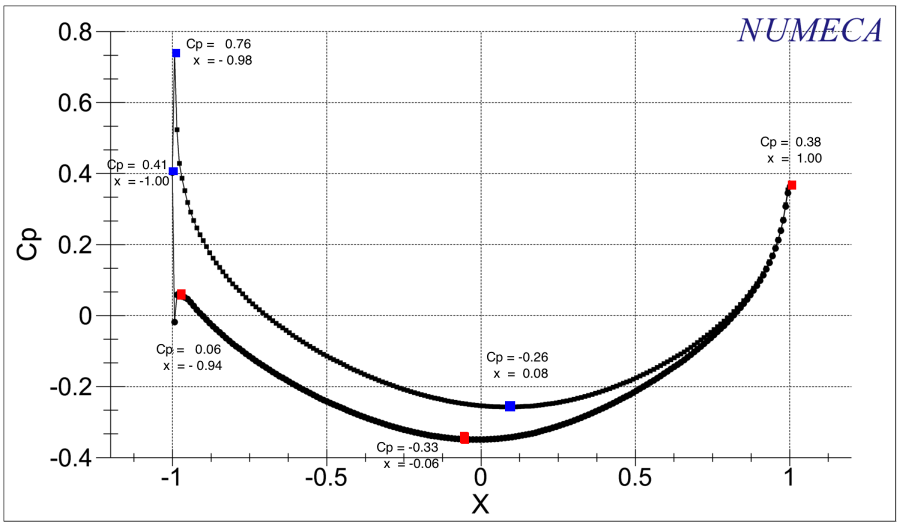
Figure 6.1: Comparison of the C_P distributions generated with the algorithm and *Numeca*; $\alpha = 0°$, $M_\infty = 0.806$



(a) *Algorithm*

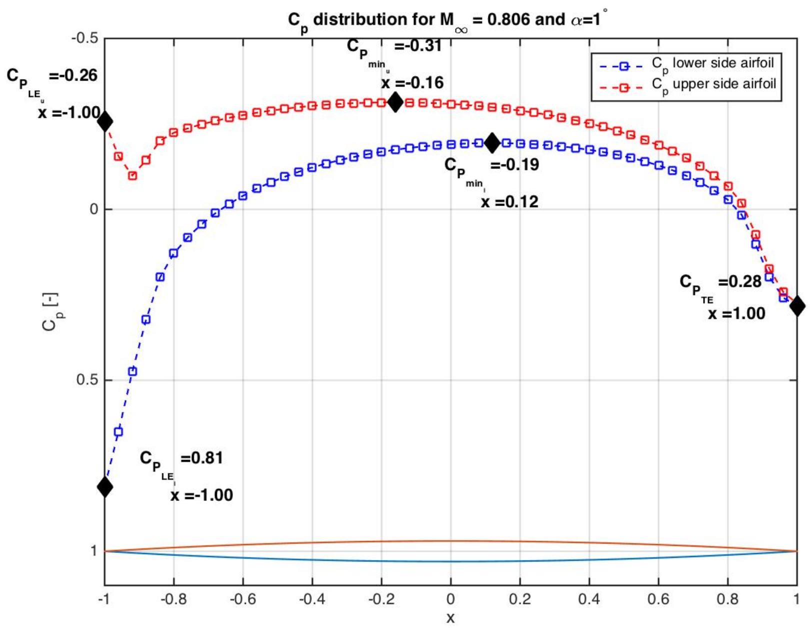


(b) Lamah

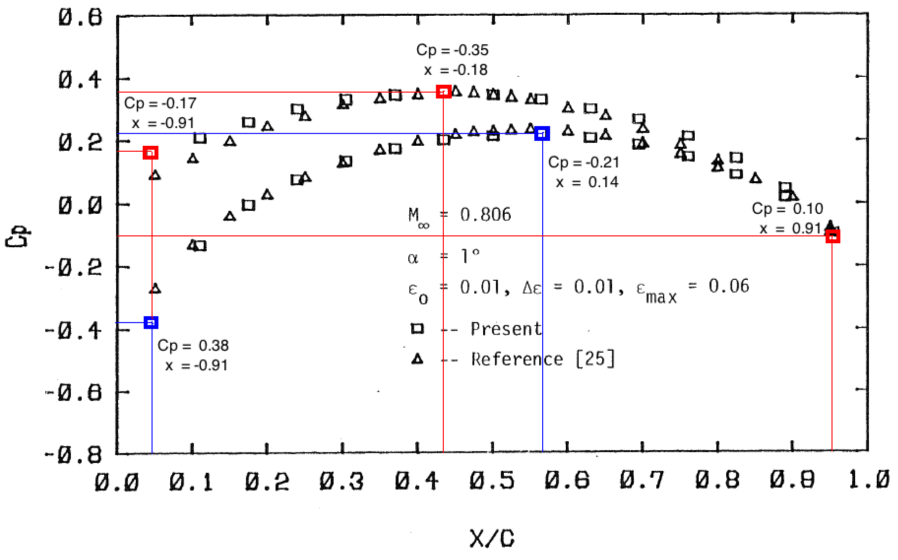


(c) *Numeca*

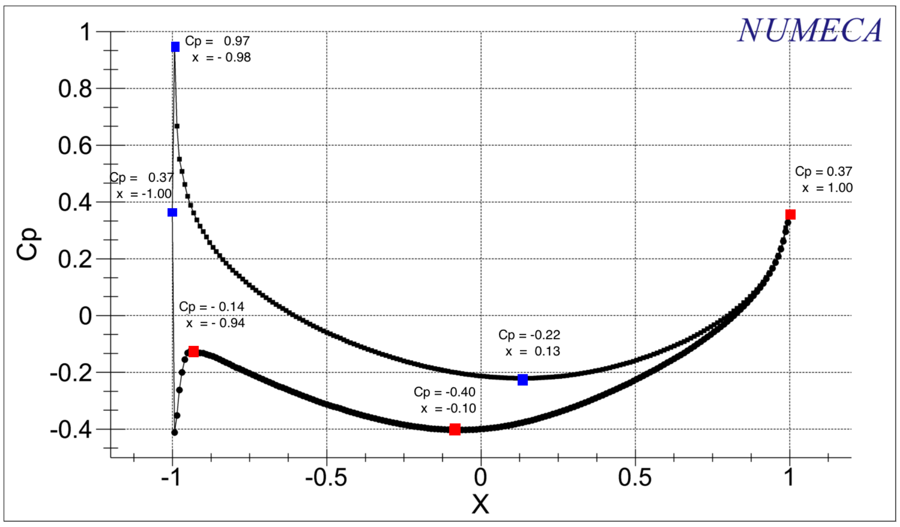
Figure 6.2: Comparison of the Cp distributions generated with the algorithm, *Numeca*, and Lamah's results; $\alpha = 0.5°$, $M_\infty = 0.806$



(a) *Algorithm*

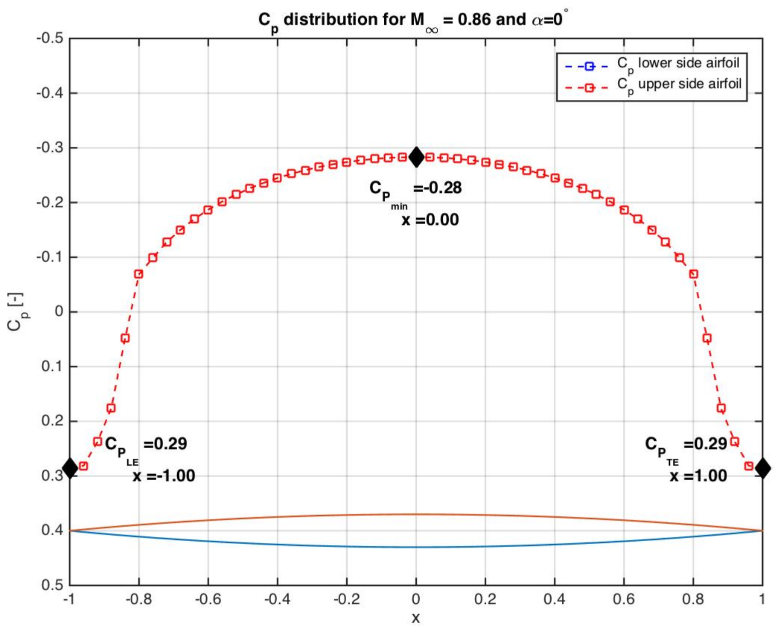


(b) Lamah

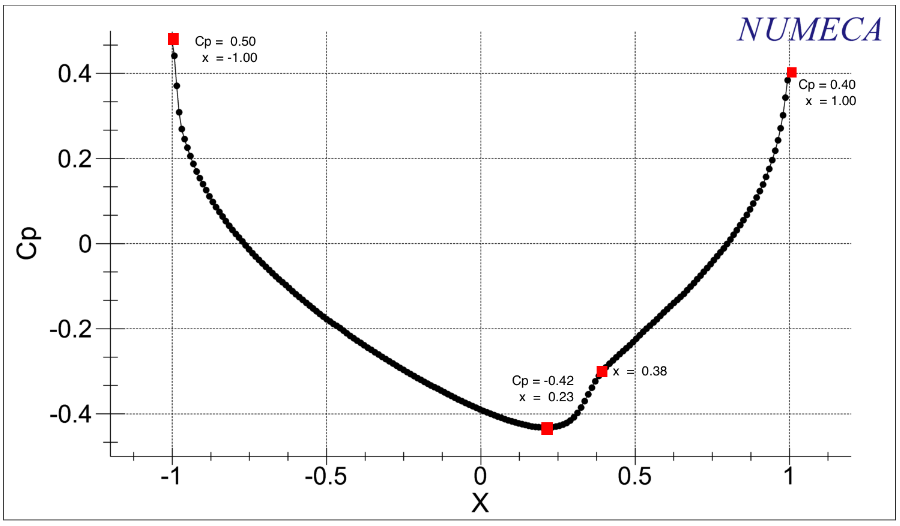


(c) *Numeca*

Figure 6.3: Comparison of the C_P distributions generated with the algorithm, *Numeca*, and Lamah's results; $\alpha = 1°$, $M_\infty = 0.806$

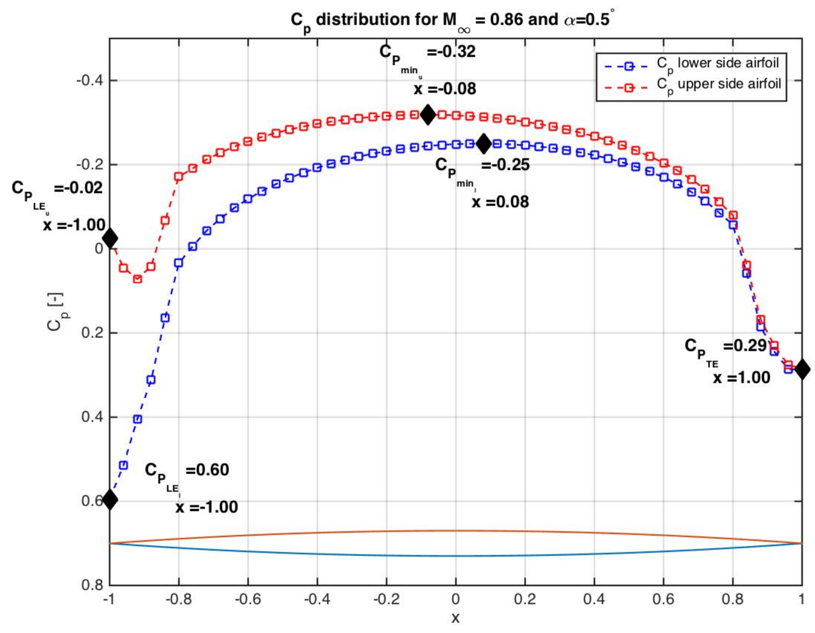


(a) *Algorithm*

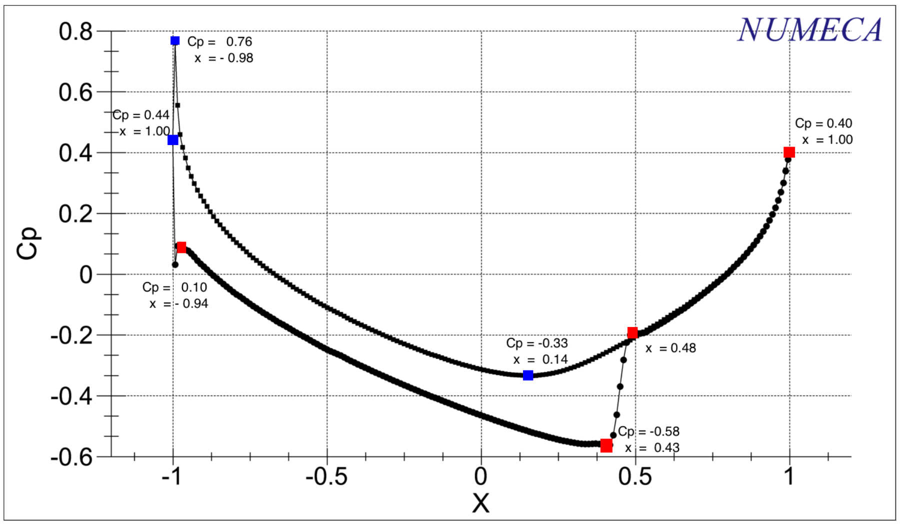


(b) *Numeca*

Figure 6.4: Comparison of the C_P distributions generated with the algorithm and *Numeca*; $\alpha = 0°$, $M_\infty = 0.86$

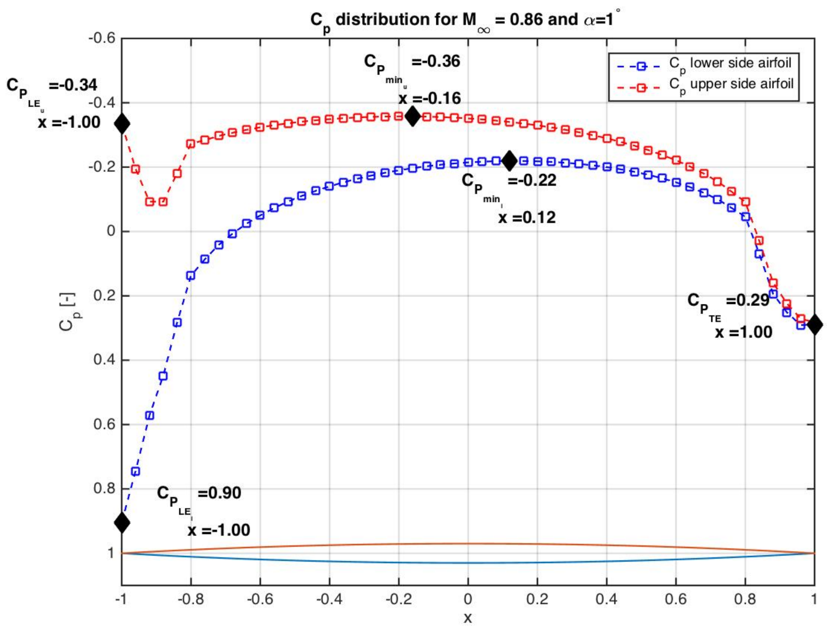


(a) *Algorithm*

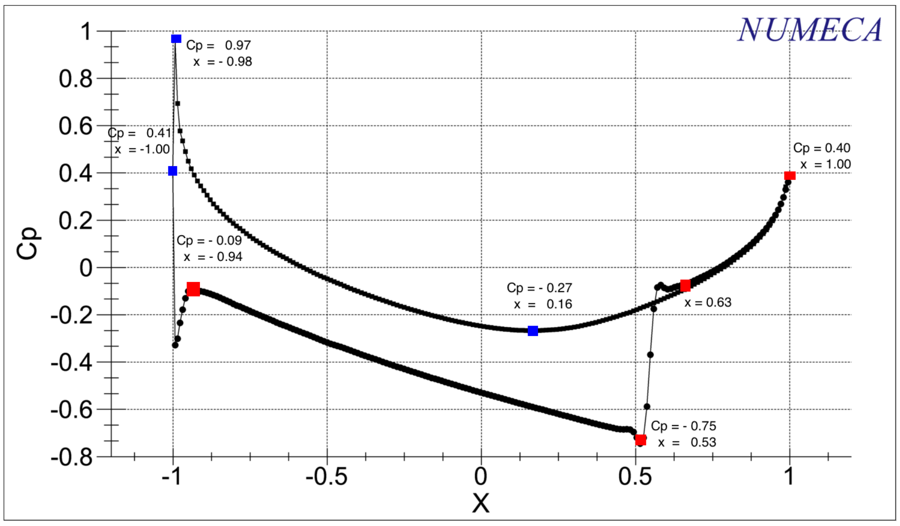


(b) *Numeca*

Figure 6.5: Comparison of the C_P distributions generated with the algorithm and *Numeca*; $\alpha = 0.5°$, $M_\infty = 0.86$



(a) *Algorithm*



(b) *Numeca*

Figure 6.6: Comparison of the C_P distributions generated with the algorithm and *Numeca*; $\alpha = 1°$, $M_\infty = 0.86$

SUMMARY

After comparing the data of the algorithm to Lamah's data and the data generated with *Numeca*, few conclusions are drawn. The most striking observation was that Lamah's data differ more from the data resulting from the algorithm, while both methods solve the TSD equations. These discrepancies are expected to arise due to the way the equations are implemented in the corresponding programming languages. Lamah does not provide the reader with information regarding the limits of the integrals, but the algorithm developed for this study has changed the lower limit of the integral equations from $x = -1$ to $x = -0.99$ because of occurring singularities. This also explains the bigger discrepancy at the leading edge, rather than the minimum pressure coefficient and the pressure coefficient at the trailing edge. Another factor which might be causing the discrepancies are the iterations. Lamah only provides information about the number of iterations for updating the perturbation velocity u with increments in the thickness ratio $\Delta\epsilon$. Starting with an initial thickness ratio of $\epsilon_0 = 0.01$, the thickness has been increased to $\epsilon = 0.06$ within six iterations (i.e. $\Delta\epsilon = 0.01$). However, nothing is stated about whether this is the optimized number of iterations. Also, no information is provided at all about the convergence of the x-derivative of the perturbation velocity potential g_x. The algorithm developed for this study consist of an inner-loop and outer-loop. The inner-loop, in which g_x is being computed until it reaches a converged value, consist of $n = 10$ iterations. The outer-loop, in which the perturbation velocity is being updated, consists of $m = 24$ iterations (i.e. $\epsilon_0 = 0.00125$, $\Delta\epsilon = 0.00125$, $\epsilon_{max} = 0.03$, for both the upper and lower sides of the airfoil). However, it should be noted that Lamah's scenario has also been investigated with the algorithm, but this was producing almost similar results as with $\epsilon_0 = 0.00125$ and $m = 24$. The most important difference was the computation time, which turned out to be 16 minutes. Also, Lamah does not use an under-relaxation factor, while the algorithm developed for this study does (i.e. $d = 0.1$). Another remarkable difference is that Lamah's code uses a different value for the airfoil semi-chord ($b = 0.5$) than the value used for the algorithm ($b = 1$). A difference in the number of mesh cells which is also a factor influencing the results. The algorithm uses 87 cells in both the x and z-directions, while Lamah does not provide any information regarding this.

The data obtained with $Numeca$ also differ from the data resulting from the algorithm, with the most significant difference being the appearance of the shock wave for $M_\infty = 0.86$. Shock waves have been captured in the pressure coefficient distributions resulting from $Numeca$, while the algorithm fails in showing any trail of a shock wave. This originates from the fact that the algorithm lacks the shock wave effects. Adding a separate shock wave integral contribution to the perturbation velocity potential will solve this problem. The tricky part in doing so is to predict the location of the shock wave analytically, without an external tool. Other evident discrepancies between the data resulting from the algorithm and $Numeca$ have been observed at the leading and trailing edges in particular (as it was the case for Lamah's results as well). Another remarkable difference between these two data, other than the magnitude of the pressure coefficients, is that the pressure coefficient distribution for the airfoil under zero angle of attack obtained with $Numeca$ is not symmetric, while this was the expected result. The data resulting from the algorithm do show symmetry. The different methodologies (equation sets) used for these tools are one of the reasons for the discrepancies in the two data sets. $Numeca$ solves the Euler equations, in which inviscid flow is assumed. However, the algorithm solves the TSD equations, for which irrotational, isentropic flow and small-disturbances have been assumed. In addition to this, the algorithm computes the pressure coefficient distribution along the chord of the airfoil ($z = 0$). The surface tangency condition is applied on the profile mean chord, rather than on the actual surface, due to thin airfoil considerations in the formulation of the boundary conditions. Thus, the additional assumptions due to the TSD equations, are probably causing the differences in the pressure coefficients resulting from the two tools. Also, the quality of the meshes of both tools differs very much, resulting in discrepancies in the results. But the same mesh as Numeca cannot be reproduced for the algorithm

Thus, after having compared the data of the three tools, it can be concluded that the product is built right for shock-free freestream Mach numbers (i.e. shock waves appear for freestream Mach numbers starting from $M_\infty = 0.86$ for the present study). The failure in capturing the shock waves originates due to a lack of a separate shock wave contribution integral, which is difficult to implement since the location of the shock wave has to be predicted analytically (without an external tool). The overall shapes of the pressure coefficient distributions obtained with the algorithm are in good agreement with the data resulting from Lamah's code and $Numeca$ up to a freestream Mach number of $M_\infty = 0.86$. However, there are some differences in the magnitudes of the pressure coefficients, along with the locations where the leading and trailing edge and minimum pressure coefficients occur. These differences are being introduced due to different methodologies used (Euler vs. TSD equations) with their corresponding assumptions, and the way the equations have been implemented in the associated programming languages.

6.2. Validation

By verifying the algorithm, the validation is also indirectly done. The output of the algorithm is the pressure coefficient distribution along the airfoil, which is the result aimed for. However, the modeling of the shock wave is lacking in the algorithm. Thus, the model does not include all the physics to approximate reality (for airfoils in flows with critical Mach numbers and beyond). The conclusion can be drawn that the right product has been built for airfoils in flows with freestream Mach numbers below the critical Mach number. Though, adding a shock wave contribution to the model will improve the algorithm, and would make it applicable to airfoils in flows with Mach numbers equal to the critical Mach number and beyond.

7 Conclusions & Recommendations

The integral equation method has been successfully adapted to the solution of the steady TSD equation, and applied to a symmetric, 6% thick, parabolic arc airfoil. The developed algorithm (within $Matlab$) computes three contributions to the perturbation velocity potential; thickness contribution, contribution due to lift, and non-linear contribution. The thickness contribution is a non-iterative process, while the lifting and non-linear contributions are being computed within loops. Two for-loops are created; an inner-loop and and an outer-loop. The inner-loop computes the x-derivative of the perturbation velocity potential g_x, and iterates until a converged solution is reached. The outer-loop computes the perturbation velocity for the desired airfoil thickness, by updating this velocity every iteration-step with g_x resulting from the inner-loop and incrementing the airfoil thickness ratio $\Delta\epsilon$ until the desired thickness is reached. Subsequently, the resulting perturbation velocity (which was calculated along the mean chord $z = 0$) was multiplied by the factor $-\frac{2}{b*v}$, outside the loops, to compute the pressure coefficient distribution along the mean airfoil chord. The incrementing index variables for the inner- and outer-loops are denoted by n and m, respectively. The optimal values were found by increasing/decreasing the number of iterations of both the inner- and outer-loop, until no (notable) differences in the pressure coefficient values were observed, and while an attempt was made to keep these numbers as low as possible (since it results in a faster algorithm). These values turned out to be $n = 10$ and $m = 24$. Thus, the initial thickness ratio is $\epsilon_0 = 0.000125$, and increases to $\epsilon_{max} = 0.03$ for both the upper- and lower-side of the airfoil in increments of $\Delta\epsilon = 0.000125$. Adding up the thickness ratios of both sides of the airfoil results in $\epsilon = 0.06$, i.e. a 6% thick airfoil. Also, an under-relaxation factor of $d = 0.1$ was introduced for stability purposes. This value was chosen, since it turned out that a small factor was satisfying.

With the algorithm, the pressure coefficient distributions were computed for the a symmetrical parabolic arc airfoil immersed in two different free-stream Mach numbers, i.e. $M_\infty = 0.806$ and $M_\infty = 0.86$. Opting for these Mach numbers will become apparent later on in this chapter. For each Mach number, three different angles of attack were investigated; $\alpha = 0°$, $\alpha = 0.5°$, and $\alpha = 1°$. These data were then compared to data obtained from Lamah's masters thesis [5] and the data resulting from the CFD tool $Numeca$. Since Lamah computes pressure coefficient distributions for a freestream Mach number of $M_\infty = 0.806$, it was decided to do that within this study too to compare the data with. The choice for $M_\infty = 0.86$ was made to explore the limitations of the algorithm and investigate the computational ability at Mach numbers where shock waves start to appear. This Mach number was explored by examining the shock appearance within $Numeca$.
Subsequently, after comparing the data of the algorithm to the data of Lamah and $Numeca$, it was observed that the overall shapes of the pressure coefficient distributions over the airfoil surfaces obtained with the algorithm are in good agreement with the data resulting from Lamah's code and $Numeca$. However, there are some differences in the magnitudes of the pressure coefficients, along with the locations where the leading and trailing edge and minimum pressure coefficients occur. It is striking though, that for both the data (algorithm vs. Lamah and algorithm vs. $Numeca$) the biggest discrepancies appear in the following order: leading edge pressure coefficient ($C_{P_{LE}}$), trailing edge pressure coefficient ($C_{P_{TE}}$), minimum pressure coefficient ($C_{P_{min}}$). These discrepancies in the pressure coefficients between the three data will be declared next.
The most striking observation resulting from the quality assessment is that the discrepancies between the data of the algorithm and Lamah's code are more pronounced than the data of the algorithm and $Numeca$. In the first place, this observation was found to be odd, since both the algorithm and Lamah solve the TSD

equations. However, the tools differ in the way the equations are implemented in the corresponding programming languages, which most probably is parenting for the discrepancies between the data resulting from these tools. The first difference regarding the way of implementing the equations, is regarding the way the limits of the integrals are treated. Lamah doesn't provide much information about the implementation but for the algorithm resulting from this study, the lower limits of the integrals have been changed to -0.99. Thus, the running parameter of the integrals within the algorithm run from -0.99 to 1 ($\int_{-0.99}^{1}$), rather than from -1 to 1 ($\int_{-1}^{1}$). This is done, because singularities were observed in the first place. This also explains the bigger discrepancy at the leading edge, rather than the minimum pressure coefficient and the pressure coefficient at the trailing edge. However, the fact that the trailing edge is associated with the upper limit of the integrals (which is associated with singularities), causes the discrepancies in these values to be slightly larger than in the minimum pressure coefficients. Another difference between the two tools is in the iterations. Lamah only provides information about the number of iterations for updating the perturbation velocity u with increments in the thickness ratio $\Delta\epsilon$. Starting with an initial thickness ratio of $\epsilon_0 = 0.01$, the thickness has been increased to $\epsilon = 0.06$ within six iterations (i.e. $\Delta\epsilon = 0.01$). However, nothing is stated about whether or not this is the optimized number of iterations. Also, no information is provided at all about the convergence of the x-derivative of the perturbation velocity potential g_x, so it seems like there is only one loop in Lamah's code. The algorithm developed for this study,however consist of an inner-loop and outer-loop. The inner-loop, in which g_x is being computed until it reaches a converged value, consist of $n = 10$ iterations. The outer-loop, in which the perturbation velocity is updated, consists of $m = 24$ iterations (i.e. $\epsilon_0 = 0.00125$, $\Delta\epsilon = 0.00125$, $\epsilon_{max} = 0.03$, for both the upper and lower sides of the airfoil). Also, the algorithm treats the lower and upper sides of the airfoil separately in updating g_x and u and adds these values up at the end of the loops, while little information regarding the way this is done in Lamah's code is provided. However, since Lamah's iteration index runs from $\epsilon_0 = 0.01$ to $\epsilon_{max} = 0.06$ within six iterations, it seems like the upper and lower sides were not treated separately. In addition to this, Lamah's code doesn't involve an under-relaxation factor, while the algorithm uses an under-relaxation factor of $d = 0.1$. Also Lamah's code uses a different value for the airfoil semi-chord ($b = 0.5$) than the value used in the algorithm ($b = 1$). A difference in the number of mesh cells is also a factor that influences the results. The algorithm uses 87 cells in both the x and z-directions, while Lamah does not provide any information regarding this. Also, due to the semi-analytical, and thus semi-numerical nature of the algorithm, the disturbance velocities in the flow field were calculated in addition to the values on the airfoil surface. Accordingly, it differs from the standard integral equation method in that the latter method only computes surface velocities and is less accurate. However, since little is known about the mesh generated in Lamah's code, and thus the way of computing, little can be said about whether and to which extent the algorithm is causing discrepancies between the two data sets. Thus, to conclude, the combination of these differences between the tools is most likely the origin of the discrepancies between the data. It should be noticed that these are observed differences, and that there are probably more differences which might be overlooked due to the lack of information about Lamah's code.

The data obtained with $Numeca$ also differ from the data resulting from the algorithm, with the most significant difference being the inability of the algorithm to capture shock waves. This is caused do to the fact that the shock-wave effects have been assumed to be included in the non-linear contribution integral, while after analysis it turned out that a separate shock-wave contribution integral is needed. However, in order to solve this integral, the location of the shock wave should be predicted analytically without using an external tool, which is a very challenging job. Another remarkable difference between these two data, other than the magnitude of the pressure coefficients, is that the pressure coefficient distribution for the airfoil under zero angle of attack obtained with $Numeca$ is not symmetric, which is an expected result. The data resulting from the algorithm on the other hand do show symmetry. The different methodologies (equation sets) used for these tools are one of the reasons for the discrepancies in the two data sets. $Numeca$ solves the Euler equations, in which inviscid flow is assumed. However, the algorithm solves the TSD equations, for which irrotational, isentropic flow and small-disturbances (i.e. thin airfoil considerations) have been assumed. In addition to this, the algorithm computes the pressure coefficient distribution along the chord of the airfoil ($z = 0$). The surface tangency condition is applied on the profile mean chord, rather then on the actual surface, due to thin airfoil considerations in the formulation of the boundary conditions. Thus, the additional assumptions due to the TSD equations, are probably causing the differences in the pressure coefficients resulting from the two tools. Another difference between the tools is that $Numeca$ solves PDEs, while the algorithm solves integral equations. These integral equations contain singularities at the lower limit of the integrals, due to which this limit is slightly changed to -0.99 (from -1). This difference in the equation types, and the handling of the singularities, are contributing to the discrepancies between the two data. Also, the quality of the meshes of both tools differs very much, resulting in discrepancies in the results.

Rather than the factors causing the discrepancies between the data resulting from the algorithm and the two validation tools (Lamah's code and *Numeca*), the limitations of this algorithm were also identified. The steady TSD formulation is valid for isentropic, irrotational flows in which viscous effects (flow separation in particular) are negligible. When a shock wave contribution is added, the algorithm would get restricted to flows containing weak shock waves (making it inapplicable to flows with strong shock waves). Thus, allowable freestream conditions range from incompressible ($M_\infty \approx 0$) to supersonic ($M_\infty \geq 1$), providing weak shock waves. The weak shock wave condition is approximately satisfied if the maximum normal shock Mach number never exceeds 1.3. However, these are theoretical limitations. In reality, the current algorithm doesn't provide results when M_∞ exceeds 0.92 (partly due to the way the equations have been implemented within the programming language *Matlab*). At this Mach number, the pressure coefficient values at the leading and trailing edges in particular, fluctuate and become very high. Reliable results are generated up to a freestream Mach number of $M_\infty = 0.86$. The algorithm also doesn't generate results for $M_\infty = 0$, since all terms of the governing equations and boundary conditions either become zero or infinite at this Mach number. Thus, this algorithm is valid for Mach numbers ranging from $M_\infty = 0.01$ to $M_\infty = 0.92$, and provides reasonable results for Mach numbers up till $M_\infty = 0.86$ since it fails in capturing the shock waves.
Having explored the sources of the discrepancies between the algorithm and the two validation data originating from, together with the operational limitations of the algorithm, some recommendations can be given on improving the algorithm and ways to decrease the dissimilarities between the data resulting from the algorithm and *Numeca* (on the condition that the same methodology will be used, i.e. TSD integral equations).
First, a method for improving the results at the leading edge will be treated. As mentioned before, the lower limit of the integral equations, which correspond to the leading edge of the airfoil, was slightly changed to compensate for the singularities occurring in this region. In order to improve the quality of the results in this region, the lower limit can be made smaller. In addition to this, the mesh in the region where the biggest discrepancies were spotted can be made finer in these regions. Thus in addition to the three regions in which the domain has been divided (the vicinity of the airfoil, an intermediate region, and the far field), three other regions can be introduced (leading edge, center of airfoil, and trailing edge). However, a finer mesh will result in higher computation times (CPU), meaning that a trade-off has to be made then between accuracy and speed.
Another significant improvement can be made by including a separate shock wave contribution integral to the perturbation velocity potential, instead of assuming the shock to be included in the non-linear contribution integral $\left(g_S(\xi,\zeta) = \frac{1}{2\pi} \int_\Sigma \frac{\delta g}{\delta n} \ln \left[(x-\xi)^2 + \zeta^2 \right]^{\frac{1}{2}} \mathrm{d}\Sigma \right)$. This would further expand the current predictive range of the algorithm. However, in order to be able to solve this shock-wave integral the location of the shock-wave should be predicted analytically first, which is a challenging job.
The algorithm developed for this study is applied to a symmetrical parabolic arc airfoil up to a thickness of 6%. In order to obtain a more accurate overview of the applicability and the limits of this algorithm, thicker symmetrical parabolic arc airfoils should be tested and compared to the data resulting from *Numeca*. It should also be applied to other airfoils of arbitrary shape, to test the capabilities of this algorithm in predicting pressure coefficient distributions over the airfoil surfaces. Recalling that for the present study the steady component of the unsteady TSD has been solved, the algorithm can be extended by including the unsteady component of the TSD. By solving the unsteady disturbance velocity potentials, aeroelastic analysis in the flutter critical transonic speed range can be carried out. The resulting algorithm can be used for educational purposes, as well as for stand-alone analysis.

A

Classification of Partial Differential Equations

This appendix will treat the classification of the partial differential equations (PDEs). Classification is needed to solve PDEs, since there are a lot of different types of PDEs and there is no general method providing great solutions to all of them. Every PDE has to be treated separately. Therefore the first step in doing so is to classify a PDE, since one should know the nature of the problem to be able to solve it. Before treating this classification, an introduction to PDEs will be given first. The second step is to categorize these PDEs.

Partial Differential Equations

PDEs are equations with two or more independent variables, and one or more dependent variable(s). In order to illustrate this, a function ϕ has been defined. This function depends on the variables x and z, which can be noted as $\phi(x,z)$. Here, x and$z$ are typically the independent variables, while f is the dependent variable. A general partial differential equation, expressed in the x and z coordinates, can be formulated as:

$$a\frac{\delta^2\phi}{\delta x^2}+b\frac{\delta^2\phi}{\delta x\delta z}+c\frac{\delta^2\phi}{\delta z^2}+d\frac{\delta\phi}{\delta x}+e\frac{\delta\phi}{\delta z}+f\phi+g=0 \tag{A.1}$$

where a, b, c, d, e, f, and g are coefficients. Although Equation A.1 is a second-order equation, the classification which will be given below will be valid for higher-order PDEs too.

Classification

This section will treat the classification of PDEs, which can be done in several ways. The methods that will be discussed here are the categorizations based on linearity and the roots of the PDEs, since this is what makes treating the transonic equations complex. First the linearity of equations of the form given by Equation A.1 will be discussed, after which they will be divided in three categories based on the coefficients a, b, and c.

Linear vs. Non-linear

Equation A.1 will be classified based on linearity in this section. A sufficient, but necessary condition for a PDE to be linear is that the unknown function ϕ and its derivatives appear to the power of one. Another condition is regarding the operators involved in the unknown function ϕ. If this function has a non-linear operator, such as a sinus function (i.e., $sin(\phi)$), the PDE itself becomes non-linear too. As an example, the unsteady transonic small disturbance equation (that will be solved in this report for the prediction of the aerodynamic loading on the airfoil) will be treated. This equation will be repeated here again:

$$\left[1-M_\infty{}^2-M_\infty{}^2(\gamma+1)\frac{\delta\phi}{\delta x}\right]\frac{\delta^2\phi}{\delta x^2}+\frac{\delta^2\phi}{\delta z^2}-2M_\infty{}^2\frac{\delta^2\phi}{\delta x\delta t}-M_\infty{}^2\frac{\delta^2\phi}{\delta t^2}=0$$

where the bars have been removed for simplicity. It is worth mentioning that the unknown function in this equation is again ϕ, but it depends on three variables (i.e., x, z, and t). This unsteady transonic small disturbance equation turns out to be non-linear, because of the first term (indicated by red). Here, the first

derivative of ϕ with respect to x is being multiplied by the second derivative of ϕ with respect to x. Without the first derivative of ϕ, this function would be linear since all terms other than ϕ are known values.
Another example is the steady component of the unsteady transonic small disturbance equation, which is given by Equation 3.4a. This equation is formulated as follows:

$$(1-{M_\infty}^2)\frac{\delta^2\phi_0}{\delta x^2}+\frac{\delta^2\phi_0}{\delta z^2}={M_\infty}^2(\gamma+1)\frac{\delta}{\delta x}\left(\frac{1}{2}\left(\frac{\delta^2\phi_0}{\delta x^2}\right)^2\right)$$

It turns out that this equation is non-linear as well. The term $\left(\frac{\delta^2\phi_0}{\delta x^2}\right)^2$ (indicated in red) is not a first-power, and causes this equation to become non-linear.

ELLIPTIC, PARABOLIC, AND HYPERBOLIC

This section treats the second way of categorizing PDEs as given by Equation A.1, which is based on the coefficients a, b, and c. In studying higher-order equations, it has been shown that solutions of equations of the form of Equation A.1 have different properties depending on the coefficients of the highest order terms (i.e., second-order terms in Equation A.1 and the corresponding coefficients a, b, and c). This means that Equation A.1 has a solution based on the Auxillary Equation A.2. The roots, and hence the number of real characteristics of this equation, can be determined as follows:

$$am^2+bm+c=0 \tag{A.2}$$

$$m=\frac{-b\pm\sqrt{b^2-4ac}}{2a} \tag{A.3}$$

The classification is performed, based on the term $\left(b^2-4ac\right)$, and can be divided in three categories:

Elliptic	:	$\left(b^2-4ac\right)<0$
Parabolic	:	$\left(b^2-4ac\right)=0$
Hyperbolic	:	$\left(b^2-4ac\right)>0$

The Transonic Small Disturbance (TSD) equation, which will be solved in this report to predict the loading on the parabolic arc airfoil in consideration, has been derived from the Navier-Stokes (NS) equations as can be seen in Chapter 3. The reader is referred to [13] for a detailed overview of this derivation. Since the computation of aerodynamic loading starts from the NS equations, the mathematical classification of the flow will be done based on these equations. The NS equations embrace the time-dependent continuity equation (conservation of mass), time-dependent momentum equation in three directions (spatial coordinates x,y,z), and the time-dependent energy equation. Those equations are a set of coupled differential equations and describe how the velocity, pressure, temperature, and density of a moving fluid are related. The three-dimensional unsteady form of the NS Equations are outlined below.

Continuity:
$$\frac{\delta\rho}{\delta t}+\frac{\delta(\rho u)}{\delta x}+\frac{\delta(\rho v)}{\delta y}+\frac{\delta(\rho w)}{\delta z}=0$$

X-momentum:
$$\underbrace{\frac{\delta(\rho u)}{\delta t}+\frac{\delta(\rho u^2)}{\delta x}+\frac{\delta(\rho uv)}{\delta y}+\frac{\delta(\rho uw)}{\delta z}}_{\text{convection terms}}=-\frac{\delta P}{\delta x}+\frac{1}{Re}\underbrace{\left(\frac{\delta\tau_{xx}}{\delta x}+\frac{\delta\tau_{xy}}{\delta y}+\frac{\delta\tau_{xz}}{\delta z}\right)}_{\text{diffusion terms}}$$

Y-momentum:
$$\underbrace{\frac{\delta(\rho v)}{\delta t}+\frac{\delta(\rho uv)}{\delta x}+\frac{\delta(\rho v^2)}{\delta y}+\frac{\delta(\rho vw)}{\delta z}}_{\text{convection terms}}=-\frac{\delta P}{\delta y}+\frac{1}{Re}\underbrace{\left(\frac{\delta\tau_{xy}}{\delta x}+\frac{\delta\tau_{yy}}{\delta y}+\frac{\delta\tau_{yz}}{\delta z}\right)}_{\text{diffusion terms}}$$

Z-momentum:
$$\underbrace{\frac{\delta(\rho w)}{\delta t}+\frac{\delta(\rho uw)}{\delta x}+\frac{\delta(\rho vw)}{\delta y}+\frac{\delta(\rho w^2)}{\delta z}}_{\text{convection terms}}=-\frac{\delta P}{\delta z}+\frac{1}{Re}\underbrace{\left(\frac{\delta\tau_{xz}}{\delta x}+\frac{\delta\tau_{yz}}{\delta y}+\frac{\delta\tau_{zz}}{\delta z}\right)}_{\text{diffusion terms}}$$

Energy:
$$\frac{\delta(E)}{\delta t}+\frac{\delta(uE)}{\delta x}+\frac{\delta(vE)}{\delta y}+\frac{\delta(wE)}{\delta z}=-\frac{\delta(uP)}{\delta x}-\frac{\delta(vP)}{\delta y}-\frac{\delta(wP)}{\delta z}-\frac{1}{RePr}\left(\frac{\delta q_x}{\delta x}+\frac{\delta q_y}{\delta y}+\frac{\delta q_z}{\delta z}\right)$$
$$+\frac{1}{Re}\left(\frac{\delta}{\delta x}(u\tau_{xx}+v\tau_{xy}+w\tau_{xz})+\frac{\delta}{\delta y}(u\tau_{xy}+v\tau_{yy}+w\tau_{yz})+\frac{\delta}{\delta z}(u\tau_{xz}+v\tau_{yz}+w\tau_{zz})\right)$$

where

u	=	velocity in x-direction
v	=	velocity in y-direction
w	=	velocity in z-direction
P	=	pressure
ρ	=	density
Re	=	Reynolds number
Pr	=	Prandtl number
q_x,q_y, q_z	=	heat flux components
E	=	total energy
τ_{xx},τ_{xy},τ_{xz},τ_{yy},τ_{yz},τ_{zz}	=	stress tensor components

The variables given above can be divided into independent and dependent variables. The independent variables consist of the spatial coordinates x, y, and z and the time t. The dependent variables, however, consist of P, ρ, T (contained through E), u, v, and w. All six dependent variables are functions of the four independent variables, which makes the differential equations given above PDEs. One can see that the momentum equations consist of convective and diffusion terms. Those terms represent physical processes that occur in a flow of gas in which some property is transported by the ordered motion of the flow and random motion of the molecules of the gas, respectively. It is the diffusion in the flow that results in the generation of boundary layers and turbulence. However, since the research treated in this report considers inviscid flow, those terms disappear and won't be elaborated on.

The categorization of PDEs in elliptic, parabolic, and hyperbolic equations can also be applied to the NS equations. The steady and unsteady NS equations are formally elliptic and parabolic, respectively [14]. However, the classification of inviscid equations is different than the classification of the full NS equations, due to the absence of the viscous higher order terms. Therefore, this classification depends on the role of compressibility, and hence the free stream Mach number. The physical meaning of each category of the inviscid equations will be treated here.
For elliptic equations information at a certain point in the (x,z)-plane influences all other regions of the domain. Therefore, the solution at that certain point must be carried out simultaneously with the solution at all other points in the domain [15]. The solution of elliptic problems depend solely on the boundary conditions (e.g., boundary value problem). Disturbances int he interior of the solution influence all other points, and the signals of disturbances travel in all directions. This results in only smooth solutions, even though the boundary conditions are discontinuous. Consequently, the numerical solution becomes easier. The action of pressure is what causes the elliptic nature of inviscid flows at Mach numbers below 1. For flows with freestream Mach numbers below 1, disturbances can be propagated by the pressure at the speed of sound (i.e. this is greater than the speed of the flow). However, if the free stream Mach number is greater than 1 the pressure will not be able to affect events in the upstream direction, since fluid velocity is greater than the propagation speed of disturbances in this case [14].
For parabolic equations information at a certain point in the (x,z)-plane influences the entire region of the plane to one side of that certain point only. The solution to parabolic equations can thus be marched in the main flow direction [14]. For hyperbolic equations information at a certain point in the (x,z)-plane influences those regions between the advancing characteristics. The reader is referred to the concept of the method of characteristics for a more detailed description of advancing characteristics.
It was mentioned above that the classification of compressible flow depends on the parameter M_∞. This is also the case for the Transonic Small Disturbance (TSD) equation (i.e. which is the governing equation of the problem in consideration), since this equation is derived from the NS equations by assuming inviscid (i.e. no diffusion terms), irrotational flow with small perturbation velocities. For unsteady, inviscid, compressible flows the system of equations is always hyperbolic, no matter whether the flow is locally subsonic or supersonic. In case of a steady motion the classification of the system depends on the fluid speed; hyperbolic if its supersonic and elliptic if its subsonic. Therefore, if the flow is steady and subsonic in one region and supersonic elsewhere, the inviscid compressible flows can be of mixed type. Especially flows with Mach numbers close to 1 are complicated to solve, since such flows may contain shock wave discontinuities and regions of subsonic (i.e. elliptic) and supersonic (i.e. hyperbolic) flow, whose exact locations are not known beforehand. Considering the totally different mathematical behavior of elliptic and hyperbolic equations, one can imagine the difficulties facing the solution of this type of mixed equations.

B Derivation TSD and Boundary Conditions

Unsteady Transonic Small Disturbance Equation

This section is concerned with the derivation of the non-linear two-dimensional unsteady TSD equation, as given by Equation 3.1. When a body is immersed in two-dimensional, irrotational, isentropic flow perturbation velocities arise. The uniform flow has a velocity V_∞ and is oriented in the x-direction, as can be seen in Figure B.1. At an arbitrary point **P** in the flow field, the velocity is $\overrightarrow{V}$ with the x and z-components given by u and w, respectively. This velocity $\overrightarrow{V}$ can be visualized as the sum of the uniform flow velocity V_∞ plus some extra increment in velocity. Thus, the x-component of velocity u in Figure B.1 can be visualized as V_∞ plus an increment in velocity $\hat{u}$. Similarly, the z-component of velocity w can be visualized as simple increment itself, because the uniform flow has zero component in the z-direction. These increments are called perturbations, where $\hat{u}$ and $\hat{w}$ are perturbation velocities:

$$u = V_\infty + \hat{u} \tag{B.1a}$$

$$w = \hat{w} \tag{B.1b}$$

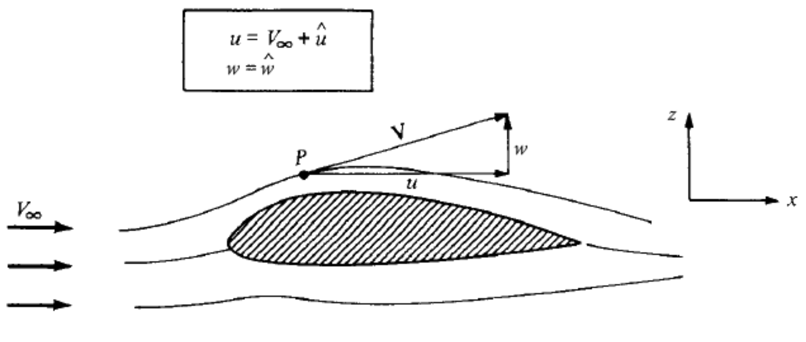


Figure B.1: Perturbation velocities induced by an airfoil immersed in a uniform flow [1]

As mentioned in Chapter 3.1, the TSD is derived by assuming irrotational flow. For irrotational flow, there exists a scalar function ϕ such that the velocity is given by the gradient of ϕ, as denoted in Equation B.3. This ϕ is denoted as the velocity potential and is a function of the spatial coordinates (i.e. $\phi = (x,z)$), whereas $\hat{\phi}$ is defined as the perturbation velocity potential, such that:

$$\phi = V_\infty x + \hat{\phi} \tag{B.2}$$

Also, due to the condition of irrotationality, and thus the concept of a potential:

$$\vec{V} = \nabla \vec{\phi} \tag{B.3}$$

where

$$\vec{V} = u\mathbf{i} + w\mathbf{k} \tag{B.4a}$$

$$\nabla\phi = \frac{\delta\phi}{\delta x}\mathbf{i} + \frac{\delta\phi}{\delta z}\mathbf{k} \tag{B.4b}$$

By substituting equations into Equation B.3, the following relations can be obtained:

$$u\mathbf{i} + w\mathbf{k} = \frac{\delta\phi}{\delta x}\mathbf{i} + \frac{\delta\phi}{\delta z}\mathbf{k} \tag{B.5a}$$

$$u = \frac{\delta\phi}{\delta x} \tag{B.5b}$$

$$w = \frac{\delta\phi}{\delta z} \tag{B.5c}$$

By assuming $V_\infty = 1$ the expressions for u and w become [16]:

$$u = \phi_x = 1 + \hat{\phi}_x \tag{B.6a}$$

$$w = \phi_z = \hat{\phi}_z \tag{B.6b}$$

From equations B.5b and B.5c it is observed that the flow-field velocities are obtained by differentiation ϕ in the same direction as the velocities. It should also be noted that the velocity potential is defined for irrotational flow only. When a flow field is irrotational, hence allowing a velocity potential to be defined, there is a tremendous simplification. Instead of dealing with the two velocity components as unknowns (u and w), there is dealt with the velocity potential ϕ as one unknown. Due to this, only one equation for the flow field is required for the solution, which makes the analysis of irrotational flows simpler than rotational flows. These type of flows, where the flow field can be described by a potential, are called potential flows. Since the present study deals with inviscid and compressible flow over a body immersed in a uniform stream, there is no mechanism to start rotating the fluid elements. Subsequently, the assumption of irrotational flow can be made.

It is important to note that the TSD will be derived with the help of the two-dimensional (specific to this study) continuity equation, which is given by:

$$\frac{\delta\rho}{\delta t} + \nabla\cdot(\underbrace{\rho\vec{V}}_{\text{perturbations to freestream mass fluxes}}) = 0$$

$$\underbrace{\frac{\delta\rho}{\delta t}}_{\text{net mass flow out of control volume}} + \underbrace{\frac{\delta(\rho u)}{\delta x} + \frac{\delta(\rho w)}{\delta z}}_{\text{time rate of change of mass inside control volume}} = 0 \tag{B.7}$$

which upon differentiation by parts becomes:

$$\frac{\delta\rho}{\delta t} + \rho\frac{\delta u}{\delta x} + u\frac{\delta\rho}{\delta x} + w\frac{\delta\rho}{\delta z} + \rho\frac{\delta w}{\delta z} = 0 \tag{B.8}$$

Substituting equations B.6 into B.8, and ignoring the hat in $\hat{\phi}$ for future computations, the continuity equation becomes:

$$\frac{\delta\rho}{\delta t} + \rho\frac{\delta^2\phi}{\delta x^2} + \frac{\delta\phi}{\delta x}\frac{\delta\rho}{\delta x} + \frac{\delta\phi}{\delta z}\frac{\delta\rho}{\delta z} + \rho\frac{\delta^2\phi}{\delta z^2} = 0$$

$$\frac{\delta\rho}{\delta t} + \rho\left(\frac{\delta^2\phi}{\delta x^2} + \frac{\delta^2\phi}{\delta z^2}\right) + \frac{\delta\phi}{\delta x}\frac{\delta\rho}{\delta x} + \frac{\delta\phi}{\delta z}\frac{\delta\rho}{\delta z} = 0 \tag{B.9}$$

Since this equation contains two unknowns (ϕ and ρ), instead of only ϕ, ρ will be eliminated next through the energy equation. The energy equation for unsteady flows can be derived with the x and z-components of the momentum equation, respectively:

$$\frac{\delta(\rho u)}{\delta t} + \nabla\cdot(\rho u \overrightarrow{V}) = -\frac{\delta P}{\delta x} + \rho f_x + (F_x)_{viscous} \tag{B.10}$$

$$\frac{\delta(\rho w)}{\delta t} + \nabla\cdot(\rho w \overrightarrow{V}) = -\frac{\delta P}{\delta z} + \rho f_z + (F_z)_{viscous} \tag{B.11}$$

In equations B.10 and B.11 the first terms on the left side of the equals signs represents the time rate of change of momentum. This change is caused by unsteady fluctuations of the flow properties inside the control volume. The second term on the left side of the equals sign depicts the net flow of momentum out of the control volume across the surface. Moving to the right side of the equals sign, the pressure force, body force, and total viscous force exerted on the control surface emerge respectively. However, in this study the body and viscous forces have been disregarded, which simplifies equations B.10 and B.11 to:

$$\frac{\delta(\rho u)}{\delta t} + \frac{\delta(\rho u^2)}{\delta x} + \frac{\delta(\rho u w)}{\delta z} = -\frac{\delta P}{\delta x} \tag{B.12}$$

$$\frac{\delta(\rho w)}{\delta t} + \frac{\delta(\rho u w)}{\delta x} + \frac{\delta(\rho w^2)}{\delta z} = -\frac{\delta P}{\delta z} \tag{B.13}$$

Upon differentiation by parts Equation B.12 turns into:

$$\begin{aligned}
\rho\frac{\delta u}{\delta t} + u\frac{\delta \rho}{\delta t} + (\rho u)\frac{\delta u}{\delta x} + u\frac{\delta(\rho u)}{\delta x} + (\rho w)\frac{\delta u}{\delta z} + u\frac{\delta(\rho w)}{\delta z} + \frac{\delta P}{\delta x} &= 0\\
\rho\frac{\delta u}{\delta t} + (\rho u)\frac{\delta u}{\delta x} + (\rho w)\frac{\delta u}{\delta z} + \frac{\delta P}{\delta x} + u\left(\underbrace{\frac{\delta \rho}{\delta t} + \frac{\delta(\rho u)}{\delta x} + \frac{\delta(\rho w)}{\delta z}}_{\text{continuity equation} = 0}\right) &= 0
\end{aligned} \tag{B.14}$$

The term between the brackets is the continuity equation, and therefore vanishes. By assuming irrotational flow ($\frac{\delta u}{\delta z} = \frac{\delta w}{\delta x}$) and rewriting the remaining terms in Equation B.14, the momentum equation transforms into:

$$\begin{aligned}
\frac{\delta u}{\delta t} + u\frac{\delta u}{\delta x} + w\frac{\delta u}{\delta z} + \frac{1}{\rho}\frac{\delta P}{\delta x} &= 0\\
\frac{\delta u}{\delta t} + \frac{\delta(\frac{1}{2}u^2)}{\delta u}\frac{\delta u}{\delta x} + \frac{\delta(\frac{1}{2}w^2)}{\delta w}\frac{\delta w}{\delta x} + \frac{1}{\rho}\frac{\delta P}{\delta x} &= 0
\end{aligned} \tag{B.15}$$

The first term in this equation can be written in terms of ϕ_x by differentiation B.5b with respect to time t:

$$\frac{\delta u}{\delta t} = \frac{\delta \phi_x}{\delta t} = \frac{\delta \phi_t}{\delta x} \tag{B.16}$$

Substituting this equation into Equation B.15 results in:

$$\frac{\delta \phi_x}{\delta t} + \frac{\delta(\frac{1}{2}u^2 + \frac{1}{2}w^2))}{\delta x} + \frac{1}{\rho}\frac{\delta P}{\delta x} = 0 \tag{B.17}$$

Also, for irrotational and inviscid flows, the following isentropic relations in a calorically perfect gas hold:

$$\frac{\rho}{\rho_\infty} = \left(\frac{P}{P_\infty}\right)^{\frac{1}{\gamma}} \tag{B.18a}$$

$$P = c\rho^\gamma \tag{B.18b}$$

where c is a constant. Using relation B.18a the third term in Equation B.17 can be rewritten as follows:

$$\rho = \rho_\infty \left(\frac{P}{P_\infty}\right)^{\frac{1}{\gamma}}$$

$$\frac{1}{\rho} = \frac{1}{\rho_\infty \left(\frac{P}{P_\infty}\right)^{\frac{1}{\gamma}}} = \frac{1}{\rho_\infty}\left(\frac{P}{P_\infty}\right)^{-\frac{1}{\gamma}} = \frac{P_\infty^{\frac{1}{\gamma}}}{\rho_\infty} P^{-\frac{1}{\gamma}} \tag{B.19}$$

Equation B.19 can be written in the $\frac{1}{\rho}\frac{\delta P}{\delta x}$ form as follows:

$$\frac{1}{\rho}\frac{\delta P}{\delta x} = \frac{P_\infty^{\frac{1}{\gamma}}}{\rho_\infty} \frac{1}{\left(-\frac{1}{\gamma}+1\right)} \frac{\delta\left(P^{-\frac{1}{\gamma}+1}\right)}{\delta x} = \frac{P_\infty^{\frac{1}{\gamma}}}{\rho_\infty}\frac{1}{\frac{\gamma-1}{\gamma}}\frac{\delta\left(P^{-\frac{1}{\gamma}+1}\right)}{\delta x} = \frac{\gamma}{\rho_\infty}\frac{P_\infty^{\frac{1}{\gamma}}}{\gamma-1}\frac{\delta\left(P^{1-\frac{1}{\gamma}}\right)}{\delta x} \tag{B.20}$$

Subsequently, substituting Equation B.20 into Equation B.17 results in:

$$\frac{\delta}{\delta x}\left[\frac{\delta\phi}{\delta t} + \frac{1}{2}\left(u^2+w^2\right) + \frac{\gamma}{\gamma-1}\frac{(P_\infty)^{\frac{1}{\gamma}}}{\rho_\infty}P^{(1-\frac{1}{\gamma})}\right] = 0 \tag{B.21}$$

$$\rho = \rho_\infty\left(\frac{P}{P_\infty}\right)^{\frac{1}{\gamma}} = \rho_\infty P^{\frac{1}{\gamma}} P_\infty^{-\frac{1}{\gamma}} = \frac{\rho_\infty}{P_\infty^{\frac{1}{\gamma}}} P^{\frac{1}{\gamma}}$$

$$\frac{\rho_\infty}{P_\infty^{\frac{1}{\gamma}}} = \frac{\rho}{P^{\frac{1}{\gamma}}}$$

$$\left(\frac{\rho_\infty}{P_\infty^{\frac{1}{\gamma}}}\right)^{-1} = \frac{P_\infty^{\frac{1}{\gamma}}}{\rho_\infty} = \frac{P^{\frac{1}{\gamma}}}{\rho} \tag{B.22a}$$

Next, this equation is substituted into Equation B.21. That is:

$$\frac{\delta}{\delta x}\left[\frac{\delta\phi}{\delta t} + \frac{1}{2}\left(u^2+w^2\right) + \frac{\gamma}{\gamma-1}\frac{P^{\frac{1}{\gamma}}}{\rho}P^{\left(1-\frac{1}{\gamma}\right)}\right] = 0$$

$$\frac{\delta}{\delta x}\left[\frac{\delta\phi}{\delta t} + \frac{1}{2}\left(u^2+w^2\right) + \frac{\gamma}{\gamma-1}\frac{P^{\left(\frac{1}{\gamma}+1-\frac{1}{\gamma}\right)}}{\rho}\right] = 0$$

$$\frac{\delta}{\delta x}\left[\frac{\delta\phi}{\delta t} + \frac{1}{2}\left(u^2+w^2\right) + \frac{\gamma}{\gamma-1}\frac{P}{\rho}\right] = 0$$

which is satisfied if the term between the brackets is an arbitrary function of z or a constant in x. That is:

$$\frac{\delta\phi}{\delta t} + \frac{u^2}{2} + \frac{w^2}{2} + \frac{\gamma}{\gamma-1}\frac{P}{\rho} = F(z) \tag{B.23}$$

where $F(z)$ is an arbitrary function of z or a constant in x. Executing similar manipulations with the z-component of the momentum equation leads to:

$$\frac{\delta\phi}{\delta t} + \frac{u^2}{2} + \frac{w^2}{2} + \frac{\gamma}{\gamma-1}\frac{P}{\rho} = F(x) \tag{B.24}$$

Here, $F(x)$ is an arbitrary function of x. Comparing equations B.23 and B.23 it can be concluded that:

$$F(z) = F(x) = C_1 \tag{B.25}$$

where C_1 is an arbitrary constant. Subsequently, the energy equation becomes:

$$\frac{\delta\phi}{\delta t} + \frac{u^2}{2} + \frac{w^2}{2} + \frac{\gamma}{\gamma-1}\frac{P}{\rho} = C_1 \tag{B.26}$$

At the freestream ϕ_t is zero, which reduces Equation B.26 to:

$$\frac{u_\infty^2}{2} + \frac{w_\infty^2}{2} + \frac{\gamma}{\gamma-1}\frac{P_\infty}{\rho_\infty} = C_1 \tag{B.27}$$

In order to eliminate the constant C_1 from equations B.26 and B.27, the following relations have been used:

$$u_\infty^2 + w_\infty^2 = V^2 = 1 \tag{B.28a}$$

$$\gamma\frac{P_\infty}{\rho_\infty} = \gamma R T_\infty = a_\infty^2 = \frac{1}{{M_\infty}^2} \tag{B.28b}$$

where the first relation in Equation B.28 results from the fact that uniform flow has zero component in z-direction (normalization). Thus, the only component is in the x-direction. This can be seen in Figure B.1. By equating equations B.26 and eq:energyequationfreestream to each other and inserting the isentropic relations given by equations B.18 and B.28b, the following equation is constructed:

$$\frac{\delta\phi}{\delta t} + \frac{u^2}{2} + \frac{w^2}{2} + \frac{1}{{M_\infty}^2(\gamma-1)}\rho^{\gamma-1} = \frac{1}{2} + \frac{1}{{M_\infty}^2(\gamma-1)} \tag{B.29}$$

which can be rewritten as:

$$\rho = \left[1 + \frac{\gamma-1}{2}{M_\infty}^2\left\{1 - 2\phi_t - u^2 - w^2\right\}\right]^{\frac{1}{\gamma-1}} \tag{B.30}$$

By inserting equations B.5b and B.5c into Equation B.30, the desired energy equation becomes:

$$\rho = \left[1 + \frac{\gamma-1}{2}{M_\infty}^2\left\{1 - 2\phi_t - \phi_x^2 - \phi_z^2\right\}\right]^{\frac{1}{\gamma-1}} \tag{B.31}$$

This energy equation in combination with Equation eq:continuitydiffbyparts, gives two equations and two unknowns (ϕ and ρ), and provides a complete set of equations for solving the unsteady potential flow problem. The density can be eliminated from the continuity equation by taking the derivative of ρ with respect to t, x, and z and subsequently substituting these into the continuity equation given by Equation B.8:

$$\frac{\delta\rho}{\delta t} = \frac{1}{\gamma-1}\left[1 + \frac{\gamma-1}{2}{M_\infty}^2\left\{1 - 2\phi_t - \phi_x^2 - \phi_z^2\right\}\right]^{\left(\frac{1}{\gamma-1}-1\right)}\left[\frac{\gamma-1}{2}{M_\infty}^2\left\{-2\phi_{tt} - 2\phi_x\phi_{xt} - 2\phi_z\phi_{zt}\right\}\right] \tag{B.32}$$

(B.33)

where the second derivatives of the potentials in the second term between brackets in Equation B.33 are obtained by:

$$\phi_x^2 = \left(\frac{\delta\phi}{\delta x}\right)^2 \rightarrow \frac{\delta}{\delta t}\left[\left(\frac{\delta\phi}{\delta x}\right)^2\right] = 2\frac{\delta\phi}{\delta x}\frac{\delta^2\phi}{\delta x\delta t} = 2\phi_x\phi_{xt}$$

$$\phi_z^2 = \left(\frac{\delta\phi}{\delta z}\right)^2 \rightarrow \frac{\delta}{\delta t}\left[\left(\frac{\delta\phi}{\delta z}\right)^2\right] = 2\frac{\delta\phi}{\delta z}\frac{\delta^2\phi}{\delta z\delta t} = 2\phi_z\phi_{zt}$$

Equation B.33 can be further simplified by observing that ρ is given by Equation B.31. Thus, the first term between brackets can be written as $\rho^{2-\gamma}$, because:

$$\left[1 + \frac{\gamma-1}{2}{M_\infty}^2\left\{1 - 2\phi_t - \phi_x^2 - \phi_z^2\right\}\right]^{-1} = \rho^{-(\gamma-1)}$$

This results in:

$$\left[1 + \frac{\gamma-1}{2}{M_\infty}^2\left\{1 - 2\phi_t - \phi_x^2 - \phi_z^2\right\}\right]^{-1} = \rho^{\left(\frac{1}{\gamma-1}-1\right)} = \rho\rho^{-(\gamma-1)} = \rho^{(1-\gamma+1)} = \rho^{(2-\gamma)} \tag{B.34}$$

Inserting Equation B.34 into Equation B.33 gives:

$$\rho_t = -\frac{1}{\gamma-1}\rho^{(2-\gamma)}\left(\gamma-1\right)M_\infty{}^2\left(\phi_{tt}+\phi_x\phi_{xt}+\phi_z\phi_{zt}\right) \tag{B.35}$$

Further simplification is possible by using equations B.5b, B.5c, B.18, and B.28b. This reduces Equation B.35 to:

$$\rho_t = -\frac{\rho}{a^2}\left(\phi_{tt}+u\phi_{xt}+w\phi_{zt}\right) \tag{B.36}$$

Similarly:

$$\rho_x = -\frac{\rho}{a^2}\left\{\phi_{xt}+u\phi_{xx}+w\phi_{xz}\right\} \tag{B.37a}$$

$$\rho_z = -\frac{\rho}{a^2}\left\{\phi_{zt}+u\phi_{xz}+w\phi_{zz}\right\} \tag{B.37b}$$

By inserting equations B.36, B.37a, and B.37b into Equation B.8, the continuity equation becomes:

$$\rho\left[-\frac{1}{a^2}\phi_{tt}-2u\frac{1}{a^2}\phi_{xt}-2w\frac{1}{a^2}\phi_{zt}-2uw\frac{1}{a^2}\phi_{xz}-u^2\frac{1}{a^2}\phi_{xx}-w^2\frac{1}{a^2}\phi_{zz}+u_x+w_z\right]=0$$

Multiplying this equation by $\left(\frac{a^2}{\rho}\right)$ and noting that $u_x = \phi_{xx}$ and $w_z = \phi_{zz}$, the following equation has been obtained:

$$\underbrace{\left(\phi_{tt}+2u\phi_{xt}+2w\phi_{zt}\right)}_{\text{unsteady part}} = \underbrace{\left(a^2-u^2\right)\phi_{xx}+\left(a^2-w^2\right)\phi_{zz}-2uw\phi_{xz}}_{\text{steady part}} \tag{B.38}$$

Equation B.38 can be further simplified by using the small disturbance assumptions. These assumptions are given by:

$$u = \phi_x \cong \mathcal{O}(1) \tag{B.39a}$$

$$w = \phi_z \ll 1 \tag{B.39b}$$

First, the steady part of Equation B.38 will be examined. By equating the energy equations given by equations B.26 and B.27, and using the isentropic relations given by Equation B.28b, a^2, $\left(a^2-u^2\right)$, $\left(a^2-w^2\right)$, and $(2uw)$ can be computed to insert into Equation B.38:

$$\phi_t + \frac{u^2}{2}+\frac{w^2}{2}+\frac{a^2}{\gamma-1} = \frac{1}{2}+\frac{1}{M_\infty{}^2(\gamma-1)} \tag{B.40}$$

Thus:

$$a^2 = -\phi_t(\gamma-1)+\frac{(\gamma-1)}{2}-\frac{u^2}{2}(\gamma-1)-\frac{w^2}{2}(\gamma-1)+\frac{1}{M_\infty{}^2} \tag{B.41}$$

Inserting equations B.6 into Equation B.41 results in:

$$a^2 = -\phi_t(\gamma-1)+\frac{(\gamma-1)}{2}-\frac{\left(1+\phi_x\right)^2}{2}(\gamma-1)-\frac{\phi_z^2}{2}(\gamma-1)+\frac{1}{M_\infty{}^2} \tag{B.42}$$

By eliminating second-degree terms, assuming low-frequency, and applying the small-disturbance assumptions given by Equation B.39 transforms Equation B.42 into:

$$a^2 \approx -\phi_t(\gamma-1)-(\gamma-1)\phi_x+\frac{1}{M_\infty{}^2} \tag{B.43}$$

Similarly:

$$\begin{aligned} a^2-u^2 &\cong \frac{1}{M_\infty{}^2}-\left(\gamma-1\right)\phi_t-\left(\gamma-1\right)\phi_x-\left(1+2\phi_x+\phi_x^2\right) \\ &\cong \frac{1}{M_\infty{}^2}+\left(\gamma+1\right)\phi_x-1 \end{aligned} \tag{B.44a}$$

$$\begin{aligned} a^2 - w^2 &\cong \frac{1}{M_\infty{}^2} - (\gamma-1)\phi_t + (\gamma-1)\phi_x - \phi_z^2 \\ &\cong \frac{1}{M_\infty{}^2} \end{aligned} \tag{B.44b}$$

$$2uw = 2(1+\phi_x)\phi_z \cong 0 \tag{B.44c}$$

Substituting these equations in the steady part of Equation B.38 transforms the right hand side of this equation, to first order accuracy, into:

$$\frac{1}{M_\infty{}^2}\left[1 - M_\infty{}^2 - (\gamma+1)M_\infty{}^2\phi_x\right]\phi_{xx} + \frac{1}{M_\infty{}^2}\phi_{zz} \tag{B.45}$$

This equation is first order accurate, because ϕ_x^2 and similar second-degree terms have been neglected in the derivation process. Continuing with the unsteady part of the continuity equation given by Equation B.38, results in:

$$\begin{aligned} \phi_{tt} + 2u\phi_{xt} + 2w\phi_{zt} &= \phi_{tt} + 2(1+\phi_x)\phi_{xt} + 2\phi_z\phi_{zt} \\ &\cong \phi_{tt} + 2(1+\phi_x)\phi_{xt} \end{aligned} \tag{B.46}$$

Subsequently, implementing the small disturbance approximations and combining equations B.45 and B.46 gives the unsteady TSD:

$$M_\infty{}^2\phi_{tt} + 2M_\infty{}^2\phi_{xt} = \left[1 - M_\infty{}^2 - M_\infty{}^2(\gamma+1)\phi_x\right]\phi_{xx} + \phi_{zz}$$

$$\left[1 - M_\infty{}^2 - M_\infty{}^2(\gamma+1)\phi_x\right]\phi_{xx} + \phi_{zz} - 2M_\infty{}^2\phi_{xt} - M_\infty{}^2\phi_{tt} = 0 \tag{B.47}$$

which is equal to Equation 3.1.

Steady Tangency Condition

The flow tangency boundary condition for a typical thin airfoil will be derived in this section. The standard tangency condition [17] can be implemented at the airfoil surface using:

$$\left(\frac{\phi_{z'}}{u}\right)^{\pm}_{z'=h'} = \left(\frac{\phi_{z'}}{\phi_{x'} + u_\infty}\right)^{\pm}_{z'=h'} = \frac{\delta h'^{\pm}}{\delta x'} \tag{B.48}$$

The subscript $z' = h'$ indicates that the boundary condition is enforced on the actual surface of the airfoil.The functions h'^{+} and h'^{-} define the upper and lower airfoil surfaces respectively. This boundary condition is converted to the small disturbance version by applying two simplifications. The first simplification involves neglecting ϕ_x relative to u_∞. Second, the flow tangency boundary condition is applied at the airfoil slit (i.e. $z' = 0$) instead of at the actual airfoil surface. By applying these simplifications to Equation B.48, the small disturbance boundary condition forms:

$$\phi'_z(x, 0^{\pm}) = u_\infty \frac{\delta h'^{\pm}}{\delta x'} \tag{B.49}$$

By rewriting u_∞ in terms of the parameters and variables given in Section 3.2 the small disturbance tangency condition is transformed into:

$$(\phi'_z)_{z'=0} = \mu h'_{x'} \tag{B.50}$$

This is equal to the tangency condition given by 3.9.

PRESSURE COMPUTATION

This section covers how the pressure is obtained from the definition of the pressure coefficient C_P, once the disturbance velocities in x-direction have been computed. Therefore, the definition of C_P is recalled first:

$$C_P \equiv \frac{P - P_\infty}{q_\infty} \tag{B.51}$$

where q_∞ is the dynamic pressure, and can be expressed in terms of M_∞ as follows:

$$q_\infty = \frac{1}{2}\rho_\infty V_\infty^2 = \frac{1}{2}\frac{\gamma P_\infty}{\gamma P_{infty}}\rho_\infty V_\infty^2 = \frac{\gamma}{2}P_\infty\left(\frac{\rho_\infty}{\gamma P_\infty}\right)V_\infty^2 \tag{B.52}$$

From the isentropic relations B.28b, the speed of sound is expressed as $a_\infty^2 = \frac{\gamma P_\infty}{\rho_\infty}$. Hence, Equation B.52 becomes:

$$q_\infty = \frac{\gamma}{2}P_\infty\frac{V_\infty^2}{a_\infty^2} = \frac{\gamma}{2}P_\infty {M_\infty}^2 \tag{B.53}$$

Substituting Equation B.53 into B.51 gives:

$$C_P = \frac{2}{\gamma {M_\infty}^2}\left(\frac{P}{P_\infty} - 1\right) \tag{B.54}$$

To obtain a linearized form of the pressure coefficient, it should be recalled that this study deals with an adiabatic flow of a calorically perfect gas; hence from a special form of the energy equation [1]:

$$T + \frac{\vec{V}^2}{2c_p} = T + \frac{V_\infty^2}{2c_p} \tag{B.55}$$

where c_p is the specific heat at constant pressure and can be expressed as [1]:

$$c_p = \frac{\gamma R}{\gamma - 1} \tag{B.56}$$

Substituting Equation B.56 into B.55 results in:

$$T - T_\infty = \frac{V_\infty^2 - \vec{V}^2}{2\gamma\frac{R}{(\gamma-1)}} \tag{B.57}$$

Also, recalling from Equation B.28b that $a_\infty = \sqrt{\gamma R T_\infty}$, Equation B.57 becomes:

$$\frac{T}{T_\infty} - 1 = \frac{\gamma - 1}{2}\frac{V_\infty^2 - \vec{V}^2}{\gamma R T_\infty} = \frac{\gamma - 1}{2}\frac{V_\infty^2 - \vec{V}^2}{a_\infty^2} \tag{B.58}$$

In terms of the perturbation velocities B.1:

$$\vec{V}^2 = (V_\infty + \hat{u})^2 + \hat{w}^2$$

Equation B.58 can be written as:

$$\frac{T}{T_\infty} = 1 - \frac{\gamma - 1}{2a_\infty^2}\left(2\hat{u}V_\infty + \hat{u}^2 + \hat{w}^2\right) \tag{B.59}$$

Since the flow is isentropic, the following relation holds:

$$\frac{P}{P_\infty} = \left(\frac{T}{T_\infty}\right)^{\frac{\gamma}{(\gamma-1)}} \tag{B.60}$$

Equation B.59 becomes:

$$\frac{P}{P_\infty} = \left[1 - \frac{\gamma - 1}{2a_\infty^2}\left(2\hat{u}V_\infty + \hat{u}^2 + \hat{w}^2\right)\right]^{\frac{\gamma}{(\gamma-1)}}$$

or

$$\frac{P}{P_\infty} = \left[1 - \frac{\gamma-1}{2a_\infty^2} {M_\infty}^2 \left(\frac{2\hat{u}}{V_\infty} + \frac{\hat{u}^2 + \hat{w}^2}{V_\infty^2}\right)\right]^{\frac{\gamma}{(\gamma-1)}} \tag{B.61}$$

When assuming small perturbations, that is:

$$\frac{\hat{u}}{V_\infty} \ll 1$$
$$\frac{\hat{u}^2}{V_\infty^2} \lll 1$$
$$\frac{\hat{w}^2}{V_\infty^2} \lll 1$$

In this case, Equation B.61 is of the form:

$$\frac{P}{P_\infty} = (1-\epsilon)^{\frac{\gamma}{(\gamma-1)}} \tag{B.63}$$

where ϵ is small. From the binomial expansion, neglecting higher-order terms, Equation B.63 becomes:

$$\frac{P}{P_\infty} = 1 - \frac{\gamma}{2} {M_\infty}^2 \left(\frac{2\hat{u}}{V_\infty} + \frac{\hat{u}^2 + \hat{w}^2}{V_\infty^2}\right) + \dots \tag{B.64}$$

Substituting Equation B.64 into the expression for the pressure coefficient, Equation B.51, results in:

$$\begin{aligned} C_P &= \frac{2}{\gamma {M_\infty}^2} \left[1 - \frac{\gamma}{2} {M_\infty}^2 \left(\frac{2\hat{u}}{V_\infty} + \frac{\hat{u}^2 + \hat{w}^2}{V_\infty^2}\right) + \dots - 1\right] \\ &= -\frac{2\hat{u}}{V_\infty} - \frac{\hat{u}^2 + \hat{w}^2}{V_\infty^2} \end{aligned} \tag{B.65}$$

Since $\frac{\hat{u}^2}{V_\infty^2} \lll 1$ and $\frac{\hat{w}^2}{V_\infty^2} \lll 1$:

$$C_P = -\frac{2\hat{u}}{V_\infty} \tag{B.66}$$

In the first section of this appendix, it was assumed that $V_\infty = 1$. Therefore:

$$C_P = -2\hat{u} \tag{B.67}$$

where $\hat{u} = \hat{\phi}_x$. Rewriting this equation in terms of the parameters and variables introduced in Chapter 3 gives:

$$C_P = -\frac{2}{\nu b} \hat{u}(x, 0^\pm) \tag{B.68}$$

Transformation of Governing Equations to ϵ-Space

Recalling the partial differential equation that governs transonic flow, as given by Equation 3.1:

$$\left[1 - {M_\infty}^2 - {M_\infty}^2(\gamma+1)\phi_x\right]\phi_{xx} + \phi_{zz} - 2{M_\infty}^2\phi_{xt} - {M_\infty}^2\phi_{tt} = 0 \tag{B.69}$$

where the bars over the spatial coordinates have been ignored for simplicity. The flow is assumed to be unsteady, inviscid, and compressible past a thin wing at small angle of attack. The wing lies in the $x-z$-plane, with the wind axis parallel to the x-axis. The thickness ratio ϵ of the wing section is used as the parameter for the parametric differentiation. The transformation of Equation B.69 to the ϵ-space with

$$g = \frac{\delta\phi}{\delta\epsilon} \tag{B.70a}$$

$$u = \frac{\delta\phi}{\delta x} \tag{B.70b}$$

results in the linear equation:

$$M_\infty{}^2\phi_{tt} + 2M_\infty{}^2\phi_{xt} - \phi_{zz} - \left(1 - M_\infty{}^2\right)\phi_{xx} = -M_\infty{}^2\left(\gamma+1\right)\phi_x\phi_{xx} \tag{B.71}$$

which upon inserting Equation B.70b becomes:

$$M_\infty{}^2\phi_{tt} + 2M_\infty{}^2\phi_{xt} - \phi_{zz} - \left(1 - M_\infty{}^2\right)\phi_{xx} = -M_\infty{}^2\left(\gamma+1\right)u\phi_{xx} \tag{B.72}$$

The $u\phi_{xx}$ term on the right hand side of Equation B.72 can be written as:

$$u\phi_{xx} = u\frac{\delta^2\phi}{\delta x^2} = u\frac{\delta}{\delta x}\left(\frac{\delta\phi}{\delta x}\right) = \frac{\delta}{\delta x}\left(u\frac{\delta\phi}{\delta x}\right) = \left(ug_x\right)_x \tag{B.73}$$

Substituting Equation B.70a into B.73 gives:

$$M_\infty{}^2 g_{tt} + 2M_\infty{}^2 g_{xt} - g_{zz} - \left(1 - M_\infty{}^2\right)g_{xx} = -M_\infty{}^2\left(\gamma+1\right)\left(ug_x\right)_x \tag{B.74}$$

It is assumed that $g(x,z,t)$ may be written as the sum of a steady component $\hat{g}(x,z)$ and an unsteady component $\tilde{g}(x,z,t)$, and $\tilde{g} \ll \hat{g}$. Consistent with the latter assumption, the velocity component parallel to the x-axis may be decomposed into a steady component $\hat{u}$ and an unsteady component $\tilde{u}$, where $\tilde{u} \ll \hat{u}$. Thus:

$$\begin{aligned} g(x,z,t) &= \hat{g}(x,z) + \tilde{g}(x,z,t) & \qquad \tilde{g} &\ll \hat{g} \\ u(x,z,t) &= \hat{u}(x,z) + \tilde{u}(x,z,t) & \qquad \tilde{u} &\ll \hat{u} \end{aligned}$$

These approximations simplify Equation B.74 to the following form:

$$\hat{g}_{zz} + \left(1 - M_\infty{}^2\right)\hat{g}_{xx} = M_\infty{}^2\left(\gamma+1\right)\left(\hat{u}\hat{g}_x\right)_x \tag{B.75}$$

$$M_\infty{}^2\tilde{g}_{tt} + 2M_\infty{}^2\tilde{g}_{xt} - \tilde{g}_{zz} - \left(1 - M_\infty{}^2\right)\tilde{g}_{xx} = -M_\infty{}^2\left(\gamma+1\right)\left(\hat{u}\tilde{g}_x\right)_x \tag{B.76}$$

These steady and unsteady equations for $\hat{g}$ and $\tilde{g}$ are weakly coupled, $\hat{u}$ being the coupling variable. The steady equation must be solved first to obtain $\hat{u}$, before solving the unsteady equation.

The focus in this study is limited to two-dimensional steady flows. Next, Green's theorem will be introduced to convert the PDEs to integral equations. Green's theorem may be written as:

$$\iint_S \left(\psi\nabla^2\Omega - \Omega\nabla^2\psi\right)\mathrm{d}S = -\int_C \left(\psi\frac{\delta\Omega}{\delta n} - \Omega\frac{\delta\psi}{\delta n}\right)\mathrm{d}C - \lim_{\sigma\to 0}\int_\sigma \left(\psi\frac{\delta\Omega}{\delta n} - \Omega\frac{\delta\psi}{\delta n}\right)\mathrm{d}C \tag{B.77}$$

where

S	=	The region bounded by a sectionally smooth curve C
σ	=	Circular cavity surrounding a point P in S
Ω	=	Function with continuous first and second derivatives in S
ψ	=	Function with continuous first and second derivatives in S, and satisfies Laplace's equation except possibly at the point P
n	=	The inward normal to C

Equation B.75 and the equation for ψ may be combined to give [11]:

$$\psi\nabla^2\hat{g} - \hat{g}\nabla^2\psi = \psi\left(\hat{u}\hat{g}_x\right)_x \tag{B.78}$$

Green's theorem B.77 can be applied to the left hand side of Equation B.78 if $\hat{g}$ corresponds to Ω, i.e.,

$$-\int_C \left(\psi\frac{\delta\hat{g}}{\delta n} - \hat{g}\frac{\delta\psi}{\delta n}\right)\mathrm{d}i - \lim_{\sigma\to 0}\left[\int_\sigma \left(\psi\frac{\delta\hat{g}}{\delta n} - \hat{g}\frac{\delta\psi}{\delta n}\right)\mathrm{d}\sigma\right] = \iint_S \psi\left(\hat{u}\hat{g}_x\right)_x \mathrm{d}S \tag{B.79}$$

The contour C consists of segments C_∞, C_W, C_B, and Σ, where

C_∞	=	contour at infinity
C_W	=	contour along wake
C_B	=	contour over the body
Σ	=	contour over the shock surface

Next, ψ will be identified with the elementary solution of $\nabla^2\psi = 0$, where

$$\psi\left(x-\xi, z-\zeta\right) = \ln\left[(x-\xi)^2 + \left(2-\zeta^2\right)^2\right]^{\frac{1}{2}} \tag{B.80}$$

and where (ξ,ζ) is the observation point. The integral equation for $\hat{g}\left(\xi,\zeta\right)$ takes the form [18]

$$\hat{g}\left(\xi,\zeta\right) = \hat{g}_B + \frac{1}{2\pi}\int_\Sigma \psi\frac{\delta\hat{g}}{\delta n}\mathrm{d}\Sigma + \frac{1}{2\pi}\int_\Sigma \psi\hat{u}\hat{g}_x\mathrm{d}z - \frac{1}{2\pi}\iint_S \psi_x\hat{u}\hat{g}_x\mathrm{d}x\mathrm{d}z \tag{B.81}$$

The term $\hat{g}_B\left(\xi,\zeta\right)$ is the classical linearized subsonic solution. This term remains the same not only for all iterations, but also for all ϵ-levels. the double integral is the contribution due to the nonlinearities of the transonic equation. The integrals along the shock surface represent the jump in $\hat{u}$ and in the derivatives of $\hat{g}$ across the shock.

C
Derivations for Numerical Implementation of the Governing Equations

Stream-wise derivative of perturbation potential

The stream-wise derivative of the perturbation potential is obtained by differentiating Equation **??** with respect to the stream-wise direction ξ, that is:

$$\begin{aligned}\frac{\delta g}{\delta \xi}(\xi,\zeta) = &-\frac{1}{2\pi}\int_{-1}^{1}\left[\Delta\frac{\delta g}{\delta z}\right]_{z=0}\frac{(x-\xi)}{(x-\xi)^2+\zeta^2}dx\\ &+\frac{1}{2\pi}\int_{-1}^{1}\left[\Delta\frac{\delta g}{\delta x}\right]_{z=0}\frac{\zeta}{(x-\xi)^2+\zeta^2}dx\\ &-\frac{1}{2\pi b}\frac{\delta}{\delta\xi}\left[\iint_S\frac{(x-\xi)}{(x-\xi)^2+(z-\zeta)^2}\left(u\frac{\delta g}{\delta x}\right)dxdz\right]\end{aligned} \tag{C.1}$$

Vortex strength

The surface distribution of the vortex strength, such that the surface tangency condition satisfies on the plane ζ, is determined by differentiating **??** with respect to ζ and taking the limits of both sides of the equation as ζ approaches ±0. This results in:

$$\frac{\delta g}{\delta\zeta}(\xi,0^+) = \frac{1}{2}\left[\Delta\frac{\delta g}{\delta\zeta}\right]_{\zeta=0}+\frac{1}{2\pi}\int_{-1}^{1}\Delta\frac{\delta g}{\delta x}\frac{1}{x-\xi}dx-\frac{1}{2\pi b}\iint_S\frac{2z(x-\xi)}{[(x-\xi)^2+z^2]^2}\left(u\frac{\delta g}{\delta x}\right)dxdz \tag{C.2a}$$

$$\frac{\delta g}{\delta\zeta}(\xi,0^-) = -\frac{1}{2}\left[\Delta\frac{\delta g}{\delta\zeta}\right]_{\zeta=0}+\frac{1}{2\pi}\int_{-1}^{1}\Delta\frac{\delta g}{\delta x}\frac{1}{x-\xi}dx-\frac{1}{2\pi b}\iint_S\frac{2z(x-\xi)}{[(x-\xi)^2+z^2]^2}\left(u\frac{\delta g}{\delta x}\right)dxdz \tag{C.2b}$$

Adding equations C.2a and C.2b to each other gives:

$$\left[\frac{\delta g}{\delta\zeta}(\xi,0^+)+\frac{\delta g}{\delta\zeta}(\xi,0^-)\right]+\frac{1}{\pi b}\iint_S\frac{2z(x-\xi)}{[(x-\xi)^2+z^2]^2}\left(u\frac{\delta g}{\delta x}\right)dxdz = \frac{1}{\pi b}\int_{-1}^{1}\Delta\frac{\delta g}{\delta x}\frac{1}{x-\xi}dx \tag{C.3}$$

This equation can be inverted using the steady surface tangency condition given by Equation 3.14b, and assuming that the surface integral on the left-hand-side of Equation C.3 is a known function of ξ. The result is given by Equation C.4:

$$\Delta\frac{\delta g}{\delta\xi}(\xi,0^\pm) = \frac{2}{\pi}\sqrt{\frac{1-\xi}{1+\xi}}\int_{-1}^{1}\frac{I(x)}{\xi-x}\sqrt{\frac{1+x}{1-x}}dx \tag{C.4}$$

Here, $I(x)$ represents the surface integral on the left-hand-side of Equation C.3, thus:

$$I(x) = \iint_S \frac{2z(x-\xi)}{[(x-\xi)^2+z^2]^2}\left(u\frac{\delta g}{\delta x}\right) dxdz \tag{C.5}$$

Integration Scheme

The numerical technique used to evaluate the integral equations on the mesh will be outlined in this section. The treatment of the line integrals appearing in the computation of the perturbation velocity $u(\xi,\zeta)$ and the thickness and lift contribution terms, as can be seen in equations 4.8 and 4.4 respectively, is carried by the numerical integration function of $MATLAB$. That is, first the integrand, thus the function that has to be integrated, is defined by:

$$fun = @(x)F \tag{C.6}$$

where F represents the integrand, and the symbol within the brackets (x in this case) represents the variable the function F depends on. Once this function fun and the variable are defined, the syntax to numerically integrate the function F from the lower limit $xmin$ to the upper limit $xmax$ is defined as:

$$q = integral(fun, xmin, xmax) \tag{C.7}$$

The double integrals however, also known as surface integrals, have to be treated using another technique. This technique consists of dividing the flow field into two-dimensional computational rectangular boxes, which can be seen in Figure C.1a.
Those boxes will then be used to perform the calculations on. A single box is shown in Figure C.1b to demonstrate this calculation process. Before delving into those computations the flow domain is cut into an upper an lower domain by introducing a cut at $z=0$, indicated by S^+ and S^{-1} respectively. This is done because of the jump in velocity potential across the plane $z=0$ downstream of the leading edge, which is a feature of inviscid, lifting flows. Since the potential g appears as a multiplying factor in the integrand of the surface integral in Equation C.1, the integrand also experiences a jump. An effective way of incorporating this jump in the integration procedure is by introducing a cut along $z=0$, explained before. This is shown in Figure C.2.
The computation process then starts with computing the value of the integrand of the surface integral (which will be denoted by F from now on) at the intersection points of the domain. Those values are indicated by F_{i-1}^{j+1}, F_i^{j+1}, F_{i+1}^{j+1}, F_{i-1}^{j}, F_i^{j}, F_{i+1}^{j}, F_{i-1}^{j-1}, F_i^{j-1}, F_{i+1}^{j-1} and so on, which is shown in Figure C.3. The integral $\iint Fdxdz$ over the area of this single rectangular element can then be written as:

$$\iint_{rectangle} Fdxdz = \frac{1}{36}\left[F_{i-1}^{j+1} + F_{i+1}^{j+1} + F_{i+1}^{j-1} + F_{i-1}^{j-1} + 4\left(F_{i-1}^{j} + F_i^{j+1} + F_{i+1}^{j} + F_i^{j-1}\right) + 16F_i^j\right]\Delta x\Delta z \tag{C.8}$$

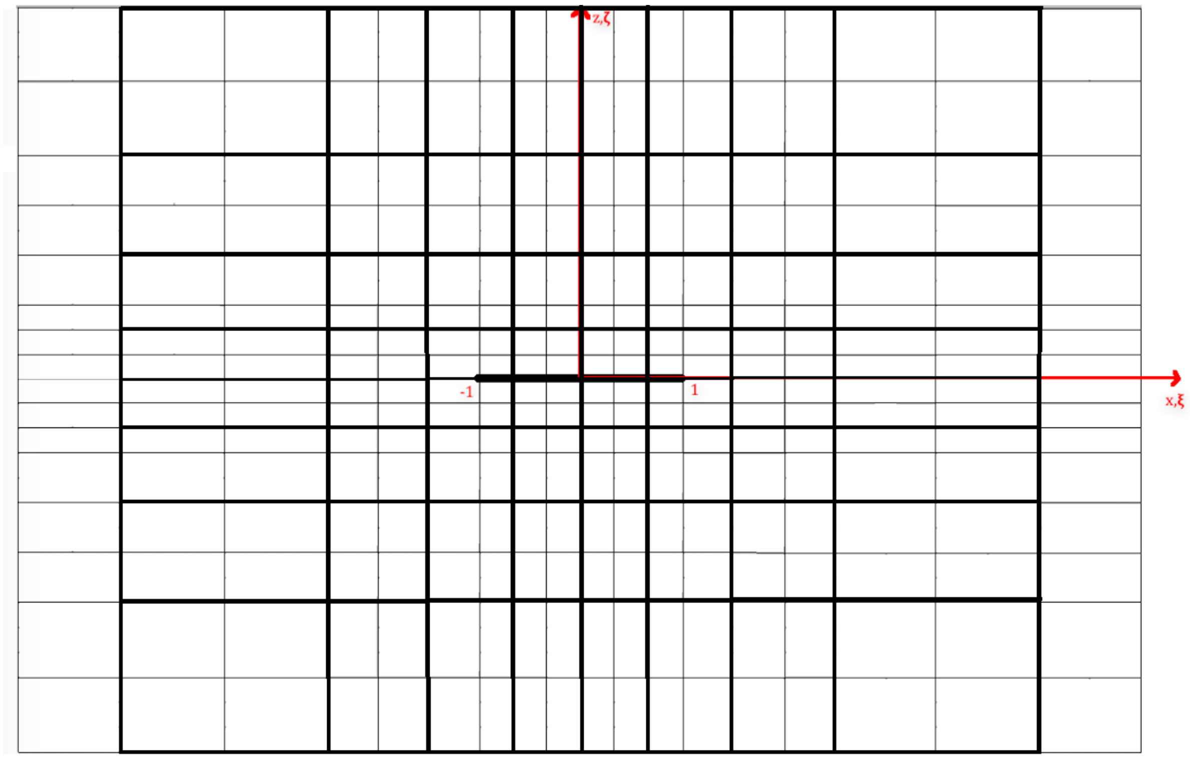
where Δx and Δz denote the horizontal and vertical sides of the rectangular box, as can be seen in Figure C.1b. This integration process is also known as Simpson's one-third rule. The surface integral over the entire flow field can then be obtained by summation over individual rectangles, that is:

$$\iint_{x,z} Fdxdz = \sum_{allrectangles}\left[\iint_{onerectangle} Fdxdz\right] \tag{C.9}$$

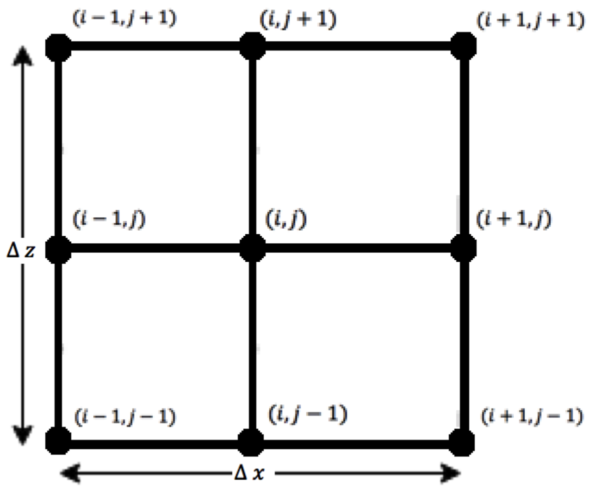
The upper S^+ and lower S^- surfaces are treated separately in doing this, because the values of the integrands at both surfaces (F^+ and F^-) are affected by the values of the potential below and above the cut $\left(\left(\frac{\delta g}{\delta x}\right)^+ and \left(\frac{\delta g}{\delta x}\right)^-\right)$. That is:

- F^+ has a contribution from $\left(\frac{\delta g}{\delta x}\right)^+$ and $\left(\frac{\delta g}{\delta x}\right)^-$ ($MATLAB$ code lines 366 - 424)
- F^- has a contribution from $\left(\frac{\delta g}{\delta x}\right)^+$ and $\left(\frac{\delta g}{\delta x}\right)^-$ ($MATLAB$ code lines 483 - 514)

The upper and lower domains, S^+ and S^- respectively, are indexed according to the indexing system shown in Figure C.4.



(a) Flowfield divided into two-dimensional rectangular boxes



(b) Single two-dimensional rectangular box

Adapting Equation C.8 to this indexing system results in equations C.10a and C.10b for the upper S^+ and lower S^- domains respectively:

$$\iint_{rectangleS^+} F dx dz = \frac{1}{36}\left[F_{i-1}^{j+2} + F_{i+1}^{j+2} + F_{i+1}^{j} + F_{i-1}^{j} + 4\left(F_{i-1}^{j+1} + F_{i}^{j+2} + F_{i+1}^{j+1} + F_{i}^{j}\right) + 16F_{i}^{j+1}\right]\Delta x \Delta z \quad \text{(C.10a)}$$

$$\iint_{rectangleS^-} F dx dz = \frac{1}{36}\left[F_{i-1}^{j-2} + F_{i+1}^{j-2} + F_{i+1}^{j} + F_{i-1}^{j} + 4\left(F_{i-1}^{j-1} + F_{i}^{j-2} + F_{i+1}^{j-1} + F_{i}^{j}\right) + 16F_{i}^{j-1}\right]\Delta x \Delta z \quad \text{(C.10b)}$$

Once the integrals are computed, they have to be differentiated with respect to ϵ, as can be seen in the third term of Equation C.1. This differentiation scheme will be treated in the next section.

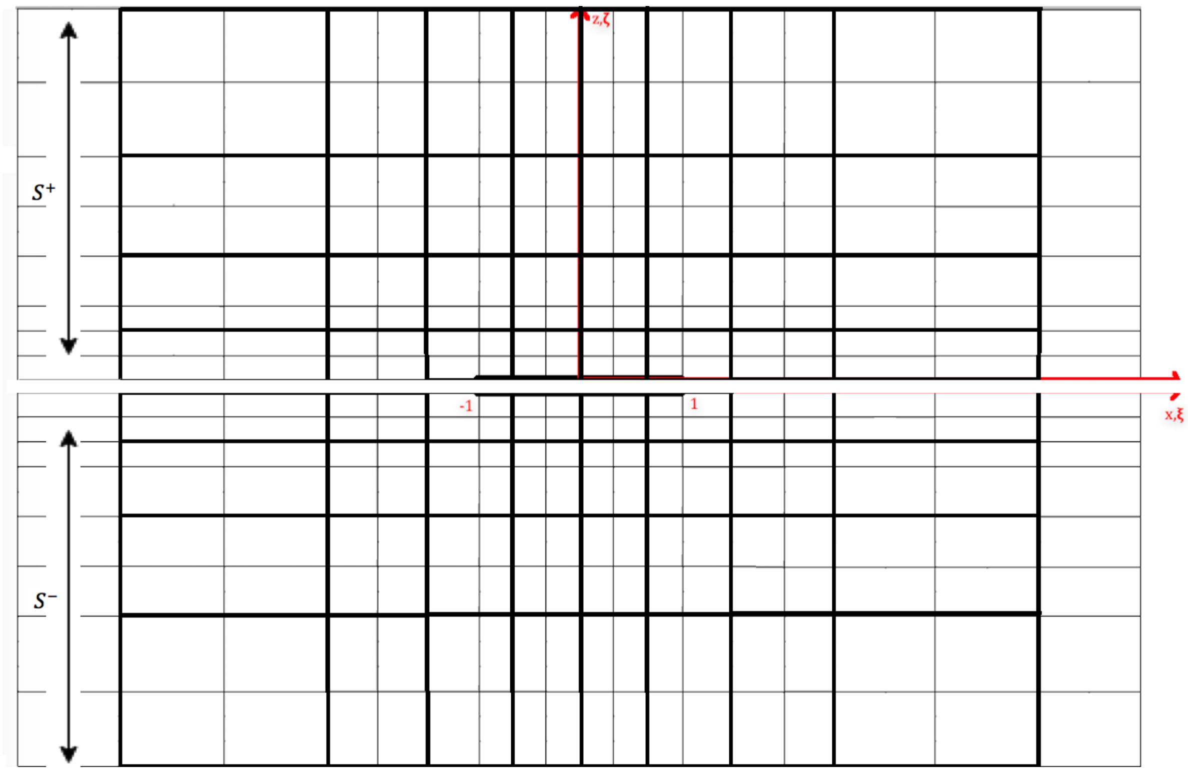


Figure C.2: Flow domain dividid into upper (S^+) and lower (S^-) surface

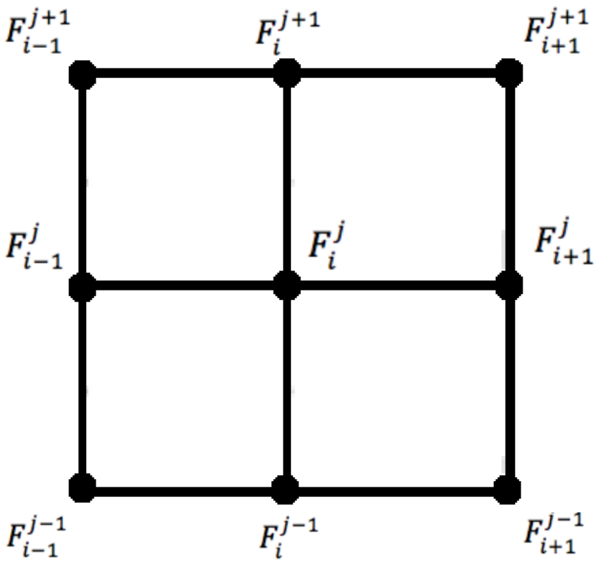


Figure C.3: Integrand computed at intersection points of the flow field domain

DIFFERENTIATION SCHEME

The differentiation process is carried out with a centered formula of order Ox^4, also known as a forth order accurate finite difference scheme:

$$\left[\frac{\delta F}{\delta x}\right]_{ij} = -\frac{1}{12\Delta x}\left[F_{i-2}^{j} + 8\left(F_{i+1}^{j} - F_{i-1}^{j}\right) - F_{i+2}^{j}\right] \tag{C.11}$$

where Δx is given by $(x_{i+2} - x_{i-2})$. However, this equation can only be used for the middle region, meaning that for the intersection points of the two cells at the left edge of the domain and two cells at the right edge of

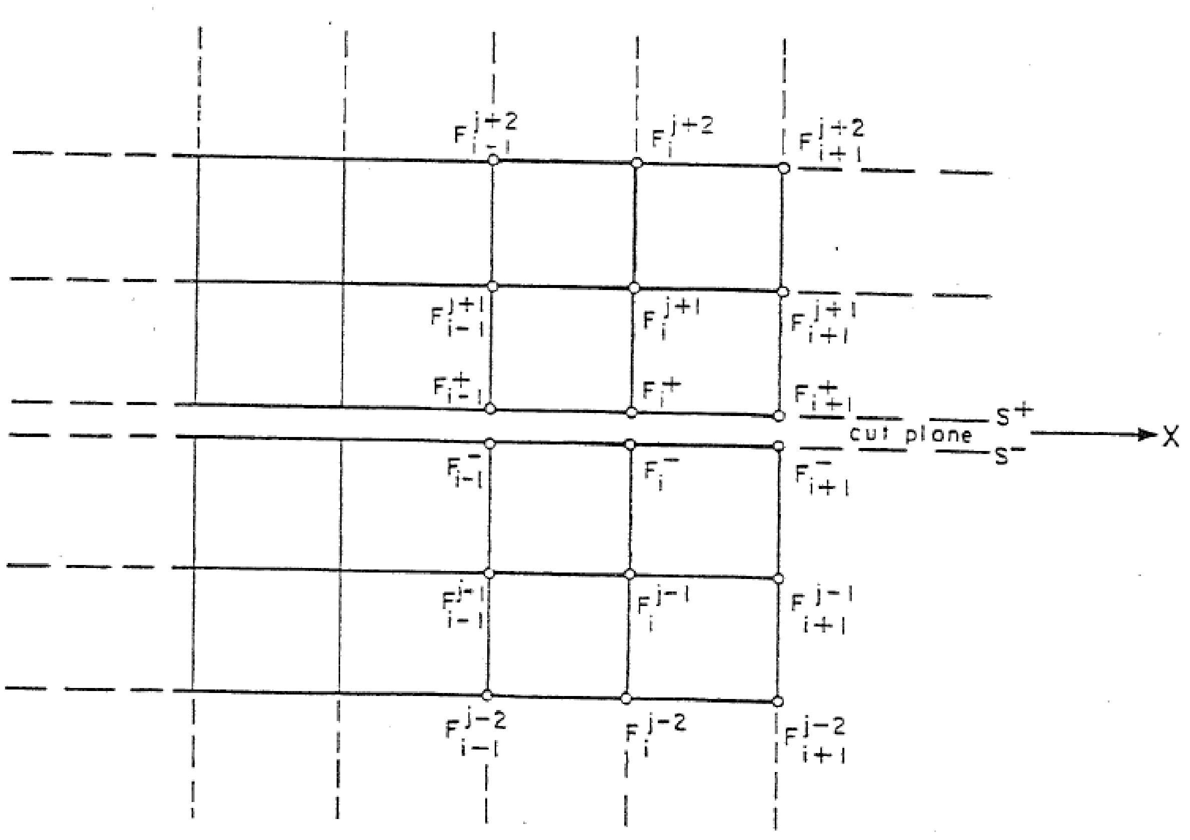


Figure C.4: Indexing of the upper and lower flow field domains

the domain different equations are needed. This is because for those points F_{i-2}^{j}, F_{i-1}^{j}, F_{i+1}^{j}, and F_{i+2}^{j} do not exist. Therefore, the differentiation process will be performed in five parts, according to Figure C.5.
The values of $\left[\frac{\delta F}{\delta x}\right]_{ij}$ on the intersection points of the red and green lines on the left side of the mesh are calculated with equations C.12a and C.12b respectively:

$$\left[\frac{\delta F}{\delta x}\right]_{ij} = \frac{1}{(x_{i+1} - x_i)}\left[F_{i+1}^{j} - F_{i}^{j}\right] \tag{C.12a}$$

$$\left[\frac{\delta F}{\delta x}\right]_{ij} = \frac{1}{(x_{i+1} - x_{i-1})}\left[F_{i+1}^{j} - F_{i-1}^{j}\right] \tag{C.12b}$$

The same process is carried out on the right side of the mesh, resulting in equations C.13a and C.13b for the values of $\left[\frac{\delta F}{\delta x}\right]_{ij}$ on the red and green lines respectively:

$$\left[\frac{\delta F}{\delta x}\right]_{ij} = \frac{1}{(x_{i} - x_{i-1})}\left[F_{i}^{j} - F_{i-1}^{j}\right] \tag{C.13a}$$

$$\left[\frac{\delta F}{\delta x}\right]_{ij} = \frac{1}{(x_{i+1} - x_{i-1})}\left[F_{i+1}^{j} - F_{i-1}^{j}\right] \tag{C.13b}$$

The computations of $\left[\frac{\delta F}{\delta x}\right]_{ij}$ for the different regions can be found in the following lines of the *MATLAB* code:

- Values on the red line on the left side of the mesh: code lines 445 – 448 and 527 – 530 for the upper side of the airfoil and lower side of the airfoil respectively.
- Values on the green line on the left side of the mes: code lines 451 – 454 and 533 – 536 for the upper side of the airfoil and lower side of the airfoil respectively.

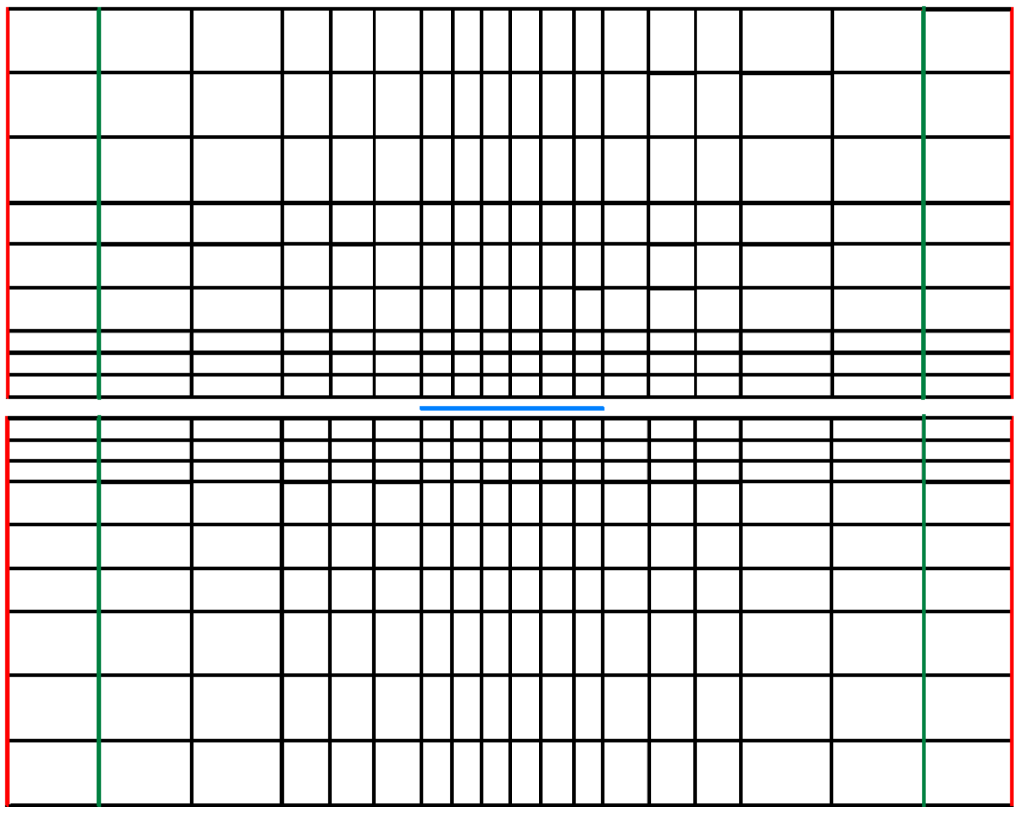


Figure C.5: Flow field domain split up into five regions for differentiation process

- Values in the middle region of the mesh: code lines 427 – 441 and 517 – 523 for the upper side of the airfoil and lower side of the airfoil respectively.
- Values on the red line on the right side of the mesh: code lines 460 – 463 and 542 – 545 for the upper side of the airfoil and lower side of the airfoil respectively.
- Values on the green line on the right side of the mes: code lines 469 – 472 and 551 – 554 for the upper side of the airfoil and lower side of the airfoil respectively.

D

Equation Form of Algorithm

This appendix serves as an extension of Chapter 4.2 and outlines the algorithm developed to predict the loading on a symmetric parabolic arc airfoil immersed in an inviscid, irrotational, compressible flow in equation form.

$$\begin{tikzcd}
\boxed{u_i(\xi,\zeta) = \frac{4\mu\epsilon_0}{\pi}\int_{-1}^{1}\frac{x(x-\xi)}{(x-\xi)^2+\zeta^2}dx + \frac{\alpha\mu}{\pi}\int_{-1}^{1}\sqrt{\frac{(1-x)}{(1+x)}}\frac{\zeta}{(x-\xi)^2+\zeta^2}dx} \arrow[d] \\
\boxed{\left(\frac{\delta g}{\delta\xi}\right)_j(\xi,\zeta) = \frac{4\mu}{\pi}\int_{-1}^{1}\frac{x(x-\xi)}{(x-\xi)^2+\zeta^2}dx} \arrow[d] \\
\boxed{I_j(\xi) = \frac{1}{\pi b}\iint_S \frac{2z(x-\xi)}{[(x-\xi)^2+z^2]^2}\left(u_i\left(\frac{\delta g}{\delta x}\right)_j\right)dxdz} \arrow[d] \\
\boxed{\left(\Delta\frac{\delta g}{\delta\xi}\right)_j(\xi,0^\pm) = \frac{2}{\pi}\sqrt{\frac{1-\xi}{1+\xi}}\int_{-1}^{1}\frac{I_i}{\xi-x}\sqrt{\frac{1+x}{1-x}}dx} \arrow[d] \\
\boxed{\left(\frac{\delta g}{\delta\xi}\right)_{j+1}(\xi,\zeta) = \frac{4\mu}{\pi}\int_{-1}^{1}\frac{x(x-\xi)}{(x-\xi)^2+\zeta^2}dx + \frac{1}{2\pi}\int_{-1}^{1}\left[\left(\Delta\frac{\delta g}{\delta\xi}\right)_j\right]\frac{\zeta}{(x-\xi)^2+\zeta^2}dx - \frac{1}{2\pi b}\frac{\delta}{\delta\xi}\left[\iint_S\frac{(x-\xi)}{(x-\xi)^2+(z-\zeta)^2}\left(u_i\left(\frac{\delta g}{\delta\xi}\right)_j\right)dxdz\right]} \arrow[d] \\
\boxed{I_{j+1}(\xi) = \frac{1}{\pi b}\iint_S \frac{2z(x-\xi)}{[(x-\xi)^2+z^2]^2}\left(u_i\left(\frac{\delta g}{\delta x}\right)_{j+1}\right)dxdz} \arrow[d] \\
\boxed{\left(\Delta\frac{\delta g}{\delta\xi}\right)_{j+1}(\xi,0^\pm) = \frac{2}{\pi}\sqrt{\frac{1-\xi}{1+\xi}}\int_{-1}^{1}\frac{I_{j+1}}{\xi-x}\sqrt{\frac{1+x}{1-x}}dx} \arrow[d] \\
\boxed{\left(\frac{\delta g}{\delta\xi}\right)_{j+2}(\xi,\zeta) = \frac{4\mu}{\pi}\int_{-1}^{1}\frac{x(x-\xi)}{(x-\xi)^2+\zeta^2}dx + \frac{1}{2\pi}\int_{-1}^{1}\left[\left(\Delta\frac{\delta g}{\delta\xi}\right)_{j+1}\right]\frac{\zeta}{(x-\xi)^2+\zeta^2}dx - \frac{1}{2\pi b}\frac{\delta}{\delta\xi}\left[\iint_S\frac{(x-\xi)}{(x-\xi)^2+(z-\zeta)^2}\left(u_i\left(\frac{\delta g}{\delta\xi}\right)_{j+1}\right)dxdz\right]} \arrow[d] \arrow[uu, bend left=60, red, "\begin{matrix} j=j+1 \\ 1\le j\le n \end{matrix}"] \\
\boxed{\Delta u = \left(\frac{\delta g}{\delta\xi}\right)_n\Delta\epsilon} \arrow[d] \\
\boxed{u_{i+1} = u_i + \Delta u} \arrow[d] \arrow[uuuu, bend right=60, blue, "\begin{matrix} i=i+1 \\ 1\le i\le m \end{matrix}"'] \\
\boxed{C_p(x,0^\pm) = -\frac{2}{bv}u_m}
\end{tikzcd}$$

Here, the red line represents the inner-loop (indexed by n), which loops until $\left(\frac{\delta g}{\delta \xi}\right)$ reaches a converged value while keeping the perturbation velocity u constant. In this study it has been found that $\left(\frac{\delta g}{\delta \xi}\right)$ converges after $n = 120$ iterations (with an under-relaxation factor $d = 0.1$ incorporated). This converged value of $\left(\frac{\delta g}{\delta \xi}\right)$ is then being used in the perturbation velocity u computations. These perturbation velocity computations depend on the thickness ratio ϵ, and are being looped until the desired thickness ratio is reached ($\epsilon = 0.03$ in this study for both the upper and lower side of the airfoil, so a total airfoil thickness of $\epsilon = 6\%$). This outer-loop (indexed by m) is indicated by the blue line. The thickness ratio is each loop increased in uniform increments $\Delta\epsilon$. Since the desired thickness for this study is $\epsilon = 0.03$ for the upper and lower side of the airfoil, and the increment in thickness ratio is $\Delta = 0.005$ starting from a value of $\epsilon_0 = 0.005$, six loops will be required to reach the desired total airfoil thickness of $\epsilon = 6\%$. Finally, the outcome of the outer-loop (i.e. the perturbation velocity u) will be used to determine the pressure coefficient C_P distribution along the upper and lower side of the airfoil.

It should also be mentioned that at the end of the inner-loop an under-relaxation constant has been introduced for stability purposes. This constant is implemented in the computation of $\frac{\delta g}{\delta \xi}$ as follows:

$$\left(\frac{\delta \tilde{g}}{\delta \xi}\right)_{j+1} = \left(\frac{\delta g}{\delta \xi}\right)_j + d\left[\left(\frac{\delta g}{\delta \xi}\right)_{j+1} - \left(\frac{\delta g}{\delta \xi}\right)_j\right]$$

Thus, $\left(\frac{\delta g}{\delta \xi}\right)_{j+1}$ in the overview above is actually $\left(\frac{\delta \tilde{g}}{\delta \xi}\right)_{j+1}$, and is the value of $\frac{\delta g}{\delta \xi}$ which is being used as the starting value of the next iteration step. This hold for every iteration step. $\left(\frac{\delta g}{\delta \xi}\right)_j$ is the value of $\frac{\delta g}{\delta \xi}$ at the beginning of an iteration step, while $\left(\frac{\delta g}{\delta \xi}\right)_{j+1}$ is the value of $\frac{\delta g}{\delta \xi}$ determined at the end of every iteration step. Thus, this means that when no under-relaxation factor is applied, the value of $\frac{\delta g}{\delta \xi}$ taken to the next level is computed as follows:

$$\left(\frac{\delta \tilde{g}}{\delta \xi}\right)_{j+1} = \left(\frac{\delta g}{\delta \xi}\right)_{j+1}$$

Applying an under-relaxation factor subsequently means that a fraction of the difference of the value of $\frac{\delta g}{\delta \xi}$ at the start of an iteration step, and the value of $\frac{\delta g}{\delta \xi}$ determined at the end of the inner-loop, are being added to the value of $\frac{\delta g}{\delta \xi}$ at the start of the iteration step. This fraction of the difference added to the value of $\frac{\delta g}{\delta \xi}$ at the beginning of the iteration step is being set by the under-relaxation factor d. A stability constant of $d = 0.1$ has been used for this study.

E

MATLAB CODE

```
% Steady problem - Application to a Parabolic Arc Airfoil
clc; clear all; close all

tic
%% Parameters
rho_inf = 1.1751;                    % [kg/m^3]
P_inf   = 101325;                    % [Pa]
R       = 8.314510;                  % [J/(mol K)]
M_air   = 0.0289645;                 % [kg/mol]
R_a     = R/M_air;
T       = 300;                       % [K]  (=27 deg)
gamma   = 1.4;                       % [-]
minf    = 0.86;                      % [-]
beta    = sqrt(1-minf^2);            % [-]
mu      = ((gamma+1)*minf^2)/beta^3; % [-]
mu_inf  = 1.789E-5;                  % [(kg/m)*s]
nu      = beta*mu;                   % [-]
C       = 2;                         % Chord length [m]
B       = C/2;                       % Span [m]
e0      = 0.00125;                   % Thickness ratio [-]
de      = 0.00125;                   % [-]
A       = 1;                       % [degrees]
aoa     = ((A*pi)/180);              % [radians]
a       = sqrt(gamma*R_a*T);
V_inf   = minf*a;
q_inf   = 0.5*rho_inf*V_inf^2;       % Dynamic pressure [Pa],[N/m^2],[kg/(m*s^2)]

%% Mesh generation
N = 25;             % Number of mesh points/volumes
Delta = 1/N;        % Uniforme ruimte stap / Equidistant paneling distance
uniform = true;     % Uniform grid or not

% Vicinity of airfoil
for i=1:N+1
    xmeshi1 = (i-1)*Delta;
    if uniform
        xmesh1(i) = xmeshi1;                      % Set up list of coordinates for uniform...
             grid
    else
        xmesh1(i) = 0.5*(1. - cos(pi*xmeshi1));   % Set up list of coordinates for non-...
            uniform grid (according cosine rule)
    end
end
ymesh1 = xmesh1';

% Intermediate region
for i=1:(N/2)
    xmeshi2 = i*2*Delta;
    if uniform
        xmesh2(i) = 1+xmeshi2;                      % Set up list of coordinates for ...
            uniform grid
    else
        xmesh2(i) = 0.5*(1. - cos(pi*xmeshi2));     % Set up list of coordinates for non-...
            uniform grid (according cosine rule)
    end
end
ymesh2 = xmesh2';

% Far field
for i=1:(N/4)
    xmeshi3 = i*4*Delta;
    if uniform
        xmesh3(i) = 2+xmeshi3;                      % Set up list of coordinates for ...
            uniform grid
        xmesh4(i) = xmesh3(end)+xmeshi3;
    else
        xmesh3(i) = 0.5*(1. - cos(pi*xmeshi3));     % Set up list of coordinates for non-...
            uniform grid (according cosine rule)
    end
end
ymesh3 = xmesh3';

xmesh = [xmesh1 xmesh2 xmesh3];% xmesh4];
ymesh = xmesh';

xmesh_tot = [flip(-xmesh(2:end)) xmesh];
ymesh_tot = xmesh_tot';

x = zeros(N+1,1);

% Set up coordinates for the middles of the cells.
xm = [xmesh(1) 0.5*(xmesh(2:N+1)+xmesh(1:N)) xmesh(end)];
xm_tot = [flip(-xm(2:end)) xm];
ym = [ymesh(1); 0.5*(ymesh(2:N+1)+ymesh(1:N)); ymesh(end)];
ym_tot = [flip(ym(2:end)); -ym];

% Set up the cell withs and heights
dmesh = xmesh(2:end)-xmesh(1:end-1);
dx_tot = xmesh_tot(2:end)-xmesh_tot(1:end-1);
dy = ymesh(2:end)-ymesh(1:end-1);
dy_tot = ymesh_tot(2:end)-ymesh_tot(1:end-1);

% Plot mesh
z = zeros(length(xmesh),length(ymesh_tot));
z2 = zeros(length(xmesh(19:end)),length(ymesh_tot));
%% Parabolic arc airfoil
xL = flip(-xmesh1);
xR = xmesh1;
x = [xL(1:end-1) xR];

%% Base solution for distrubation velocity u
% Numeric integration
xmesh = xmesh_tot;
ymesh = [0.01; ymesh(2:end)];
xmesh_modified2 = transpose(xmesh);
ymesh_modified2 = transpose(ymesh);
for i=1:length(ymesh)
    xmesh_modified(:,i) = xmesh_modified2;
end
for i=1:length(xmesh)
    ymesh_modified(i,:) = ymesh_modified2;
end
ymesh_l = -ymesh;
ymesh_l_modified = -ymesh_modified;

% u_base upper side airfoil
for i=1:length(xmesh) %xi
    for j=1:length(ymesh) %zeta
        fun = @(x) ((4*mu*e0)/pi)*((x.*(x-xmesh(i)))./((x-xmesh(i)).^2+ymesh(j).^2))+...
```

```
            ((mu*aoa)/pi)*(sqrt((1-x)./(1+x)).*(ymesh(j)./((x-xmesh(i)).^2+ymesh(j).^2)));
        u_base(i,j) = integral(fun,-0.9999,1);
    end
end

xmesh_transpose = transpose(xmesh);
for i=1:length(u_base(1,:))
    xmesh_int(:,i) = xmesh_transpose;
end

ymesh_transpose = transpose(ymesh);
for i=1:length(u_base(:,1))
    ymesh_int(i,:) = ymesh_transpose;
end

v = zeros(length(u_base(:,1)),length(u_base(1,:)));

% u_base lower side airfoil
for i=1:length(xmesh) %xi
    for j=1:length(ymesh_l) %zeta
        fun = @(x) ((4*mu*e0)/pi)*((x.*(x-xmesh(i)))./((x-xmesh(i)).^2+ymesh_l(j).^2))+...
            ((mu*aoa)/pi)*(sqrt((1-x)./(1+x)).*(ymesh_l(j)./((x-xmesh(i)).^2+ymesh_l(j)....
                ^2)));
        u_base_l(i,j) = integral(fun,-0.9999,1);
    end
end

xmesh_transpose = transpose(xmesh);
for i=1:length(u_base_l(1,:))
    xmesh_int(:,i) = xmesh_transpose;
end

ymesh_l_transpose = transpose(ymesh_l);
for i=1:length(u_base_l(:,1))
    ymesh_l_int(i,:) = ymesh_l_transpose;
end

v_l = zeros(length(u_base_l(:,1)),length(u_base_l(1,:)));

%% Thickness contribution

% Upper side airfoil
for i=1:length(xmesh)
    for j=1:length(ymesh)
        for k=1:length(x)
            g_e_t_2(i,j,k) = ((4*mu)/pi)*((x(k).*(x(k)-xmesh(i)))./((x(k)-xmesh(i)).^2+...
                ymesh(j)^2));
        end
    end
end

dx = x(2:end)-x(1:end-1);
dx2 = [dx, dx(end)];

dx3= zeros(length(g_e_t_2(:,1,1)),length(g_e_t_2(1,:,1)),length(g_e_t_2(1,1,:)));
for i=1:length(g_e_t_2(:,1,1))
    for j=1:length(g_e_t_2(1,:,1))
        dx3(i,j,:) = dx2;
    end
end

g_e_t_3 = g_e_t_2.*dx3;

for i=1:length(xmesh)
    for j=1:length(ymesh)
        g_e_t(i,j) = sum(g_e_t_3(i,j,:));
    end
end

% Lower side airfoil
for i=1:length(xmesh)
```

```
    for j=1:length(ymesh)
        for k=1:length(x)
            g_e_t_2_l(i,j,k) = ((4*mu)/pi)*((x(k).*(x(k)-xmesh(i)))./((x(k)-xmesh(i)).^2+...
                ymesh_l(j)^2));
        end
    end
end

g_e_t_3_l = g_e_t_2_l.*dx3;

for i=1:length(xmesh)
    for j=1:length(ymesh)
        g_e_t_l(i,j) = sum(g_e_t_3_l(i,j,:));
    end
end

%% Looping

for i=1:2:length(xmesh)-2
    dxi3(i) = xmesh(i+2)-xmesh(i);
end

dxi3 = [0,dxi3];

dxi4 = zeros(length(xmesh),length(xmesh)-1);
for i=1:length(xmesh)
    dxi4(i,:) = dxi3;
end

for k=1:length(ymesh)-3
    dxi5(:,:,k) = dxi4;
end

for i=1:2:length(ymesh)-2
    dz(i) = ymesh(i+2)-ymesh(i);
end

for i=1:2:length(ymesh_l)-2
    dz_l(i) = ymesh_l(i+2)-ymesh_l(i);
end

for i=1:length(xmesh)
    for j=1:length(xmesh)-1
        for k=1:length(dz)
            dz2(i,j,:) = dz;
        end
    end
end

for i=1:length(g_e_t(:,1))
    for j=1:length(g_e_t(:,1))-1
        for k=1:length(dz_l)
            dz_l2(i,j,:) = dz_l;
        end
    end
end

m = 24;                         % Index outer-loop
u_cell =cell(m+1,1);
u_cell{1,1} = u_base;
u_cell_l = cell(m+1,1);
u_cell_l{1,1} = u_base_l;

n = 10;                         % Index inner-loop
d = 0.1;                        % Onderrelaxatie constante/relaxation constant

g_x_cell = cell(n+1,1);
g_x_cell{1,1} = g_e_t;

g_x_cell_l = cell(n+1,1);
g_x_cell_l{1,1} = g_e_t_l;
```

```

dg_x_cell = cell(n,1);
dg_x_cell_pc = cell(n,1);

dg_x_cell_l = cell(n,1);
dg_x_cell_l_pc = cell(n,1);
for len2=1:m                                % Outer-loop, update for u with Δ epsilon
    u = u_cell{len2,1};
    u_l = u_cell_l{len2,1};
    for len=1:n                             % Inner-loop, update for Δ gx
        g_x = g_x_cell{len,1};
        g_x_l = g_x_cell_l{len,1};

        %% I(xi)
        % Effect upper mesh on upper side airfoil
        F_1_u_uu = zeros(length(xmesh),length(xmesh),length(ymesh));
        for i=1:length(xmesh)
            F_1_u_uu(i,:,:) = (((2*ymesh_modified.*(xmesh_modified-xmesh(i)))./((...
                xmesh_modified-xmesh(i)).^2+ymesh_modified.^2).^2).*u.*g_x);
        end

        % Effect lower mesh on upper side airfoil
        F_1_u_lu = zeros(length(xmesh),length(xmesh),length(ymesh));
        for i=1:length(xmesh)
            F_1_u_lu(i,:,:) = (((2*ymesh_l_modified.*(xmesh_modified-xmesh(i)))./((...
                xmesh_modified-xmesh(i)).^2+ymesh_l_modified.^2).^2).*u_l.*g_x_l);
        end

        F_1_u_tot = F_1_u_uu + F_1_u_lu;

        % Effect lower mesh on lower side airfoil
        F_1_l_ll = zeros(length(xmesh),length(xmesh),length(ymesh));
        for i=1:length(xmesh)
            F_1_l_ll(i,:,:) = (((2*ymesh_l_modified.*(xmesh_modified-xmesh(i)))./((...
                xmesh_modified-xmesh(i)).^2+ymesh_l_modified.^2).^2).*u_l.*g_x_l);
        end

        % Effect upper mesh on lower side airfoil
        F_1_l_ul = zeros(length(xmesh),length(xmesh),length(ymesh));
        for i=1:length(xmesh)
            F_1_l_ul(i,:,:) = (((2*ymesh_modified.*(xmesh_modified-xmesh(i)))./((...
                xmesh_modified-xmesh(i)).^2+ymesh_modified.^2).^2).*u.*g_x);
        end

        F_1_l_tot = F_1_l_ll + F_1_l_ul;

        F_2_u_tot = zeros(length(xmesh),1);
        for j=2:2:length(xmesh)-1
            for k=1:2:length(ymesh)-3
                F_2_u_tot = F_2_u_tot + (((1/36)*(F_1_u_tot(:,j-1,k)+F_1_u_tot(:,j+1,k)+...
                    F_1_u_tot(:,j-1,k+2)+F_1_u_tot(:,j+1,k+2)+4*(F_1_u_tot(:,j,k)+...
                    F_1_u_tot(:,j-1,k+1)+F_1_u_tot(:,j+1,k+1)+F_1_u_tot(:,j,k+2))+16*...
                    F_1_u_tot(:,j,k+1))).*dxi5(:,j,k).*dz2(:,j,k));
            end
        end

        F_2_l_tot = zeros(length(xmesh),1);
        for j=2:2:length(xmesh)-1
            for k=1:2:length(ymesh)-3
                F_2_l_tot = F_2_l_tot + (((1/36)*(F_1_l_tot(:,j-1,k)+F_1_l_tot(:,j+1,k)+...
                    F_1_l_tot(:,j-1,k+2)+F_1_l_tot(:,j+1,k+2)+4*(F_1_l_tot(:,j,k)+...
                    F_1_l_tot(:,j-1,k+1)+F_1_l_tot(:,j+1,k+1)+F_1_l_tot(:,j,k+2))+16*...
                    F_1_l_tot(:,j,k+1))).*dxi5(:,j,k).*dz_l2(:,j,k)); % dz2 = dz_l2?
            end
        end

        I_u_tot = F_2_u_tot/(pi*B);
        I_l_tot = F_2_l_tot/(pi*B);
        I_ul_tot = I_u_tot + I_l_tot;
        %% dgxi

        % Upper +lower side airfoil (same)
        for i=1:length(xmesh)
            for j=1:length(x)
                f(i,j) = ((1/(xmesh(i)-x(j))).*sqrt((1+x(j))./(1-x(j)))).*I_ul_tot(18+j);
            end
        end

        f(isinf(f))=0;
        f(isnan(f))=0;

        dx_tot4 = zeros(length(xmesh),length(x));
        for i=1:length(xmesh)
            dx_tot4(i,:) = dx2;
        end

        f2 = f.*dx_tot4;

        for i=1:length(xmesh)
            f3(i) = sum(f2(i,1:end-1));
        end

        for i=1:length(xmesh)
            dgxi(i) = (2/pi)*sqrt((1-xmesh(i))./(1+xmesh(i))).*f3(i);
        end

        dgxi(isinf(dgxi))=0;
        dgxi(isnan(dgxi))=0;

        %% Lift contribution

        % Upper side airfoil
        for i=1:length(xmesh)
            for j=1:length(ymesh)
                for k=1:length(x)
                    g_e_L(i,j,k) = (1/(2*pi))*(dgxi(18+k).*(ymesh(j)./((x(k)-xmesh(i)).^2+...
                        ymesh(j).^2)));
                end
            end
        end

        g_e_L_2 = g_e_L.*dx3;

        for i=1:length(xmesh)
            for j=1:length(ymesh)
                g_e_l(i,j) = sum(g_e_L_2(i,j,:));
            end
        end

        % Lower side airfoil
        for i=1:length(xmesh)
            for j=1:length(ymesh_l)
                for k=1:length(x)
                    g_e_L_l(i,j,k) = (1/(2*pi))*(dgxi(18+k).*(ymesh_l(j)./((x(k)-xmesh(i))...
                        .^2+ymesh_l(j).^2)));
                end
            end
        end

        g_e_L_l_2 = g_e_L_l.*dx3;

        for i=1:length(xmesh)
            for j=1:length(ymesh_l)
                g_e_l_l(i,j) = sum(g_e_L_l_2(i,j,:));
            end
        end

        %% Non-Linear contribution

        % Upper side airfoil
        % Effect upper domain on upper side airfoil
        g_e_NL = zeros(length(xmesh),length(ymesh),length(xmesh),length(ymesh));
        for k=1:length(xmesh)
            for l=1:length(ymesh)
            g_e_NL(:,:,k,l) = ((xmesh(k)-xmesh_modified)./((xmesh(k)-xmesh_modified).^2+(...
                ymesh(l)-ymesh_modified).^2)).*u.*g_x;
            end
        end

        % Effect lower domain on upper side airfoil
        g_e_NL_lower = zeros(length(xmesh),length(ymesh),length(xmesh),length(ymesh));
        for k=1:length(xmesh)
            for l=1:length(ymesh)
            g_e_NL_lower(:,:,k,l) = ((xmesh(k)-xmesh_modified)./((xmesh(k)-xmesh_modified)...
                .^2+(ymesh_l(l)-ymesh_modified).^2)).*u_l.*g_x_l; %*u.*g_x; (ymesh_l(l)-...
                ymesh_l_modified).^2
            end
        end

        g_e_NL(isnan(g_e_NL))=0;
        g_e_NL_lower(isnan(g_e_NL_lower))=0;

        dxi6 = zeros(length(xmesh),length(ymesh),length(xmesh)-1);
        for i=1:length(xmesh)
            for j=1:length(ymesh)
                for k=1:length(xmesh)-1
                    dxi6(i,j,:) = dxi3;
                end
            end
        end

        dz3 = zeros(length(xmesh),length(ymesh),length(ymesh)-3);
        for i=1:length(xmesh)
            for j=1:length(ymesh)
                for l=1:length(ymesh)-3
                    dz3(i,j,:) = dz;
                end
            end
        end

        dz3_l = zeros(length(xmesh),length(ymesh_l),length(ymesh_l)-3);
        for i=1:length(xmesh)
            for j=1:length(ymesh_l)
                for l=1:length(ymesh)-3
                    dz3_l(i,j,:) = dz_l;
                end
            end
        end

        g_e_NL2 = zeros(length(xmesh),length(ymesh));
        for k=2:2:length(xmesh)-1
            for l=1:2:length(ymesh)-3
                g_e_NL2 = g_e_NL2 + ((1/36)*(g_e_NL(:,:,k-1,l)+g_e_NL(:,:,k+1,l)+g_e_NL...
                    (:,:,k-1,l+2)+g_e_NL(:,:,k+1,l+2)+4*(g_e_NL(:,:,k,l)+g_e_NL(:,:,k-1,l...
                    +1)+g_e_NL(:,:,k+1,l+1)+g_e_NL(:,:,k,l+2))+16*g_e_NL(:,:,k,l+1))).*...
                    dxi6(:,:,k).*dz3(:,:,l);
            end
        end

        g_e_NL2_lower = zeros(length(xmesh),length(ymesh));
        for k=2:2:length(xmesh)-1
            for l=1:2:length(ymesh)-3
                g_e_NL2_lower = g_e_NL2_lower + ((1/36)*(g_e_NL_lower(:,:,k-1,l)+...
                    g_e_NL_lower(:,:,k+1,l)+g_e_NL_lower(:,:,k-1,l+2)+g_e_NL_lower(:,:,k...
                    +1,l+2)+4*(g_e_NL_lower(:,:,k,l)+g_e_NL_lower(:,:,k-1,l+1)+...
                    g_e_NL_lower(:,:,k+1,l+1)+g_e_NL_lower(:,:,k,l+2))+16*g_e_NL_lower...
                    (:,:,k,l+1))).*dxi6(:,:,k).*dz3(:,:,l); %dz3_l
            end
        end

        % Middle region
        i=3:length(xmesh)-2;
        j=1:length(ymesh);
        g_e_NL6(i,j) = -(g_e_NL2(i-2,j)+8*(g_e_NL2(i+1,j)-g_e_NL2(i-1,j))-g_e_NL2(i+2,j));
        g_e_NL6_lower(i,j) = -(g_e_NL2_lower(i-2,j)+8*(g_e_NL2_lower(i+1,j)-g_e_NL2_lower(...
            i-1,j))-g_e_NL2_lower(i+2,j));

        for i=3:length(xmesh)-2
            d_xmesh(i) = xmesh(i+2)-xmesh(i-2);
        end

        for i=1:length(g_e_NL6(1,:))
            d_xmesh3(:,i) = d_xmesh;
        end

        g_e_nl = (1/(2*pi*B))*(g_e_NL6./(12*d_xmesh3));
        g_e_nl_lower = (1/(2*pi*B))*(g_e_NL6_lower./(12*d_xmesh3));

        % Left boudary
        i=1;
        j=1:length(ymesh);
        g_e_NL6_lb(i,j) = (1/(2*pi*B))*((g_e_NL2(i+1,j)-g_e_NL2(i,j))./(xmesh(i+1)-xmesh...
            (1))); % g_e_NL2 = g_e_NL5
        g_e_NL6_lb_lower(i,j) = (1/(2*pi*B))*((g_e_NL2_lower(i+1,j)-g_e_NL2_lower(i,j))./(...
            xmesh(i+1)-xmesh(1)));

        % Next to left boundary
        i=2;
        j=1:length(ymesh);
        g_e_NL6_lb2(i,j) = (1/(2*pi*B))*((g_e_NL2(i+1,j)-g_e_NL2(i-1,j))./(xmesh(i+1)-...
            xmesh(i-1)));
        g_e_NL6_lb2_lower(i,j) = (1/(2*pi*B))*((g_e_NL2_lower(i+1,j)-g_e_NL2_lower(i-1,j))...
            ./(xmesh(i+1)-xmesh(i-1)));

        g_e_NL6_lb2 = g_e_NL6_lb2(2,:);
        g_e_NL6_lb2_lower = g_e_NL6_lb2_lower(2,:);

        % Rigth boundary
        i=length(xmesh);
        j=1:length(ymesh);
        g_e_NL6_rb(i,j) = (1/(2*pi*B))*((g_e_NL2(i,j)-g_e_NL2(i-1,j))./(xmesh(i)-xmesh(i...
            -1)));
        g_e_NL6_rb_lower(i,j) = (1/(2*pi*B))*((g_e_NL2_lower(i,j)-g_e_NL2_lower(i-1,j))./(...
            xmesh(i)-xmesh(i-1)));

        g_e_NL6_rb = g_e_NL6_rb(end,:);
        g_e_NL6_rb_lower = g_e_NL6_rb_lower(end,:);

        % Next to right boundary
        i=length(xmesh)-1;
        j=1:length(ymesh);
        g_e_NL6_rb2(i,j) = (1/(2*pi*B))*((g_e_NL2(i+1,j)-g_e_NL2(i-1,j))./(xmesh(i+1)-...
            xmesh(i-1)));
        g_e_NL6_rb2_lower(i,j) = (1/(2*pi*B))*((g_e_NL2_lower(i+1,j)-g_e_NL2_lower(i-1,j))...
            ./(xmesh(i+1)-xmesh(i-1)));

        g_e_NL6_rb2 = g_e_NL6_rb2(end,:);
        g_e_NL6_rb2_lower = g_e_NL6_rb2_lower(end,:);

        % Total computational domain
        g_e_nl = [g_e_NL6_lb; g_e_NL6_lb2; g_e_nl(3:end,:); g_e_NL6_rb2; g_e_NL6_rb];
        g_e_nl_lower = [g_e_NL6_lb_lower; g_e_NL6_lb2_lower; g_e_nl_lower(3:end,:); ...
            g_e_NL6_rb2_lower; g_e_NL6_rb_lower];
        g_e_nl_tot_u = g_e_nl + g_e_nl_lower;

        % Lower side airfoil
        % Effect lower domain on lower side airfoil
        g_e_NL_l = zeros(length(xmesh),length(ymesh_l),length(xmesh),length(ymesh_l));
        for k=1:length(xmesh)
            for l=1:length(ymesh_l)
            g_e_NL_l(:,:,k,l) = ((xmesh(k)-xmesh_modified)./((xmesh(k)-xmesh_modified)....
                ^2+(ymesh_l(l)-ymesh_l_modified).^2)).*u_l.*g_x_l;
            end
        end

        % Effect upper domain on lower side airfoil
        g_e_NL_l_upper = zeros(length(xmesh),length(ymesh_l),length(xmesh),length(ymesh_l)...
            );
        for k=1:length(xmesh)
            for l=1:length(ymesh_l)
            g_e_NL_l_upper(:,:,k,l) = ((xmesh(k)-xmesh_modified)./((xmesh(k)-...
                xmesh_modified).^2+(ymesh(l)-ymesh_l_modified).^2)).*u.*g_x;%u_l.*g_x_l; (...
                ymesh(l)-ymesh_modified).^2
            end
        end

        g_e_NL_l(isnan(g_e_NL_l))=0;
        g_e_NL_l_upper(isnan(g_e_NL_l_upper))=0;

        g_e_NL2_l = zeros(length(xmesh),length(ymesh_l));
        for k=2:2:length(xmesh)-1
            for l=1:2:length(ymesh_l)-3
                g_e_NL2_l = g_e_NL2_l + ((1/36)*(g_e_NL_l(:,:,k-1,l)+g_e_NL_l(:,:,k+1,l)+...
                    g_e_NL_l(:,:,k-1,l+2)+g_e_NL_l(:,:,k+1,l+2)+4*(g_e_NL_l(:,:,k,l)+...
                    g_e_NL_l(:,:,k-1,l+1)+g_e_NL_l(:,:,k+1,l+1)+g_e_NL_l(:,:,k,l+2))+16*...
                    g_e_NL_l(:,:,k,l+1))).*dxi6(:,:,k).*dz3(:,:,l);
            end
        end

        g_e_NL2_l_upper = zeros(length(xmesh),length(ymesh_l));
        for k=2:2:length(xmesh)-1
            for l=1:2:length(ymesh_l)-3
                g_e_NL2_l_upper = g_e_NL2_l_upper + ((1/36)*(g_e_NL_l_upper(:,:,k-1,l)+...
                    g_e_NL_l_upper(:,:,k+1,l)+g_e_NL_l_upper(:,:,k-1,l+2)+g_e_NL_l_upper...
                    (:,:,k+1,l+2)+4*(g_e_NL_l_upper(:,:,k,l)+g_e_NL_l_upper(:,:,k-1,l+1)+...
                    g_e_NL_l_upper(:,:,k+1,l+1)+g_e_NL_l_upper(:,:,k,l+2))+16*...
                    g_e_NL_l_upper(:,:,k,l+1))).*dxi6(:,:,k).*dz3(:,:,l);
            end
        end

        % Middle region
        i=3:length(xmesh)-2;
        j=1:length(ymesh_l);
        g_e_NL6_l(i,j) = -(g_e_NL2_l(i-2,j)+8*(g_e_NL2_l(i+1,j)-g_e_NL2_l(i-1,j))-...
            g_e_NL2_l(i+2,j));
        g_e_NL6_l_upper(i,j) = -(g_e_NL2_l_upper(i-2,j)+8*(g_e_NL2_l_upper(i+1,j)-...
            g_e_NL2_l_upper(i-1,j))-g_e_NL2_l_upper(i+2,j));

        g_e_nl_l = (1/(2*pi*B))*(g_e_NL6_l./(12*d_xmesh3));
        g_e_nl_l_upper = (1/(2*pi*B))*(g_e_NL6_l_upper./(12*d_xmesh3));

        % Left boudary
        i=1;
        j=1:length(ymesh);
        g_e_NL6_lb_l(i,j) = (1/(2*pi*B))*((g_e_NL2_l(i+1,j)-g_e_NL2_l(i,j))./(xmesh(i+1)-...
            xmesh(1)));
        g_e_NL6_lb_l_upper(i,j) = (1/(2*pi*B))*((g_e_NL2_l_upper(i+1,j)-g_e_NL2_l_upper(i,...
            j))./(xmesh(i+1)-xmesh(1)));

        % Next to left boundary
        i=2;
        j=1:length(ymesh_l);
        g_e_NL6_lb2_l(i,j) = (1/(2*pi*B))*((g_e_NL2_l(i+1,j)-g_e_NL2_l(i-1,j))./(xmesh(i...
            +1)-xmesh(i-1)));
        g_e_NL6_lb2_l_upper(i,j) = (1/(2*pi*B))*((g_e_NL2_l_upper(i+1,j)-g_e_NL2_l_upper(i...
            -1,j))./(xmesh(i+1)-xmesh(i-1)));

        g_e_NL6_lb2_l = g_e_NL6_lb2_l(2,:);
        g_e_NL6_lb2_l_upper = g_e_NL6_lb2_l_upper(2,:);

        % Rigth boundary
        i=length(xmesh);
        j=1:length(ymesh_l);
        g_e_NL6_rb_l(i,j) = (1/(2*pi*B))*((g_e_NL2_l(i,j)-g_e_NL2_l(i-1,j))./(xmesh(i)-...
            xmesh(i-1)));
        g_e_NL6_rb_l_upper(i,j) = (1/(2*pi*B))*((g_e_NL2_l_upper(i,j)-g_e_NL2_l_upper(i-1,...
            j))./(xmesh(i)-xmesh(i-1)));

        g_e_NL6_rb_l = g_e_NL6_rb_l(end,:);
        g_e_NL6_rb_l_upper = g_e_NL6_rb_l_upper(end,:);

        % Next to right boundary
        i=length(xmesh)-1;
        j=1:length(ymesh_l);
        g_e_NL6_rb2_l(i,j) = (1/(2*pi*B))*((g_e_NL2_l(i+1,j)-g_e_NL2_l(i-1,j))./(xmesh(i...
            +1)-xmesh(i-1)));
        g_e_NL6_rb2_l_upper(i,j) = (1/(2*pi*B))*((g_e_NL2_l_upper(i+1,j)-g_e_NL2_l_upper(i...
            -1,j))./(xmesh(i+1)-xmesh(i-1)));

        g_e_NL6_rb2_l = g_e_NL6_rb2_l(end,:);
        g_e_NL6_rb2_l_upper = g_e_NL6_rb2_l_upper(end,:);

        % Total computational domain
        g_e_nl_l = [g_e_NL6_lb_l; g_e_NL6_lb2_l; g_e_nl_l(3:end,:); g_e_NL6_rb2_l; ...
            g_e_NL6_rb_l];
        g_e_nl_l_upper = [g_e_NL6_lb_l_upper; g_e_NL6_lb2_l_upper; g_e_nl_l_upper(3:end,:)...
            ; g_e_NL6_rb2_l_upper; g_e_NL6_rb_l_upper];
        g_e_nl_tot_l = g_e_nl_l + g_e_nl_l_upper;
        %% Total g_xi

        % Upper side airfoil
        g_e = g_e_t+g_e_l-g_e_nl_tot_u;

        g_x_cell{len+1,1} = g_x_cell{len,1} + d * (g_e -g_x_cell{len,1});

        % Lower side airfoil
        g_e_ls = g_e_t_l+g_e_l_l-g_e_nl_tot_l;

        g_x_cell_l{len+1,1} = g_x_cell_l{len,1} + d * (g_e_ls -g_x_cell_l{len,1});
    end                                                                            % ...
        Inner-loop, update for g_xi till it converges

    %% Increment in u

    % Upper side airfoil
    du_tot = g_x_cell{end,1}*de;

    % Lower side airfoil
    du_tot_l = g_x_cell_l{end,1}*de;

    %% Final u

    % Upper side airfoil
    u_tot = u_cell{len2,1}+du_tot;

    u_cell{len2+1,1} = u_tot;
    g_x_cell{1,1} = g_x_cell{n+1,1};

    % Lower side airfoil
    u_tot_l = u_cell_l{len2,1}+du_tot_l;

    u_cell_l{len2+1,1} = u_tot_l;
    g_x_cell_l{1,1} = g_x_cell_l{n+1,1};
end

%% Pressure coefficient Cp
Cp_tot = (-((2)/(B*nu))*u_cell{m,1}(:,1));
Cp_tot_l = (-((2)/(B*nu))*u_cell_l{m,1}(:,1));
dCp = (Cp_tot - Cp_tot_l);

h_u =   (2*(12*de)*(1-xmesh(19:69).^2)-0*xmesh(19:69))+1;
h_l = (- 2*(12*de)*(1-xmesh(19:69).^2)-0*xmesh(19:69))+1;

figure(1)
plot(xmesh(19:69),Cp_tot_l(19:69),'b--s') %20:68
grid on
```

```
hold on
plot(xmesh(19:69),Cp_tot(19:69),'r--s')
grid on
hold on
h = plot(xmesh(19),Cp_tot_l(19),'kd',xmesh(19),Cp_tot(19),'kd',xmesh(47),Cp_tot_l(47),'kd'...
    ,xmesh(40),Cp_tot(40),'kd',xmesh(69),Cp_tot_l(69),'kd');
set(h,'MarkerSize', 10, 'MarkerFaceColor','k')
grid on
hold on
plot(xmesh(19:69),h_u,xmesh(19:69),h_l)
axis ij
title('C_p distribution for M_{\infty} = 0.86 and \alpha=1^{\circ} ')
xlabel('x')
ylabel('C_p [-]')
t1 = text(xmesh(19)+0.1,Cp_tot_l(19), ['C_{P_{{LE}_l}} =' sprintf('%.02f',Cp_tot_l(19))]);...
     %22
t2 = text(xmesh(19)+0.19,Cp_tot_l(19), ['x =' sprintf('%.02f',xmesh(19))]); %24
t3 = text(xmesh(69)-0.13,Cp_tot_l(69), ['C_{P_{TE}} =' sprintf('%.02f',Cp_tot_l(69))]); %...
    66
t4 = text(xmesh(69)-0.12,Cp_tot_l(69), ['x =' sprintf('%.02f',xmesh(69))]); %66
t5 = text(xmesh(47),Cp_tot_l(47)+0.15, ['C_{P_{{min}_l}} =' sprintf('%.02f',Cp_tot_l(47))...
    ]);
t6 = text(xmesh(47)+0.05,Cp_tot_l(47)+0.15, ['x =' sprintf('%.02f',xmesh(47))]);
t7 = text(xmesh(19)-0.2,Cp_tot(19)-0.03, ['C_{P_{{LE}_u}} =' sprintf('%.02f',Cp_tot(19))])...
    ;
t8 = text(xmesh(19)-0.15,Cp_tot(19)-0.03, ['x =' sprintf('%.02f',xmesh(19))]);
t9 = text(xmesh(40),Cp_tot(40)-0.08, ['C_{P_{{min}_u}} =' sprintf('%.02f',Cp_tot(40))]);
t10 = text(xmesh(40)+0.05,Cp_tot(40)-0.08, ['x =' sprintf('%.02f',xmesh(40))]);
t1.HorizontalAlignment = 'left';      % set horizontal alignment to center
t1.VerticalAlignment = 'bottom';      % set vertical alignment to top
t1.FontSize = 11;                     % make the text larger
t1.FontWeight = 'bold';
t2.HorizontalAlignment = 'left';      % set horizontal alignment to center
t2.VerticalAlignment = 'top';         % set vertical alignment to top
t2.FontSize = 11;                     % make the text larger
t2.FontWeight = 'bold';
t3.HorizontalAlignment = 'right';     % set horizontal alignment to center
t3.VerticalAlignment = 'bottom';      % set vertical alignment to top
t3.FontSize = 11;                     % make the text larger
t3.FontWeight = 'bold';
t4.HorizontalAlignment = 'right';     % set horizontal alignment to center
t4.VerticalAlignment = 'top';         % set vertical alignment to top
t4.FontSize = 11;                     % make the text larger
t4.FontWeight = 'bold';
t5.HorizontalAlignment = 'center';    % set horizontal alignment to center
t5.VerticalAlignment = 'bottom';      % set vertical alignment to top
t5.FontSize = 11;                     % make the text larger
t5.FontWeight = 'bold';
t6.HorizontalAlignment = 'center';    % set horizontal alignment to center
t6.VerticalAlignment = 'top';         % set vertical alignment to top
t6.FontSize = 11;                     % make the text larger
t6.FontWeight = 'bold';
t7.HorizontalAlignment = 'center';    % set horizontal alignment to center
t7.VerticalAlignment = 'bottom';      % set vertical alignment to top
t7.FontSize = 11;                     % make the text larger
t7.FontWeight = 'bold';
t8.HorizontalAlignment = 'center';    % set horizontal alignment to center
t8.VerticalAlignment = 'top';         % set vertical alignment to top
t8.FontSize = 11;                     % make the text larger
t8.FontWeight = 'bold';
t9.HorizontalAlignment = 'center';    % set horizontal alignment to center
t9.VerticalAlignment = 'bottom';      % set vertical alignment to top
t9.FontSize = 11;                     % make the text larger
t9.FontWeight = 'bold';
t10.HorizontalAlignment = 'center';   % set horizontal alignment to center
t10.VerticalAlignment = 'top';        % set vertical alignment to top
t10.FontSize = 11;                    % make the text larger
t10.FontWeight = 'bold';
ylim([-0.6 1.1])
legend('C_p lower side airfoil','C_p upper side airfoil','Location','northeast');
OutputName = 'Cp_aoa1_M086_matlab';
```

```
saveas(figure(1), ['/Users/emelcankaya/Desktop/Thesis/Thesis/Pictures/' OutputName '.jpg'...
    ]);

toc
```

F
$NumecaFine^{TM}/Open$ SETTINGS

This appendix serves as a manual for the application of $NumecaFine^{TM}/Open$ to the study presented in this report. All steps required in computing the pressure coefficient distribution along a symmetric parabolic arc airfoil will be explained here.

The first window that opens up when $Numeca$ is started is the 'Options' window asking whether one wants to import a mesh, create a mesh or open an existing project. Choosing for the second option, thus creating a mesh, a folder has to be defined to which all files will saved together with the name of the project. After having completed these steps, $HEXPRESS$ will be started automatically. This is an unstructured, full hexahedral, body fitted grid generator for arbitrary geometries. The first thing this generator asks for is to import the object on which the computations have to be performed, as can be seen in Figure F.3a. This object can be imported as a Parasolid model ($< .x_t >$) or a CATIA V5 model ($< .CATPart >$). For the current study, the symmetrical parabolic arc airfoil the computations will be performed on is created within $CATIA$. However, due to license restrictions it was not possible to import the object as a $CATIA$ model. Therefore, the object had to be saved as a Parasolid model, but this is not possible within $CATIA$. Therefore, another design tool called $SolidWorks$ has been used to save the object which was created within $CATIA$ as a Parasolid.
Once the object has been imported, a domain has to be created. For this study, two options were tested: creating both the airfoil and domain within $CATIA$, and creating the airfoil in $CATIA$ but the domain in $NUMECA$. For the second case, the computational domain around the airfoil has been created by first drawing a box around it. This is done by selecting the 'Create box' option from the 'Create/Edit' window of the CAD Manipulation menu on the top left corner of the screen, as can be seen in Figure F.3b. Then, the airfoil has been subtracted from this box with the 'Subtract' button from the same 'Create/Edit' window, so the domain on which the mesh will be created becomes one object (with the hole of the airfoil in it). This is displayed in the 'Visualization/Selection' window on the left side of the screen, below the 'Create/Edit' window. The next step is to create the domain with the 'Create Domain' button, which means that the domain will be divided into triangles. However, before doing so the Faceting settings have to be set. For this study, the following values have been used:

- Minimum length: 1e-007
- Maximum length: 1
- Curve chordal tolerance: 1e-007
- Surface plane tolerance: 1
- Curve resolution: 3.0
- Surface resolution: 3.0
- Export parameters
  Merge tolerance: 1e-020

After having chosen the faceting values, first the 'Apply' and then the 'Create' buttons have been pressed. The created domain has to be saved to the folder defined before. *HEXPRESS* then asks whether or not to import the domain, to which the answer should be 'yes'. Once the domain is imported, the boundary conditions have to be set by selecting the cubic 'BC' icon from the bar on the top of the screen. This can be seen in Figure F.15c. The boundary conditions are set as follows:

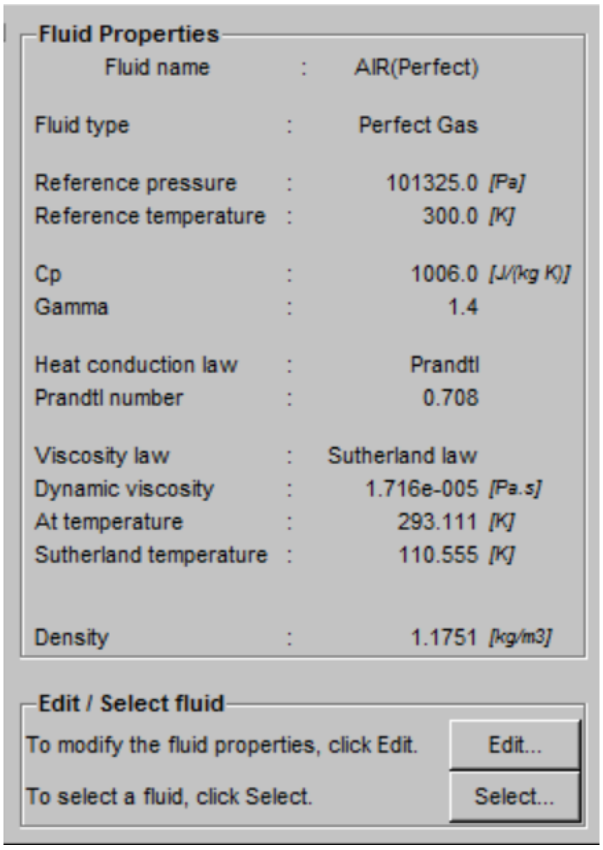
- The upper and lower surfaces of the airfoil are set to be of 'Solid' (SOL) face type.
- The two planes in the same ($x - z$ plane) as the chord of the airfoil are set to be of 'Mirror' (MIR) type.
- The remaining four faces are set to be of 'External' (EXT) face type.

Then , the Grid generation mode has to be set to 2D (since two-dimensional flow has to be solved for this study). This mode is selected from the bar on top of the screen, to the right of the 'BC' icon. The Mesh wizard menu on the left of the screen will be finished off. This menu consists of five steps, can be seen in Figure F.15c. These steps, and the according settings, are as follows:

- **Initial mesh** → Initial mesh parameters → Subdivide the domain bounding box → Set the number of cells along Cartesian axes:
  - X axis: 80
  - Y axis: 80
  - Nb of Cells: 6400
- **Adapt to geometry:**
  Refinement and trimming parameters:
  - *Global*
    Refinement:
    Maximum number of refinements:
  - *Curve refinement:* set to be inactive
  - *Surface refinement*
    Refinement
    Maximum nb of refinements: 7
    Adaptation criteria
    Target cell sizes:
    - X axis: 0.01
    - Y axis: 0.01
  - *Box refinement*
    Active: Draw boxes around the leading and trailing edges of the airfoil.
    - Refinement
      Max. nb of refinements: 7
    - Target cell sizes
      X axis: 0.005
      Y axis: 0.005
  - *Trimming:* inactive
- **Snap to geometry**
- **Optimize**
- **Viscous layers:** not selected

After completing all these steps the domain has been created. *HEXPRESS* can be closed at this point, whereafter *Fine*/*Open* will be opened automatically. This is the Navier-Stokes solver, and can be seen in Figure F.16. Before starting the computations, the project parameters have to be set. The project parameter menu is subdivided into seven parts, and will be run through now.

- **Physical Configurations**
  - **General Properties**
    Time configuration: Steady vs. Unsteady → Steady selected
    Black properties: Fluid block vs. Solid block → Fluid block selected
    Optional properties: Combustion, Radiation, Porous → not selected, since they are not applicable to the current study
  - **Fluid Model:** The fluid properties for the current study can be found in Figure F.1.



Fluid Properties

Fluid name	:	AIR(Perfect)
Fluid type	:	Perfect Gas
Reference pressure	:	101325.0 [Pa]
Reference temperature	:	300.0 [K]
Cp	:	1006.0 [J/(kg K)]
Gamma	:	1.4
Heat conduction law	:	Prandtl
Prandtl number	:	0.708
Viscosity law	:	Sutherland law
Dynamic viscosity	:	1.716e-005 [Pa.s]
At temperature	:	293.111 [K]
Sutherland temperature	:	110.555 [K]
Density	:	1.1751 [kg/m3]

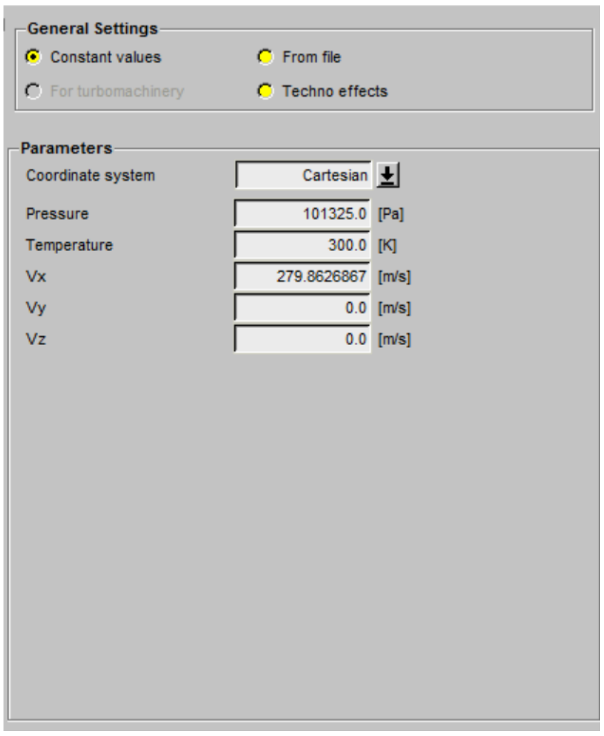
Edit / Select fluid

To modify the fluid properties, click Edit.

To select a fluid, click Select.

Figure F.1: Fluid Model: fluid properties used for this study

  - **Flow Model**
    - *Mathematical Model*
      Flow model: Euler, Laminar Navier-Stokes, Turbulent Navier-Stokes → Euler selected
      Gravity forces: inactive
      Low speed Flow ($M < 0.3$): inactive
    - *Reference Parameters*
      Reference length: 2.0 [m] → which is the chord length
      Reference velocity: $[\frac{m}{s}]$ → depends on the freestream Mach number and can be calculated with $v = M_\infty * a$
      Reynolds: 3.9142E+007 (not set by the user)
  - **Solid Model:** There's no block of solid type, so not applicable for this study
  - **Rotating Machinery:** Not applicable
  - **Heat Source:** No active heat source defined
  - **Harmonic fModel:** The 'Non-linear Harmonic' Module is not activated
- **Optional Models:** Combustion, Radiation, Porous Media, Coupling, Fluid-Particle Interaction, Non-Deterministic Data → not activate, since they are not applicable to this study



General Settings

[✓ Constant values] From file

For turbomachinery Techno effects

Parameters

Coordinate system	Cartesian	
Pressure	101325.0	[Pa]
Temperature	300.0	[K]
Vx	279.8626867	[m/s]
Vy	0.0	[m/s]
Vz	0.0	[m/s]

Figure F.2: Initial Solution settings

- **Boundary Conditions:** consists of 'Solid' and 'External', as set before. Both boundary condition types, and their settings, can be seen in Figure F.17. The terms $arctg(Vy/Vx)$ and $arctg(Vz/Vx)$ in Figure F.17b define the angle of attack of the airfoil.

- **Initial Solution:** the initial solution parameters have been set here, as can be seen in Figure F.2. The same values of pressure and temperature as in Figure F.1 have been chosen. Also, it should be noted there there's only a velocity component (of the freestream flow) in the x-direction. The angles of attack (when the airfoil is set under an angle of attack) have to be indicated in the 'Boundary Conditions' settings, as can be seen in Figure F.17b.

- **Numerical Parameters**

  – *General Prameters* → Multigrid Parameters
    Number of grid(s):
    Correction damping: active
    Coarse grid initialization: active

  – *Numerical Schemes*
    Discretization scheme: Central
    Scheme accuracy: 2nd order
    CFL number: 3

  – *Mesh Adaptation*
    Number of adaptations: 0

- **Computational Control:**

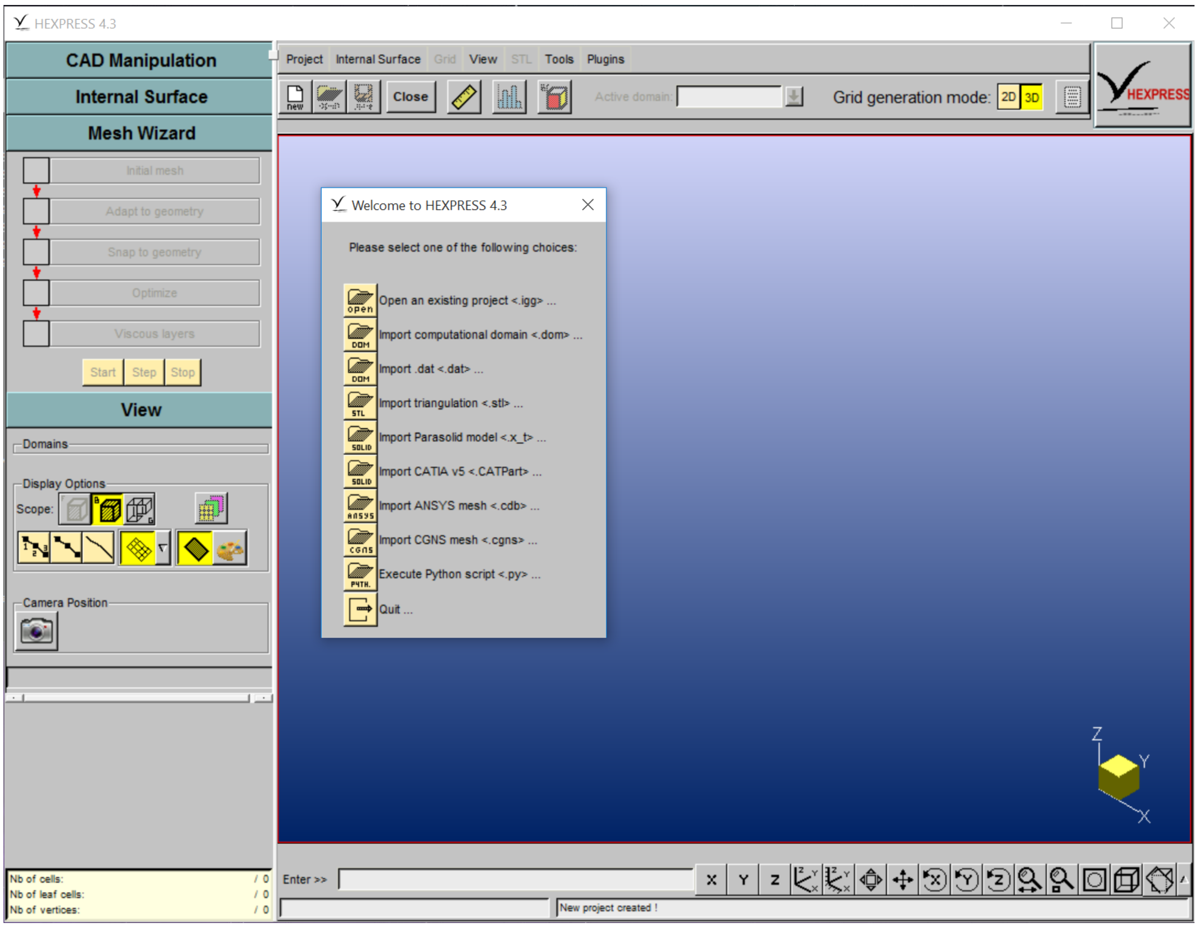
  – *Outputs:* this consists of the general outputs, external flow outputs, and advanced outputs. They can be viewed in Figure F.18.

- *Control Variables:*
  Parameters
  Number of iterations: 1000

Convergence criteria: -6.0
Save solution every: 100

- *Launching Mode:* Serial vs. Parallel → Serial selected
- *Ansys Output:* inactive
- *Non-Deterministic Results:* inactive

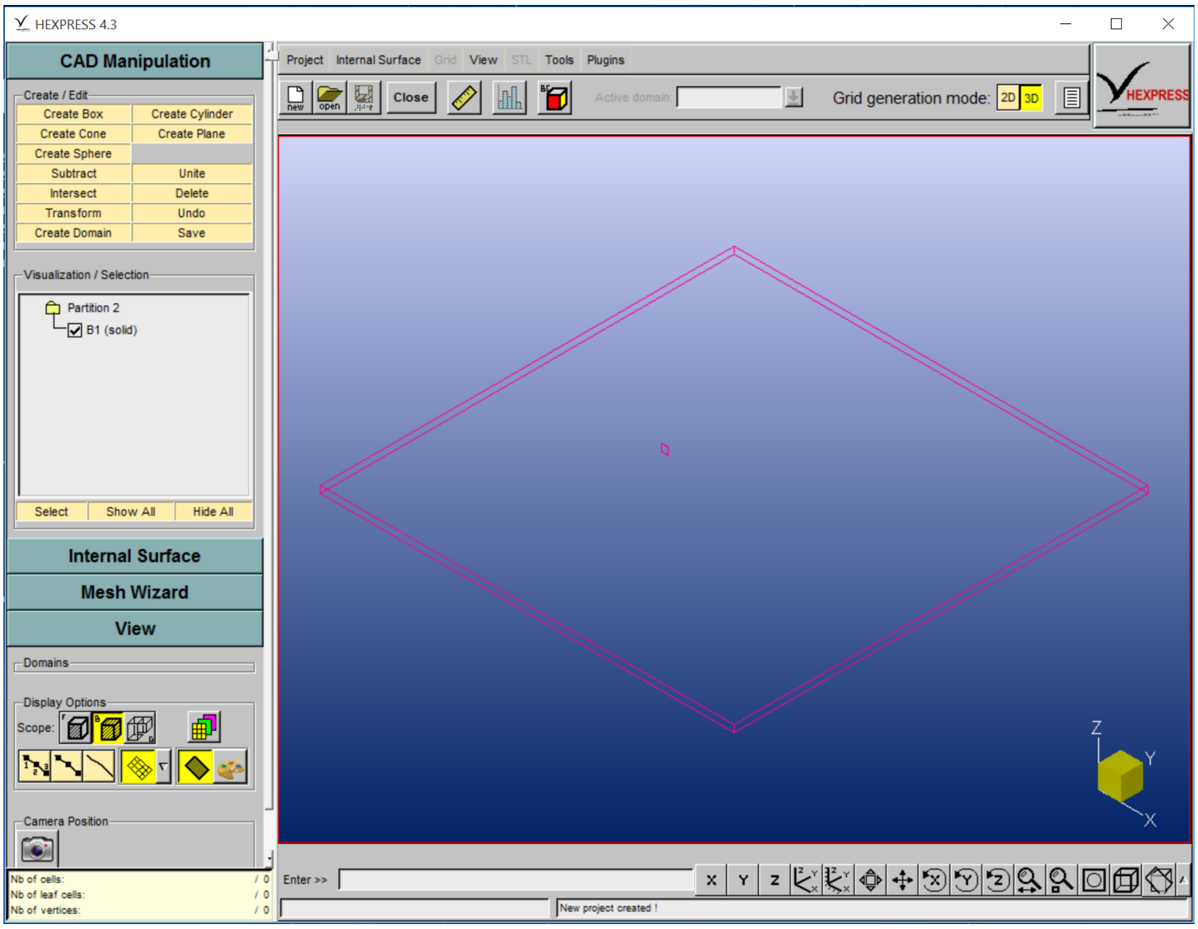
domain
number of mesh cells setup simulation settings



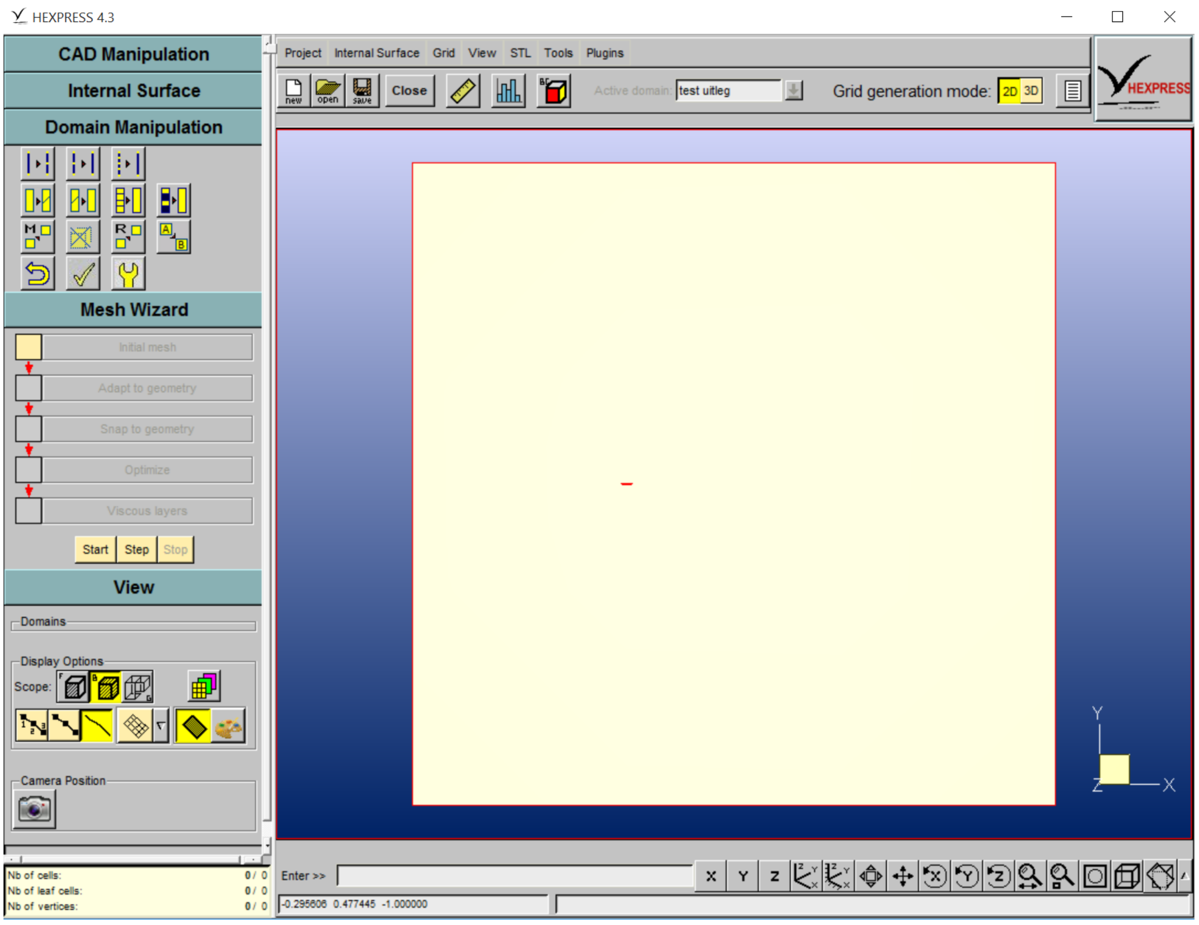
Please select one of the following choices:

Open an existing project <.igg> ...
Import computational domain <.dom> ...
Import .dat <.dat> ...
Import triangulation <.stl> ...
Import Parasolid model <.x_t> ...
Import CATIA v5 <.CATPart> ...
Import ANSYS mesh <.cdb> ...
Import CGNS mesh <.cgns> ...
Execute Python script <.py> ...
Quit ...

(a) *HEXPRESS* - Object importing options



(b) *HEXPRESS* - Domain creating options



Active domain: test uitleg

Grid generation mode: 2D 3D

Nb of cells: 0 / 0
Nb of leaf cells: 0 / 0
Nb of vertices: 0 / 0

-0.295606 0.477445 -1.000000

(c) *HEXPRESS* - Meshing procedure

Figure F.15: *HEXPRESS* - Grid Generator

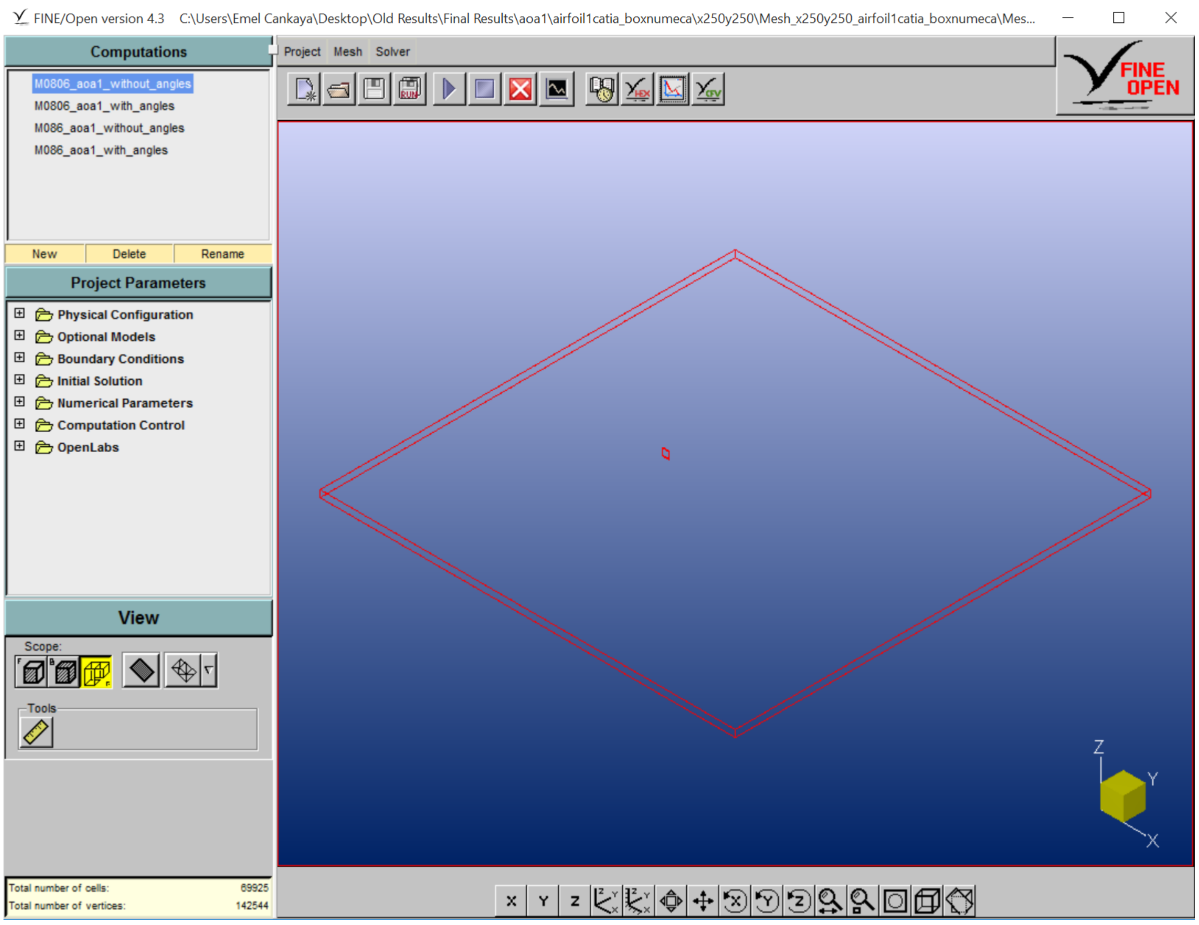
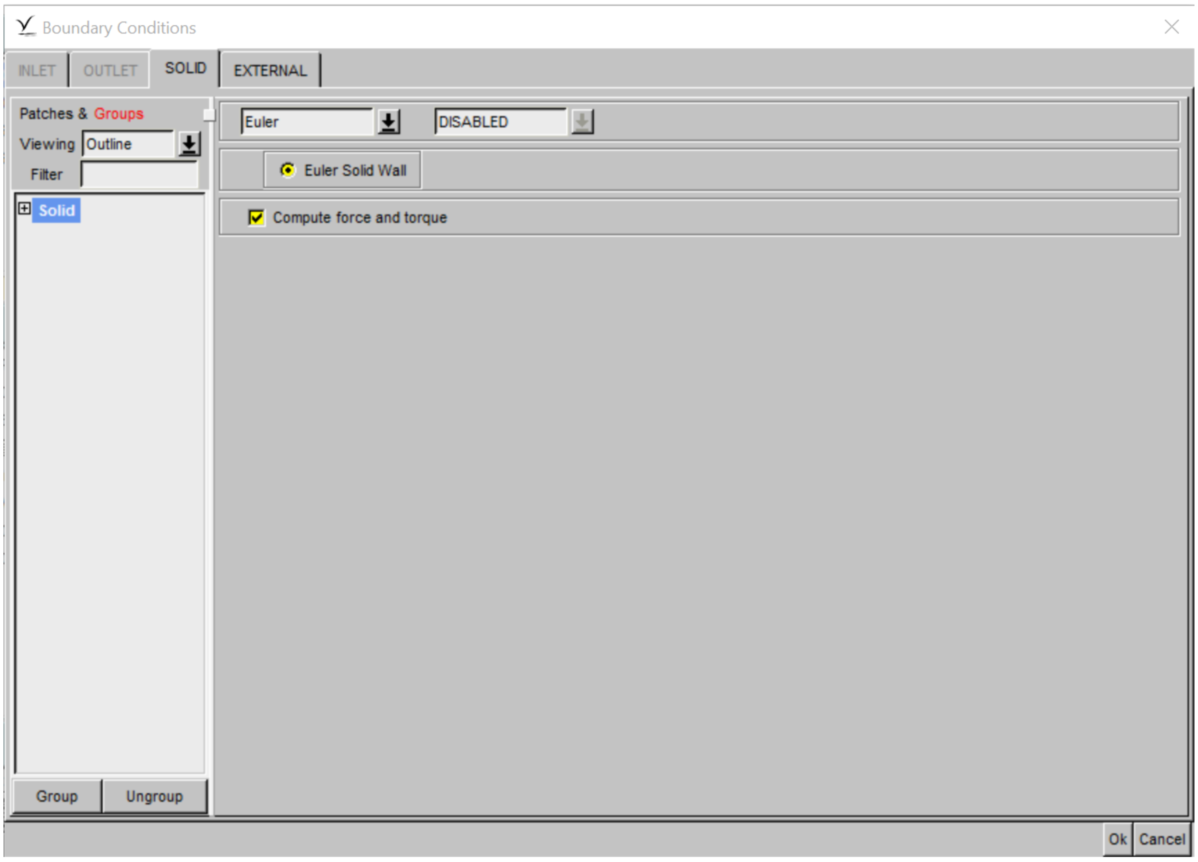


Figure F.16: *Fine*/*Open* - Solver



Viewing: Outline

Filter:

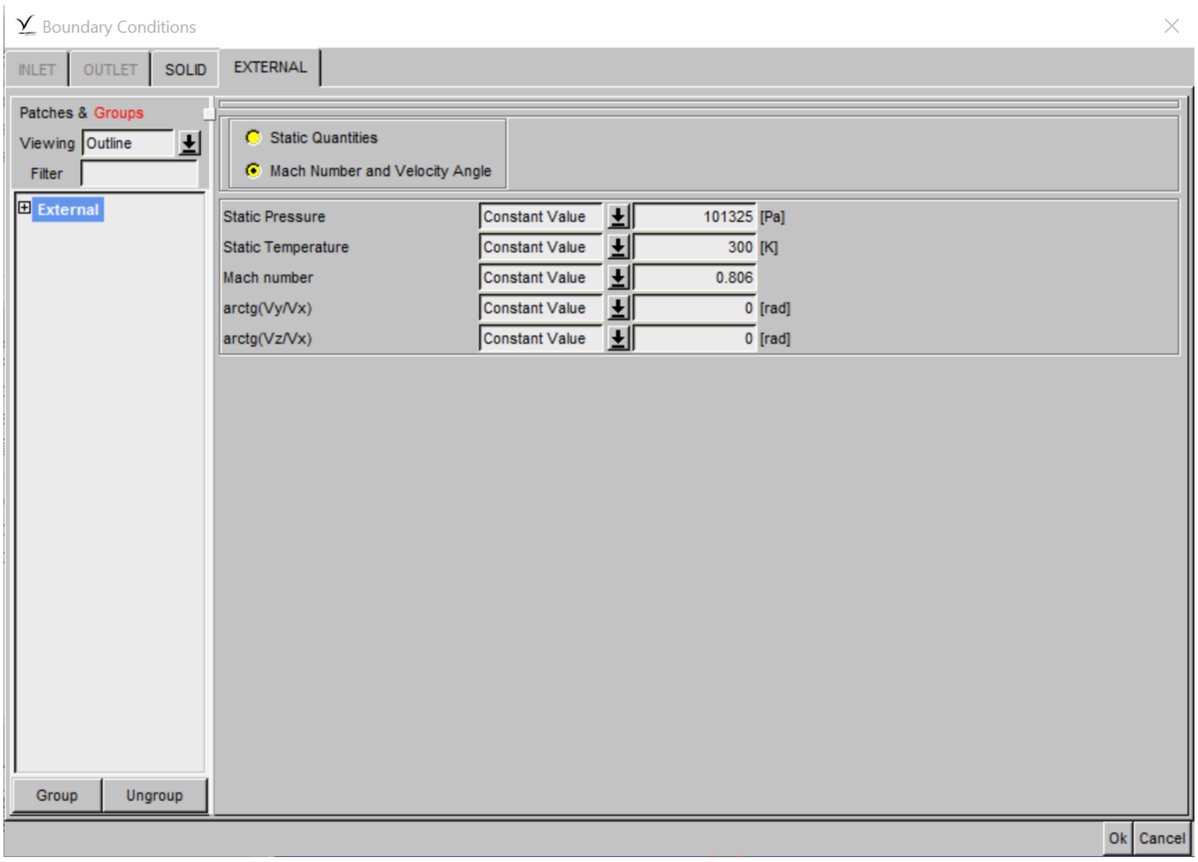
Solid

Euler | DISABLED

[✓] Euler Solid Wall

[✓] Compute force and torque

(a) BC - Solid



Viewing: Outline

Filter:

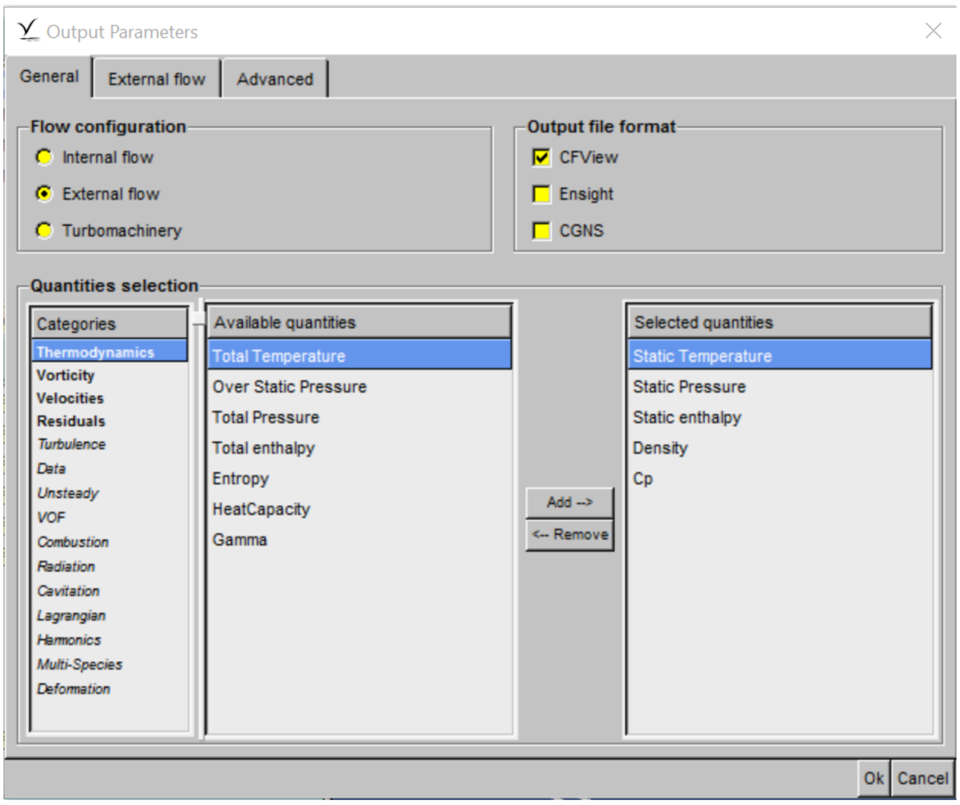
External

Static Quantities [✓ Mach Number and Velocity Angle]

Static Pressure	Constant Value	101325	[Pa]
Static Temperature	Constant Value	300	[K]
Mach number	Constant Value	0.806	
arctg(Vy/Vx)	Constant Value	0	[rad]
arctg(Vz/Vx)	Constant Value	0	[rad]

(b) BC - External

Figure F.17: Boundary Condition settings



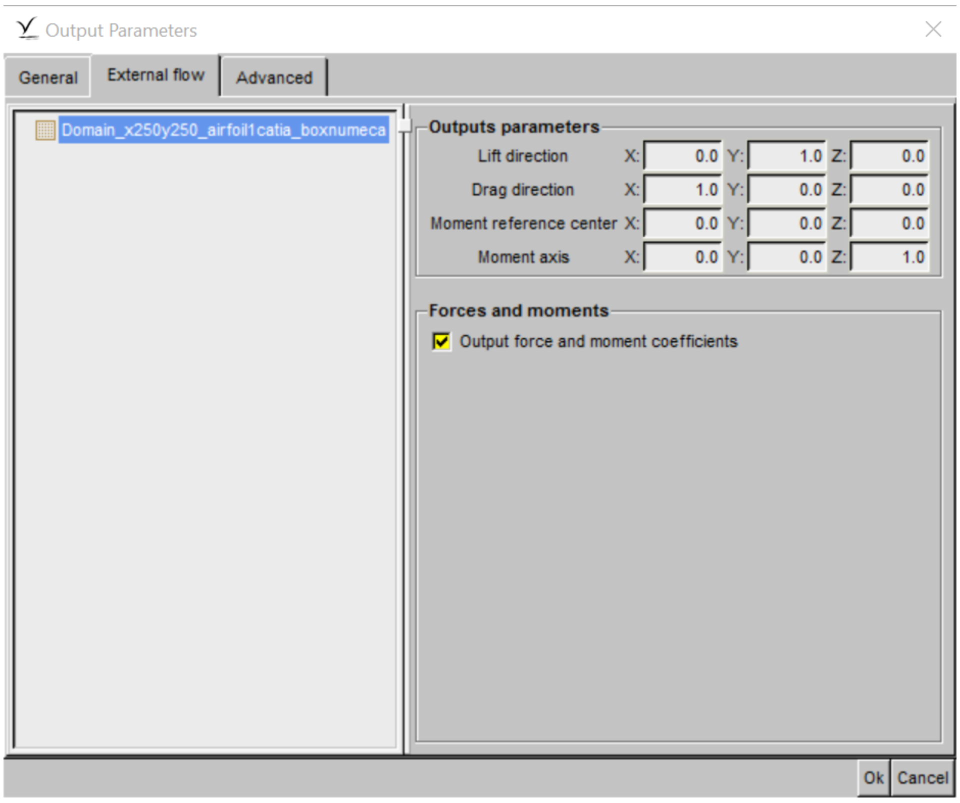
Flow configuration
- Internal flow
- [✓ External flow]
- Turbomachinery

Output file format
- [✓ CFView]
- Ensight
- CGNS

Quantities selection

Categories	Available quantities	Selected quantities
Thermodynamics	Total Temperature	Static Temperature
Vorticity	Over Static Pressure	Static Pressure
Velocities	Total Pressure	Static enthalpy
Residuals	Total enthalpy	Density
Turbulence	Entropy	Cp
Data	HeatCapacity	
Unsteady	Gamma	
VOF		
Combustion		
Radiation		
Cavitation		
Lagrangian		
Harmonics		
Multi-Species		
Deformation		

(a) Outputs - General



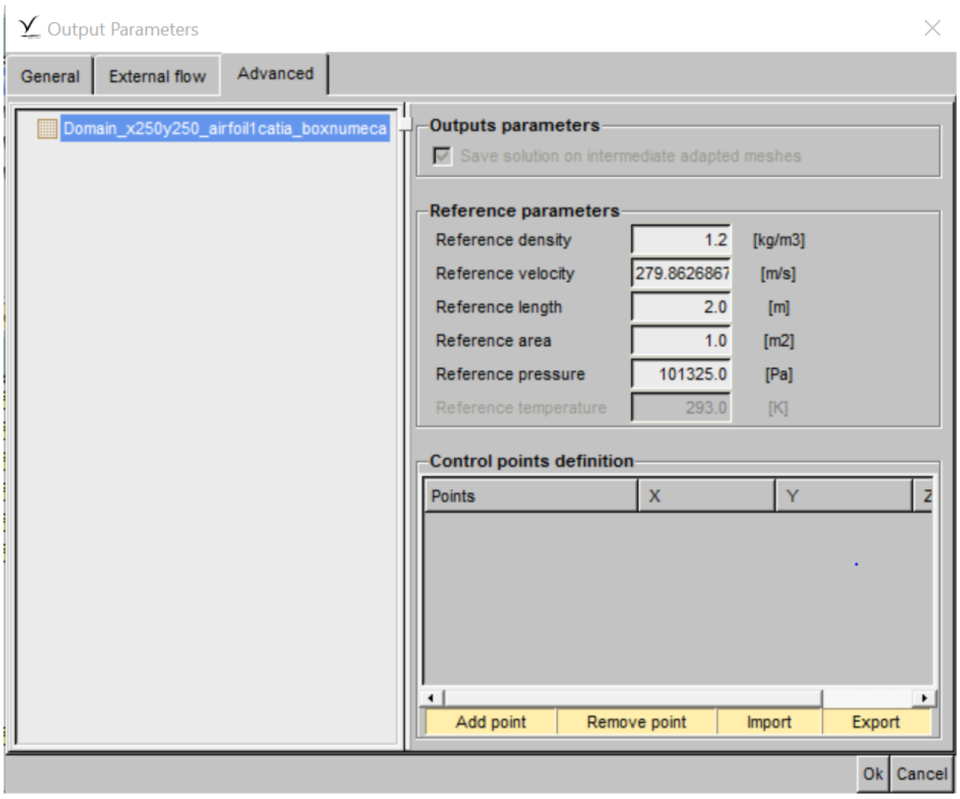
Domain_x250y250_airfoil1catia_boxnumeca

Outputs parameters

	X:	Y:	Z:
Lift direction	0.0	1.0	0.0
Drag direction	1.0	0.0	0.0
Moment reference center	0.0	0.0	0.0
Moment axis	0.0	0.0	1.0

Forces and moments
- [✓ Output force and moment coefficients]

(b) Outputs - External Flow


Output Parameters

General | External flow | Advanced

Domain_x250y250_airfoil1catia_boxnumeca

Outputs parameters

☑ Save solution on intermediate adapted meshes

Reference parameters

Reference density	1.2	[kg/m3]
Reference velocity	279.8626867	[m/s]
Reference length	2.0	[m]
Reference area	1.0	[m2]
Reference pressure	101325.0	[Pa]
Reference temperature	293.0	[K]

Control points definition

Points	X	Y	Z

(c) Outputs - Advanced

Figure F.18: Outputs

G
$NumecaFine^{TM}/Open$ Results

The pressure coefficient distributions resulting from $NumecaFine^{TM}/Open$, which are obtained for different setups, will be shown in this section.

$\alpha = 0°$

This section treats the pressure coefficient distribution of the symmetric parabolic arc airfoil under zero angle of attack. First, the results of the airfoil immersed in a flow with freestream Mach number $M_\infty = 0.806$ will be elaborated on, followed by the results for $M_\infty = 0.86$.

$M_\infty = 0.806$

The computations with $Numeca$ are performed for two different setups as explained in Appendix F. The results corresponding to the first setup, thus creating both the airfoil and borders of the domain within $CATIA$, are displayed on the left side of Figure G.7. The results of the second setup, i.e. creating the airfoil in $CATIA$ and the borders of the domain in $NUMECA$, can be seen on the right side of Figure G.7. The lift and drag values corresponding to those pressure distributions can be found in Table G.1.
Based on this figure, the following conclusions can be drawn:

- The shape and overall values of the C_P distributions don't change with changing amount of cells. The minimum value of the pressure coefficient is $C_{P_{min}} \approx -0.30$ for all test cases, at the location $x = 0$. The values of C_P at the leading and trailing edges of the airfoil, i.e. $x = -1$ and $x = 1$ respectively, do increase with increasing number of cells and converge to a value of $C_{P_{LE}} \approx 0.53$ and $C_{P_{TE}} \approx 0.47$ at the leading and trailing edges respectively. Another changing feature of the pressure coefficient distributions is that the plots seems to become thicker/darker when increasing the number of cells. This is due to the increasing number of points on which C_P is being calculated.

- The results of both setups are identical to each other, which means that it doesn't matter whether the domain is created within $CATIA$ or $NUMECA$.

- The pressure distributions for the upper and lower side of the airfoil lie on top of each other. This is as expected, since those pressure distributions are computed for a symmetric parabolic arc airfoil under zero angle of attack, for which zero lift is expected. In order to achieve zero lift, the difference in C_P between the upper and lower sides of the airfoil must be zero according to

$$C_L = \int (C_{P_l} - C_{P_u}) \mathrm{d}\frac{x}{c}$$

  In terms of the pressure coefficient distribution curve, this means that the C_P distributions of the upper and lower side of the airfoil should be identical to each other, and thus must lie on each other. Another remarkable feature is, as mentioned earlier, that the pressure coefficient increases from $C_{P_{LE}} \approx 0.35$ to $C_{P_{LE}} \approx 0.53$ at the leading edge and from $C_{P_{TE}} \approx 0.25$ to $C_{P_{TE}} \approx 0.47$ at the trailing edge. This increasing trend is an indication of reliable results, since typical C_P at the stagnation point in compressible flow is somewhat larger than unity [1]. Also, the lower C_P at the trailing edge (compared

to the leading edge) is an indication of good performance. This is a result of the Kutta condition of finite velocities, which means that on the trailing edge the flow on the upper surface (where the velocities are higher than on the lower surface) decelerates and converge with the flow on the lower surface. The pressure at the trailing edge is also related to the airfoil thickness and shape near the trailing edge. For thick airfoils the pressure here is slightly positive, as in the current study.
However, it is remarkable that the C_P values at the leading edge are far below unity. This could be due to numerical issues related to the absence of a real nose of the airfoil in consideration. The parabolic arc airfoil used to obtain the current results has a 'wedge' shaped nose, instead of a 'real' rounded nose. A sketch of this airfoil can be seen in Figure G.1. The nose of the airfoil in this figure can be found on the left side where the upper surface and lower surface coincide, meaning that it consists of only one point instead of a 'surface'. This is illustrated with Figure G.1. It is very difficult to capture the nose within the mesh and determine the velocity and C_P at exactly this point, which is indicated by the blue dot. The velocities slightly away from this point are being captured instead, which are indicated by the red dots. Since the velocities V at those points aren't zero, but a higher value (due to the acceleration of the flow along the airfoil), the determined C_P values are less than unity.

- The shape of the overall pressure distribution in all cases looks reasonable and is in accordance with a typical $\alpha 0°$ pressure distribution, which shows a decrease in pressure when moving away from the stagnation point near the leading edge, and a minimum pressure (i.e. called the suction peak) which depends on the airfoil shape and angle of attack. The thickest point of the airfoil mainly determines where the suction peak occurs [19]. The suction peak is followed by an increase in pressure towards the leading edge, which is called the pressure recovery [20].
  This trend can also be seen in the C_P distributions in Figure G.7. For now, this will be illustrated with reference to the pressure coefficient distribution of Figure G.5g due to reasons that will be made clear later on in this chapter. The C_P distribution in this figure starts out at the value of $C_{P_{LE}} \approx 0.45$ at the nose, and drops as the flow expands around the nose. The pressure P decreases below P_∞, yielding a minimum value of $C_{P_{min}} \approx -0.30$ downstream of the nose. This happens at the thickest point of the airfoil, which is at $x = 0$ for the parabolic arc airfoil under consideration. Further downstream the pressure tries to recover and approaches a value of $C_{P_{TE}} \approx 0.37$ at the trailing edge. Such a region of increasing pressure in the direction of the flow is called an adverse pressure gradient. A too severe adverse pressure gradient indicates boundary layer transition, and possibly separation. The value of C_P at the trailing edge is slightly smaller than the value at the leading edge due to the Kutta condition of finite velocities, which is satisfied at the trailing edge. Because of this condition, the velocity V at the trailing edge will always be slightly higher than the velocity at the stagnation point close to the leading edge. This causes C_P at the trailing edge to be slightly lower than C_P at the leading edge. However, the flow at the trailing edge is still attached, since the pressure coefficient of separated flow is $C_P < 0$. This is not the case for the current setup.

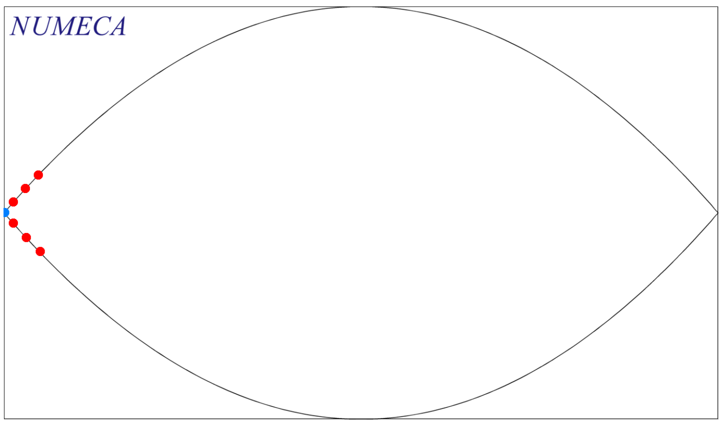


Figure G.1: Points on the airfoil where C_p is calculated

The values of the lift L, drag D, and computation time T corresponding to the C_P distributions in Figure G.7 can be seen in the three columns on the left side of Tables G.1. For simplicity, those values are visualized by inserting them into a graph with the number of the mesh cells on the horizontal axis, and the lift and drag on the vertical axis. The values corresponding to the current setup are indicated by the red lines/symbols in figures G.4a and G.4b. By focussing on Figure G.4a one can see that the values of the lift change abruptly and oscillate till the number of mesh cells of 250. After this point, the lift still oscillates, but with a much lower amplitude. It looks like the lift converges towards a certain value around $L \approx 666$ $[N]$. The drag values are analyzed in the same way, by focussing on Figure G.4b. From this figure it can be seen that the drag keeps decreasing, and that this decrease consists of three parts; number of cells 20-80, number of cells 100-750, and number of cells 750-1000. The slope coefficients corresponding to those regions are determined to be $\left(\frac{dL}{dx}\right)_{(\#cells:20-80)} = \frac{195.6-248.2}{80-20} = -0.8767$, $\left(\frac{dL}{dx}\right)_{(\#cells:100-750)} = \frac{83.4-195.8}{750-100} = -0.1729$, and $\left(\frac{dL}{dx}\right)_{(\#cells:750-1000)} = \frac{61.73-83.4}{1000-750} = -0.0867$. The highest decrease slope corresponds to the first region, while the lowest slope corresponds to the third region. Thus, from those results one can conclude that the drag coefficient keeps decreasing, with a decreasing slope. However, it is hard to predict to which value the drag is converging to, since there is no predictable and logical trend in the progression of the development of $\frac{dL}{dx}$, as can be seen from Figure G.2.

When looking at the values of those lift and drag data, rather than at whether they are converging, one can see that the values are striking. First of all, the lift values should be nearly zero for an airfoil under zero angle of attack. The values shown in Table G.1 and the corresponding Figure G.4a are far above $L = 0$, except for a number of cells equal to 20 and 80. Those high values are most likely caused by the asymmetrical meshes created for the number of cells 40,60,100,250,500,750, and 1000. These meshes can be seen in Figure G.3. Inspecting those figures, one can see in Figure G.3a that the meshes on the upper side and lower side of the airfoil are identical. This is not true for the mesh in Figure G.3b, where the meshes for the upper and lower side of the airfoil aren't exactly the same. This probably causes nonzero lift. However, when converting those lift values to lift coefficients, this situation becomes less dramatic. The lift coefficients C_L have been determined with Equation G.1, and the results are displayed in Table G.1. When looking at those C_L values, one can see that they are so small that they can be neglected. The same approach is applied for the discussion of the drag values. The drag coefficient C_D is determined with Equation G.2, and the values are shown in Table G.1. Again, the C_D turns out to be so small that they can be neglected.

Another parameter with which the performance of an airfoil can be assessed is the lift-to-drag ratio $\frac{L}{D}$. Those values can be found in Table G.1. Since a symmetric mesh is important for a proper simulation, and at the same time as many points on the airfoil have to be captured, the lift-to-drag ratio of the simulation corresponding to a number of mesh cells of 80 will be reviewed. This value turns out to be $\frac{L}{D} \approx -0.0047$, and is a reasonable value since the value of lift should be near zero for an airfoil under zero angle of attack.

	Domain created in *CATIA*						Domain created in *NUMECA*					
# mesh cells	L [N]	C_L [-]	D [N]	C_D [-]	$\frac{L}{D}$ [-]	Run Time	L [N]	C_L [-]	D [N]	C_D [-]	$\frac{L}{D}$ [-]	Run Time
20	-2.781e+000	-0.0000	2.482e+002	0.0013	-0.0112	0h 0m 50.582s	-3.156e+000	-0.0000	2.482e+002	0.0013	-0.0127	0h 0m 41.467s
40	1.258e+003	0.0068	2.191e+002	0.0012	5.7417	0h 1m 15.89s	-1.256e+003	-0.0068	2.191e+002	0.0012	-5.7325	0h 1m 6.384s
60	9.917e+002	0.0054	2.050e+002	0.0011	4.8376	0h 1m 46.874s	-9.909e+002	-0.0054	2.050e+002	0.0011	-4.8337	0h 1m 35.692s
80	-1.438e+000	-0.0000	1.956e+002	0.0011	-0.0074	0h 2m 9.705s	-2.156e+000	-0.0000	1.956e+002	0.0011	-0.0110	0h 2m 3.834s
100	8.673e+002	0.0047	1.958e+002	0.0011	4.4295	0h 3m 21.408s	-8.652e+002	-0.0047	1.958e+002	0.0011	-4.4188	0h 2m 46.453s
250	5.873e+002	0.0032	1.768e+002	0.0010	3.3218	0h 10m 26.315s	-5.858e+002	-0.0032	1.768e+002	0.0010	-3.3133	0h 10m 2.163s
500	5.968e+002	0.0032	1.252e+002	0.0007	4.7668	0h 37m 53.955s	-5.949e+002	-0.0032	1.252e+002	0.0007	-4.7516	0h 36m 43.676s
750	4.647e+002	0.0025	8.340e+001	0.0005	5.5719	1h 42m 11.351s	-4.626e+002	-0.0025	8.339e+001	0.0005	-5.5474	0h 52m 29.285s
1000	6.017e+002	0.0033	6.173e+001	0.0003	9.7473	3h 11m 46.675s	-5.992e+002	-0.0033	6.173e+001	0.0003	-9.7068	3h 21m 56.865s

Table G.1: Lift, lift coefficient, drag, drag coefficient, lift-to-drag ratio, and run time of simulations for $\alpha = 0°$ and $M_\infty = 0.806$

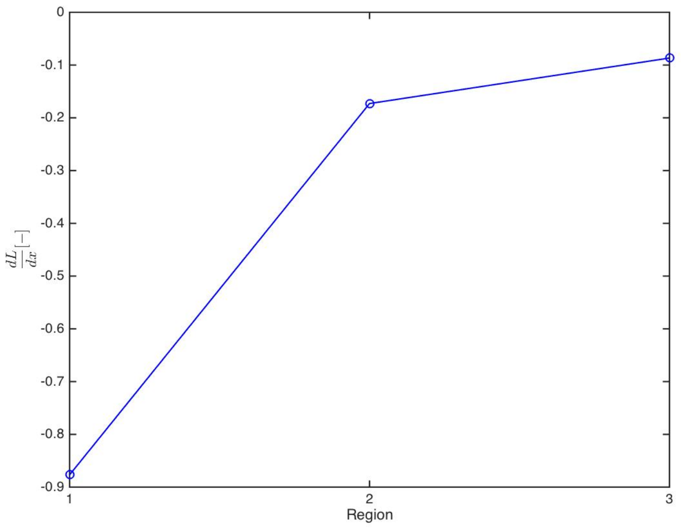
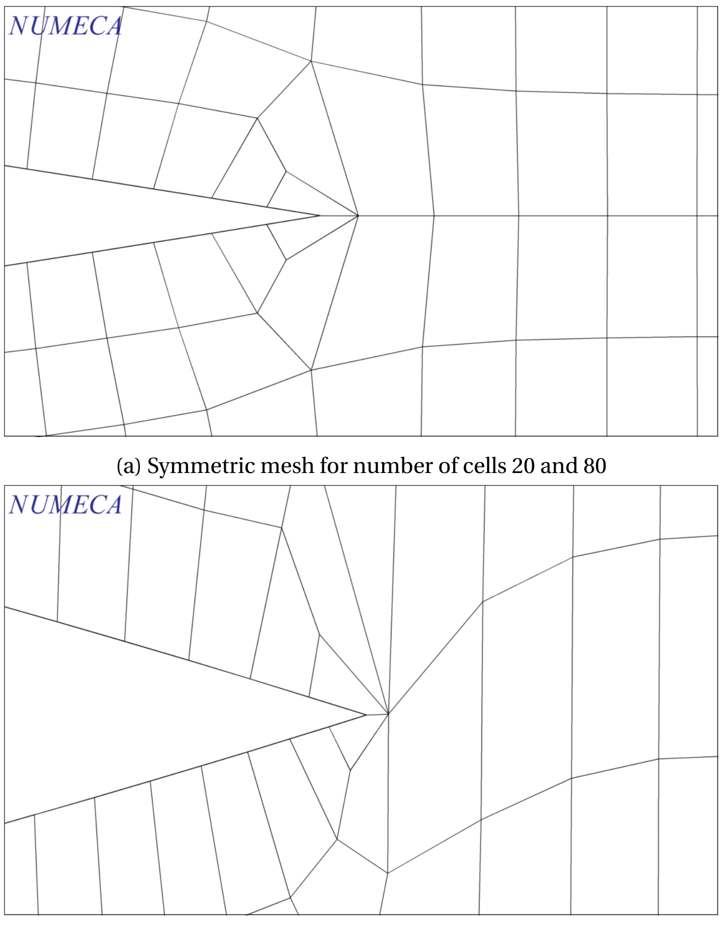


Figure G.2: Progress of $\frac{dL}{dx}$



(a) Symmetric mesh for number of cells 20 and 80

(b) Asymmetric mesh for number of cells 40,60,100,250,500,750, and 1000

Figure G.3: Symmetric and asymmetric meshes

$$C_L = \frac{L}{\frac{1}{2}\rho_\infty V_\infty^2 S} \tag{G.1}$$

$$C_D = \frac{D}{\frac{1}{2}\rho_\infty V_\infty^2 S} \tag{G.2}$$

Although it was concluded that there were no differences between the pressure coefficient distribution between the first and second setup, a remarkable difference has been identified in Figure G.4a. Here, the values of the lift are mirrored with respect to the horizontal axis. This might be caused by the fact that the orientation within $NUMECA$ is opposite to the orientation within $CATIA$. The flow within $NUMECA$ goes from right to left, meaning that positive flow points to the left. The opposite is true for $CATIA$, that is that positive flow is directed to the right. Due to this difference, the lift and drag coefficients as well as the lift-to-drag ratios for the setup where the domain box has been created in $NUMECA$ are the same as for the setup where the domain box is created in $CATIA$, but with an opposite sign. The lift-to-drag ratio's $\frac{L}{D}$, lift coefficients C_L, and drag coefficients C_D for this setup can be seen on the right half of Table G.1.

$M_\infty = 0.86$

The C_P distributions for the airfoil under zero angle of attack immersed in a flow with freestream Mach number $M_\infty = 0.86$ are displayed in Figure G.8. The graphics on the left side have been computed on the domain of which the surrounding box is created within $CATIA$, and for an increasing number of mesh cells (increasing from the top down). The right side of Figure G.8 contains the C_P distributions for the domain of which the surrounding box is created within $NUMECA$, as well as for an increasing number of mesh cells. The lift and drag values corresponding to those pressure distributions can be found in Table .

The conclusions drawn form this figure are outlined below:

- The shape and overall values of the C_P distributions don't change with changing amount of cells. The suction peak is $C_{P_{min}} \approx -0.50$ for all test cases, at the location $x \approx 0.23$. The values of C_P at the leading and trailing edges of the airfoil, i.e. $x = -1$ and $x = 1$ respectively, do increase with increasing number of cells and converge to a value of $C_{P_{LE}} \approx 0.59$ and $C_{P_{TE}} \approx 0.50$ at the leading and trailing edges respectively.

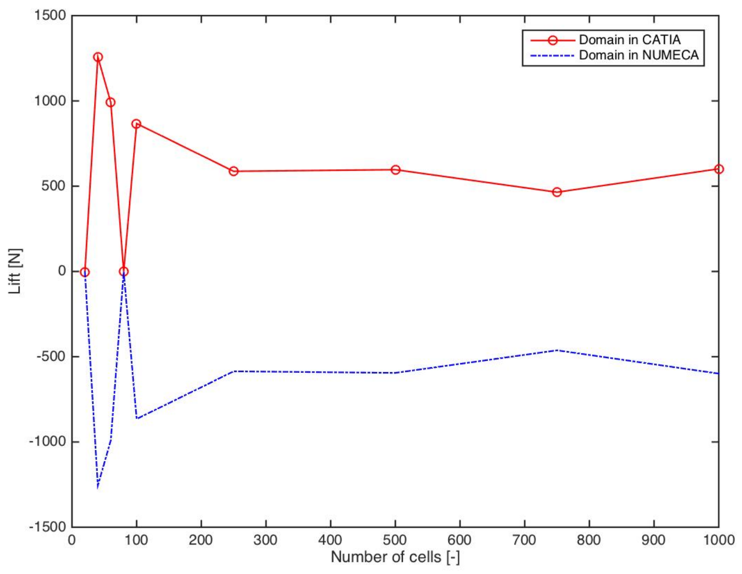
- The location of the suction peak is shifted further downstream with respect to $M_\infty = 0.806$ (i.e. from $x = 0$ to $x \approx 0.23$), while the magnitude of the suction peak has decreased (i.e. from $C_{P_{min}} = -0.30$ to $C_{P_{min}} = -0.50$.

- The C_P distributions resulting from the domain box created in $CATIA$ and $NUMECA$ are identical to each other, so this factor doesn't influence the results.

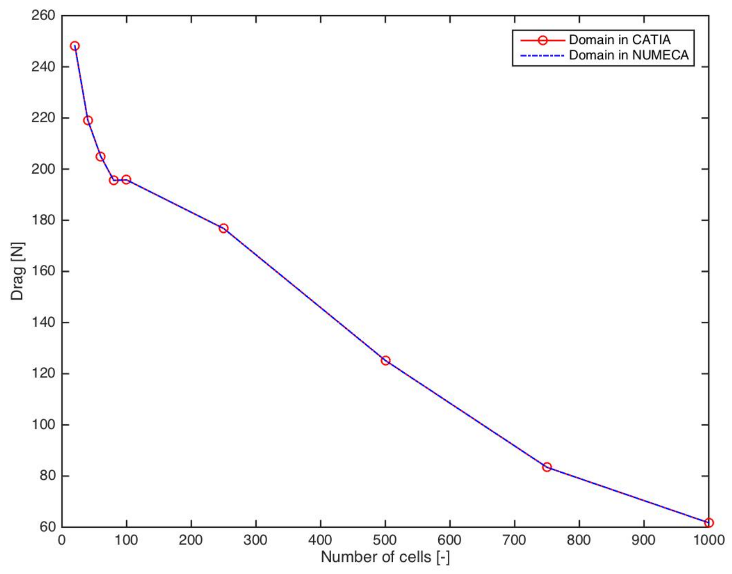
- The appearance of shocks where the flow transitions from supersonic to subsonic is evident in the data at the location $x \approx 0.23$, which is identical to the location of the suction peak.

- The pressure distributions for the upper and lower side of the airfoil lie on top of each other. This is as expected, since those pressure distributions are computed for a symmetric parabolic arc airfoil under zero angle of attack, for which zero lift is expected. However, a negligible difference between the C_P of the upper and lower sides of the airfoil has been detected right after the location of the shock wave.

- The C_P distribution, beyond the shock wave appearance, is consistent with a typical pressure coefficient distribution for a symmetrical parabolic arc airfoil under zero angle of attack, immersed in a flow with freestream Mach number $M_\infty = 0.86$ [20]. As it was the case for $M_\infty = 0.806$, the pressure coefficient distribution displayed in Figure G.8h will be interpreted here. The C_P distribution in this figure starts out at the value of $C_{P_{LE}} \approx 0.50$ at the nose, and drops as the flow expands around the nose. The pressure P decreases below P_∞, yielding a minimum value of $C_{P_{min}} \approx -0.42$ downstream of the nose, at the location $x \approx 0.23$. This is also the location where a shock wave has been identified. Further downstream the pressure stops with increasing abruptly, and starts recovering smoothly at the location $x \approx 0.38$. It finally converges to a value of $C_{P_{TE}} \approx 0.40$ at the trailing edge. The value of C_P at the trailing edge is again slightly smaller than the value at the leading edge due to the Kutta condition



(a) Lift varying with the number of cells for $\alpha = 0°$ and $M_\infty = 0.806$



(b) Drag varying with the number of cells for $\alpha = 0°$ and $M_\infty = 0.806$

Figure G.4: Lift and drag varying with the number of mesh cells for $\alpha = 0°$ and $M_\infty = 0.806$

of finite velocities, which is satisfied at the trailing edge. However, the flow at the trailing edge is still attached, since the pressure coefficient of separated flow is $C_P < 0$.
The same trend as for $M_\infty = 0.806$ of increasing C_P at the leading and trailing edges with an increasing number of mesh cells can be seen here as well. The pressure coefficients at the leading and trailing edges increase from $C_{P_{LE}} \approx 0.33$ to $C_{P_{LE}} \approx 0.59$ and from $C_{P_{TE}} \approx 0.27$ to $C_{P_{TE}} \approx 0.50$, respectively. This increasing trend is an indication of reliable results, since typical C_P at the stagnation point in compressible flow is somewhat larger than unity [1]. Also, the lower C_P at the trailing edge (compared to the leading edge) is an indication of good performance. This is a result of the Kutta condition of finite velocities.

- It is again remarkable that the C_P values at the leading edge are far below unity, which can be explained with the same reasoning as for $M_\infty = 0.806$ (i.e. the pointy 'nose' of the airfoil) .

	Domain created in $CATIA$						Domain created in $NUMECA$					
# mesh cells	L [N]	C_L [-]	D [N]	C_D [-]	$\frac{L}{D}$ [-]	Run Time	L [N]	C_L [-]	D [N]	C_D [-]	$\frac{L}{D}$ [-]	Run Time
20	-5.000e+000	-0.0000	6.057e+002	0.0029	-0.0083	0h 1m 13.203s	-5.781e+000	-0.0000	6.057e+002	0.0029	-0.0095	0h 0m 48.669s
40	1.795e+003	0.0086	5.741e+002	0.0027	3.1266	0h 1m 17.596s	-1.792e+003	-0.0086	5.740e+002	0.0027	-3.1220	0h 1m 9.362s
60	1.423e+003	0.0068	5.579e+002	0.0027	2.5506	0h 1m 44.319s	-1.422e+003	-0.0068	5.579e+002	0.0027	-2.5488	0h 1m 33.680s
80	-2.313e+000	-0.0000	5.488e+002	0.0026	-0.0042	0h 2m 9.417s	-2.938e+000	-0.0000	5.488e+002	0.0026	-0.0054	0h 2m 0.510s
100	1.270e+003	0.0061	5.493e+002	0.0026	2.3120	0h 2m 49.664s	-1.267e+003	-0.0060	5.493e+002	0.0026	-2.3066	0h 2m 43.210s
250	9.014e+002	0.0043	5.227e+002	0.0025	1.7245	0h 10m 38.265s	-8.981e+002	-0.0043	5.227e+002	0.0025	-1.7182	0h 10m 12.885s
500	9.888e+002	0.0047	4.013e+002	0.0019	2.4640		-9.851e+002	-0.0047	4.012e+002	0.0019	-2.4554	
750	8.111e+002	0.0039	2.799e+002	0.0013	2.8978		-8.063e+002	-0.0038	2.799e+002	0.0013	-2.8807	
1000	1.060e+003	0.0051	2.207e+002	0.0011	4.8029		-1.055e+003	-0.0050	2.207e+002	0.0011	-4.7802	

Table G.2: Lift, lift coefficient, drag, drag coefficient, lift-to-drag ratio, and run time of simulations for $\alpha = 0°$ and $M_\infty = 0.86$

$\alpha = 0.5°$

This section contains the pressure coefficient distributions for a symmetric parabolic arc airfoil under an angle of attack of $\alpha = 0.5°$, immersed in flows with freestream Mach numbers $M_\infty = 0.806$ and $M_\infty = 0.86$. Six setups have been tested with *NUMECA* for each of the Mach numbers, as explained in Section 6.1.1.2. First, the C_P distributions of $M_\infty = 0.806$ will be treated in Section G, followed by the results of $M_\infty = 0.86$ in Section G.

$M_\infty = 0.806$

Figure G.7 contains the C_P distributions of the setups for which the airfoil has been drawn in *CATIA* under zero angle of attack, after which it is given an angle of attack $\alpha = 0.5°$ in *NUMECA* (i.e. setups 1 & 2 in Section 6.1.1.2). On the left side of this figure the C_P distributions resulting from the setup for which the borders of the domain have been created in *CATIA* are shown, while on the right side the C_P has been computed on a domain of which the borders have been created in *NUMECA*. The discussion of these figures is as follows:

- The C_P distributions on both sides of the figure are the same, meaning that it doesn't matter in which program the borders of the domain are created.
- The simulations have only been performed for 20, 40, 60, 80, and 100 mesh cells, since from a number of 80 mesh cells not a big difference of the C_P at the leading and trailing edges was noticed (i.e. converged solution reached).
- The overall values of the C_P distribution don't change with the number of mesh cells, except the pressure coefficients at the leading and trailing edges. The pressure coefficient at exactly the leading edge (i.e. $x = -1$) increases from $C_{P_{LE}} \approx 0.33$ to $C_{P_{LE}} \approx 0.41$, while the pressure coefficient at the trailing edge increases from $C_{P_{TE}} \approx 0.25$ to $C_{P_{TE}} \approx 0.39$. The values of the location on the airfoil immediately next to the leading edge, also change with an increasing number of mesh cells. Along the lower side of the airfoil (i.e. upper curve) C_P increases from $C_{P_{LE_l}} \approx 0.50$ to $C_{P_{LE_l}} \approx 0.77$, and along the upper side of the airfoil (i.e. lower curve) it decreases from $C_{P_{LE_u}} \approx 0.08$ to $C_{P_{LE_u}} \approx 0.06$. This indicates that the velocities along the upper side of the airfoil reach higher values than on the lower side of the airfoil.
- The overall shape of the C_P distribution doesn't change with the number of mesh cells. However, a small cove arises at the leading edge along the upper side of the airfoil (i.e. the lower curve) for a number of mesh cells of 80 and 100. This might be due to computational issues regarding the nose of the airfoil. The symmetric parabolic arc airfoil in consideration doesn't have a real nose, but a wedge instead. Therefore it becomes difficult to catch this nose which basically consists of a point (since a mesh node should be placed at exactly this point). Another possibility is that singularities could occur at the leading edge.
- The pressure coefficient distributions along the upper and lower side of the airfoil have been separated from each other starting at the leading edge, due to an increase in the angle of attack. However, further downstream these C_P distributions start to flock again. Since it doesn't matter whether the borders of the domain have been created in *CATIA* or *NUMECA*, Figure G.9g has been picked to elaborate on. The pressure coefficient at the leading edge starts with $C_{P_{LE}} \approx 0.41$, after which it immediately reduces to $C_{P_{LE_u}} \approx 0.06$ on the upper side of the airfoil and increases to $C_{P_{LE_l}} \approx 0.73$ on the lower side of the airfoil.
  The pressure coefficient along the upper side of the airfoil starts with $C_{P_{LE_u}} \approx 0.06$ at the location $x \approx -0.94$, and decreases to a minimum value of $C_{P_{min_u}} \approx -0.33$ at the location $x \approx -0.06$. Then, it starts increasing and reaches a value of $C_{P_{TE}} \approx 0.38$ at the trailing edge.
  Along the lower side of the airfoil, the pressure coefficient starts with $C_{P_{LE_l}} \approx 0.73$ at the location $x \approx -0.98$. Subsequently it gradually decreases and reaches a minimum value of $C_{P_{min_l}} \approx -0.26$ at the location $x \approx 0.08$, after which it increases and merges with the C_P distribution along the upper side at the trailing edge. Here it reaches a value of $C_{P_{TE}} \approx 0.38$.
- The increase of the angle of attack from $\alpha = 0°$ to $\alpha = 0.5°$ has caused the suction peak to move forward toward (toward the leading edge) on the upper side of the airfoil (i.e. from $x = 0$ to $x \approx -0.06$), and further downstream (i.e. from $x = 0$ tot $x \approx 0.08$) on the lower side of the airfoil. The magnitudes of the C_P values has also changed with an increasing angle of attack. On the upper side, the pressure

coefficient at the leading edge had decreased from $C_{P_{LE_u}} \approx 0.45$ to $C_{P_{LE_u}} \approx 0.06$, while it has increased from $C_{P_{LE_l}} \approx 0.45$ to $C_{P_{LE_l}} \approx 0.73$ on the lower side of the airfoil. The magnitude of the suction peak has decreased (i.e. more negative) on the upper side of the airfoil from $C_{P_{min}} \approx -0.30$ to $C_{P_{min}} \approx -0.33$, and increased (i.e. more positive) on the lower side of the airfoil from $C_{P_{min}} \approx -0.30$ to $C_{P_{min}} \approx -0.26$. The value of the pressure coefficient at the leading edge however, didn't change ($C_{P_{TE}} \approx 0.38$).

The results of setups 3 & 5 can be seen in Figure G.10. For this setup, the airfoil was created in $CATIA$ under an angle of attack $\alpha 0.5°$. It is remarkable that for the setup for which the borders of the domain have been created in $CATIA$, more simulations have been performed than for the setup for which the borders of the domain have been created in $NUMECA$. This was due to problems with $NUMECA$, since the program was crashing while performing simulations for a number of mesh cells starting from 100. However, since the C_P distributions of setup 3 and 5 up till a number of mesh cells of 80 are identical to each other as can be seen from Figure G.10, this is not a big problem. Also, the same discussion as the discussion of Figure G.9 is valid for Figure G.10, since the C_P distributions are almost similar. The only difference occurs at the leading edge of the airfoil, where higher C_P values can be found. This might be due to the way the airfoil is put under an angle of attack. Since the airfoil for this case is given an angle of attack in $CATIA$, rather than in $NUMECA$, there might be slight differences in the mesh at the leading edge. However, this is not dramatic, since the rest of the C_P distributions are almost similar to the ones in Figure G.9. The pressure coefficient distribution in Figure G.10g shows that the C_P value at the leading edge starts with $C_{P_{LE}} \approx 0.42$, after which it immediately splits up along the upper and lower sides of the airfoil. The pressure coefficient distribution along the upper side of the airfoil starts at $C_{P_{LE_u}} \approx 0.06$ at the location $x = -0.94$, after which it decreases to $C_{P_{min_u}} \approx -0.33$ at the location $x \approx -0.06$. Further downstream C_P gradually increases and reaches $C_{P_{TE}} \approx 0.37$ at the trailing edge at the location $x = 1$.
The C_P distribution along the lower side of the airfoil look similar as the upper side, but its level is higher. It starts with $C_{P_{LE_l}} \approx 0.75$ at the leading edge (i.e.$x = -0.98$), decreases to $C_{P_{min_l}} \approx -0.26$ at the location $x \approx 0.08$, and increases further downstream to $C_{P_{TE}} = 0.37$ at the trailing edge (i.e. $x = 1$).
Finally, the results of setups 4 & 6 are displayed in Figure G.11. The difference of these setups with setups 3 & 5 is that the aerodynamic forces (i.e. lift and drag) are decomposed in a $x-$ and z-direction in $NUMECA$. However, it is striking that these results are similar to the results shown in Figure G.10, which means that decomposition of the forces is not needed if the airfoil is already drawn under an angle of attack within $CATIA$. Since the C_P distributions are similar to the ones of setups 3 & 5, the same discussion as the discussion of Figure G.10 holds for Figure G.11.

$M_\infty = 0.86$

This section covers the pressure coefficient distribution of the symmetric arc airfoil under an angle of attack $\alpha = 0.5°$, immersed in a flow with freestream Mach number $M_\infty = 0.86$. The simulations have been performed for six different setups, as was the case for $M_\infty = 0.806$. The results of setups 1 & 2, 3 & 5, and 4 & 6 are displayed in figures G.12, G.13, and G.14 respectively. It is clear that the results of the different setups don't differ much from each other.
Starting with Figure G.12g, one can see that the pressure coefficient at the leading edge starts with $C_{P_{LE}} \approx 0.44$. Subsequently, the pressure coefficient distributions of the upper and lower side of the airfoil split up. On the upper side of the airfoil the pressure coefficient starts with $C_{P_{LE_u}} \approx 0.10$ at the location $x \approx -0.94$. Then, it starts decreasing further downstream and reaches a minimum value of $C_{P_{min_u}} \approx -0.58$ at the location $x \approx 0.43$. This is also the location where the shock wave on the upper side of the airfoil appears, which is more evident than it is the case for $\alpha = 0°$. It should be noted that the location of the start of the shock wave, which is also the location of the suction peak, is shifted towards the trailing edge with increasing angle of attack (i.e. from $x \approx 0.23$ to $x \approx 0.43$). Also, the magnitude of the suction peak along the upper side of the airfoil has decreased (became more negative) from $C_{P_{min}} \approx -0.42$ to $C_{P_{min}} \approx -0.58$. Another remarkable feature is that the recovery phase of the shock wave (i.e. the distance between the start of the shock wave and point where it merges with the C_P distribution of the lower side of the airfoil) decreases with increasing angle of attack. The shock wave along the upper side of the airfoil under zero angle of attack starts at $x \approx 0.23$ and ends at $x \approx 0.38$ ($\Delta x \approx 0.15$), while for the airfoil under angle of attack $\alpha = 0.5°$ these locations are $x \approx 0.43$ and $x \approx 0.48$ ($\Delta x \approx 0.05$).
Along the lower side of the airfoil, the pressure coefficient starts at $C_{P_{LE_l}} \approx 0.76$ at the location $x \approx -0.98$, and reaches a minimum value of $C_{P_{min_l}} \approx -0.33$ at the location $x \approx 0.14$. This means that the suction peak of the lower side of the airfoil has moved forward (towards the leading edge) with respect to the C_P distribution of

the airfoil under zero angle of attack. Also, the magnitude of the suction peak has increased (became less negative). The pressure coefficient distributions of the upper and lower side of the airfoil merge at the trailing edge and reach a value of $C_{P_{TE}} \approx 0.40$.
Figure G.13g is almost similar to Figure G.12g. The only difference is in the value of the pressure coefficient at the leading edge along the lower side of the airfoil, which is $C_{P_{LE}} \approx 0.78$ instead of $C_{P_{LE}} \approx 0.76$. The other values are similar to the values of C_P in Figure G.12g. The same holds for Figure G.14g, which means that no decomposition of the aerodynamic forces in *NUMECA* is needed when the airfoil is drawn under an angle of attack within *CATIA*.

$\alpha = 1°$

The pressure coefficient distribution of the symmetric parabolic arc airfoil under $\alpha = 1°$ angle of attack, immersed in a flow with freestream Mach numbers $M_\infty = 0.806$ and $M_\infty = 0.86$ will be covered in sections G and G, respectively. Six different setups have been tested with *NUMECA* for each Mach number. The results can be found in the corresponding sections.

$M_\infty = 0.806$

This section covers the C_P distribution of the airfoil under $\alpha = 1°$ angle of attack, immersed in a flow with freestream Mach number $M_\infty = 0.806$. The results of setups 1 & 2 are displayed in Figure G.15. Figures G.16 and G.17 display the results of setups 3 & 5 and 4 & 6, respectively.
Starting with Figure G.15, the following conclusions can be drawn:

- The C_P distributions on both the left and right side of the figure are the same. This indicates that the tool with which the borders of the domain have been created doesn't matter.
- The overall shape and values of the C_P distributions don't change with increasing number of mesh cells, except the values at the leading and trailing edges. The pressure coefficient at exactly the leading edge (i.e. $x = -1$) increases from $C_{P_{LE}} \approx 0.30$ to $C_{P_{LE}} \approx 0.37$, while the pressure coefficient at the trailing edge increases from $C_{P_{TE}} \approx 0.23$ to $C_{P_{TE}} \approx 0.39$. The values of the location on the airfoil immediately next to the leading edge, also change with an increasing number of mesh cells. Along the lower side of the airfoil (i.e. upper curve) C_P increases from $C_{P_{LE_l}} \approx 0.66$ to $C_{P_{LE_l}} \approx 1.00$, and along the upper side of the airfoil (i.e. lower curve) it decreases from $C_{P_{LE_u}} \approx -0.18$ to $C_{P_{LE_u}} \approx -0.41$. This indicates that the velocities along the upper side of the airfoil reach higher values than on the lower side of the airfoil.
- A cove has been identified between $x = -1$ and $x \approx -0.97$ along the upper side of the airfoil. In this region, the values of the pressure coefficient along the upper side of the airfoil reach very low (i.e. negative) values. From $x \approx -0.97$ on a reliable C_P distribution starts to arise. The cove again might be caused due to computational issues regarding the nose of the airfoil and singularities occurring at the leading edge of the airfoil.
- Focussing on Figure G.15g, the pressure coefficient at the leading edge starts with $C_{P_{LE}} \approx 0.37$, after which it immediately reduces to $C_{P_{LE_u}} \approx -0.14$ on the upper side of the airfoil and increases to $C_{P_{LE_l}} \approx 0.97$ on the lower side of the airfoil.
  The pressure coefficient along the upper side of the airfoil starts with $C_{P_{LE_u}} \approx -0.14$ at the location $x \approx -0.94$, and decreases to a minimum value of $C_{P_{min_u}} \approx -0.40$ at the location $x \approx -0.10$. Then, it starts increasing and reaches a value of $C_{P_{TE}} \approx 0.37$ at the trailing edge.
  Along the lower side of the airfoil, the pressure coefficient starts with $C_{P_{LE_l}} \approx 0.97$ at the location $x \approx -0.98$. Subsequently it gradually decreases and reaches a minimum value of $C_{P_{min_l}} \approx -0.22$ at the location $x \approx 0.13$, after which it increases and merges with the C_P distribution along the upper side at the trailing edge. Here it reaches a value of $C_{P_{TE}} \approx 0.37$.
- When comparing Figure G.15g to Figure Figure G.9g, it turns out that increasing the angle of attack from $\alpha = 0.5°$ to $\alpha = 1°$ causes the suction peak along the upper side of the airfoil to decrease (i.e. from $C_{P_{min_u}} \approx -0.33$ to $C_{P_{min_u}} \approx -0.40$) and along the lower side of the airfoil to increase (i.e. from $C_{P_{min_u}} \approx -0.26$ to $C_{P_{min_u}} \approx -0.22$). As a result, the C_P distributions along the upper and lower side of the airfoil move further apart. The location of the suction peaks also changes with increasing angle of attack. This location moves forward towards the leading edge (i.e. from $x \approx -0.06$ to $x \approx -0.10$) along the upper side of the airfoil, and further downstream towards the trailing edge (i.e. from $x \approx 0.08$ to $x \approx 0.13$) along the lower side of the airfoil.

When examining Figure G.16, the similarity of the pressure coefficient distributions in this figure with the ones in Figure G.15 is observed. The only difference is regarding to the pressure coefficient at the trailing edge, which is negligibly small ($C_{P_{TE}} \approx 0.34$ instead of $C_{P_{TE}} \approx 0.37$ for 80 mesh cells). It also turns out that Figure G.17 is similar to Figure G.16, which leads to the conclusion that when the airfoil is given an angle of attack within $CATIA$ no decomposition of the aerodynamic forces is needed since this doesn't make any difference.

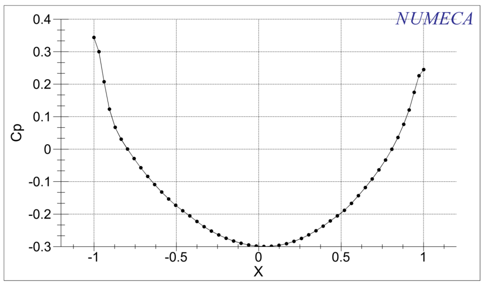
$M_\infty = 0.86$

The results displayed in this section belong to the airfoil under $\alpha = 1°$ angle of attack, immersed in a flow with freestream Mach number $M_\infty = 0.86$. Six different setups have been tested. The results of setups 1 & 2, 3 & 5, and 4 & 6 are outlined in figures G.18, G.19, and G.20 respectively. Comparing these figures to each other, it is observed that the pressure coefficients are similar to each other. Therefore, only Figure G.18g will be elaborated on.
The following conclusions can be drawn after observing Figure G.18:

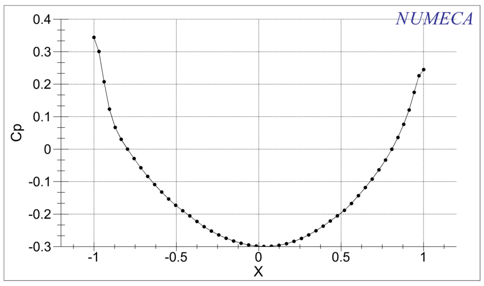
- The overall shape of the pressure coefficients in Figure G.18 doesn't change with the increasing number of mesh cells, except the values at the leading and trailing edges. The magnitude of C_P at the leading edge starts at $C_{P_{LE}} \approx 0.34$ (for 20 mesh cells) and increases to $C_{P_{LE}} \approx 0.42$ (for 100 mesh cells). Also, the values next to the leading edge along the upper and lower side of the airfoil do change with an increasing number of mesh cells. Along the upper side of the airfoil, the pressure coefficient at the location $x \approx -0.97$ holds a value of $C_{P_{LE_u}} \approx -0.11$ for 20 mesh cells, and decreases to $C_{P_{LE_u}} \approx -0.09$ when the number of mesh cells increases to 100. Along the lower side of the airfoil the opposite trend is observed. The pressure coefficient at the location $x \approx -0.97$ increases from $C_{P_{LE_l}} \approx 0.67$ to $C_{P_{LE_l}} \approx 0.99$ with an increasing number of mesh cells (i.e. from 20 to 100). Finally, the value of the pressure coefficient at the trailing edge increases from $C_{P_{TE}} \approx 0.27$ to $C_{P_{TE}} \approx 0.41$.

- When examining Figure G.18g more closely, it can be observed that the pressure coefficient starts with $C_{P_{LE}} \approx 0.41$ at the leading edge (i.e. $x = -1$). Subsequently, it splits up in pressure coefficient distributions along the upper and lower side of the airfoil.
The pressure coefficient along the upper side of the airfoil starts with $C_{P_{LE_u}} \approx -0.09$ at the location $x \approx -0.97$, after which it decreases to a minimum value of $C_{P_{min_u}} \approx -0.75$ at the location $x \approx 0.53$. This is also the location where the shock wave arises. The recovery of the shock wave takes place, and ends at $x \approx 0.63$ (i.e. $\Delta x \approx 0.10$). From this point on, the pressure coefficient distribution gradually increases towards a value of $C_{P_{TE}} \approx 0.40$ at the trailing edge.
Along the lower side of the airfoil, the pressure coefficient distribution starts with $C_{P_{LE_l}} \approx 0.97$ at the location $x \approx -0.97$, whereafter it decreases towards a minimum value of $C_{P_{min_l}} \approx -0.27$ at the location $x \approx 0.16$. At this point, the pressure coefficient distribution along the lower side of the airfoil merges with the pressure coefficient distribution along the upper side of the airfoil and increases towards a value of $C_{P_{TE}} \approx 0.40$ at the trailing edge.

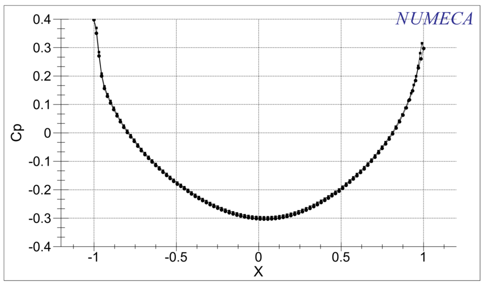
- As the angle of attack increases further from $\alpha = 0.5°$ to $\alpha = 1°$, the location of the shock wave moves aft (i.e. from $x \approx 0.43$ to $x \approx 0.53$).



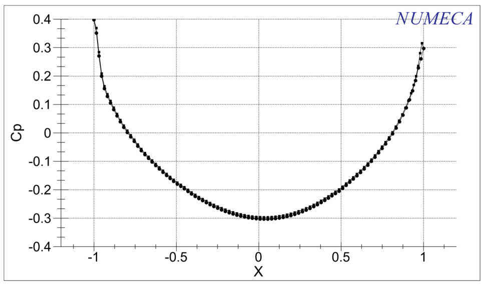
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



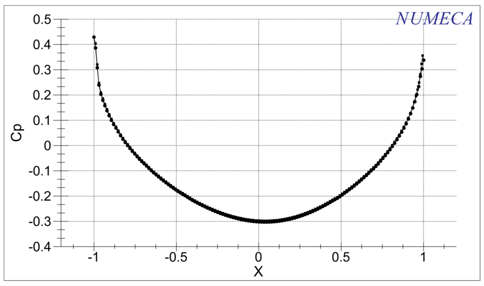
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



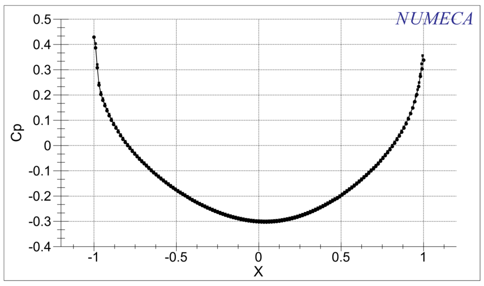
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



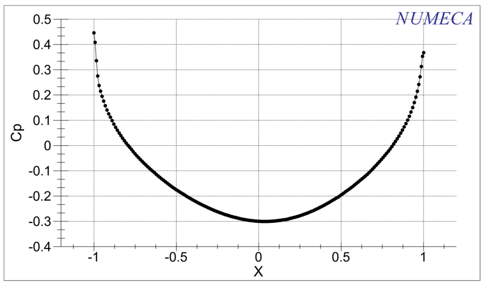
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



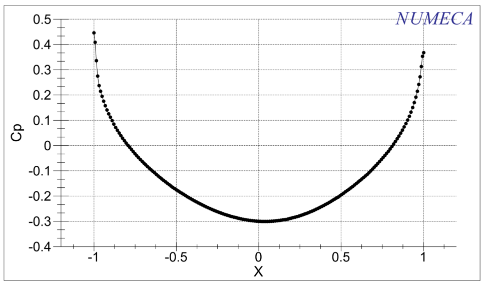
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



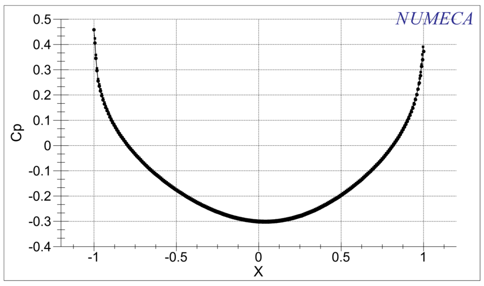
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



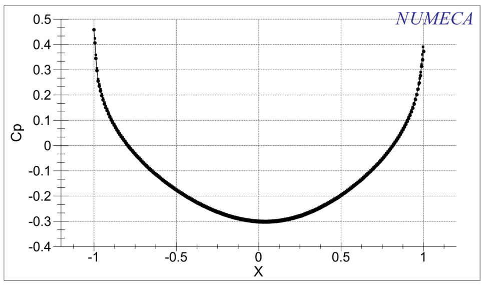
(g) Number of cells in x and z direction: 80, domain created in $CATIA$



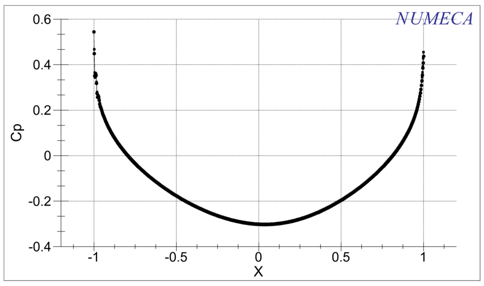
(h) Number of cells in x and z direction: 80, domain created in $NUMECA$



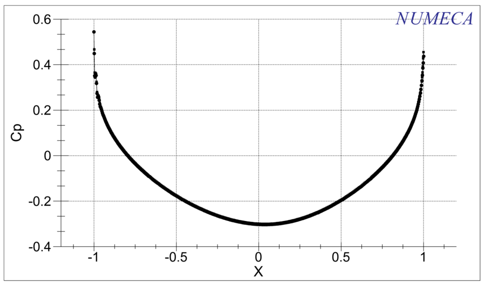
(i) Number of cells in x and z direction: 100, domain created in $CATIA$



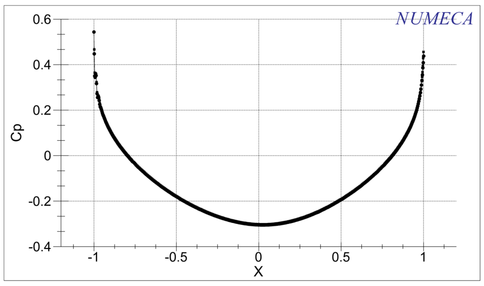
(j) Number of cells in x and z direction: 100, domain created in $NUMECA$



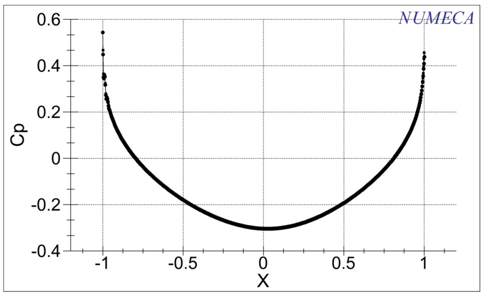
(k) Number of cells in x and z direction: 250, domain created in $CATIA$



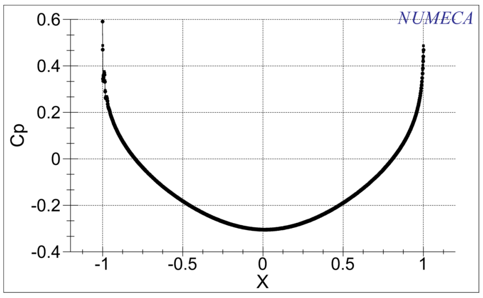
(l) Number of cells in x and z direction: 250, domain created in $NUMECA$



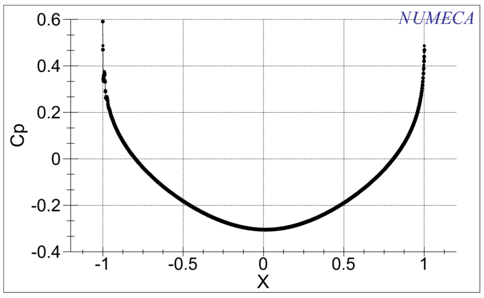
(m) Number of cells in x and z direction: 500, domain created in $CATIA$



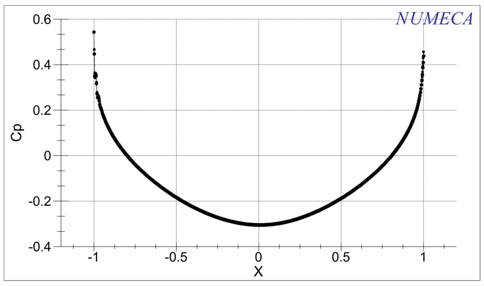
(n) Number of cells in x and z direction: 500, domain created in $NUMECA$



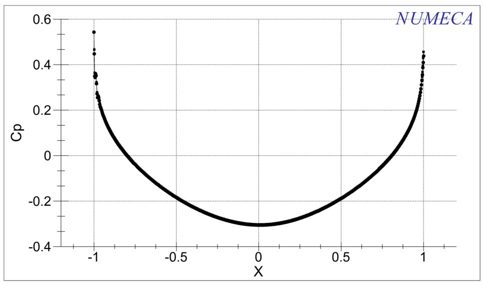
(o) Number of cells in x and z direction: 750, domain created in $CATIA$



(p) Number of cells in x and z direction: 750, domain created in $NUMECA$

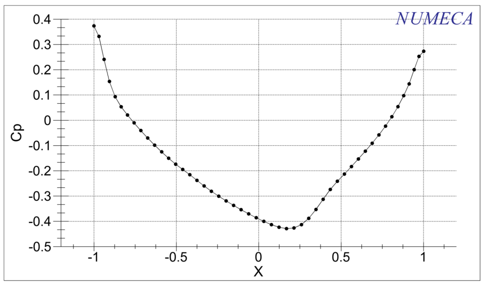


(q) Number of cells in x and z direction: 1000, domain created in $CATIA$

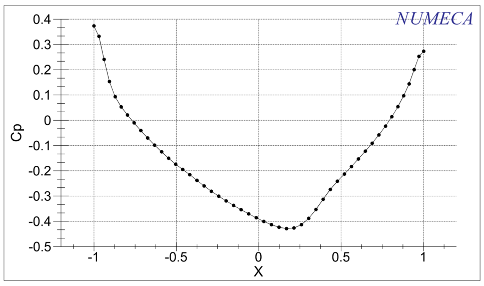


(r) Number of cells in x and z direction: 1000, domain created in $NUMECA$

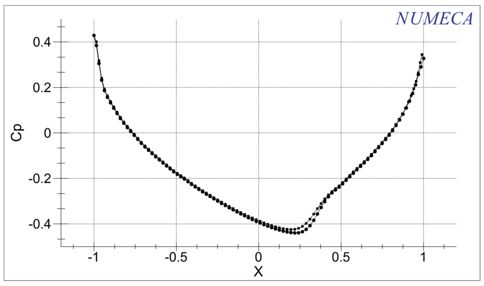
Figure G.7: C_p distribution for $\alpha = 0°$ and $M_\infty = 0.806$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed.



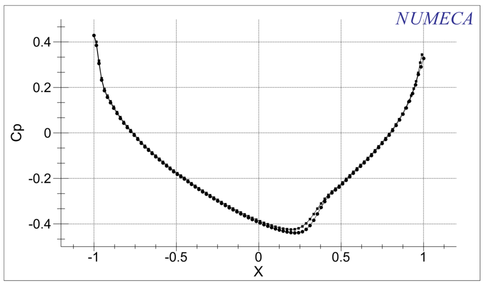
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



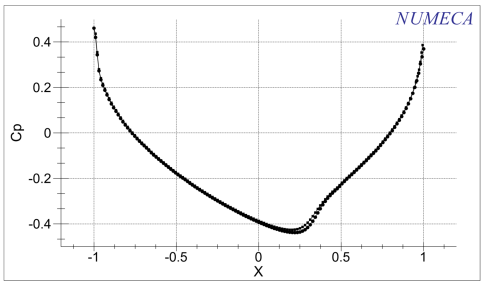
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



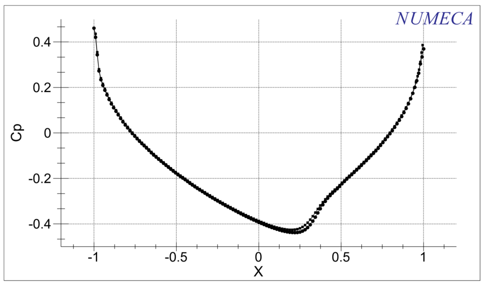
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



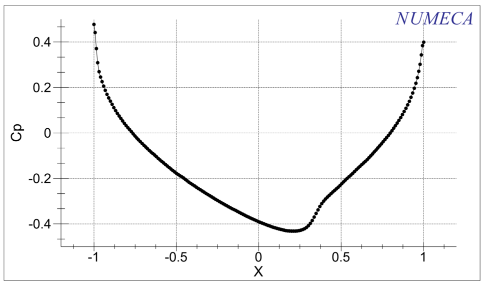
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



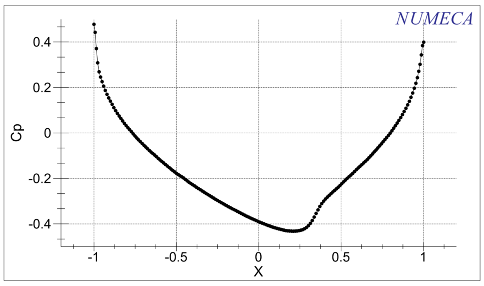
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



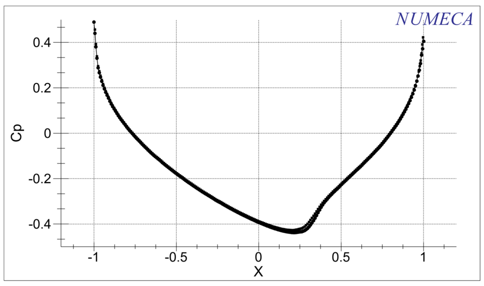
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



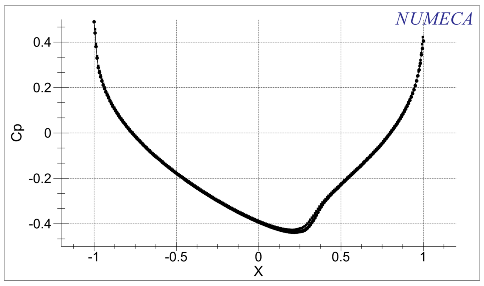
(g) Number of cells in x and z direction: 80, domain created in $CATIA$



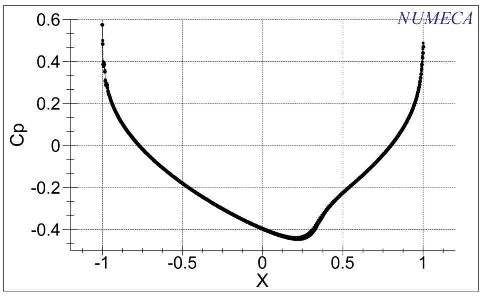
(h) Number of cells in x and z direction: 80, domain created in $NUMECA$



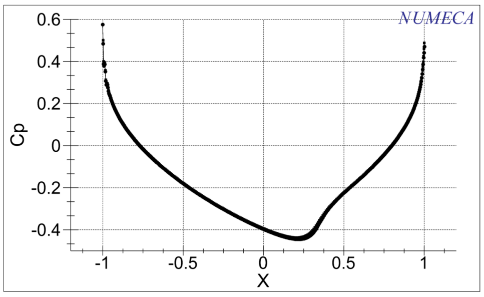
(i) Number of cells in x and z direction: 100, domain created in $CATIA$



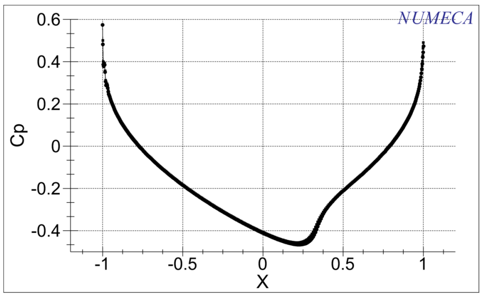
(j) Number of cells in x and z direction: 100, domain created in $NUMECA$



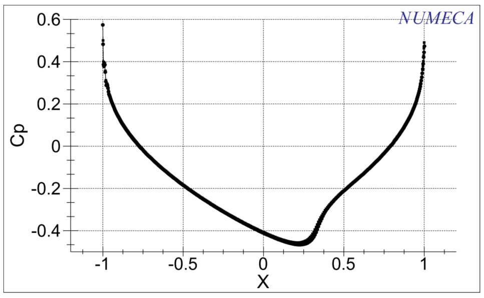
(k) Number of cells in x and z direction: 250, domain created in $CATIA$



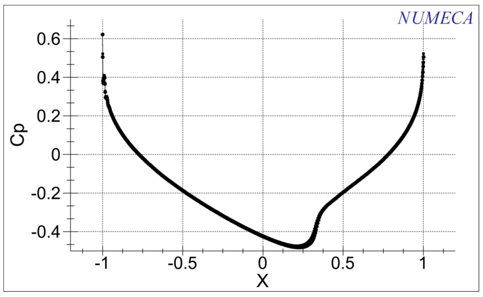
(l) Number of cells in x and z direction: 250, domain created in $NUMECA$



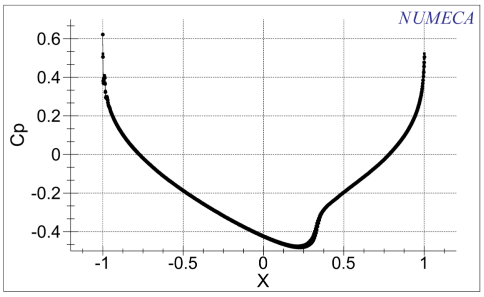
(m) Number of cells in x and z direction: 500, domain created in $CATIA$



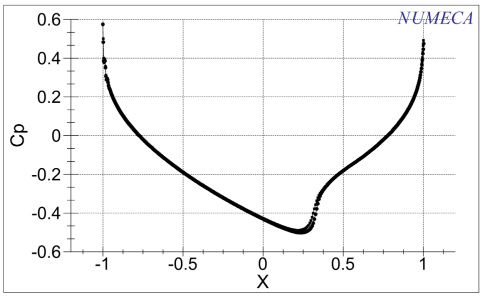
(n) Number of cells in x and z direction: 500, domain created in $NUMECA$



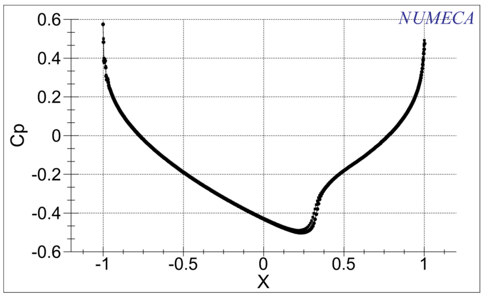
(o) Number of cells in x and z direction: 750, domain created in $CATIA$



(p) Number of cells in x and z direction: 750, domain created in $NUMECA$

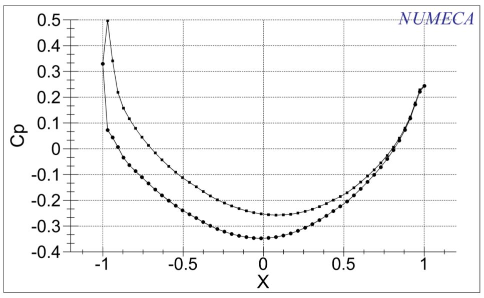


(q) Number of cells in x and z direction: 1000, domain created in $CATIA$

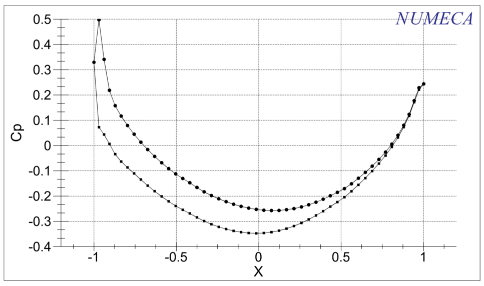


(r) Number of cells in x and z direction: 1000, domain created in $NUMECA$

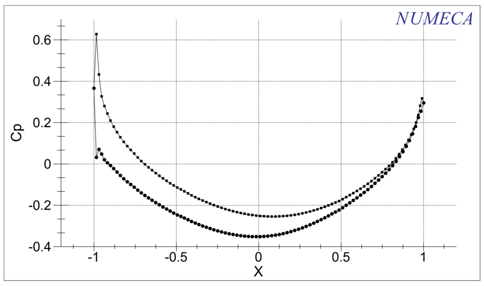
Figure G.8: C_p distribution for $\alpha = 0°$ and $M_\infty = 0.86$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed.



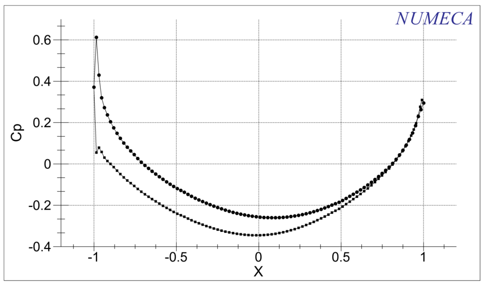
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



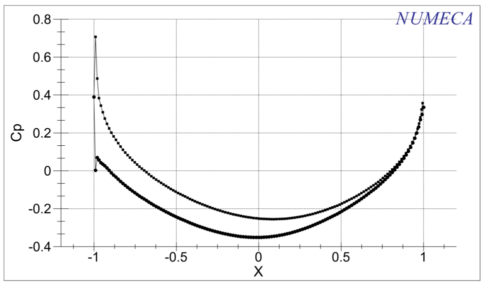
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



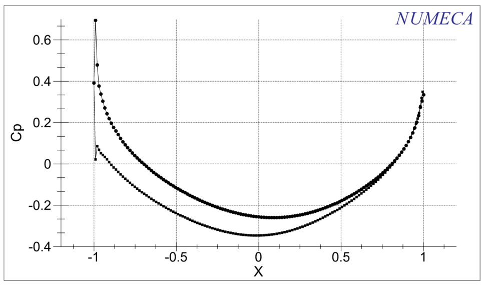
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



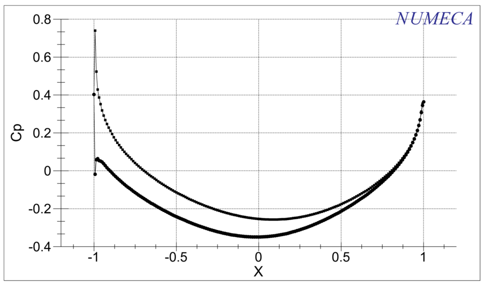
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



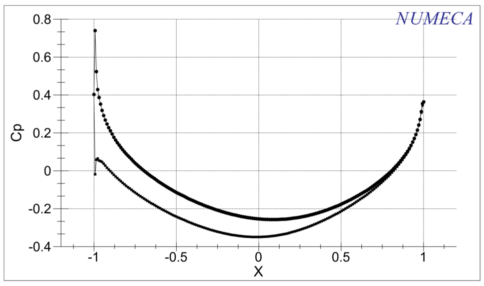
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



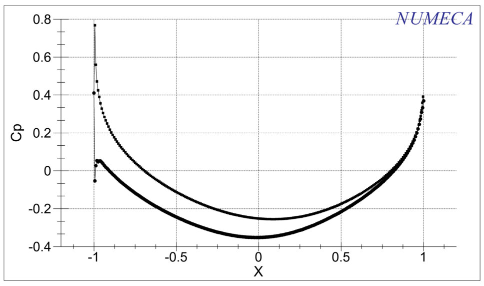
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



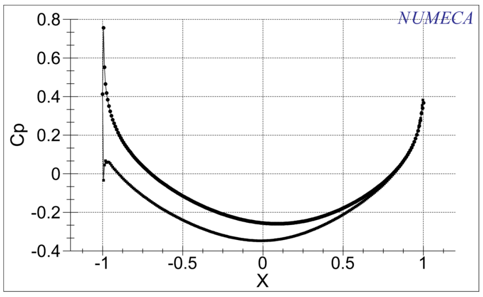
(g) Number of cells in x and z direction: 80, domain created in $CATIA$



(h) Number of cells in x and z direction: 80, domain created in $NUMECA$

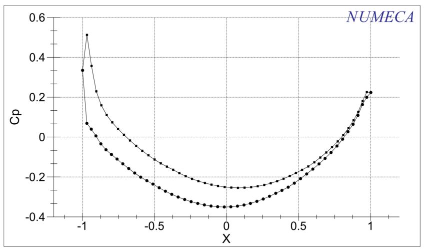


(i) Number of cells in x and z direction: 100, domain created in $CATIA$

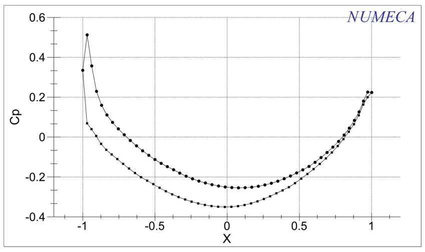


(j) Number of cells in x and z direction: 100, domain created in $NUMECA$

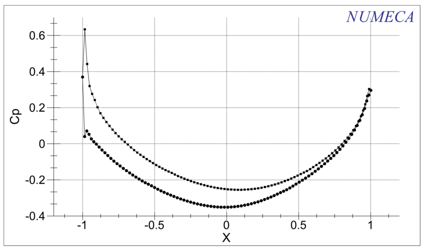
Figure G.9: C_p distribution for $\alpha = 0.5°$ and $M_\infty = 0.806$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under zero angle of attack, and is set under an angle of attack $\alpha = 0.5°$ in $NUMECA$.



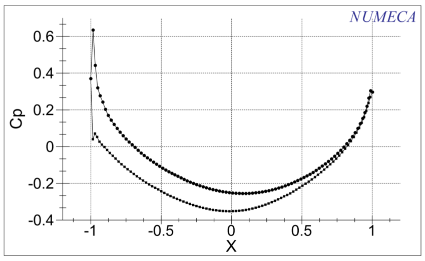
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



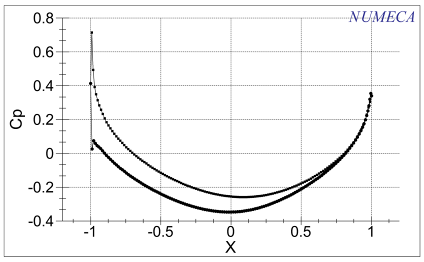
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



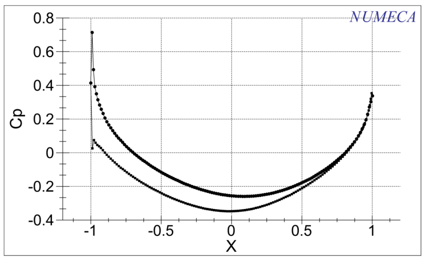
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



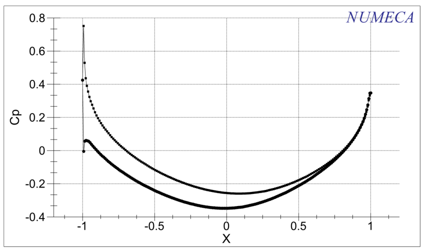
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



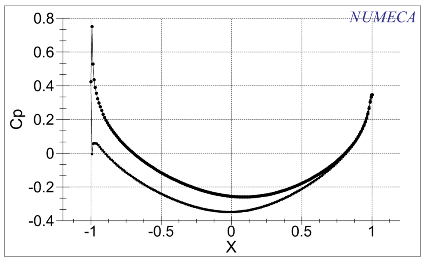
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



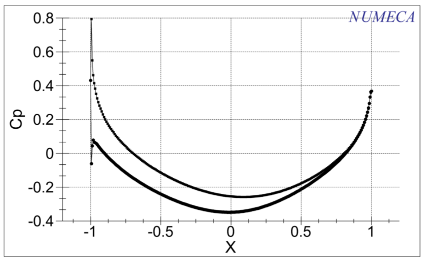
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



(g) Number of cells in x and z direction: 80, domain created in $CATIA$

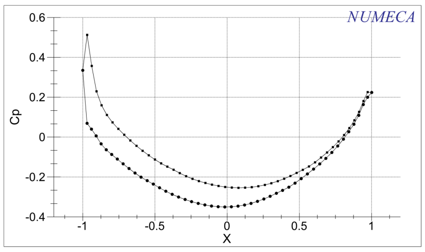


(h) Number of cells in x and z direction: 80, domain created in $NUMECA$

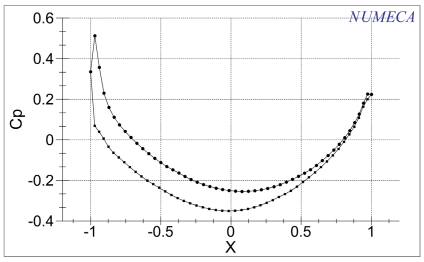


(i) Number of cells in x and z direction: 100, domain created in $CATIA$

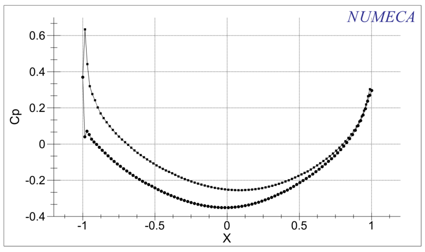
Figure G.10: C_p distribution for $\alpha = 0.5°$ and $M_\infty = 0.806$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under $\alpha = 0.5°$.



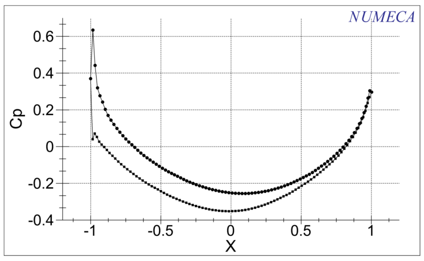
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



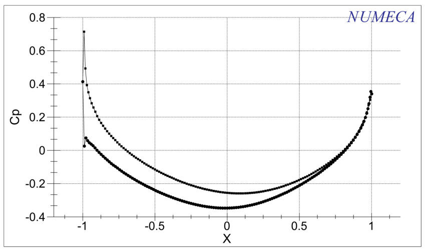
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



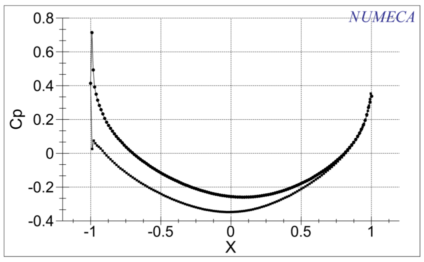
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



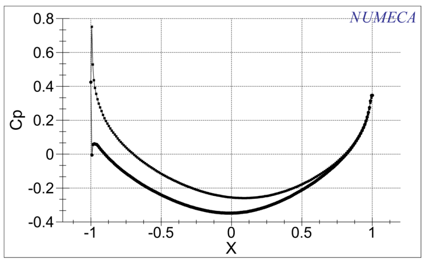
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



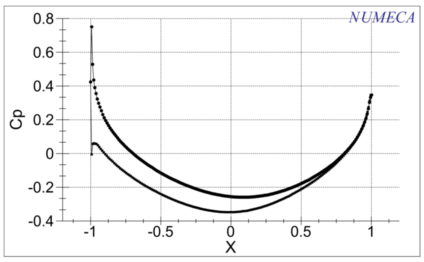
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



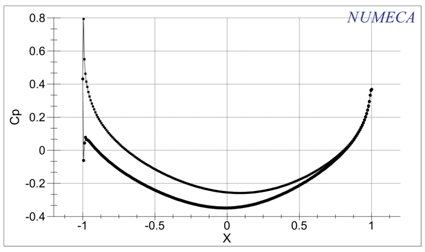
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



(g) Number of cells in x and z direction: 80, domain created in $CATIA$

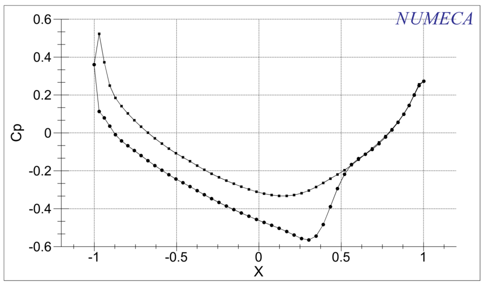


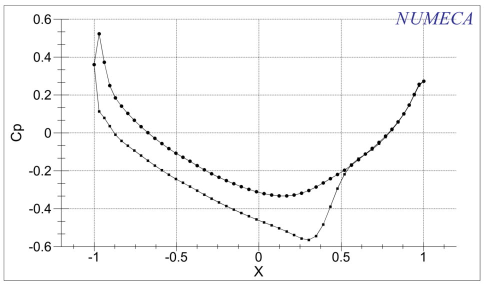
(h) Number of cells in x and z direction: 80, domain created in $NUMECA$



(i) Number of cells in x and z direction: 100, domain created in $CATIA$

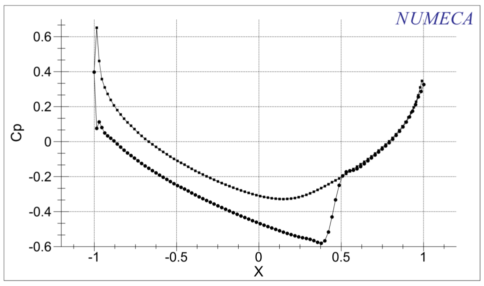
Figure G.11: C_p distribution for $\alpha = 0.5°$ and $M_\infty = 0.806$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under $\alpha = 0.5°$, and the aerodynamic forces are decomposed in a $x-$ and z-direction.

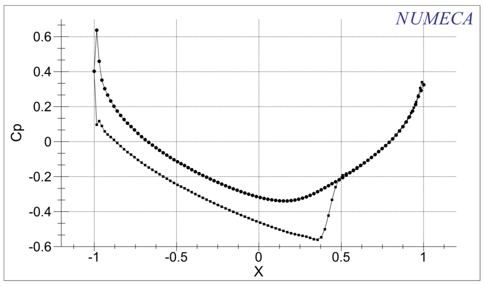




(a) Number of cells in x and z direction: 20, domain created in $CATIA$

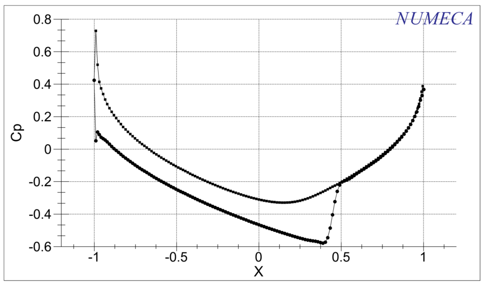
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$

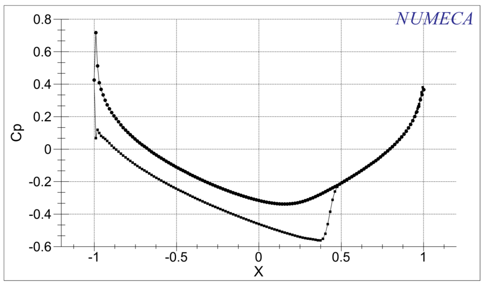




(c) Number of cells in x and z direction: 40, domain created in $CATIA$

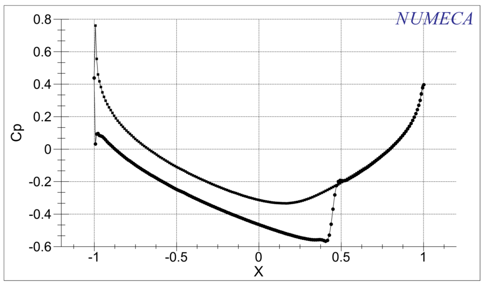
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$

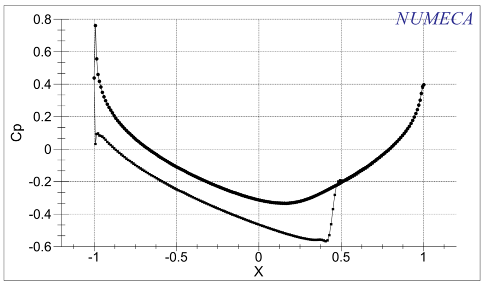




(e) Number of cells in x and z direction: 60, domain created in $CATIA$

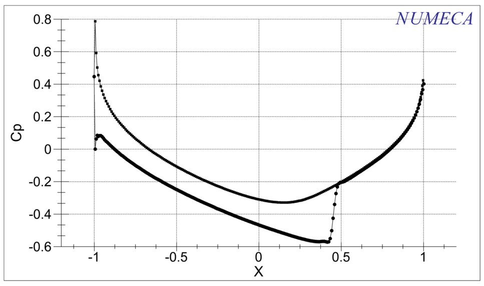
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$

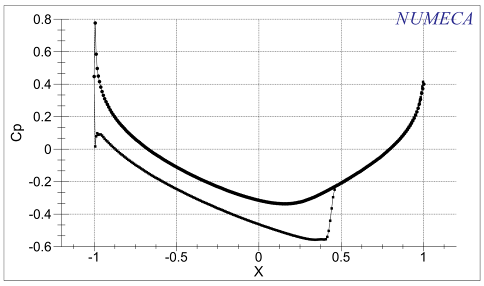




(g) Number of cells in x and z direction: 80, domain created in $CATIA$

(h) Number of cells in x and z direction: 80, domain created in $NUMECA$

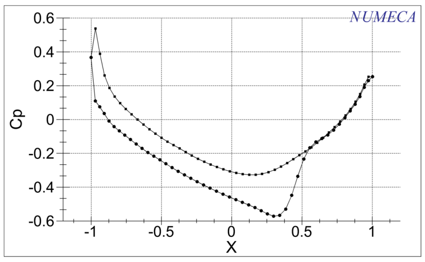




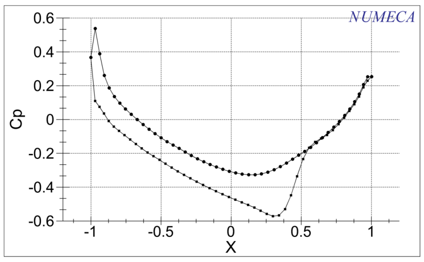
(i) Number of cells in x and z direction: 100, domain created in $CATIA$

(j) Number of cells in x and z direction: 100, domain created in $NUMECA$

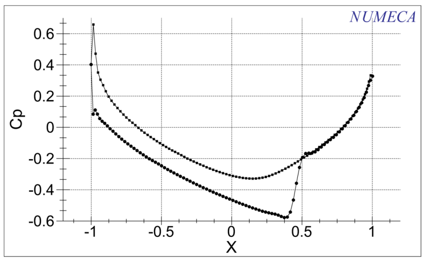
Figure G.12: C_p distribution for $\alpha = 0.5°$ and $M_\infty = 0.86$.The left side contains the C_p distributions corresponding to the domain created in $CATIA$, while on the right side the C_p distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under zero angle of attack, and is set under an angle of attack $\alpha = 0.5°$ in $NUMECA$.



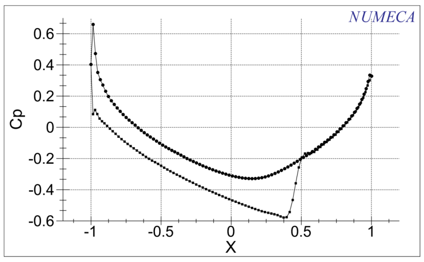
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



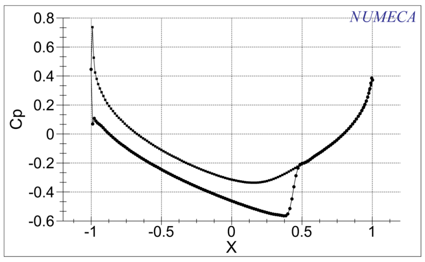
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



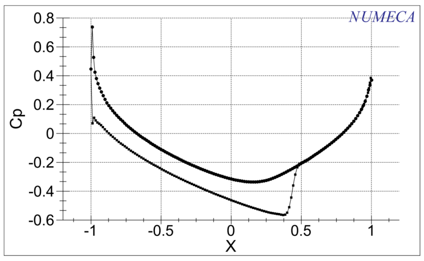
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



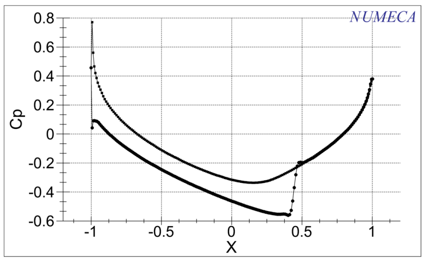
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



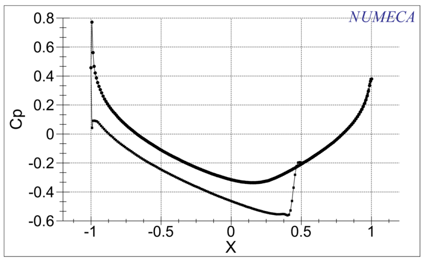
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



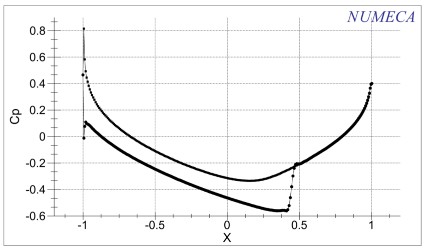
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



(g) Number of cells in x and z direction: 80, domain created in $CATIA$

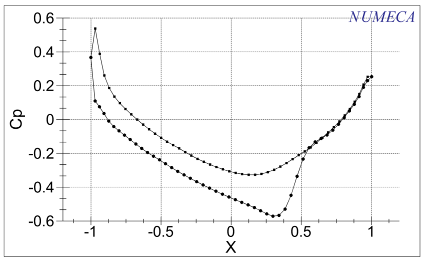


(h) Number of cells in x and z direction: 80, domain created in $NUMECA$

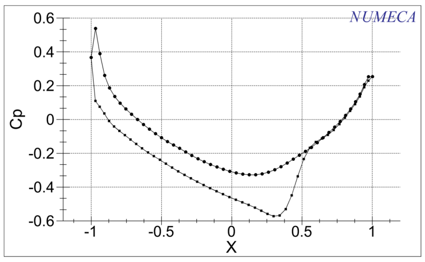


(i) Number of cells in x and z direction: 100, domain created in $CATIA$

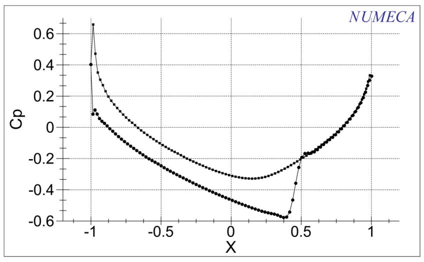
Figure G.13: C_p distribution for $\alpha = 0.5°$ and $M_\infty = 0.86$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under $\alpha = 0.5°$.



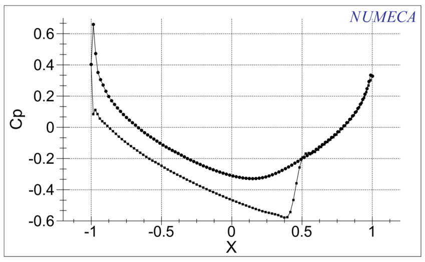
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



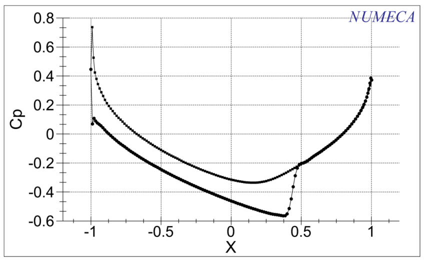
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



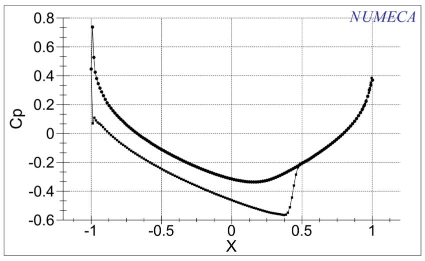
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



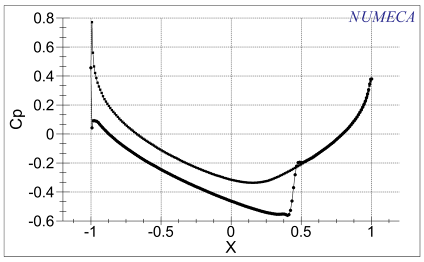
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



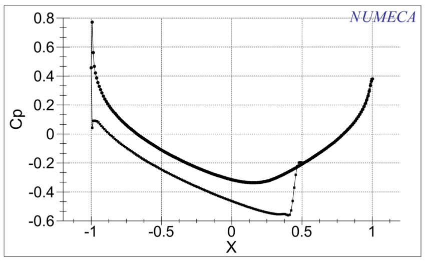
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



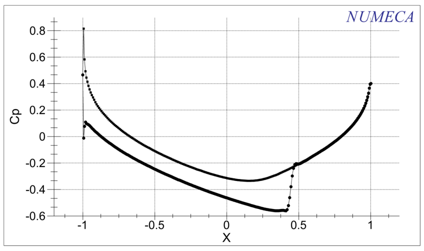
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



(g) Number of cells in x and z direction: 80, domain created in $CATIA$

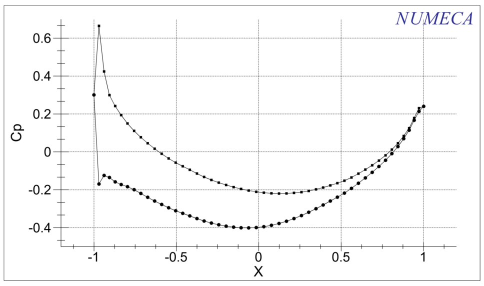


(h) Number of cells in x and z direction: 80, domain created in $NUMECA$

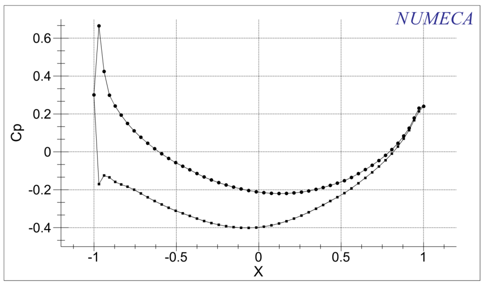


(i) Number of cells in x and z direction: 100, domain created in $CATIA$

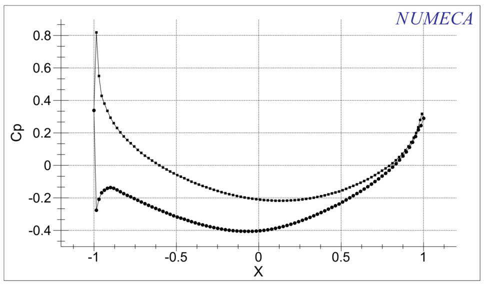
Figure G.14: C_p distribution for $\alpha = 0.5°$ and $M_\infty = 0.86$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under $\alpha = 0.5°$, and the aerodynamic forces are decomposed in a $x-$ and z-direction.



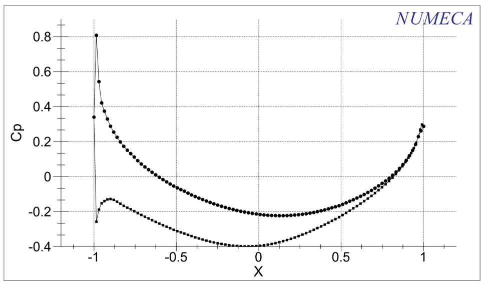
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



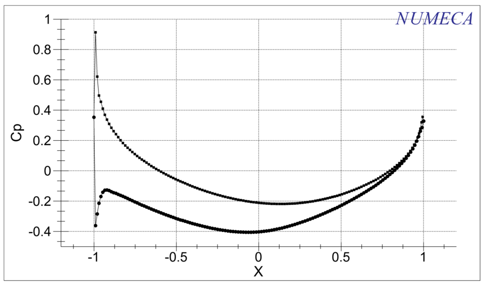
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



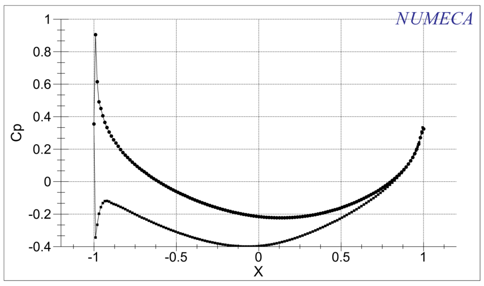
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



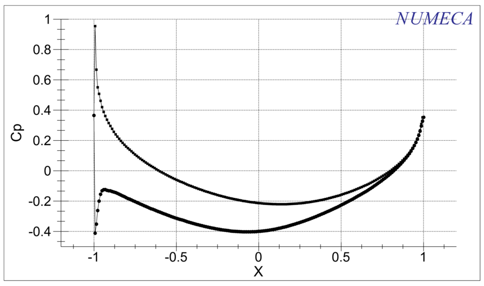
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



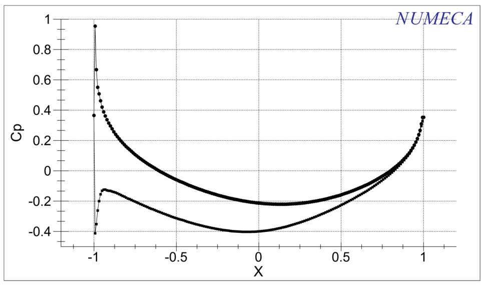
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



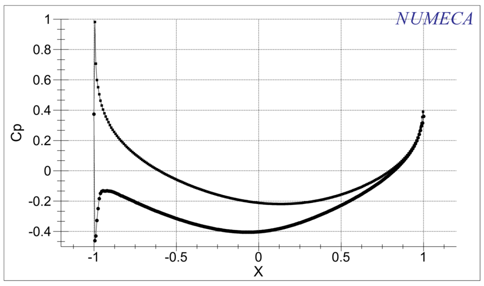
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



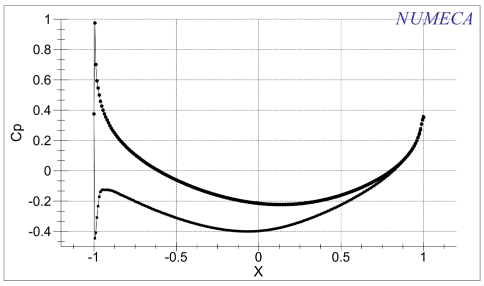
(g) Number of cells in x and z direction: 80, domain created in $CATIA$



(h) Number of cells in x and z direction: 80, domain created in $NUMECA$

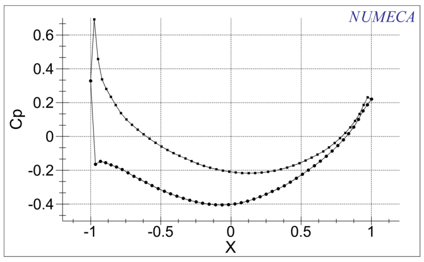


(i) Number of cells in x and z direction: 100, domain created in $CATIA$

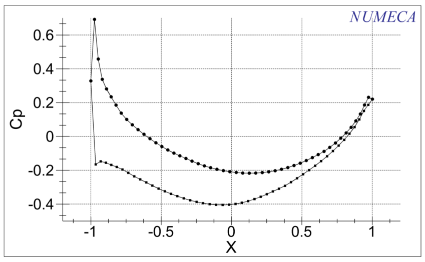


(j) Number of cells in x and z direction: 100, domain created in $NUMECA$

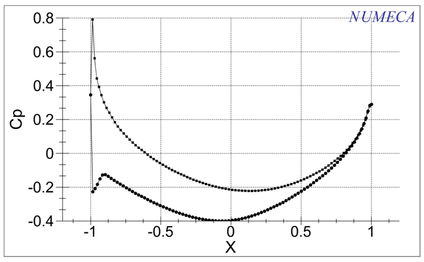
Figure G.15: C_p distribution for $\alpha = 1^\circ$ and $M_\infty = 0.806$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under zero angle of attack, and is set under an angle of attack $\alpha = 1^\circ$ in $NUMECA$.



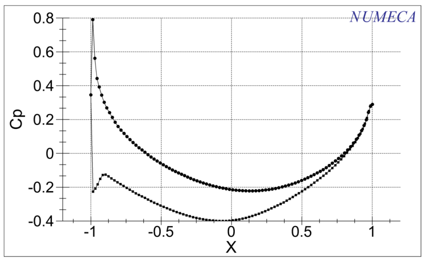
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



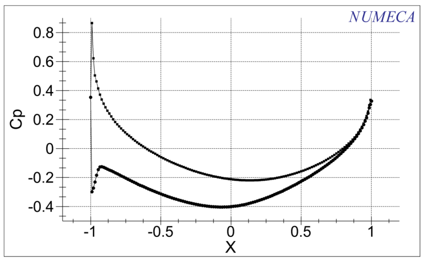
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



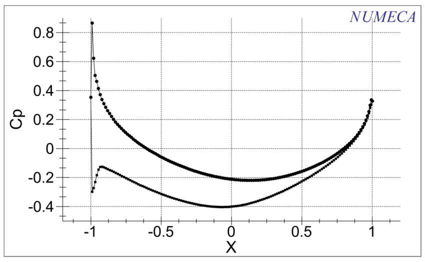
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



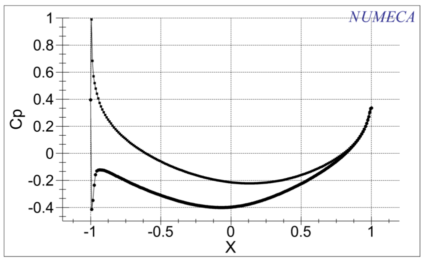
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



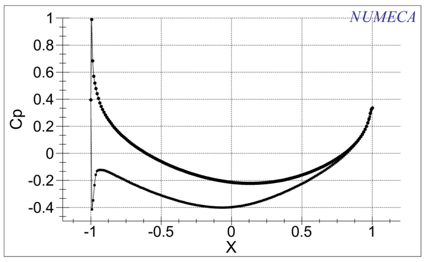
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



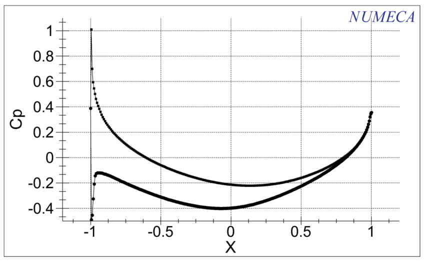
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



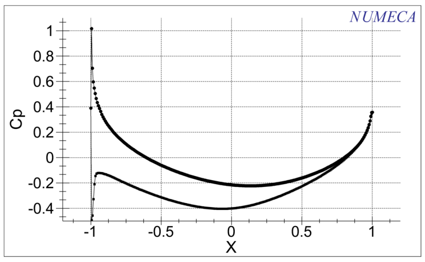
(g) Number of cells in x and z direction: 80, domain created in $CATIA$



(h) Number of cells in x and z direction: 80, domain created in $NUMECA$

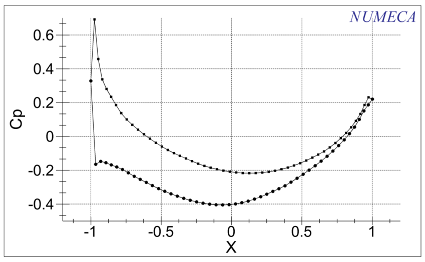


(i) Number of cells in x and z direction: 100, domain created in $CATIA$

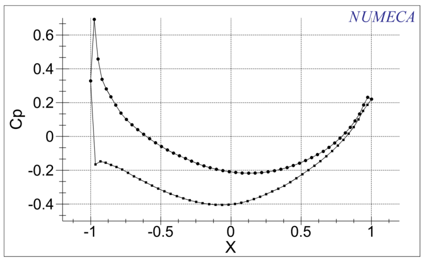


(j) Number of cells in x and z direction: 100, domain created in $NUMECA$

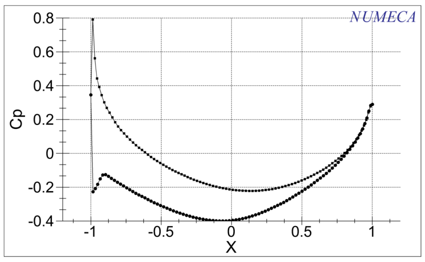
Figure G.16: C_p distribution for $\alpha = 1°$ and $M_\infty = 0.806$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under $\alpha = 1°$.



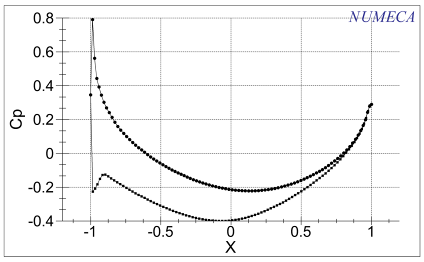
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



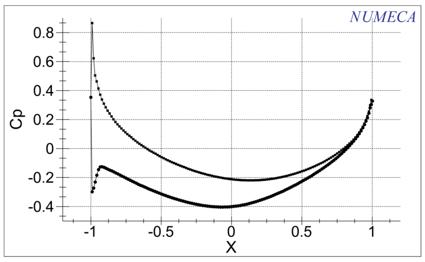
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



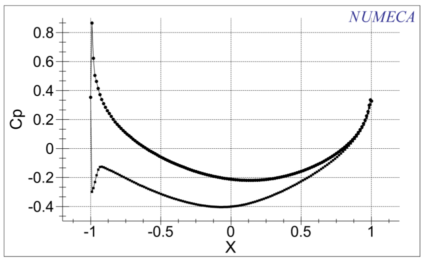
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



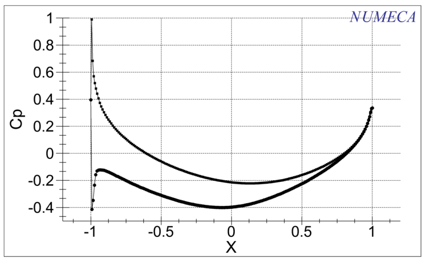
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



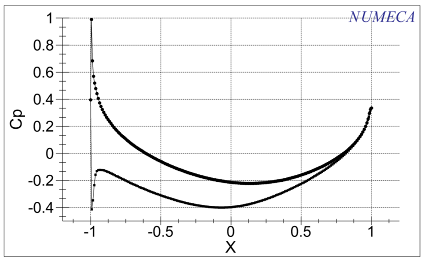
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



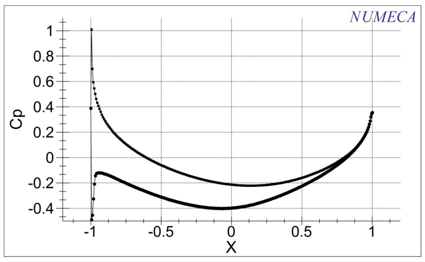
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



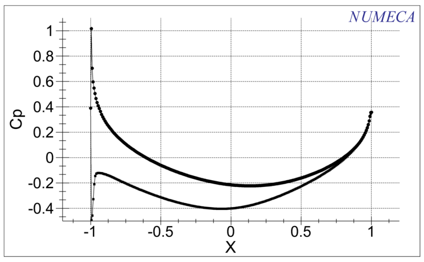
(g) Number of cells in x and z direction: 80, domain created in $CATIA$



(h) Number of cells in x and z direction: 80, domain created in $NUMECA$

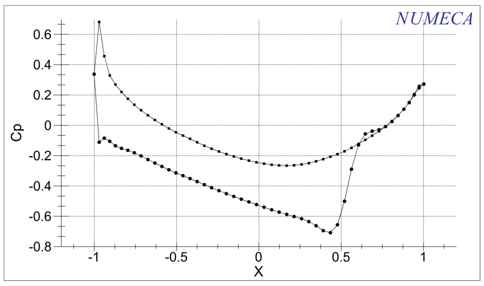


(i) Number of cells in x and z direction: 100, domain created in $CATIA$

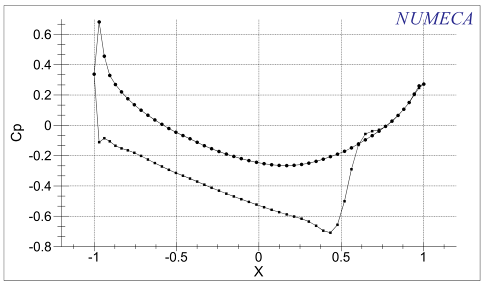


(j) Number of cells in x and z direction: 100, domain created in $NUMECA$

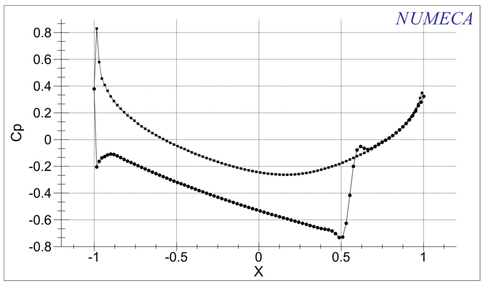
Figure G.17: C_p distribution for $\alpha = 1°$ and $M_\infty = 0.806$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under $\alpha = 1°$, and the aerodynamic forces are decomposed in a $x-$ and z-direction.



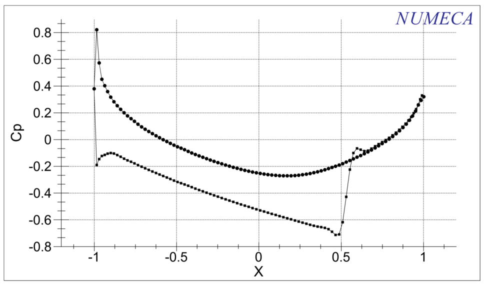
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



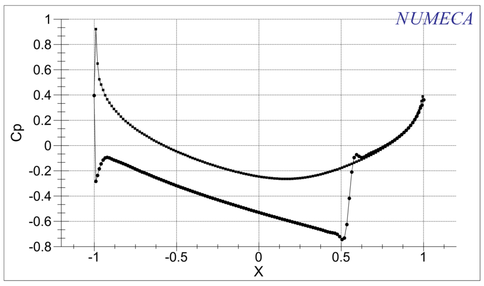
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



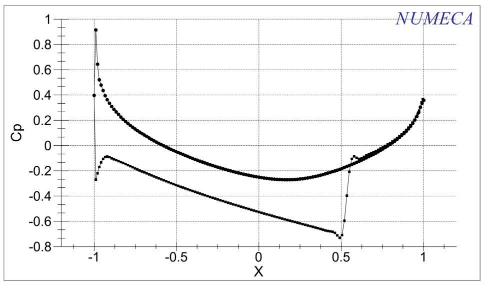
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



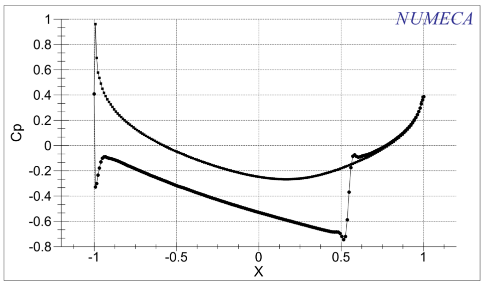
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



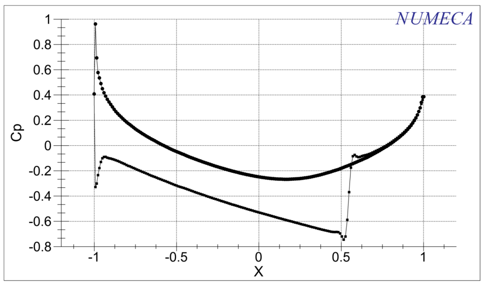
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



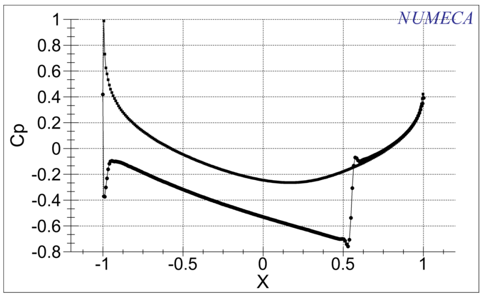
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



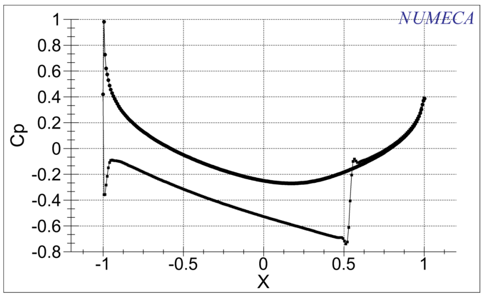
(g) Number of cells in x and z direction: 80, domain created in $CATIA$



(h) Number of cells in x and z direction: 80, domain created in $NUMECA$

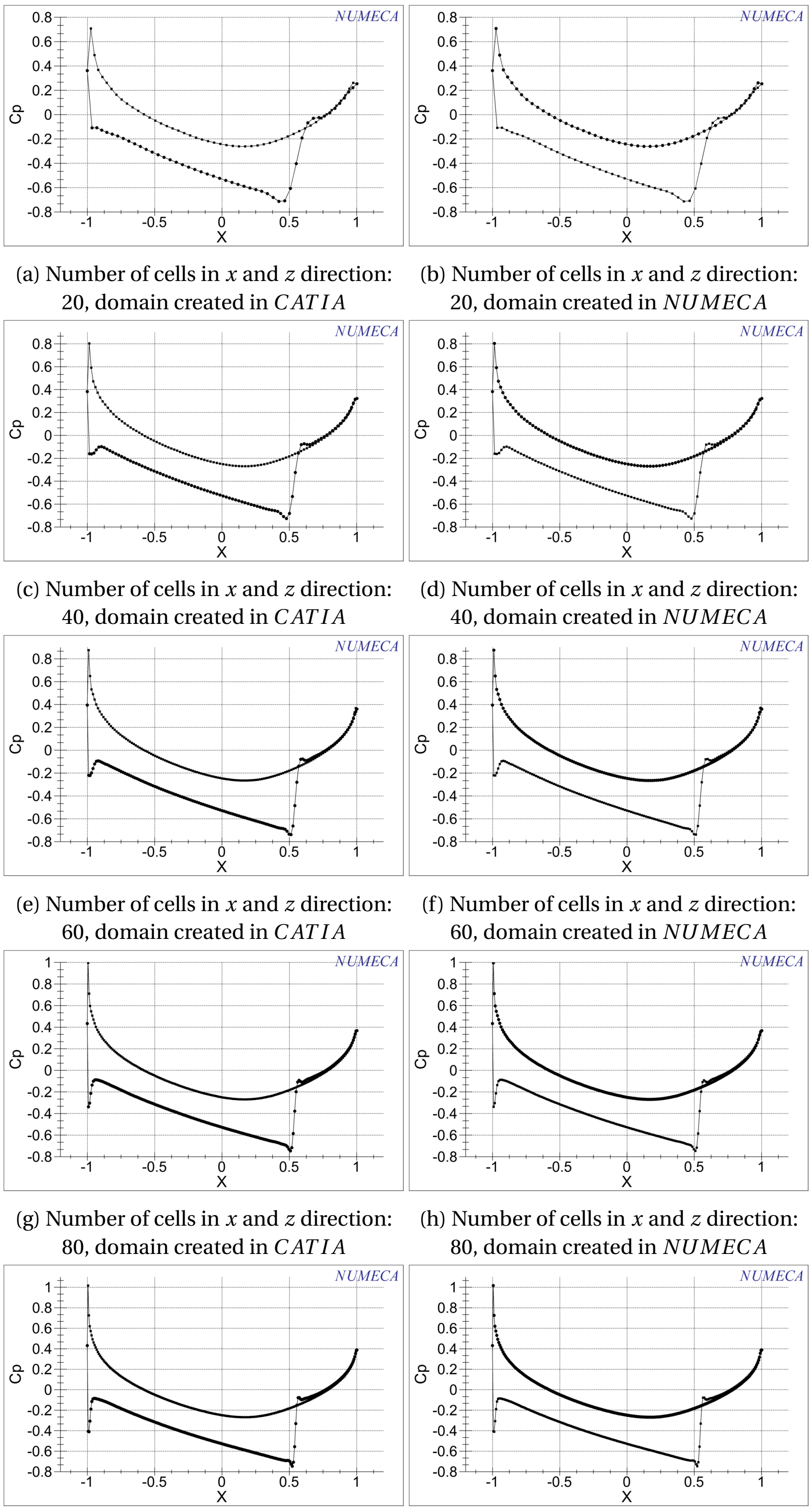


(i) Number of cells in x and z direction: 100, domain created in $CATIA$



(j) Number of cells in x and z direction: 100, domain created in $NUMECA$

Figure G.18: C_P distribution for $\alpha = 1°$ and $M_\infty = 0.86$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under zero angle of attack, and is set under an angle of attack $\alpha = 1°$ in $NUMECA$.



(a) Number of cells in x and z direction: 20, domain created in $CATIA$

(b) Number of cells in x and z direction: 20, domain created in $NUMECA$

(c) Number of cells in x and z direction: 40, domain created in $CATIA$

(d) Number of cells in x and z direction: 40, domain created in $NUMECA$

(e) Number of cells in x and z direction: 60, domain created in $CATIA$

(f) Number of cells in x and z direction: 60, domain created in $NUMECA$

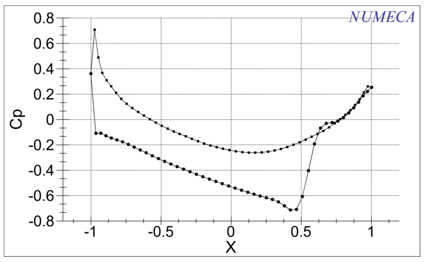
(g) Number of cells in x and z direction: 80, domain created in $CATIA$

(h) Number of cells in x and z direction: 80, domain created in $NUMECA$

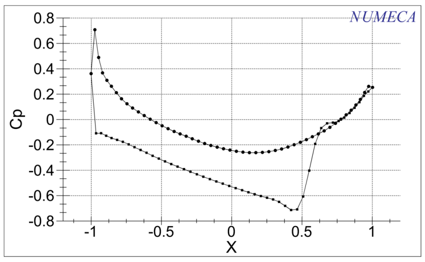
(i) Number of cells in x and z direction: 100, domain created in $CATIA$

(j) Number of cells in x and z direction: 100, domain created in $NUMECA$

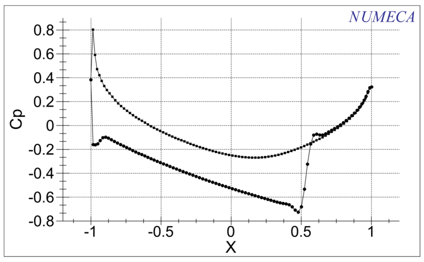
Figure G.19: C_P distribution for $\alpha = 1°$ and $M_\infty = 0.86$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under $\alpha = 1°$.



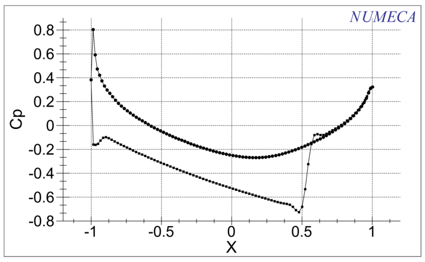
(a) Number of cells in x and z direction: 20, domain created in $CATIA$



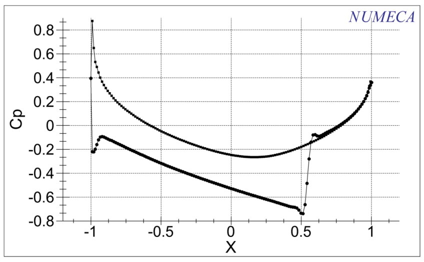
(b) Number of cells in x and z direction: 20, domain created in $NUMECA$



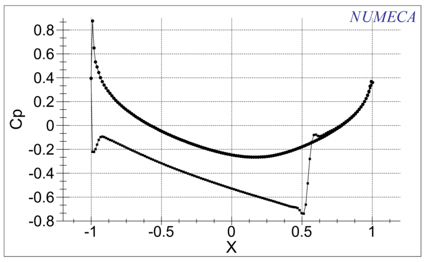
(c) Number of cells in x and z direction: 40, domain created in $CATIA$



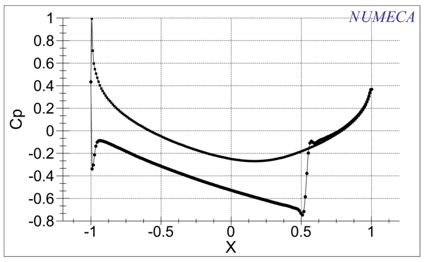
(d) Number of cells in x and z direction: 40, domain created in $NUMECA$



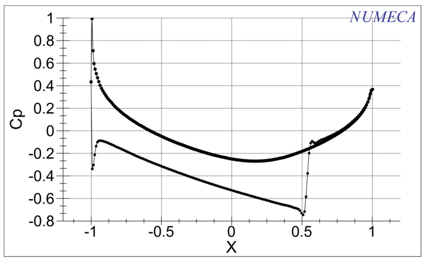
(e) Number of cells in x and z direction: 60, domain created in $CATIA$



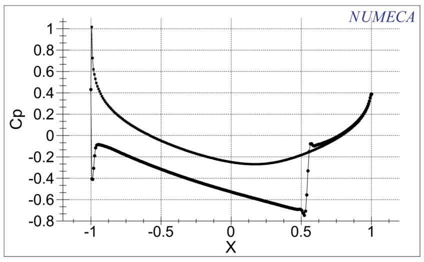
(f) Number of cells in x and z direction: 60, domain created in $NUMECA$



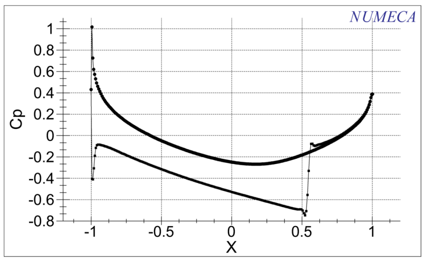
(g) Number of cells in x and z direction: 80, domain created in $CATIA$



(h) Number of cells in x and z direction: 80, domain created in $NUMECA$



(i) Number of cells in x and z direction: 100, domain created in $CATIA$



(j) Number of cells in x and z direction: 100, domain created in $NUMECA$

Figure G.20: C_P distribution for $\alpha = 1^\circ$ and $M_\infty = 0.86$.The left side contains the C_P distributions corresponding to the domain created in $CATIA$, while on the right side the C_P distributions corresponding to the domain created in $NUMECA$ are displayed. The airfoil is created in $CATIA$ under $\alpha = 1^\circ$, and the aerodynamic forces are decomposed in a $x-$ and z-direction.

Bibliography

[1] J. John D. Anderson, *Fundamentals of Aerodynamics* (McGraw-Hill, Avenue of the Americas, New York, NY, 10020, 2001).

[2] S. S. Ray, R. K. Bera, A. Kiliçman, O. P. Agrawal, and Y. Khan, *Analytical and numerical methods for solving partial differential equations and integral equations arising in physical models 2014,* Abstract and Applied Analysis **2015**, 2 (2015).

[3] J. Flores, J. Barton, T. Holst, and T. Pulliam, *Comparison of the full-potential and euler fomrulations for computing transonic airfoil flows,* NASA TM (1984).

[4] C. Raymond, *Definition of a continuum,* Earth & Space Sciences, University of Washington, Course Notes Glaciology ESS511 (2002).

[5] C. A. Lamah, *Application of the Method of Parametric Differentiation to the Solution of the Transonic Integral Equation*, Master of science thesis, Massachusetts Institute of Technology (1982).

[6] *Navier-stokes equations,* https://www.grc.nasa.gov/www/k-12/airplane/nseqs.html.

[7] S. Koziel and X.-S. Yang, *Computational Optimization, Methods and Algorithms* (Springer - Verlag Berlin Heidelberg, 2011).

[8] R. Levicky, *Potential flow,* NYU Tandon School of Engineering - Handout Bio-interfacial Engineering & Diagnostics Group.

[9] *Applied aerodynamics: A digital textbook,* http://docs.desktop.aero/appliedaero/fundamentals/equations.html.

[10] G. Strang, *Calculus* (Wellesley - Cambridge Press, Box 82-279 Wellesley MA 02181, 1991).

[11] N. T. Sivenari and W. L. Harris, *Numerical soltuion of transonic flows by parametric differentiation and integral equation techniques,* AIAA **18** (1980).

[12] A. Bakker, *Applied computational fluid dynamics - meshing,* http://www.bakker.org.

[13] L. N. Sankar, *Small disturbance equation derivation,* Georgia Tech Lecture Notes High Speed Aerodynamics.

[14] H. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics - The Finite Volume Method*, 2nd ed. (Pearson Education Limited, 2007).

[15] V. K. Garg, *Applied Computational Fluid Dynamics* (Marcel Dekker, Inc., 270 Madison Avenue, New York, NY 10016, 1998).

[16] L.N.Sankar and M.J.Smith, *Unsteady transonic potential flow,* Advanced Compressible FLow II (1995).

[17] T. L. Holst, *Transonic flow computations using nonlinear potential methods,* Progress In Aerospace Sciences **1-61** (2000).

[18] N. Sivenari, *Transoninc Flows by Parametric Differentiation and Integral Equation Techniques*, Master's thesis, Massachusetts Institute of Technology (1978).

[19] R. Vos and S. Farokhi, *Introduction to Transonic Aerodynamics* (Springer, 2015).

[20] D. McLean, *Understanding Aerodynamics - Arguing From the Real Physics* (John Wiley & Sons, Ltd., 2013).